A fast implementation of DES and Triple-DES on PA-RISC 2.0

Francisco Corella
Hewlett Packard Co.

1. Introduction

Encryption is an essential tool for protecting the confidentiality of data. Network security protocols such as SSL or IPSec use encryption to protect Internet traffic from eavesdropping. Encryption is also used to protect sensitive data before it is stored on non-secure disks or tapes.

Encryption, however, is computationally expensive. A computer server that must encrypt data for thousands of clients before sending it over the network can easily become crypto-bound. The capacity of the server is then determined by the speed at which it can perform encryption. This is especially the case when slow encryption protocols such as the Digital Encryption Standard (DES) or Triple-DES are employed. Since DES and Triple-DES are very widely used, it is important to optimize the performance of these algorithms.

We describe an implementation of DES and Triple-DES in PA-RISC 2.0 assembly language that outperforms other practical (non bit-sliced) implementations by large margins. It is based on a technique due to Eli Biham of the Technion that takes advantage of 64-bit registers, with substantial improvements developed at Hewlett Packard.

We assume that the reader is familiar with the details of DES and Triple-DES, which are described in [1].

2. Earlier software implementations

Most software implementations of DES and Triple-DES are written for machines having 32-bit registers. In 1997, Eli Biham published an implementation, written in C, specifically targeted for the 64-bit Alpha architecture [2].

Biham took advantage of the 64-bit register width as follows. He stored the two block halves that each round of DES operates on in two separate 64-bit registers. However, instead of storing them in their standard 32-bit format, he stored them in the 48-bit format that results from applying the expansion permutation to a 32-bit array, with zeros in the twelve remaining bit positions. Each round then proceeds as follows. The right half, which is already in expanded form in a 64-bit register, is xored with the subkey, which is also contained in a 64-bit register. Then the resulting 48-bit array is divided into eight groups of six bits, each of which is used as the index into an S-box.

Biham also changed the format of the S-boxes. He turned each S-box entry into a 64-bit array, derived from the 4-bit array of the original S-box by the following procedure. First the 4-bit array is placed in its proper position within a 32-bit (unexpanded) block half, the other bit positions in the array being filled with zeros. Then the 32-bit permutation is applied to the 32-bit array. Finally, the expansion permutation is applied to this 32-bit array, producing a 48-bit array, which is completed with zeros in the twelve remaining bit positions. Biham's algorithm does an S-box look up as a straightforward DES implementation would, but the look up produces a 64-bit array, mostly filled with zeros, rather than a 4-bit array. After the eight S-boxes have been looked up, the eight resulting 64-bit results are ored together and the result is then xored with the left half, which is also stored in a 64-bit register in the expanded 48-bit format.

With this implementation, Biham achieved a throughput of 46 Mb/s for DES and 22 Mb/s for Triple-DES on a 300 MHz Alpha 8400 processor. He compared this performance to that of Eric Young's libdes library on the same machine, which presumably does not take advantage of the 64-bit register width. The performance of libdes was quoted by Biham as 28 Mb/s for DES.

Recently, however, other 32-bit implementations have achieved better performance. The popular BSAFE cryptographic toolkit of RSASecurity, can serve as a good benchmark for comparison purposes. At the 1999 RSA conference, substantial performance improvements for several BSAFE algorithms were announced. The new performance figures quoted for DES and Triple-DES were 39.2 Mb/s and 15.2 Mb/s respectively, on a 233 MHz Pentium II.

In the same paper [2], Biham also describes a bit-sliced implementation, which has higer throughput. However, a bit-sliced implementation is not practical because application software would have to be restructured to take advantage of the 64-way parallelism that yields the increased throughput.

	Young's 32-bit implementation	Biham's 64-bit implementation	BSAFE 32-bit implementation	Our 64-bit implementation
Throughput	28 Mb/s	46 Mb/s	39.2 Mb/s	183 Mb/s
CPU frequency	300 MHz	300 MHz	233 MHz	550 MHz
Relative speed	686 clocks/block	417 clocks/block	380 clocks/block	192 clocks/block

Table 1. Comparison of four DES implementations (CBC mode)

3. Our implementation

Biham provides an instruction count for his standard DES implementation. Processing one DES block takes 634 instructions. A majority of these instructions (61%) are concerned with S-box look-up. Looking up an S-box takes three instructions, and this is done 8 times per each of the 16 rounds, for a total of 16x8x3 = 384 instructions.

We have made two key improvements to Biham's implementation. First, we have grouped the eight S-boxes into four pairs of boxes, and merged the two 6-input boxes in each pair into a single 12-input box.[1] The resulting four double S-boxes occupy 1 MB of memory.

Then, writing the code in PA-RISC 2.0 assembly language [3], we have reduced the number of instructions needed to look up an S-box from 3 to 2. An S-box look-up is performed by the following two-instruction code fragment:

```
EXTRD,U     gr1,pos,12,gr2
LDD,S       gr2(gr3),gr4
```

The first instruction extracts an unsigned 12-bit value from bit position pos of general register gr1, and places it in general register gr2. In the implementation, the general register gr1 contains the result of xoring the expanded right half with the subkey for the round. The immediate value pos selects one of the four groups of 12 bits that are used to look up the four 12-input S-boxes.

The second instruction shifts left by three positions (i.e. multiplies by 8) the value contained in gr2, adds the result (a displacement) to the contents of general register gr3 (a memory address), and loads the 64-bit word stored at the resulting memory address into general register gr4. In the implementation, gr3 contains the base address of the 12-input S-box, gr2 contains the 12-bit index into the S-box, and gr4 receives the 64-bit entry read from the referenced S-box entry.

The cumulative result of these two improvements is that S-box look up only requires a total of 16x4x2 = 128 instructions, thus saving 256 instructions. A variety of other improvements saved another 108 instructions, resulting in a count of only 270 instructions for a DES computation. Table 1 shows the performance achieved by our DES implementation, comparing it to that of the other implementations mentioned above. Our Triple-DES implementation takes only 471clocks/block (74.8 Mb/s on a 550 MHz CPU), while Biham's and BSAFE take 873 and 981 clocks/block respectively.

Our implementations have been incorporated in the IPSec/9000 product available with HP-UX, and are currently being incorporated in a cryptographic toolkit.

[1] This idea is due to Peter Markstein.

Acknowledgements

Peter Markstein contributed the idea of merging two 6-input S-boxes into a single 12-input box. Ruby Lee provided motivation for the project and drew attention to Eli Biham's paper [2].

References

[1] FIPS 46-3, Data Encryption Standard (DES), National Institute of Standards and Technology (NIST), http://csrc.nist.gov/cryptval/des.htm.

[2] Eli Biham, *A Fast New DES Implementation in Software*, Fast Software Encryption 4, Haifa, Israel, January 1997. Available from Eli Biham's home page, http://www.cs.technion.ac.il/~biham/.

[3] G. Kane, *PA-RISC 2.0 Architecture*, Prentice Hall, 1996.