# 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

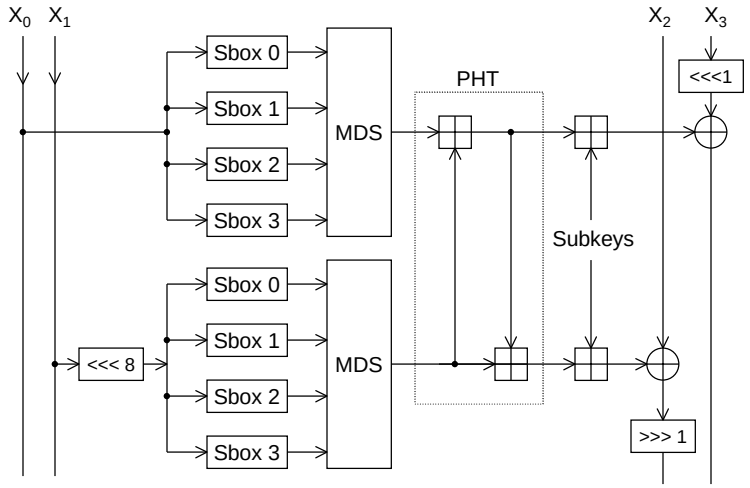
- 128-bit block Feistel network
- 16 rounds (nominal)
- Pre- and post-whitening
- Key-dependent S-boxes
- Key schedule computable “on-the-fly”
- Wide range of speed/cost tradeoffs.


## 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.

COUNTERPANE SYSTEMS hifn 3

## Twofish Round Function Block Diagram




COUNTERPANE SYSTEMS hifn 4

## 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):

$s_0(x) = q_1[q_0[q_0[x] \wedge k_0] \wedge k_1]$

$s_1(x) = q_0[q_0[q_1[x] \wedge k_2] \wedge k_3]$

$s_2(x) = q_1[q_1[q_0[x] \wedge k_4] \wedge k_5]$

$s_3(x) = q_0[q_1[q_1[x] \wedge k_6] \wedge k_7]$

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): $\underline{v} = M\,\underline{u}$ .
- Maximum Distance Separable (MDS) property guarantees that there are at least five nonzero bytes in $\underline{u},\underline{v}$.
- 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 $\underline{v}$ (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

COUNTERPANE SYSTEMS hifn 15

## Twofish in C on Pentium or Pentium Pro

| 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* |

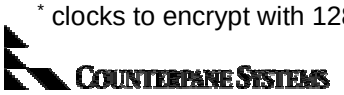
* clocks to encrypt with 128-bit key (larger keys are slower)

COUNTERPANE SYSTEMS hifn 16

## Twofish in ASM on Pentium or Pentium Pro

| Keying Option | RAM bytes | Key Setup Clocks | | | Clocks to Encrypt |
|---|---|---|---|---|---|
| | | 128-bit | 192-bit | 256-bit | |
| Compiled | 4500 | 12700 | 15400 | 18100 | 285 |
| Full | 4500 | 7800 | 10700 | 13500 | 315 |
| Partial | 1400 | 4900 | 7600 | 10500 | 460 |
| Minimal | 1400 | 2400 | 5300 | 8200 | 720 |
| Zero | 200 | 1250 | 1600 | 2000 | 860* |

* 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 clcoks/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

| RAM (bytes) | ROM (bytes) | Clocks per block | Throughput @ 4MHz |
|---|---|---|---|
| 60 | 2200 | 26500 | 19.3 Kbps |
| 60 | 2000 | 35000 | 14.6 Kbps |
| 60 | 1760 | 37100 | 13.8 Kbps |

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

- q0,q1 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

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





## 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.


## Conslusions

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





## Conslusions: 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.



27

## 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.



28

## Twofish Paper

- 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: Crptanalysis

## Section 11.8: Family Key Variant

- Allows for non-imeroperable 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-bu-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.