AN ANALYSIS OF NEWDES: A MODIFIED VERSION OF DES

Charles Connell

ADDRESS: 79 Locust Street, Burlington MA 01803 USA and Computer Science Department, Boston University, 111 Cummington St., Boston MA 02215 USA.

ABSTRACT: This paper examines an encryption algorithm designed by Robert Scott[3] that is a modified form of the Data Encryption Standard. Scott's goal in varying DES is to improve two aspects of it that are alleged to be weaknesses: the length of the key, and the "secretness" of the design of the S-boxes. An ancillary goal of his is to provide an algorithm that is easy to implement in software on a microcomputer.

Scott's algorithm is indeed simple to implement. One of the main reasons for this is that it uses only operations on entire bytes, so there is no individual bit manipulation. Also, as promised, the design of the entire f-function is open for examination. The increase in key length, however, may not be as significant as it first appears to be.

KEYWORDS: Cryptography, cryptanalysis, Data Encryption Standard, DES, key length.

A. INTRODUCTION

The Data Encryption Standard is a widely used encoding/decoding algorithm that has received a great deal of analysis. While the algorithm is believed to be sufficiently secure for many applications, some aspects of it keep DES from being more strongly trusted. The two main criticisms of it have been the length of the key and the design of its internal S-boxes.[1, pp. 97-98]

The objection to the key length is simply that the key is too short (56 bits). A number of researchers have pointed out that someone can build a machine to break DES by exhaustive search. The machine would try every key on a known plaintext/ciphertext pair and discover the key in approximately one day. The objection to the S-boxes is twofold. First, they may contain a *trap door*. A trap door would allow a cryptanalyst to discover information about the key by examining pairs of plaintext and ciphertext. Second, the design of the S-boxes is classified, so independent researchers cannot verify that no such trap door exists.

Scott's algorithm, NEWDES, is a variation of DES that attempts to improve these two weaknesses.[3]

B. THE NEWDES ALGORITHM

B.1. Encryption

NEWDES is based on the overall structure of DES. NEWDES operates on 64-bit message blocks and produces 64-bit cipher blocks. It uses a series of encryption *rounds* that progressively mix the message with the key, and distribute input bits throughout the output.

NEWDES varies from DES in several ways, though. It is considerably simpler than DES. It does not use initial and final permutations. All operations are on entire bytes – at no time does the algorithm read any particular bits. The central f-function is much, much simpler than in DES. The key in NEWDES is longer than in DES: 120 bits or 15 bytes. The key is also used just as it is entered; there is no series of internal keys that must be generated.

It is important to note however, that despite these major differences, the overall structure of NEWDES is the same as that of DES. The core of the NEWDES algorithm is shown in Figure 1. The algorithm contains the familiar exclusive-or of key segments with data segments, and an f-function that operates on the results of these exclusive-ors. All data items in the figure are bytes. The 15 key bytes are used in order, then repeated until each has been used four times.

Notice that one line of the algorithm is different from the others. It takes two data bytes as input to the f-function, rather than a data byte and a key byte. The purpose of this line is to break a symmetry that results from all data bytes being the same as each other and all key bytes being the same as each other.

B.2. Design of the f-Function

What about the f-function? In NEWDES, the f-function is simply a fixed, pseudo-random permutation of bytes to bytes. The input is a number between 0 and 255 and the output is another, probably different, number in the same range. No two distinct inputs produce the same output. So, the f-function is just a table with 256 entries.

The particular method used to generate the f-function's permutation table is not important. As long as it is open for public examination, it is not likely to contain a trap door known only to the algorithm's designer. Scott chose to use a long series of swapping operations, driven by the text of the Declaration of Independence, to mix up the numbers between 0 and 255. While Scott's method appears acceptable, NEWDES works equally well with any pseudo-random permutation table.

```
i := 1;
WHILE (i=1) DO
  BEGIN

  block[5] := block[5] XOR f(block[1] XOR key_sched[i]);
  i := i + 1;
  block[6] := block[6] XOR f(block[2] XOR key_sched[i]);
  i := i + 1;
  block[7] := block[7] XOR f(block[3] XOR key_sched[i]);
  i := i + 1;
  block[8] := block[8] XOR f(block[4] XOR key_sched[i]);
  i := i + 1;

  IF i >= 60 THEN GOTO 10;

  block[2] := block[2] XOR f(block[5] XOR key_sched[i]);
  i := i + 1;
  block[3] := block[3] XOR f(block[6] XOR block[5]);
  block[4] := block[4] XOR f(block[7] XOR key_sched[i]);
  i := i + 1;
  block[1] := block[1] XOR f(block[8] XOR key_sched[i]);
  i := i + 1;

  END;
10: ;
```

