Date: Thu, 15 Apr 1999 14:55:20 +1000
From: Geoff Keating <geoffk@ozemail.com.au>
To: AESFirstRound@nist.gov
Subject: Revised version of my AES2 paper.

Please find below the revised version of the paper I presented at AES2. It includes some corrections and some new data. It is in LaTeX2e format. It is available on the Web in PDF form at

<http://www.ozemail.com.au/~geoffk/aes-6805/paper.pdf>

--
Geoffrey Keating geoffk@ozemail.com.au

Performance Analysis of AES candidates on the 6805 CPU core

Geoffrey Keating

15 April 1999

Abstract

The AES candidate block ciphers Crypton, Mars, RC6, Rijndael, and Serpent were implemented on the Motorola 6805 series 8-bit architecture. Their performance, including ROM and RAM sizes and time to encrypt a single block, was measured in simulation, and the results presented and compared with results for the other NIST cryptography algorithms SHA and DEA and previously published results for AES candidate Twofish. Rijndael was found to be the clear "winner", but the ciphers Crypton, Serpent, and Twofish also performed acceptably.

The NIST is currently evaluating block cipher algorithms as part of its Advanced Encryption Standard development effort. Among the requirements for the AES is that it should be efficient on small 8-bit processors as found in smart cards. Unfortunately, although most of the AES submissions presented performance estimates (sometimes even timings of actual implementations) for some kind of 8-bit processor, there were almost as many 8-bit processors used as there were submissions. In this paper, we hope to rectify this by implementing the most likely AES candidates for a single 8-bit platform, the Motorola 6805 series [3] and measuring their performance in simulation.

The candidates chosen were Crypton, Mars, RC6, Rijndael, and Serpent. The authors of the Twofish AES submission [8] have already implemented Twofish on a 6805 CPU, so we simply quote their results below. These include the fastest five algorithms on the reference platform. There is some discussion below of the next two fastest candidates, CAST and E2.

1 The 6805 processors

The processor family we chose is based around the Motorola HC05 core. There are a large number of variants, all of which use the same instruction set and tim-

Table 1: 68HC05-series processors.

Part	RAM (bytes)	EEPROM (bytes)	ROM (bytes)	typical package	price (USD)
MC68HC705KJ1	64	0	1240	16-pin PDIP	0.84
MC68HC705P6A	176	0	4672	28-pin PDIP	1.89
MC68HC705JJ7	224	0	6160	20-pin PDIP	2.19
MC68HC705C8A	304	0	7774	40-pin PDIP	3.95
MC68HC705C9A	352	0	15932	40-pin PDIP	4.95
MC68HC05SC41	128	3008	6144	sawn wafer	
MC68HC05SC42	384	8192	32040	sawn wafer	

ings but which vary in ROM, RAM, ancillary logic, and packaging.

Table 1 lists some members of the family (on the cheaper side), with their RAM and ROM capacities. The 68HC705 parts have EPROM instead of ROM and are thus suitable for prototyping work. They can often also be programmed by the user's code, which would be useful for (for instance) loading a key schedule in the field, so that the key can be stored in EPROM.

The smartcard variants (the 68HC05SC models) in the family are not available with EPROM, and data books are not available to the casual enquirer. They do use the same CPU core as the general-purpose microcontrollers so the resource requirements for an encryption algorithm should be the same.

The prices are Motorola's North American prices for shipments of small quantities (around 50) to distributors.

The processors can run at cycle rates of up to 5Mhz, depending on the particular model and operating voltage. 2Mhz is a typical maximum.

The 6805 CPU core has four registers: 8-bit accumulator, index register, and stack pointer, and a 16-bit program counter (of which the high bits may be unimplemented). There are four broad classes of instructions:

- Register/Memory instructions, that operate between the accumulator and another value stored in memory, for instance, ADD;
- Read-modify-write instructions, that operate on a memory location, the accumulator, or the index register, for instance, INCX, which increments the index register;
- Branch instructions.
- Bit manipulation instructions, which can set, clear, or test a single bit in memory.

Table 2: Memory requirements of the algorithms.

Cipher	RAM (bytes)			ROM (bytes)	
	Scheduled key	Encrypt	Schedule	Encrypt	Encrypt+ Schedule
Crypton	32	53	52	1101	1349
MARS	23	91	74	4059	4077
MARS (2)	160	33	32	3329	4136
RC6	56	55	38	1342	1374
RC6 (2)	176	24	30	639	933
Rijndael	16	34	0	879	879
Rijndael-d	16	37	1	976	1049
Serpent	16	85	0	1056	1056
Twofish	24	36	?	?	2200
DES	96	21	19	680	1036
SHA	20	98	0	379	419

There are also some instructions that do not fit in these categories.

