Cover Sheet

Name of submitted algorithm	:	SERPENT
Principal submitter	:	Ross J. Anderson

Tel: +44 1223 334733
Fax: +44 1223 334678
email: Ross.Anderson@cl.cam.ac.uk
URL: www.cl.cam.ac.uk/users/rja14

University of Cambridge
Computer Laboratory
Pembroke Street
Cambridge CB2 3QG
England

Auxiliary submitters	:	Eli Biham
	:	Lars R Knudsen
Inventors, and owners of the patent application for the algorithm	:	Ross J. Anderson
	:	Eli Biham
	:	Lars R. Knudsen

Signature

Ross J. Anderson

2.B. Algorithm Specification and Supporting Documentation

2.B.1 Written specification

This is contained in the attached paper "Serpent: A Proposal for the Advanced Encryption Standard of which an initial version (entitled "Serpent: A new block cipher proposal") appeared in *Fast Software Encryption — proceedings of fifth international workshop*, Springer Lecture Notes in Computer Science v 1372 pp 222–238

2.B.2 Computational efficiency

Computational efficiency estimates are included in the paper. We summarise them here.

1. The number of cycles required to encrypt or decrypt is independent of the key size; all keys shorter than 256 bits are padded to that length and used as a 256 bit key.

2. It takes about 1830–1940 instructions to encrypt 128 bits using Serpent. The exact figure depends on the processor used, and encryption is very slightly faster than decryption in the optimised implementation, although it needs a few more instructions. On a Pentium processor, the 1940 instructions required take about 1738 clock cycles. That the clock cycle count is less than the instruction count is due to efficient use of pipelining. The time required to set up or change a key is approximately the time required to perform one encryption; there is no extra time required to set up the algorithm such as by building internal tables.

3. On a 133MHz Pentium/MMX processor running Linux, our optimised C implementation achieves an encryption throughput of 9,791,000 bits per second which corresponds to about 1738 clock cycles per block. We therefore expect that on a 200 MHz Pentium as specified in section 6.B, Serpent will have a throughput of about 14.7 Mbit/sec unless the test software or choice of operating system imposes a significant performance penalty.

4. Our Java implementation performs 10,000 encryptions in 3.3 seconds on a 133 MHz Pentium MMX. This translates to 388 kbit/s, and we expect 583 kbit/s on the NIST 200 MHz machine (though just-in-time compilation should speed things up). In each case, this translates to about 44,000 clock cycles per block.

5. On 8-bit processors, a compact implementation should take less than 1Kbyte but 34,000 clock cycles, giving a throughput of about 12.8 kbit/s on a 3.5 MHz 6805 as used in low-cost smartcards. An implementation optimised for speed should take 11,000 clock cycles and thus deliver 40.7 kbit/s, but occupy about 2K of memory. This is comparable with common DES implementations and more than adequate for typical applications.

6. We expect that a fully pipelined hardware implementation would take about 100,000 gates. If piplined key scheduling is not a requirement (it would almost never be), then this falls to 67,000. If pipelining of eight stages at a time is adequate (as it would usually be) then the gate count falls to about 18,000. With no pipelining, it falls to about 4,500. In addition it is possible to construct highly efficient hardware/software versions of our algorithm by adding an extra instruction called 'BITSLICE' to existing processors at a cost of about 3,200 gates.

2.B.3 KAT and MCT Tests

These are included on diskette

2.B.4 Expected strength

The following workload figures are for the best attack that we expect to be possible on our cipher:

Block Size	Key Size	Workload	Type of attack	Chosen/Known Texts
128	128	2^{128}	Exhaustive Search	1
128	192	2^{192}	Exhaustive Search	2
128	256	2^{256}	Exhaustive Search	2

We do not believe that either differential or linear attacks are possible.

Indeed, with the current state of the art we do not believe that they give useful attacks even if Serpent is reduced from 32 rounds to 16. In any case, we advise users to change keys well before birthday attacks are possible (i.e. well before 2^{64} texts have been encrypted). In that case, no shortcut attacks are possible using any techniques known to us and we believe that any such attack would require a major theoretical breakthrough.

We decided to use twice the number of rounds that are necessary to guard against all presently known attacks, because if DES serves as a reasonable guide, the cipher selected for AES may have to withstand attack for 50 years or more (25 years as a standard, and 25 years in legacy systems). If Moore's law continues to hold, then such attacks might involve hardware capable of searches in excess of 2^{100} as well as considerable advances in the techniques of cryptanalysis.

2.B.5 Resistance against known attacks

Our cipher resists all known attacks. The details are in the paper which also includes references.

2.B.6 Advantages and limitations

An extremely important advantage of Serpent is the bitslice design technique which enables us to use the S-boxes from DES, and thus benefit from the extensive analysis already done on DES, while avoiding the poor software performance from which DES suffers. The result is a cipher which, on the main target platform, is as fast as single DES while being more secure than three-key triple-DES.

In order to design a cipher to withstand 50 years of attacks, we did not believe it prudent to use novel and untested ideas, especially as the AES algorithm will be accepted after a short review period and used to protect enormous volumes of financial transactions, health records and government information.

To design a cipher using only well understood components and techniques, while simultaneously delivering enough performance improvement to justify a move away from triple-DES, required an innovative idea. However this innovative idea had to be such as to enable the existing cryptanalytic apparatus to be used, rather than a new construction whose weakness might only become apparent to the cryptanalytic community after some years. Our bitslice construction is that idea.

Other more specific advantages of Serpent include:

a. When Serpent is implemented on the industry standard architecture (Intel Pentium/MMX or Pentium II) it is possible to perform simultaneous CBC encryption and MAC computation on a message using different keys.

b. Serpent can also be used in standard modes of operation, such as in Output Feedback Mode to give a very efficient and strong pseudorandom bit generator for stream cipher and other applications.

c. Serpent provides adequate performance as a hash function. We are aware of no way in which it differs from a pseudorandom permutation, and it will thus provide a one-way collision-free hash function when used in feedforward mode, subject to the limits imposed by its block length. On many modern processors, it allows several streams of data to be hashed simultaneously.

d. Serpent runs quickly enough on smartcards and other 8-bit processors for the key management, value transfer and similar protocols commonly implemented on such devices. The fact that key set up or change requires computation equivalent to only one encryption makes it also suitable for applications such as ATM encryption where key agility may be required.

e. Serpent can be implemented in hardware in a variety of ways depending on the precise performance requirements. There is a particularly efficient way to add it to an existing CPU by means of a special purpose instruction that speeds up software implementations considerably. Serpent also supports highly pipelined hardware implementations for applications such as ATM/B-ISDN, and very compact hardware implementations for applications such as pay-TV decoder ASICs.

2.C Magnetic Data

2.C.1 Reference Implementation

Enclosed.

2.C.2 Mathematically Optimized Implementation

ANSI C and Java enclosed.

2.C.3 Test values

Enclosed.

2.D Intellectual Property Statements

2.D.1 Statement by the Submitter

Enclosed.

2.D.2 Statement by Patent Application Owner

Enclosed.

2.D.3 Statement by the owners of the Reference and Mathematically Optimized Implementations

Enclosed.