Serpent and Smartcards

Ross Anderson[1] Eli Biham[2] Lars Knudsen[3]

[1] Cambridge University, England; email rja14@cl.cam.ac.uk
[2] Technion, Haifa, Israel; email biham@cs.technion.ac.il
[3] University of Bergen, Norway; email lars.knudsen@ii.uib.no

Abstract. We proposed a new block cipher, Serpent, as a candidate for the Advanced Encryption Standard. This algorithm uses a new structure that simultaneously allows a more rapid avalanche, a more efficient bitslice implementation, and an easy analysis that enables us to demonstrate its security against all known types of attack. Although designed primarily for efficient implementation on Intel Pentium/MMX platforms, it is also suited for implementation on smartcards and other 8-bit processors. In this note we describe why. We also describe why many other candidates are not suitable.

1 Introduction

For many applications, the Data Encryption Standard algorithm is nearing the end of its useful life. Its 56-bit key is too small, as the Electronic Frontier Foundation's keysearch machine has vividly demonstrated [15]. Although triple-DES can solve the key length problem, the DES algorithm was also designed primarily for hardware encryption, yet the great majority of applications that use it today implement it in software, where it is relatively inefficient.

For these reasons, the US National Institute of Standards and Technology has issued a call for a successor algorithm, to be called the *Advanced Encryption Standard* or *AES*. The AES call is aimed at Intel Pentium processors, but one of the evaluation criteria is that the algorithm should have the flexibility to be implemented on other platforms. As the majority of fielded block cipher systems use smartcards and other 8-bit processors and are found in applications such as electronic banking [2, 3], prepayment utility meters [4] and pay-TV subscription enforcement systems [6], the performance of the Advanced Encryption Standard on such platforms will be critical to its acceptance worldwide.

2 Serpent

In [5], we presented Serpent, a candidate for AES. Our design philosophy was highly conservative; we did not feel it appropriate to use novel and untested ideas in a cipher which, if accepted after a short review period, will be used to protect enormous volumes of financial transactions, health records and government information over a period of decades. We therefore devised a new structure which enabled us to use the S-boxes from DES, which have been studied intensely for many years and whose properties are thus well understood, in a manner suitable for efficient implementation on modern processors while simultaneously allowing us to apply the extensive analysis already done on DES.

The resulting design gave an algorithm (which we call Serpent-0) that was as fast as DES and yet resists all currently known attacks at least as well as triple-DES. It was published at the 5th International Workshop on Fast Software Encryption [9] in order to give the maximum possible time for public review. After that we worked hard to strengthen the algorithm and improve its performance, especially on 8-bit processors. The final submission has new, stronger, S-boxes and a slightly different key schedule. We can now show that our design (which we call Serpent-1, or more briefly, Serpent) resists all known attacks, including those based on both differential [10] and linear [20] techniques, with very generous safety margins. The improved key schedule ensures that it is also highly suitable for smartcards; in this context, we acknowledge useful discussions with Craig Clapp and Markus Kuhn on the hardware and low-memory implementation costs respectively.

The Serpent ciphers were inspired by recent ideas for bitslice implementation of ciphers [8]. However, unlike (say) the bitslice implementation of DES, which encrypts 64 different blocks in parallel in order to gain extra speed, Serpent is designed to allow a single block to be encrypted efficiently by bitslicing. This allows the usual modes of operations to be used, so there is no need to change the environment to gain the extra speed.

The basic idea is that a single round of Serpent has a key mixing step followed by 32 identical 4-bit S-boxes in parallel and finally a linear transformation (see [5] for full details). This conventional design enables the existing block cipher analysis machinery to be applied with ease. The radical performance improvement comes from the fact that, on a 32 bit processor such as the Pentium, one can implement each set of 32 S-boxes by bitslicing. Just as one can use a 1-bit processor to implement an algorithm such as DES by executing a hardware description of it (using one logical instruction to emulate each gate), so one can also use a 32-bit processor to compute 32 different DES blocks in parallel — in effect, using the CPU as a 32-way SIMD machine.

Serpent is optimised for bitsliced processing, and compared with DES, the performance gain is significant. 16-round Serpent resists all currently known attacks at least as well as triple-DES, and yet runs three times as quickly as single DES (about 50 Mbit/sec versus 15 Mbit/sec on a 200MHz Pentium). But as AES may persist for 25 years as a standard and a further 25 years in legacy systems, and will have to protect data for the lifetime of individuals thereafter, it will need a cover time of at least a century. In order to ensure that Serpent could withstand advances in both engineering and cryptanalysis for that length of time, we proposed 32 rounds to put its security beyond question. This gives a cipher that is faster than DES (i.e., about 26 Mbit/sec on a 200MHz Pentium Pro) but very much more secure than 3DES.

This is all very well for Pentium software, but what about smartcards?