Some of the features of the core are:

- Single-bit rotates and shifts only;
- An $8 \times 8 \rightarrow 16$-bit multiply instruction, which takes its inputs in the accumulator and index register and returns the result in the same places;
- Faster addressing modes that operate on the first 256 bytes of memory;
- The number of cycles required for any instruction is independent of the value of the data operated on.

The author constructed a simulation environment for this processor for the purposes of this evaluation. The simulator keeps track of the number of cycles executed, so it is possible to obtain exact cycle counts for the algorithms. The simulator, and the algorithms implemented, are available from the author's web site [5].

2 Results

The five ciphers chosen, the DES and the SHA were implemented, tested, and their performance measured. The results are shown in table 2 for the memory requirements and table 3 for the execution time.

Table 3: Execution time of the algorithms.

Cipher	Time (cycles)	
	Encrypt	Schedule
Crypton	31524	5075
MARS	358240	212735
MARS (2)	34163	110066
RC6	105981	78808
RC6 (2)	32731	82167
Rijndael	9464	0
Rijndael-d	13538	2278
Serpent	126074	0
Twofish	26500	1750
DES	17458	12320
SHA	67244	478

In the table, 'Encrypt' is the resources required to encrypt a single 128-bit block (64 bits for DES), or to step the hash function once (hashing 512 bits). 'Schedule' is the resources required to schedule a 128-bit key (56 bits for DES), or to set up the initial hash.

Note that the 'Encrypt' resource requirements for Twofish include the capability of performing decryption; for the other ciphers, this is not the case. Also, all RAM requirements assume that the input data is stored in RAM; for some key schedules this is not necessary.

In general, the algorithms were implemented to fit within 120 bytes of RAM including the key schedule. The algorithms were implemented to take about 1024 bytes of ROM, but flexibility was allowed where this would cause a large speed penalty.

The algorithms were implemented from the specification in the AES submissions. The author did not spend large amounts of time studying each algorithm, and it is quite possible that a significant optimisation may have been missed if it was not discussed in the AES submission. Certainly, there are minor improvements that can be made to the implementations.

2.1 Crypton

Crypton is unexpectedly slow. It turns out that the π transformation takes 1140 clock cycles to execute because of the single accumulator, accounting for 43% of the execution time. The implementation above executes π twelve times.

2.2 RC6

Two variants of RC6 were implemented. The first satisfies the 120-byte RAM requirement, the second would be used if more RAM was available.

RC6 includes a variable-size rotate, which takes 260 clock cycles when implemented to be constant-time; it accounts for about 40% of the encryption time of the first version, and 30% of the second version. About half, on average, of the time taken for the variable-size rotate could be eliminated if a constant-time implementation was unnecessary.

Note that RC6 will be much slower to decrypt using 120 bytes of RAM, because the RC6 key schedule produces the results in a less convenient order. It is likely to take about 700000 clock cycles to decrypt one block.

2.3 MARS

As for RC6, two variants of MARS were implemented. The first satisfies the 120-byte RAM requirement, the second would be used if more RAM was available. The first is very slow because of the need to repeat the expensive key schedule five times.

MARS suffers somewhat from the requirement for a constant-time implementation. More than 25% of the time of the first implementation is spent computing values which will almost never (less than 1% of the time) be needed, to 'fix' key words.

The large S-boxes in MARS preclude implementation in less than 2K of ROM and 120 bytes of RAM. MARS is also sufficiently complex that it would not fit in 1K of ROM even given unlimited RAM (computing S-box entries using SHA).

2.4 Rijndael

Rijndael was the smallest and fastest cipher of those implemented, being more than three times as fast (per byte) as DES. Rijndael also deserves special mention for being implemented in 64 bytes of RAM.

The decryption process in Rijndael is listed as Rijndael-d. It is more expensive, by about 30%, but not enough so that it is slower than the other algorithms. The `InvMixColumn` transformation is coded as

```
tmp = a[0] ^ a[1] ^ a[2] ^ a[3];
xe = xtime(a[0] ^ a[2]); xo = xtime(a[1] ^ a[3]);
tmp2 = xtime(xtime(xo ^ xe)) ^ tmp;
a[0] = xtime(a[0] ^ a[1] ^ xe) ^ tmp2 ^ a[0];
a[1] = xtime(a[1] ^ a[2] ^ xo) ^ tmp2 ^ a[1];
```

```
a[2] = xtime(a[2] ^ a[3] ^ xe) ^ tmp2 ^ a[2];
a[3] = a[0] ^ a[1] ^ a[2] ^ tmp;
```

compared to `MixColumn` coded as

```
tmp = a[0] ^ a[1] ^ a[2] ^ a[3];
a[0] = xtime(a[0] ^ a[1]) ^ tmp ^ a[0];
a[1] = xtime(a[1] ^ a[2]) ^ tmp ^ a[1];
a[2] = xtime(a[2] ^ a[3]) ^ tmp ^ a[2];
a[3] = a[0] ^ a[1] ^ a[2] ^ tmp;
```

and this makes the entire difference in timing and RAM use.

