Performance of Symmetric Ciphers and One-way Hash Functions

Michael Roe

Cambridge University Computer Laboratory

1 Rationale

An alarmingly large number of different cryptosystems have been proposed for use with Internet Privacy Enhanced Mail [1]. These include the hash functions MD-n (for various values of n) and a combinatorial explosion of new DES modes.

The criteria for choosing which algorithm is most suitable for a particular application include vulnerability to attacks, performance, and availability of hardware implementations. There is already extensive literature on vulnerabilities. The objective of this paper is to provide some concrete information on performance, in order that an algorithm with a suitable combination of both security and performance may be chosen.

2 Experimental Approach

For each class of algorithm (one-way hash functions; symmetric ciphers; and asymmetric ciphers) a generic application program interface (API) was defined that could be used to access any algorithm in that class. For each algorithm that we tested, an implementation was written conforming to this API.

This met two objectives. Firstly, a common API allows different algorithms to be substituted into real applications with a minimum of effort. Secondly, for performance measurements, using a common API reduces the influence of the API on perceived performance. Different APIs vary greatly in efficiency, and so measurements of different algorithms with different APIs are hard to compare in a fair manner.

For each class of algorithm, three generic programs were written; one to test correctness, and two to measure performance. For each particular algorithm, these generic programs were linked with a subroutine library containing the specific algorithms, and then run.

The correctness test reads in from a file test cases taken from the algorithm specification, and checks that the implementation under test gives the correct results. (The published specifications of MD2, MD4, MD5, SHA and RIPE-MD all contain test cases).

The first performance test for hash algorithms measures the time taken to hash a message containing 6,400,000 zero octets. This is implemented by clearing a 64 character buffer, calling the routine to initialise hashing, calling the routine to hash a buffer 100 000 times, and then calling the routine to finish hashing.

The second performance test for hash algorithms measures the time taken to hash a 64 character message. This is implemented by clearing a 64 character buffer, and then 10,000 times calling the routines to start hashing, hash the buffer and finish hashing.

In general, the time taken to hash a message is a constant (the time needed to call the start and stop hashing routines) plus an amount proportional to the length of the message. Thus, the time taken to hash a message of any particular length can be estimated by interpolating from between the above two measurements.

The first performance test for symmetric key algorithms measures the time taken to encrypt a message of 6,400,000 octets with a fixed key. The second performance test measures the time taken to encrypt a single data block under 10,000 different keys. The size of a block depends upon the particular algorithm, but is typically 64 bits (8 octets).

The time taken to encrypt one or more message under a fixed key is approximately a constant (the time taken to expand the key into a *key schedule* of subkeys) plus an amount proportional to the total number of data blocks to be encrypted. Thus, the time taken to encrypt a message of any particular length can be estimated by interpolating the results of the above two tests.

3 Apparatus

These measurements where carried out on Unix workstations from two different manufacturers:

1. A DEC 3000/400 "Sandpiper". This uses an Alpha CPU clocked at 133 MHz. There is 2 times 8KB of primary cache memory and 512KB of secondary cache.
2. A Sun SparcStation ELC. This uses a Sparc CPU clocked at 33 MHz.

Both CPUs are RISC processors with a large number of registers. On other types of CPU, such as those with a small number of registers or where the registers are only 8 bits wide, the relative speeds of these algorithms may be significantly different.

For example, the SAFER-K64 algorithm uses only 8-bit operations, while most of the other algorithms use 32 bit operations. It is likely that on an 8 bit processor SAFER-K64 will have a much greater speed advantage over the other algorithms. The NIST Secure Hash Algorithm involves a large number of temporary variables; it is likely that on machines with only a small number of registers this algorithm will be at a greater disadvantage.

4 Hash Algorithms

The algorithms chosen were MD2 [4], MD4 [10], MD5 [11], SHA [9] and RIPE-MD [3].

With the exception of MD2, all of these algorithms are variants of MD4. The MD4 variants were implemented by writing a single subroutine which was copied and edited to produce each of the variants. As a result, exactly the same optimisations and speed-up techniques were used in all the variants. This makes it easier to compare the results, as some optimisations can make a tremendous difference to the performance of these hash functions.

The platforms chosen for measurement were a Sun IPC (Sparc architecture) and a DEC 3000 (Alpha architecture).

