Twofish
A Block Encryption Algorithm

Bruce Schneier, Counterpane Systems
John Kelsey, Counterpane Systems
Doug Whiting, Hi/fn
David Wagner, UC Berkeley
Chris Hall, Counterpane Systems
Niels Ferguson, Counterpane Systems

http://www.counterpane.com/twofish.html

Twofish Overview

- 128-bit block Feistel network
- 16 rounds (nominal)
- Pre- and post-whitening
- Key-dependent S-boxes
- Key schedule computable “on-the-fly”
- Wide range of speed/cost tradeoffs.
Twofish Performance Sampler

- Encrypt/decrypt data at 285 clocks/block on a Pentium Pro CPU (i.e., 90 Mbit/sec at 200 MHz), after a 12700 clock key setup.
- Encrypt/decrypt data at 860 clocks/block on a Pentium Pro, after a 1250 clock key setup.
- Almost identical throughput on Pentium CPU.
- Encrypt/decrypt at 26,500 clock cycles/block on 6805 after a 1750 clock key setup (20 Kbit/sec at 4 MHz).
- Encrypt/decrypt at 80 Mbits/sec with 14K gates, 1200 Mbits/sec with 30K gates.

Twofish Round Function Block Diagram
Building Block: Keyed S-boxes

- Twofish builds four bijective key-dependent 8x8-bit S-boxes using a key/permutation "sandwich" (shown for a 128-bit key):

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

  where \( q_0, q_1 \) are two fixed 8-bit permutations.

Why Keyed S-boxes?

- Fixed S-boxes (e.g., DES) allow attackers to study S-boxes and find weak points.
- With key-dependent S-boxes, attacker doesn’t know what the S-boxes are.
- Defense against “unknown attacks.”
- Complexity of keyed S-box depends on the length of the key.
- Downside: takes longer to set up for a key, since S-boxes have to be built for each key.
What About Weak S-boxes?

- Based on two fixed S-boxes with strong properties.
- Keyed S-boxes can be tested for desired security properties:
  - Exhaustive testing for 128-bit keys.
  - Monte Carlo testing for 192- and 265-bit keys.

Building Block: MDS Matrix

- 4x4 matrix multiply over GF(256): $v = Mu$.
- Maximum Distance Separable (MDS) property guarantees that there are at least five nonzero bytes in $u,v$.
- Used as the main diffusion mechanism in Twofish.
- Minimum binary Hamming weight output difference for single byte input difference = 8 bits.
- Preserves MDS property for single byte input difference even after single-bit “rotate right” of $v$ (treated as a 32-bit quantity).
**Building Block: PHT**

- Pseudo-Hadamard Transform: simple, fast, reversible diffusion mechanism.
  
  \[
  a' = a + b \\
  b' = a + b^*2
  \]

- Twofish uses 32-bit PHT on pairs on MDS outputs.

---

**Building Block: 1-bit Rotation**

- Used in each Twofish round to break up the byte-aligned nature of other operations.

- Each of the four 32-bit quantities in the block is used once in each of the eight possible bit positions (mod 8).
Key Schedule

- Design the key schedule for the cipher
- Reuse the same primitives
- Use all key Bytes in the same way
- Make it hard to attack both the S-box and the subkey generation processes

Key Schedule

- Key-dependent S-boxes
- Round subkeys:
  - Based on same construction as key-dependent S-boxes.
  - Can be precomputed or constructed on the fly.
- Every key bit affects every round.
Setup time vs. Throughput

- Basic tradeoffs involve how much of the S-box and MDS matrix multiply we “precompute”.
- The more we precompute, the faster we can encrypt, but setup takes longer.
- Several levels of trade-offs available.