3 Performance in Smartcard Applications

In a typical smartcard application, one performs a small number of encryptions or decryptions (typically 1–10 of them) in a transaction which should complete in well under a second in order not to inconvenience the user. In some applications, there is a smartcard at each end — one held by the customer, and another embedded in the merchant terminal, or parking meter, or whatever. So an encryption algorithm for use in smartcards should take at most a few tens of milliseconds to encrypt or decrypt a block. But that is not the only constraint.

3.1 EEPROM usage versus speed

The most critical performance requirement is often memory usage rather than raw speed. In prepayment electricity meters, for example, code size was a determining factor in block cipher choice [4]. Fortunately, there is an implementation strategy for Serpent which yields a small code size. This is a bitslice implementation which uses table lookups for each S-box, thus avoiding the code size of the Boolean expression of the S-box.

An Ada implementation that uses this strategy indicates a code size of just under 1K and a computational cost of 34,000 clock cycles. Thus on the 3.5 MHz 6805 processor typical of low-cost smartcards, we expect a throughput of about 100 encryptions per second or 12.8 Kbit/s. A full bitslice implementation — with an unrolled Boolean implementation of the S-boxes rather than table lookup — should occupy 2K of memory but take 11,000 clock cycles and thus deliver 40.7 Kbit/s. These will be the relevant figures for some microcontroller applications; but as most smartcards do not have enough RAM to store Serpent's round keys, these will have to computed afresh with each block. Fortunately, round key computation takes only as many cycles as a single encryption, so the throughput figures for smartcards are about 6.4 and 20 Kbit/sec for 1K and 2K code size respectively.

This is still much faster than needed in most applications. In passing, we note that in applications where we only encrypt and never decrypt (such as some pay-TV and road toll tag systems), we can save 256 bytes from the 1K code size of the compact implementation by not including the inverse S-boxes.

3.2 RAM usage

RAM requirements are also a critical factor in mass market applications as they make the difference between a 50 cent processor and a 90 cent processor. The sales projections of Gemplus, which has 40% of the smartcard market, indicate that in 2003 the world market will be about 1,885 million cards with small RAM and shared-key cryptography, 400 million with large RAM and shared key cryptography, and 405 million with large RAM and public key cryptography [21]. Serpent can be implemented using under 80 bytes of RAM (current text, new text, prekey, working space); this is close to the minimum for a cipher with 128 bit blocks and a 256 byte key. It means that of the 128 bytes typically available in low cost smartcards, at least 48 bytes are left for the calling application.

Other candidates are not as frugal. For example, RC6 [23] takes 176 bytes for the round keys, 32 for the current and next text, plus storage for a number of transient variables, so it could be implemented in large RAM cards but not in low cost ones. MARS [13] is similar, needing 160 bytes for the round keys. (The MARS round keys can in theory be computed on the fly, but at the cost of a huge performance penalty; doing this appears to reduce the throughput by about an order of magnitude.)

3.3 Key agility

Key set-up time is critical in many smartcard applications. The UEPS system described at Cardis 94 is fairly typical, and has since been adapted by VISA as its COPAC product [2, 3]. UEPS uses a three pass protocol in which a customer card transfers value to a merchant card. This involves two block encryptions by the customer card, then two decryptions and two encryptions by the merchant card, then two decryptions, two MAC computations and two encryptions in the customer card, and finally two decryptions by the merchant card. This makes a total of fourteen block cipher operations. But there are ten key set-up operations, and unless there is enough RAM to cache round keys, there are fourteen of them. If the protocol were re-implemented with AES, the longer key and block sizes might allow this number to be reduced to nine.

With Serpent, each key setup costs the same as one encryption, so fourteen of each will cost less than 300ms, and nine of each will cost less then 200ms, on a 3.5 MHz card. However, E2 [22] requires the equivalent of three encryptions per key setup, while LOKI-97 requires over 400,000 instructions to set up a key on an 8-bit processor [12] — which at an average of 8 clock cycles per instruction would take almost a second at 3.5 Mhz. This would be unacceptable.

On-the-fly round key computation introduces asymmetries as well as delays. For example, CAST-256 computes its round keys using a separate block cipher, so in order to decrypt, implementers would have to execute the whole key schedule and then reverse it a step at a time. Unless a separate decryption key is available, decryption should take about 246,000 operations versus 136,000 for encryption (based on the figures of [1]).

3.4 Embedded crypto ASIC

In some smartcards, such as the Sky-TV series 10 subscriber card, a hardware encryption circuit has been bundled along with the CPU. For such applications, Serpent has a hardware implementation that iteratively applies one round at a time, while computing the S-boxes on the fly from table lookup. We estimate that the number of gates required for this is about 4,500 — roughly the same as the Sky-10 crypto ASIC.