Figure 1 shows the speed in Mbit/s when hashing a 6,400,000 octet message.

Algorithm	Sparc (Mbit/s)	Alpha (Mbits/s)
MD2	0.212	0.755
MD4	15.36	78.77
MD5	10.97	60.02
SHA	6.707	41.51
RIPE-MD	9.143	48.00

Fig. 1. Hash Algorithms: Long messages

Figure 2 shows the number of 64 octet long messages that can be hashed per second.

Algorithm	Sparc (hashes/s)	Alpha (hashes/s)
MD2	876	2906
MD4	19934	63160
MD5	16043	50002
SHA	10381	35504
RIPE-MD	13575	45456

Fig. 2. Hash Algorithms: Short messages

5 Symmetric Ciphers

The ciphers examined were DEA-1 [7], GOST 28147 [12] and SAFER-K64 [5].

DEA-1 is the subject of a U.S. Federal Information Processing standard, and is widely used in financial applications in many countries. GOST 28147 is a Soviet standard, and is widely used within the countries of the former Soviet

Union. SAFER-K64 is a new algorithm developed by James Massey for Cylink corporation. A full description of SAFER-K64 is given elsewhere in these proceedings.

Figure 3 shows the throughput of these algorithms when encrypting in Electronic Code Book (ECB) mode. For all these algorithms, the speed of decryption is approximately the same as the speed of encryption.

Algorithm	Sparc (Mbit/s)	Alpha (MBit/s)
DEA-1	0.487	1.855
GOST 28147	0.713	2.909
SAFER-K64	2.259	7.680

Fig. 3. Symmetric Ciphers: Long messages

Figure 4 shows the rate at which keys can be changed. The "ratio" column gives the number of bits than can be encrypted (without changing the key) in the same time that it takes to change the key. This ratio gives a measure of the complexity of computing the key schedule for each algorithm. DEA-1 has the most complex key schedule, involving permutations and rotations by varying numbers of bits. GOST 28147 has an extremely simple key schedule. The use of a complex key schedule can do much to increase the security of a cryptographic algorithm. Furthermore, in many applications key changes are fairly infrequent, and so increasing the complexity of the key schedule has little impact on overall performance.

DEA-1, GOST 28147 and SAFER-K64 all have the property that the subkey used in each round is a linear function of the original key. Designers of new encryption algorithm should consider using non-linear functions in the key schedule, as this can increase security without affecting performance.

Algorithm	Sparc (changes/s)	Alpha (changes/s)	Sparc ratio	Alpha ratio
DEA-1	896	4000	543	464
GOST 28147	8696	37501	82	78
SAFER-K64	6000	24000	377	320

Fig. 4. Symmetric Ciphers: Short messages

For encrypting bulk data, it is conventional to use a chaining mode such as Cipher Block Chaining (CBC) [8] rather than ECB mode. For any block cipher, CBC mode will be slightly slower than ECB mode, as an additional exclusive-or operation is required for each block. However, the time taken to perform the exclusive-or is very small compared to the total time taken to encrypt a block, and so the speeds to the two modes are approximately the same. The following table compares the throughput of ECB and CBC mode for DEA-1.

Mode	Sparc (Mbit/s)	Alpha (MBit/s)
ECB	0.487	1.855
CBC	0.463	1.706

Fig. 5. DEA-1: Chaining modes

The key size of DEA-1 is sometimes considered to be too small to provide sufficient security for some applications. To increase the effective key space, a new mode called EDE (Encrypt-Decrypt-Encrypt) was invented [6]. In this mode, the data is first encrypted with key 1, then decrypted with key 2, and finally encrypted with key 1 again. This provides the same functionality as ECB mode, but doubles the size of the keys. (And so, it is hoped, greatly increases the security).

DEA-1 in EDE mode is somewhat faster than a third of the speed of ECB mode, because in a optimised implementation some of the internal permutations can be combined. This optimisation depends upon the internal structure of DEA-1, and is not necessarily possible with other cipher algorithms used in EDE mode.

Two new modes, EDE-CBC and CBC-EDE, deserve some explanation. Both of these modes use three independent DES keys, in order to guard against known plaintext attacks using very large parallel machines. They are also both chaining modes, in order that messages longer than 64 bits may be encrypted.

