AES Performance Comparisons

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

COUNTERPANE SYSTEMS

Performance

- There are as many different measures of "performance" as there are platforms to measure it on.
- As a standard, AES will have to perform on all of them.
- We concentrate on the common ones and the general ones.

COUNTERPANE SYSTEMS

hi/fn

How the Candidates Approached Key Lengths and Performance

- Some algorithms are slower for larger keys.
- Some algorithms have slower key setup for larger keys.
- Some algorithms have slower key setup AND encryption for larger keys.
- Some algorithms have constant speeds and key setup for all keys.
- One algorithm has slower key setup for smaller keys!!!

COUNTERPANE SYSTEMS

hifn

Speed Comparison For Different Key Lengths

Algorithm Name	Key Setup	Encryption
Cast-256 [Ada98]	constant	constant
Crypton [Lim98]	constant	constant
DEAL [Knu98]	increasing	128,192: 6 rounds 256: 8 rounds
DFC [GGH+98]	constant	constant
E2 [NTT98]	constant	constant
Frog [GLC98]	constant	constant
HPC [Sch98]	constant	constant
Loki97 [BP98]	decreasing	constant
Magenta [JH98]	increasing	128,192: 6 rounds 256: 8 rounds
Mars [BCD+98]	constant	constant
RC6 [RRS+98]	constant	constant
Rijndael [DR98a]	increasing	128: 10 rounds 192: 12 rounds 256: 14 rounds
SAFER+ [CMK+98]	increasing	128: 8 rounds 192: 12 rounds 256: 14 rounds
Serpent [ABK98a]	constant	constant
Twofish [SKW+98a]	increasing	constant

Speed of AES candidates for different key lengths

COUNTERPANE SYSTEMS

hifn

Speed on Different Processors

- Processor architectures stick around forever.
  - The lesson of the past twenty years is that this high-end always gets better, but the low end never goes away.
- The AES standard will have to work on all processors: small 8-bit embedded CPUs and smart cards, 32-bit CPUs and smart cards, 64-bit CPUs, etc., etc., etc.
- Performance on the low end is much more important that performance on the high end.

Languages

- Performance is only important in assembly language.
- It makes no sense to compare performance in C or Java.
  - Any application which has speed as a requirement will code the encryption algorithm in assembly.
  - An encryption algorithm is an ideal piece of code to hand optimize.
  - Optimized assembly implementations of AES will be available on the Internet.
- If performance is critical, it will be in assembly.

32-Bit Comparisons

- 32-bit machines will be used forever.
- The Intel Pentium Pro/II architecture has some oddities not present in other 32-bit processors, either low-end processors or other high-end processors.
- Most important is performance on generic 32-bit processors.

COUNTERPANE SYSTEMS hifn

Pentium/Pro/II Comparison

Algorithm Name	Key Setup Pentium Pro C (clocks)	Encrypt Pentium Pro C (clocks)	Encrypt Pentium Pro ASM (clocks)	Encrypt Pentium ASM (clocks)
Cast-256	4300	660	600*	600*
Crypton	955	476	345	390
DEAL	4000*	2600	2200	2200
DFC	7200	1700	750	?
E2	2100	720	410	410*
Frog	1386000	2600	?	?
HPC	120000	1600	?	?
Loki97	7500	2150	?	?
Magenta	50	6600	?	?
Mars	4400	390	320*	550*
RC6	1700	260	250	700*
Rijndael	850	440	291	320
SAFER+	4000	1400	800*	1100*
Serpent	2500	1030	900*	1100*
Twofish	8600	400	258	290

AES candidates' performance with 128-bit keys on Pentium-class CPUs

COUNTERPANE SYSTEMS hifn

Things to Note

- Performance varies greatly.
- Some algorithms depend heavily on the particular details of the 32-bit CPU, while others are largely CPU-independent.
- Fastest (in order): Twofish, Rijndael, Crypton, E2, Mars, RC6.
- Note that these speeds are for 128-bit keys.

Bulk Encryption versus Real Speed

- These speeds are for encryption, and do not take into account key setup.
- For bulk encryption this is a reasonable simplification, but not for smaller messages.
- We looked at total performance (key setup + encryption) for different message sizes, for the fastest algorithms (plus Serpent).