Twofish In Software

- Algorithm uses only “simple” RISC operations and table lookups (i.e., runs equally fast on Pentium and Pentium Pro CPU families).
- Fastest version requires less than 5K bytes of table space per key.
- Code fits easily in cache of modern CPUs (less than 2500 bytes each for encryption and decryption on a Pentium Pro).
- Encryption/decryption throughput is independent of key size (for long key setup).
Key Setup Time vs. Throughput

- Basic tradeoffs involve how much of the S-box and MDS multiply we “precompute”.
- More precomputation: faster encryption, longer setup.
- Many possible levels of precomputation:
  - Zero -- no S-box precomputation
  - Minimal -- precompute part of S-box
  - Partial -- precompute full S-box
  - Full -- precompute full S-box plus MDS
  - Compiled -- Full + subkey-specific compiled code

Twofish in C on Pentium or Pentium Pro

<table>
<thead>
<tr>
<th>Keying Option</th>
<th>RAM bytes</th>
<th>Key Setup</th>
<th>Clocks</th>
<th>Clocks to Encrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>128-bit</td>
<td>192-bit</td>
<td>256-bit</td>
</tr>
<tr>
<td>Full</td>
<td>4500</td>
<td>8000</td>
<td>11200</td>
<td>15700</td>
</tr>
<tr>
<td>Partial</td>
<td>1400</td>
<td>7100</td>
<td>9700</td>
<td>14100</td>
</tr>
<tr>
<td>Minimal</td>
<td>1400</td>
<td>3000</td>
<td>7800</td>
<td>12200</td>
</tr>
<tr>
<td>Zero</td>
<td>200</td>
<td>2450</td>
<td>3200</td>
<td>4000</td>
</tr>
</tbody>
</table>

* clocks to encrypt with 128-bit key (larger keys are slower)
### Twofish in ASM on Pentium or Pentium Pro

<table>
<thead>
<tr>
<th>Keying Option</th>
<th>RAM bytes</th>
<th>Key Setup Clocks</th>
<th>Clocks to Encrypt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>128-bit</td>
<td>192-bit</td>
<td>256-bit</td>
</tr>
<tr>
<td>Compiled</td>
<td>4500</td>
<td>12700</td>
<td>15400</td>
</tr>
<tr>
<td>Full</td>
<td>4500</td>
<td>7800</td>
<td>10700</td>
</tr>
<tr>
<td>Partial</td>
<td>1400</td>
<td>4900</td>
<td>7600</td>
</tr>
<tr>
<td>Minimal</td>
<td>1400</td>
<td>2400</td>
<td>5300</td>
</tr>
<tr>
<td>Zero</td>
<td>200</td>
<td>1250</td>
<td>1600</td>
</tr>
</tbody>
</table>

* clocks to encrypt with 128-bit key (larger keys are slower)

---

### Speed: Key Setup + Encryption

- 16 bytes - 140 clocks/byte
- 64 bytes - 73 clocks/byte
- 256 bytes - 48 clocks/byte
- 1K bytes - 27 clocks/byte
- 4K bytes - 21 clocks/byte
- 16K bytes - 19 clocks/byte
- 64K bytes - 18 clocks/byte
Twofish on Smartcards

- “On-the-fly” key schedule implies no RAM needed to hold subkeys, small key setup time (less than 1/10 of block encrypt time).
- If 160 bytes of RAM available to hold precomputed subkeys, throughput doubles.
- q0,q1 ROM tables = 512 bytes, no RAM needed for S-boxes
- Compute MDS matrix explicitly, with several possible speed-space tradeoffs.

Twofish on a 6805

<table>
<thead>
<tr>
<th>RAM (bytes)</th>
<th>ROM (bytes)</th>
<th>Clocks per block</th>
<th>Throughput @ 4MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>2200</td>
<td>26500</td>
<td>19.3 Kbps</td>
</tr>
<tr>
<td>60</td>
<td>2000</td>
<td>35000</td>
<td>14.6 Kbps</td>
</tr>
<tr>
<td>60</td>
<td>1760</td>
<td>37100</td>
<td>13.8 Kbps</td>
</tr>
</tbody>
</table>

Notes:
- RAM includes 32 bytes for block and 128-bit key.
- ROM includes code and tables
- If key is in EEPROM, then only 36 RAM bytes are required.
Twofish on Even Less Powerful Platforms

- $q_0, q_1$ can be calculated from eight 16-element permutations.
- All subkeys can be calculated as needed.
- No table storage required.
- This will be very slow.