EDE-CBC mode takes EDE mode as a building block and uses it in CBC mode. CBC-EDE mode (invented by Carl Ellison) is CBC mode applied three times with different keys. EDE-CBC mode is generally considered to be more resistant to cryptanalytic attacks [2].

Figure 6 shows the speed of the different triple encryption modes when encrypting a 64,000,000 octet message under a fixed key. EDE-CBC mode is only slightly slower than EDE mode, as the time taken to perform an additional exclusive-or operation is very small. CBC-EDE mode is significantly slower, as it has a chaining exclusive-or operation in between the internal permutations. This means that it is not so easy to speed up the algorithm by combining these permutations. It is theoretically possible to do an equivalent optimisation for CBC-EDE mode, using the fact that the exclusive-or operation commutes with permutations. This was not considered to be worth implementing, as CBC-EDE mode had been discredited on security grounds [2].

Mode	Sparc (Mbit/s)	Alpha (MBit/s)
ECB	0.487	1.855
CBC	0.463	1.706
EDE	0.274	1.200
EDE-CBC	0.269	1.122
CBC-EDE	0.151	0.557

Fig. 6. DEA-1: Triple encryption modes

6 Conclusions

Conventional algorithms (such as DEA-1) are surprisingly fast when run on the current generation of fast processors. For many applications, they have more than adequate performance.

Perhaps we should be concentrating on more secure algorithms (with the same performance) rather than faster, but less secure algorithms? The following techniques can be effective:

- Improving the key schedule
  GOST 28147, DEA-1 and SAFER-K64 all have the property that the subkey for each round is a linear function of the original key. Using a non-linear transformation can improve security, and has little effect on performance.
- Larger lookup tables
  GOST 28147, DEA-1 and SAFER-K64 all use lookup tables to make each round non-linear. Increasing the size of these tables can improve security. Provided that the tables still fit within the CPU's cache memory, there is essentially no performance penalty in making the tables bigger.
  GOST 28147 uses 4-bit wide tables, which are probably too small to provide adequate protection against attacks such as differential cryptanalysis. DEA-1 uses 6-bit wide tables; this was probably an optimal compromise between security and memory requirements when DEA-1 was designed (in the 1970s). SAFER-K64 uses 8-bit wide tables. This is certainly large enough, but the function used to generate the tables ($45^x mod 257$) is not as non-linear as it could be; $45^{(x+y)} = 45^x.45^y$ As the contents of the tables has no effect on performance, a function less helpful to the cryptanalyst could have been chosen.
- Triple encryption
  Triple encryption is a good way of taking an algorithm which has withstood attack for many years (such as DEA-1) and increasing its key size to protect against exhaustive search. If a chaining mode is desired, it is better to apply the chaining around three applications of the block cipher than to apply the chaining mode three times.

References

1. D. Balenson. *RFC 1423: Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers*, February 1993.
2. E. Biham. On modes of operation. In *Fast Software Encryption : Cambridge Security Workshop*, December 1993.
3. CWI, Amsterdam. *RIPE Integrity Primitives — Final Report of RACE Integrity Primitives Evaluation (R1040)*, June 1992.
4. B. Kaliski. *RFC 1319 : The MD2 Message-Digest Algorithm*, April 1992.
5. J. L. Massey. SAFER K-64: A byte-oriented block-ciphering algorithm. In *Fast Software Encryption : Cambridge Security Workshop*, December 1993.
6. C. H. Meyer and S. M. Matyas. *Cryptography: a new dimension in computer data security*. John Wiley and Sons, 1982.
7. National Bureau of Standards. *Federal Information Processing Standard — Publication 46: Data Encryption Standard*, 1977.
8. National Bureau of Standards. *Federal Information Processing Standard — Publication 81: DES Modes of Operation*, 1977.
9. National Bureau of Standards. *Federal Information Processing Standard — Publication 180: Secure Hash Standard*, 1993.
10. R. L. Rivest. *RFC 1320 : The MD4 Message-Digest Algorithm*, April 1992.
11. R. L. Rivest. *RFC 1321 : The MD5 Message-Digest Algorithm*, April 1992.
12. [Russian] State Committee for Standardization, Metrology and Certification. *GOST 28147 : Cryptographic Protection for Data Processing Systems — Cryptographic Transformation Algorithm*, 1990.