Clock Cycles, Pentium

Text Size (bytes)	Crypton	E2	Mars	RC6	Rijndael	Serpent	Twofish
16	73	100	260	146	59	205	175
32	49	63	147	95	39	137	119
64	37	44	91	69	30	103	91
128	30	35	63	57	25	86	70
256	27	30	48	50	22	77	48
512	26	38	41	47	21	73	38
2^{10}	25	27	38	45	21	71	31
2^{11}	25	26	36	45	20	70	25
2^{12}	25	26	35	44	20	69	22
2^{13}	24	26	35	44	20	69	21
2^{14}	24	26	35	44	20	69	20
2^{15}+	24	26	34	44	20	69	19

Clock cycles, per byte, to key and encrypt different text sizes on a Pentium

Clock Cycles, Pentium pro/II

Text Size (bytes)	Crypton	E2	Mars	RC6	Rijndael	Serpent	Twofish
16	70	100	246	118	53	193	132
32	46	63	133	67	36	125	93
64	34	44	76	41	27	90	73
128	28	35	48	28	23	73	64
256	25	30	34	22	20	65	48
512	23	28	27	19	19	61	33
2^{10}	22	27	24	17	19	58	25
2^{11}	22	26	22	16	18	57	20
2^{12}	22	26	21	16	18	57	18
2^{13}	22	26	20	16	18	57	17
2^{14}	22	26	20	16	18	56	17
2^{15}+	22	26	20	16	18	56	16

Clock cycles, per byte, to key and encrypt different text sizes on a Pentium Pro/II

Things to Note

- Algorithms settle down pretty quickly:
  - For a 1K message, speeds are within 15% of fastest speeds.
  - Fastest algorithms for small blocks are Rijndael and Crypton.
  - Note these speeds are for 128-bit keys: Rijndael will be slower with larger keys.

Hash Functions

- Block ciphers can be used as hash functions.
- Hash function constructions require one key setup and one encryption per block hashed.

Hash-Function Comparison

Algorithm Name	Hash Speed Pentium Pro ASM (clocks)	Hash Speed Pentium ASM (clocks)
Cast-256	282*	282*
Crypton	46*	49*
DEAL	349*	349*
DFC	245*	?
E2	100*	100*
Frog	?	?
HPC	?	?
Loki97	?	?
Magenta	?	?
Mars	246*	260*
RC6	118*	146*
Rijndael	32*	34*
SAFER+	193*	212*
Serpent	193*	205*
Twofish	132	175

Hash-function performance, per byte, of AES candidates (128-bit key) on Pentium and Pentium Pro/II

Hash Functions and Key Schedules

- Encryption algorithms do not automatically make good hash functions; they must be analyzed.
- Simple key schedules are much efficient, but may also be much less secure.
- Like all measures in this paper, these ignore security.

Minimum Secure round Performance

- Biham has invented this measure in an attempt to “normalize” the submissions.
- He takes his estimate of the number of rounds that is secure, and then adds a standard two cycles.
- This metric is not necessarily useful or interesting.

COUNTERPANE SYSTEMS

hifn

Minimum Secure round Performance

Algorithm Name	Rounds	Minimal Secure Rounds	MSR Encrypt Pentium Pro ASM (clocks)	MSR Encrypt Pentium ASM (clocks)
Cast-256	48	40	500*	500*
Crypton	12	11	316	358
DEAL	6	9	3300	3300
DFC	8	9	844	?
E2	12	10	342*	342*
Frog	8	?	?	?
HPC	8	?	?	?
Loki97	16	>36	?	?
Magenta	6	>10	?	?
Mars	32	20	200*	344*
RC6	20	20	250	700*
Rijndael	10	8	233	256
SAFER+	8	7	700*	963*
Serpent	32	17	478*	584*
Twofish	16	12	194	218

Minimum secure round performance of AES candidates with 128-bit keys on Pentium-class CPUs

COUNTERPANE SYSTEMS

hifn

Things to Note

- Twofish and Rijndael are the fastest.
- E2 and Mars are also fast.
- RC6 is fast on the Pentium Pro/II only.

64-Bit CPUs

- Again, algorithms that depend heavily on processor architecture are hurt on 64-bit CPUs.
- Our data is for the Dec Alpha.
- DFC is fastest, followed by RIjndael, Twofish, and HPC.
- We have some performance comparison's on the PA-RISC and Merced architectures. These will be discussed during the rump session.