The decryption code shares 266 bytes of tables (the 256-byte forward s-box and a table containing the round constants) with the encryption code; otherwise, the implementations are completely distinct.

The `xtime` primitive, could, if not implemented carefully, allow for a timing attack. Making it constant-time cost about 432 clock cycles for encryption, and 1008 cycles for decryption.

2.5 Serpent

The Serpent implementation uses a bitslice implementation, as this seemed to best fit the available ROM. Despite this, it turned out to be quite slow.

About 40% of the time is used to perform the "affine transformation" in the key schedule. About 30% of the time is spent computing S-boxes. The linear round function takes about 17%.

Serpent takes a more conservative attitude to security than the other algorithms. It seems likely that 16 rounds, instead of 32, would be sufficient for 128-bit keys. If this is the case, the running time would be halved.

The S-box implementations used were based on Brian Gladman's C implementation. It is probable that a 10-20% improvement in running time and code size (of the S-boxes) could be obtained by using S-boxes optimised for this particular processor. The S-boxes account for 525 bytes of ROM and average about 560 cycles each.

In the Serpent submission, [1], timing of about 11000 cycles was estimated for an implementation similar to the one presented. This is of the same order as the time required for 32 fully expanded round S-boxes, without the linear transformation or any key schedule, and does not seem to be achievable in practise.

Serpent does have quite reasonable RAM and ROM requirements. It is possible (and, for decryption, mandatory) to save 32 bytes of RAM by using separate S-box routines for the key schedule and round function, at a cost of an extra 500 bytes of ROM.

2.6 Twofish

The Twofish AES submission [8] also gives a range of alternative Twofish implementations, varying between the one given in the table and one that uses 1760 bytes of ROM and takes about 37100 clock cycles per block (this implementation was not constant-time).

2.7 DES

The DES implementation performs a partial key schedule to fit in 120 bytes of RAM. The Twofish authors quote a DES implementation on the 6805 that takes about 2k code, 23 bytes of RAM, and about 20000 clocks/block. This fits well with our estimates.

The bit permutations in DES were implemented to be constant-time. If this was not necessary an average of 2048 cycles could be saved in encryption.

2.8 CAST

CAST was not implemented. However, CAST seems unlikely to be suitable for implementation on these processors because it requires 4096 bytes of S-box ROM, which is unreasonably large.

2.9 E2

E2 was not fully implemented due to time constraints. The author estimates that the f function in E2 would take about 200 cycles, based on a preliminary implementation. Because E2's key schedule does not produce values in-order, it will need to be run about once every three rounds; thus, the f-function computations in the key schedule alone will account for about 48000 clock cycles. So E2 will not be one of the fastest candidates on this architecture, and it is reasonable to suspect it will be of about the same speed as Serpent.

3 Future Work

Further work would concentrate on the AES second round candidates, particularly any that have been overlooked in this paper, the implemention of other block/key sizes than 128 bits, and the investigation of other tradeoffs between ROM, RAM, and speed.

4 Conclusions

It is difficult to draw a direct conclusion from the above results, because of the question (which is not addressed here) of the relative security of the algorithms.

It does seem than Rijndael performs remarkably well in a quite reasonable amount of ROM (even allowing for the need for separate encryption and decryption algorithms), and, more importantly, in a very restricted amount of RAM. Rijndael also performs quite well on the AES reference platform.

Other suitable Round 2 AES candidates would include Crypton, Serpent, and Twofish. RC6 is probably not suitable because of its very slow decryption. MARS and CAST are not suitable because of their high ROM requirements.

References

[1] Ross Anderson, Eli Biham, and Lars Knudsen. *Serpent: A Proposal for the Advanced Encryption Standard*, 1998. AES submission.

[2] Carolynn Burwick, Don Coppersmith, Edward D'Avignon, Rosario Gennaro, Shai Halevi, Charanjit Jutla, Stephen M. Matyas Jr., Luke O'Connor, Mohammad Peyravi, David Stafford, and Nevenko Zunic. *MARS—a candidate cipher for AES*. IBM Corporation, June 1998. AES submission.

[3] CISC System Design Group, Motorola Inc., Austin, Texas. *68HC705P6A General Release Specification*, July 1996. Rev. 1.0.

[4] Joan Daernen and Vincent Rijmen. *AES Proposal: Rijndael*, June 1998. AES submission.

[5] `http://www.ozemail.com.au/%7Egeoffk/aes-6805/`.

[6] Chae Hoon Lim. *CRYPTON: A New 128-bit Block Cipher*, 1998. AES submission.

[7] Ronald L. Rivest, M.J.B. Robshaw, R. Sidney, and V. L. Yin. *The RC6 Block Cipher*, 1998. AES submission.

[8] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. *Twofish: A 128-Bit Block Cipher*, June 1998. AES submission.