Figure 1. NEWDES Encryption/decryption algorithm.

```
0 0 0 0 0 0 0 1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
152 52 219 139 200 230 187 62

0 0 0 0 0 0 1 0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
13 170 27 247 114 232 127 173

0 0 0 0 0 1 0 0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
176 200 46 193 80 168 174 175

0 0 0 0 0 1 0 0
2 2 3 4 5 6 7 8 9 10 11 12 13 14 15
140 200 211 171 208 38 203 248

0 0 0 0 0 1 0 0
3 2 3 4 5 6 7 8 9 10 11 12 13 14 15
85 143 22 157 41 150 72 184
```

Figure 2. Example of bit distribution in NEWDES.

B.3. Decryption

NEWDES decrypts a ciphertext in the same way that DES does. The algorithm is re-applied, with a reversed key schedule. Only one slight twist is required during decryption. The keys cannot be re-applied in strictly backwards order, or they would be used to decrypt different parts of the ciphertext than they encrypted. Note that byte 8 of the text is the last one touched during encryption, but the fourth during decryption. The decryption key schedule handles this by matching key bytes to the cipher bytes they actually encrypted.

C. Using NEWDES

As promised, it is easy to program NEWDES on a microcomputer. One implementation, NEWDES.PAS, was written in Turbo-Pascal. It took only a few hours to write, and occupies approximately 2000 bytes. Despite this small size, much of the space is consumed by redundant code and data that were added for program clarity. On a standard (8088-based) IBM-PC, NEWDES.PAS encrypts or decrypts 100 64-bit blocks of data in approximately 2 seconds (author's test). (For information about obtaining NEWDES.PAS, see the note at the end of this article.)

NEWDES also exhibits excellent bit distribution. A change of just one bit in either the data or the key can change an arbitrary number of bits in the output. The bit distribution occurs quite early in the algorithm, so that any bit in the input can affect any bit in the output a number of times. Figure 2 shows sets of inputs to NEWDES that vary by one bit from the previous input, and their resulting outputs. The first line of each set is the data to be encrypted, the second line is the key, and the third line is the encryption. In addition to this empirical evidence, Scott presents an impressive amount of analysis of bit distribution in his paper.

As a corollary to good bit distribution, there does not appear to be any way that a cryptanalyst can know that he is *close* to finding the key. Given a known plaintext attack, an analyst could test a key that is wrong by only one bit, but receive output that bears no resemblance to the known ciphertext.

D. KEY SECURITY

D.1. Brute-Force Attacks

One of the goals of NEWDES is to improve on the key security of DES. NEWDES is obviously secure against a brute-force attempt to test all possible

keys on a known plaintext/ciphertext pair. There are 2^{120} keys.

As is the case with DES, however, the actual number of keys to test in a chosen plaintext attack can be reduced by one-half. This is because both DES and NEWDES share the property that if $C = E_k(M)$, then $C' = E_{k'}(M')$. (M' is the bit-by-bit complement of M, etc.) While this property of DES is well known, the fact that it is also true of NEWDES is not noted in Scott's paper. To exploit this property, a chosen plaintext can be constructed consisting of two 64-bit message blocks, M and M'. These are enciphered with the unknown key (K), giving C_1 and C_2.

To find K, all keys that are not complements of each other are used to encipher M. A key that enciphers M into C_1, is K. A key that enciphers M into C_2', is K'.[2, pp. 116-118] NEWDES is obviously still secure against this type of attack, however, since one-half of 2^{120} is 2^{119}.

Potentially, there exists a much more serious weakness to the keys used in NEWDES, which also is not noted in Scott's paper. The problem is that there may be too many keys.

D.2. Secondary Keys

During encryption, each 64-bit message block is enciphered into a 64-bit ciphertext. A given message block, therefore, may be encrypted into 2^{64} different ciphertexts. There are, however, 2^{120} different keys. Each key cannot uniquely map a given plaintext block to a different ciphertext. There are vastly more keys than possible mappings.