DEC Alpha Comparison

Algorithm Name	Cycles
Cast-256	600
Crypton	408
DEAL	2528*
DFC	304
E2	471
Frog	?
HPC	376
Loki97	?
Magenta	?
Mars	478
RC6	467*
Rijndael	340*
SAFER+	656
Serpent	915
Twofish	360*

AES candidate performance on the DEC Alpha

Smart Cards

- Relative performance on 32-bit smart cards is approximately the same as on the Pentium.
- We concentrated on 8-bit smart cards.
- Numbers in the various papers are not good comparisons, because the assumptions vary greatly.
- Someone needs to code the leading candidates on several standard smart-card chips.

Smart Cards (cont.)

- Memory requirements are essential..
  - Most smart cards sold have 128 to 265 bytes of RAM.
  - All of this RAM is not available to the encryption engine.
- This is not a temporary problem; requirements to fit in a very small software footprint will always be there.
- High end smart cards will get better, but the low end will just get cheaper.

Smart Card RAM Requirements

Algorithm Name	Smart Card RAM (bytes)
Cast-256	60*
Crypton	52*
DEAL	50*
DFC	200
E2	300
Frog	2300+
HPC	?
Loki97	?
Magenta	?
Mars	195*
RC6	210*
Rijndael	52
SAFER+	50*
Serpent	50*
Twofish	60

AES candidates' smart card RAM requirements

Things to Note

- Some AES submissions CANNOT fit on small smart cards: DFC, E2, Mars, RC6. Frog cannot fit on any smart cards.

Hardware Performance

- We did not try to count gates for the different submissions.
- We concentrated on switching speeds in hardware applications.
- An algorithm should encrypt two blocks with two keys in no more time than it takes to encrypt two blocks with the same key.

Hardware Key-Context RAM Requirements

Algorithm Name	Key Context RAM (bytes)
Cast-256	0
Crypton	0
DEAL	0
DFC	0
E2	256
Frog	2300+
HPC	?
Loki97	?
Magenta	?
Mars	160
RC6	176
Rijndael	0
SAFER+	0
Serpent	0
Twofish	0

Hardware key-context RAM requirements

Algorithm-Specific Comments

CAST-256

- 32 bit: Slow. Uniform performance across CPUs.
- Fits in small smart cards; on-the-fly key schedule generation hurts performance.

Crypton

- 32bit: Uniform across CPUs
- Fits in small smart cards.
- Most hardware-friendly algorithm.
- Most hash-function friendly algorithm.

DEAL

- Performance of DES.
- Fits on small smart cards.

DFC

- 32 bit: Multiplication over $2^{64}+13$ slow; hurts performance. Performance strongly depends on CPU.
- Can fit on small smart cards with significant performance penalties.
- Fastest on 64-bit CPUs.
- Key schedule makes decryption slower.

E2

- 32 bit: Uniform across CPUs.
- Expanded key cannot fit on small smart cards.

Frog

- VERY slow key schedule.
- Expanded key cannot fit on any smart card.

HPC

- Heavy use of 64-bit operations hurt performance on other CPUs.
- Expanded key cannot fit on small smart cards.

Loki97

- Use of bit-level permutations hurts performance on all CPUs.
- Large tables makes it hard to fit on smart cards; expanded key cannot fit on small smart cards.

Magenta

- Slowest of all the candidates.
- Fits on small smart cards.

Mars

- 32 bit: Use of data-dependent rotations and modular multiplications hurts performance on most CPUs.
- 64-bit: Again, the use of data-dependent rotations and modular multiplications hurts performance.
- Expanded key cannot fit on small smart cards.

RC6

- 32 bit: Use of data-dependent rotations and modular multiplications hurts performance on most CPUs.
- 64-bit: Again, the use of data-dependent rotations and modular multiplications hurts performance. (A 600 MHz Alpha runs RC6 at a slower absolute speed than a 400 MHz Pentium II.)
- Expanded key cannot fit on small smart cards.

Rijndael

- 32 bit: Uniform across CPUs.
- Fits on small smart cards.
- Very fast on 64-bit CPUs.
- Efficient in hardware.
- Most efficient across all platforms.

SAFER+

- 32-bit: Byte structure hurts performance. Uniform across CPUs.
- Fits on small smart cards.

Serpent

- 32-bit: Slow. Uniform performance across CPUs.
- C performance closest to ASM performance.
- Fits on small smart cards.

Twofish

- 32-bit: Uniform performance across CPUs. Very fast.
- Fits on small smart cards; performance improvements on larger smart cards.
- Efficient in hardware.

Conclusions

- Draw your own.
- Full paper is on: http://www.counterpane.com.