Twofish in Hardware

- *Wide* variety of area-speed tradeoffs.
- Performance estimates assume “commodity” 0.35 micron CMOS.
- No custom design required for high performance (even higher performance possible for custom layouts).
- Highest throughput achievable in ECB or interleaved chaining mode.
Twofish Hardware Estimates

<table>
<thead>
<tr>
<th>Gate count</th>
<th>Clocks per block</th>
<th>Interleave level</th>
<th>Clock Speed</th>
<th>Throughput (Mbits/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>14000</td>
<td>64</td>
<td>1</td>
<td>40 MHz</td>
<td>80</td>
</tr>
<tr>
<td>19000</td>
<td>32</td>
<td>1</td>
<td>40 MHz</td>
<td>160</td>
</tr>
<tr>
<td>23000</td>
<td>16</td>
<td>1</td>
<td>40 MHz</td>
<td>320</td>
</tr>
<tr>
<td>26000</td>
<td>32</td>
<td>2</td>
<td>80 MHz</td>
<td>640</td>
</tr>
<tr>
<td>28000</td>
<td>48</td>
<td>3</td>
<td>120 MHz</td>
<td>960</td>
</tr>
<tr>
<td>30000</td>
<td>64</td>
<td>4</td>
<td>150 MHz</td>
<td>1200</td>
</tr>
<tr>
<td>80000</td>
<td>16</td>
<td>1</td>
<td>80 MHz</td>
<td>640</td>
</tr>
</tbody>
</table>

Cryptanalysis of Twofish

- 5-round Twofish (without pre- and post-whitening) can be broken with $2^{22.5}$ chosen plaintext pairs and $2^{51}$ work.
- 10-round Twofish (without the pre- and post-whitening) can be broken with a chosen-key attack, requiring $2^{32}$ chosen plaintexts and about $2^{11}$ adaptive chosen plaintexts, with $2^{32}$ work.
Conclusions

- Twofish offers a unique combination of:
  - Conservative Design
  - Fast
  - Flexible

Conclusions: Conservative Design

- Based on well-understood primitives:
  - Feistel networks
  - S-boxes
  - More than enough rounds
  - Nothing with obvious timing problems
- Twofish’s design is easy to extend:
  - longer keys
  - up to 124 rounds.
Conclusions: Fast and Flexible

- Twofish is fast: can encrypt data at 17.8 clock cycles per byte on Pentium-class CPUs.
- Twofish is flexible: there are many tradeoffs of key-setup versus encryption speed.
- Twofish is suited for smart cards: minimal table requirements, efficient on 8-bit CPUs.
- Twofish is suited for hardware: many tradeoffs of gates versus speed.

Twofish Source Code

- Source code is available in optimized C and assembly (for Pentium, Pentium Pro, and Pentium II), 6805 assembly, and Java.
  - Other implementations coming soon.
- Available on the Counterpane website:
  - http://www.counterpane.com/twofish.html
- Available outside the U.S. from several websites
  - See the Counterpane website for pointers.
On the CD (and on our website) is a LONG paper describing Twofish, our design rationale, and our analysis.

- It’s easy to read (and we think it’s interesting).
- Additional cryptanalysis will be published as “Twofish Technical Reports.”
  - The first one is available outside.

---

**Twofish Paper (cont.)**

- Section 1: Introduction
- Section 2: Twofish Design Goals
- Section 3: Twofish Building Blocks
- Section 4: Twofish Description
- Section 5: Performance
- Section 6: Twofish Design Philosophy
- Section 7: Design Details
- Section 8: Cryptanalysis
Section 11.8: Family Key Variant

- Allows for non-imperative variants of Twofish, with the addition of a “family key.”
- No family key variant is weaker than the original cipher.
- Related-key attacks against unknown but related family keys should be hard.
- The family key should not merely reorder the 128-by-128-bit permutations provided by the cipher; it should change the set.

More Information

- See our homepage for more details:
  - http://www.counterpane.com/twofish.html
- You can also sign up for email notification of new Twofish results.