How many keys will map a given 64-bit message block to the same ciphertext? Each key must map the plaintext to some ciphertext. There are 2^{120} keys and 2^{64} possible ciphertexts. So, on average, $2^{56} = 2^{120}/2^{64}$ keys will all map a given 64-bit message block to the same ciphertext. As we will see, it might be possible to exploit this fact to reduce the security of NEWDES.

We refer to the key that the user chooses as the *primary key*; and to other keys that produce the same result, but were not chosen by the user, as *secondary keys*. Note that secondary keys work during decipherment; any key that maps a plaintext to a ciphertext will decrypt that ciphertext as well.

We now pose two questions:

1. Consider a 64-bit message (M_1), a primary key (K_1), and the encipherment of M_1 using K_1 (C_1). Assume that K_2 is a secondary key for this plaintext/ciphertext pair. That is, K_2 also deciphers C_1 into M_1.

   Now consider a distinct 64-bit message (M_2) enciphered using the same primary key (K_1). Call the result of this encipherment C_2.

Is K_2 also a secondary key for M_2 and C_2?

2. Consider a 64-bit message (M_1), a primary key (K_1), and the encipherment of M_1 using K_1 (C_1). Is there an efficient method of generating secondary keys that also map M_1 to C_1?

If the answer to question (1) is "Yes", then a secondary key for one 64-bit message block is, in fact, a key for all blocks that use the same primary key. This means that every multi-block message encoded with NEWDES has, on average, 2^{56} correct keys. In this case, NEWDES has an effective key length of 2^{64}: the number of distinct ciphertexts that one message block can map to.

If the answer to question (1) is "No", but the answer to question (2) is "Yes", we may still be able to compromise the key length. Assume a cryptanalyst has a multi-block plaintext/ciphertext pair. The analyst could find a key (K) that works for one 64-bit message block (M). This might be the primary key for the entire message, or it might be a secondary key that only works for M. Assume the latter, which is more likely. The cryptanalyst could then exhaustively generate secondary keys from K and try these on other blocks of the message. One of the secondary keys derived from K is the primary key and will decrypt the entire message.

It might be possible to do this in a way that would require a maximum of $2^{64} + 2^{56}$ tests. 2^{64} tests might be able to find one secondary key for one message block, and 2^{56} variations of this key could find the primary key for all blocks. This would result in a total number of tests that is less than 2^{65}.

E. CONCLUSION

We have seen that NEWDES is a simple, elegant algorithm. It is straightforward to program in a high-level language, and it yields programs that run quickly. Its design is completely public and open for inspection. NEWDES gives excellent bit distribution for plaintexts and keys.

The effective key length of NEWDES may be less than what the user enters, however. Each message block has secondary keys that work as well as the key that the user chose. These secondary keys may compromise the effective key length for multi-block messages also, reducing it significantly below 120 bits.

REFERENCES

1. Denning, Dorothy E. R. 1983. *Cryptography and Data Security*. Reading MA: Addison-Wesley Publishing Company.

2. Meyer, Carl H. and Stephen M. Matyas. 1982. *Cryptography: A New Dimension in Data Security*. New York: John Wiley and Sons.

3. Scott, Robert. 1985. Wide Open Encryption Design Offers Flexible Implementations. *Cryptologia*. 9(1): 75-90.

ACKNOWLEDGEMENTS

I am grateful to Steven Homer for an excellent course on cryptology and for valuable suggestions about this paper; to Robert Scott for a helpful discussion of his original work and for his review of this paper; and to the anonymous reviewer for his or her careful reading of this paper and comments about it. Any errors in this article, of course, remain mine.

SOFTWARE

To obtain a copy of NEWDES.PAS (the author's Turbo-Pascal implementation of this algorithm) please send the author the following: 1) a 5 1/4 inch diskette formatted for a standard IBM-PC (double-sided, double-density, 360K); 2) a stamped, self-addressed diskette mailer.

BIOGRAPHICAL SKETCH

Charles Connell earned a BA in linguistics from Hampshire College in 1979 and an MA in computer science from Boston University in 1984. He is currently in the PhD program in computer science at Boston University. His research interests include approximation algorithms for hard problems and software engineering for safety-critical systems. Mr. Connell also works as an independent consultant in the Boston area.