It may be argued that given leading-edge fabrication technologies, custom cryptoprocessors containing tens of thousands of gates can be embedded in smartcard CPUs. This again ignores the realities of the smartcard industry: what can be done in theory, and what is economic in the mass market, are different animals. In particular, we know of no smartcard containing a crypto ASIC for symmetric key cryptography with more than 10,000 gates.

4 Smartcard specific security issues

There has been much publicity recently about specialist attacks on smartcards, and ways in which these can interact with the design of algorithms and protocols [7]. In this section (which has been significantly expanded in the period between the conference and the final paper deadline) we will try to deal with these issues.

The main attacks on smartcards were described in [6] and are based on microprobing, timing analysis, power analysis and fault induction.

Most fielded attacks that did not exploit a protocol failure or other gross design flaw started off with an invasive attack on the card using microprobing techniques [6, 18]. Once the card software had been extracted, it was usually possible to design non-invasive attacks that could be carried out quickly and with low cost equipment.

One example is the timing attack [16]. Here, one uses the fact that the time some encryption algorithms take to execute depends on the key, on the plaintext, or on some intermediate value which can help an attacker derive either the plaintext or the key. In the AES context, one possible timing attack would be on the data dependent rotations used, for example, in RC6; most smartcard processors support only a single bit shift, so a variable shift will typically be implemented as multiple single bit shifts in a loop. It is possible to design a constant time implementation, but this imposes a performance penalty.

In the case of Serpent, the number of instructions used to encrypt or decrypt does not depend on either the data or the key, and even cache accesses cannot help the attacker as we do not access different locations in memory for different plaintexts or keys. It follows that timing attacks are not applicable.

The second class of noninvasive attacks, known to the smartcard community as power analysis and to the military under the codenames 'Tempest', 'Hijack' and 'Nonstop', utilises the fact that many encryption devices leak key material electromagnetically, whether by variations in their power consumption [17, 24] or by compromising electromagnetic radiation [19].

Two relevant papers were published at the second AES candidate conference. In the first, Chari et al. report a successful power analysis attack on the 6805 reference code for Twofish, implemented on an ST16 smartcard [14]; this exploits the key whitening process. In the second, Biham and Shamir observe that if the Hamming weight of the current data word on the bus can be observed, then the key schedule of many AES candidate algorithms can be attacked.

This is only the tip of the iceberg. Firstly, different instructions have different power signatures, and so algorithms where the sequence of executed opcodes is a function of the data or the key (such as RC6 and MARS) might be particularly vulnerable. Secondly, the easiest parameter to measure on the typical card using power analysis appears to be the number of bits on the bus which change polarity at a given clock edge. This is not quite the same as parity, but given knowledge of the opcode sequence, still amounts to about one bit of information about each data byte. Thirdly, values measured using these techniques are noisy, and so a real attack is less like simple linear algebra and more like the reconstruction of a noisy shift register sequence (this may give an advantage to algorithms such as Serpent where the linear relations in the key schedule are of relatively high Hamming weight). Fourthly, a number of defensive programming techniques are being increasingly used as a first line of defence against power analysis; the computation can be obscured in various ways such as by intercalating it with a pseudorandom computation. However, these techniques carry a performance cost – the typical slowdown can vary from two to five times.

Fault induction is the last main attack technique, and typically involves inserting transients ('glitches') into the clock or power lines to the card in order to cause targeted instructions to fail. In a typical attack, the power supply might be reduced to 2V for 40nS either side of the clock edge that triggers a jump instruction, which in some processors causes the instruction to become a no-op [6]. In this way, an access control check can be bypassed, or an output loop extended. This will usually be the preferred attack, but if for some reason the attacker wants to extract a key directly, then he can reduce the number of rounds in the cipher progressively and extract the round keys directly [7]. None of the AES candidates appears to offer much more resistance to this kind of attack than any other.

The main defences against attacks on smartcards are the concern of electrical and chemical engineers (e.g., refractory chip coatings, top layer metal grids, glitch and clock frequency sensors that reset the chip [18]), while secondary defences include defensive programming techniques that minimise the number of single instructions which are possible points of failure [6]. We suggest that a useful assessment of the AES finalists will consist of implementing them using defensive programming techniques, measuring their performance and doing actual attacks on them. We believe that as most operations in Serpent use all bits in their operands actively, as the avalanche is fast and as the code is independent of the data and the key, Serpent will do well in this exercise.

5 Conclusion

Serpent was engineered to satisfy the AES requirements. It is faster than DES, and we believe that it would still be as secure as three-key triple-DES against all currently known attacks if its number of rounds were reduced by half to sixteen. Using twice the needed number of rounds gives a margin of safety which we believe to be prudent practice in a cipher with a cover time of a century.

Serpent's good performance comes from allowing an efficient bitslice implementation on a range of processors, including the market leading Intel/MMX and compatible chips. However, the 32-bit performance was not bought at the expense of imposing penalties on designers of low cost smartcard applications. The same cannot be said for many other AES candidates: some of them cannot be implemented on low cost smartcards at all, while many more would impose such constraints on system developers that they would undermine the goals of the AES programme.

Further information on Serpent, including the AES submission, the FSE and AES conference papers, this paper, and implementations in various languages, can be found on the Serpent home page at:

`http://www.cl.cam.ac.uk/~rja14/serpent.html`

References

1. CM Adams, "The CAST-256 Encryption Algorithm", available online from: `http://www.entrust.com/resources/pdf/cast-256.pdf`
2. RJ Anderson, "UEPS — a Second Generation Electronic Wallet" in *Computer Security — ESORICS 92*, Springer LNCS vol 648 pp 411–418
3. RJ Anderson, "Making Smartcard Systems Robust", in *Proceedings of Cardis 94* (Lille, October 1994) pp 1–14
4. RJ Anderson, SJ Bezuidenhoudt, "On the Reliability of Electronic Payment Systems", in *IEEE Transactions on Software Engineering* v 22 no 5 (May 1996) pp 294–301
5. RJ Anderson, E Biham, LR Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard", available from `http://www.cl.cam.ac.uk/~rja14/serpent.html`
6. RJ Anderson, MG Kuhn, "Tamper Resistance — a Cautionary Note", in *The Second USENIX Workshop on Electronic Commerce Proceedings* (Nov 1996) pp 1–11
7. RJ Anderson, MG Kuhn, "Low Cost Attacks on Tamper Resistant Devices" in *Security Protocols – Proceedings of the 5th International Workshop* (1997) Springer LNCS vol 1361 pp 125–136
8. E Biham, "A Fast New DES Implementation in Software", in *Fast Software Encryption — 4th International Workshop, FSE '97*, Springer LNCS v 1267 pp 260–271
9. E Biham, RJ Anderson, LR Knudsen, "Serpent: A New Block Cipher Proposal", in *Fast Software Encryption — FSE 98*, Springer LNCS vol 1372 pp 222–238
10. E Biham, A Shamir, *'Differential Cryptanalysis of the Data Encryption Standard'* (Springer 1993)
11. E Biham, A Shamir, "Power Analysis of the Key Scheduling of the AES Candidates", *AES Second Candidate Conference*, `http://csrc.nist.gov/encryption/aes/round1/conf2/papers/papers/biham3.pdf`
12. L Brown, J Pieprzyk, "Introducing the new LOKI97 Block Cipher", `http://www.adfa.oz.au/~lpb/research/loki97/`
13. C Burwick, D Coppersmith, E D'Avignon, R Gennaro, S Halevi, C Jutla, SM Matyas Jr., L O'Connor, M Peyravian, D Safford, N Zunic, "MARS — a candidate cipher for AES", July 17th 1998; `http://www.research.ibm.com/security/mars.html`

14. S Chari, C Jutla, JR Rao, P Rohatgi, "A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards", *AES Second Candidate Conference*, `http://csrc.nist.gov/encryption/aes/round1/conf2/papers/chari.pdf`
15. Electronic Frontier Foundation, *'Cracking DES — Secrets of Encryption Research, Wiretap Politics & Chip Design'*, O'Reilly (July 98) ISBN 1-56592-520-3
16. PC Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", in *Advances in Cryptology — Crypto 96*, Springer LNCS v 1109 pp 104–113
17. PC Kocher, "Differential Power Analysis", available from: `http://www.cryptography.com/dpa/`
18. O Kömmerling, MG Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors", USENIX Workshop on Smartcard Technology, Chicago, Illinois, USA (to appear); `http://www.cl.cam.ac.uk/~mgk25/sc99-tamper.pdf`
19. MG Kuhn, RJ Anderson, "Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations", in *Proceedings of the Second International Workshop on Information Hiding* (Portland, Apr 98), Springer LNCS vol 1525 pp 126–143
20. M Matsui, "Linear Cryptanalysis Method for DES Cipher", in *Advances in Cryptology — Eurocrypt 93*, Springer LNCS v 765 pp 386–397
21. D Naccache, *private communication*, 17 Aug 98
22. Nippon Telegraph and Telephone Corporation, "The 128-bit Block Cipher E2", July 1998, `http://info.isl.ntt.co.jp/e2/`
23. RL Rivest, MJB Robshaw, R Sidney, YL Lin, "The RC6 Block Cipher", July 1998, `http://theory.lcs.mit.edu/~rivest/publications.html`
24. P Wright, *'Spycatcher — The Candid Autobiography of a Senior Intelligence Officer'*, William Heinemann Australia, 1987, ISBN 0-85561-098-0