Chapter 4 - The Data Encryption Standard

Dr. Morton

§4.1 Introduction

1. In 1973 the National Bureau of Standards (NBS, now NIST=National Institute of Standards and Technology) made a public call for a cryptographic algorithm to become the national standard. IBM - submitted LUCIFER in 1974. NBS forwards it to NSA, which looks at it, modifies it and returns it - what they returned was (basically) DES = Data Encryption Standard. In 1975, NBS released DES with a free license for use, and made it the national standard in 1977. After DES was made into national standard, it gets recertified every five years.

2. In 1990 Eli Biham and Adi Shamir used differential cryptanalysis to attack DES. Biham and Shamir found that it would be much easier to use this type of attack if algorithm had used at most 15 rounds. I should note that DES uses 16 rounds of an algorithm. (IBM did admit that they had constructed DES to be resistant to differential cryptanalysis.)

3. In 1992 DES was recertified, but by 1997-1998, NIST seriously wanted a replacement - because in 1998 a group broke DES in a few hours. Thus DES is no longer adequate (which is why the US has now officially adopted AES = Advanced Encryption Standard).

4. DES is a block cipher. The plaintext is broken into blocks of 64 bits, and we encrypt each block separately. Each of the 16 rounds is called a Feistel network (after Horst Feistel - part of IBM team that made LUCIFER). The idea of Feistel network is to give a function which is guaranteed to be invertible and to be its own inverse.

5. From the wikipedia page on differential cryptanalysis (don't believe everything you read...): Differential cryptanalysis is a general form of cryptanalysis applicable primarily to block ciphers, but also to stream ciphers and cryptographic hash functions. In the broadest sense, it is the study of how differences in an input can affect the resultant difference at the output. In the case of a block cipher, it refers to a set of techniques for tracing differences through the network of transformations, discovering where the cipher exhibits non-random behaviour, and exploiting such properties to recover the secret key.

   The discovery of differential cryptanalysis is generally attributed to Eli Biham and Adi Shamir in the late 1980s, who published a number of attacks against various block ciphers and hash functions, including a theoretical weakness in the Data Encryption Standard (DES). It was noted by Biham and Shamir that DES is surprisingly resistant to differential cryptanalysis, in the sense that even small modifications to the algorithm would make it much more susceptible.

   In 1994, a member of the original IBM DES team, Don Coppersmith, published a paper stating that differential cryptanalysis was known to IBM as early as 1974, and that defending against differential cryptanalysis had been a design goal. According to author Steven Levy, IBM had discovered differential cryptanalysis on its own, and the NSA was apparently well aware of the technique. IBM kept some secrets, as Coppersmith explains: "After discussions with NSA, it was decided that disclosure of the design considerations would reveal the technique of differential cryptanalysis, a powerful technique that could be used against many ciphers. This in turn would weaken the competitive advantage the United States enjoyed over other countries in the field of cryptography." Within IBM, differential cryptanalysis was known as the "T-attack" or "Tickle attack".

§4.2 A Simplified DES-Type Algorithm

1. This example, like DES, is also a block cipher. Since each block is encrypted separately, we assume that the full message consists of exactly one block.

   (a) Fix a positive integer n, (in DES, $n = 32$). Given a string of $2n$ bits, group them as two parts, the left and right halves, L_0 and R_0.

   - **Example:** (This will actually be one long example, which will be split up and analyzed over the next few pages:) $n = 6$, message is 100110101101. Find L_0, R_0.

   (b) The key K is chosen. (In DES, the message has 64 bits and the key K has 56 bits, but is expressed as a 64-bit string - the 8th, 16th, 24th,...bits are parity bits - arranged so that each block of 8 bits has an odd number of 1's)

   - **Example:** In our case, let K have 9 bits. $K = 101100001$

   (c) The ith round of the algorithm: input $L_{i-1}R_{i-1}$, outputs L_i, R_i using a key K_i derived from K. (In DES K_i is a string of 48 bits obtained from K.)

   - **Example:** In our case, let K_i have 8 bits. Thus the ith round of the algorithm transforms an input $L_{i-1}R_{i-1}$ to the output L_iR_i using an 8-bit key derived from K.

   (d) Again, in the ith round: The main part of the encryption process in **every** round is a function $f(R_{i-1}, K_i)$, which is a function f defined on the pair R_{i-1}, K_i. If R_{i-1} has n bits, then $f(R_{i-1}, K_i)$ has n bits. The function f itself will be described later.

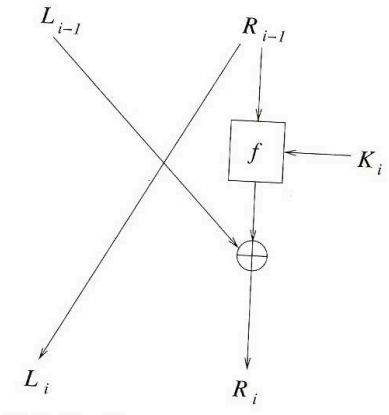
   - **Example:** Our function $f(R_{i-1}, K_i)$ will take a 6-bit input R_{i-1} and an 8-bit input K_i and produces a 6-bit output. (Note: We will describe f in gory detail later, so suppose that we know how f is defined.)

   (e) The output for the ith round is defined:

   $$L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

   where $\oplus$ is bit-wise addition - i.e. XOR.

   Here is a graphical illustration of the ith round of the algorithm, also called a Fiestel network.



   (f) This operation is performed for m rounds (16 rounds in DES) and produces the ciphertext L_mR_m.

2. To decrypt:

   (a) Start with L_mR_m and switch to get R_mL_m. (Note: will not need to do this step in DES - it is built in.)

(b) Now repeat above process, but with keys in opposite order $K_m, ...K_1$. Why does this work?

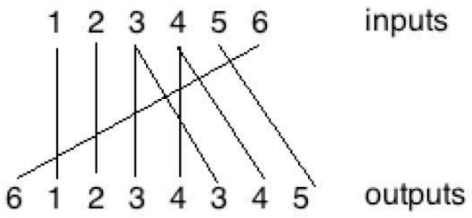
(c) The next step of decryption sends $R_{n-1}L_{n-1} \rightarrow R_{n-2}L_{n-2}$. We continue until we get back to R_0L_0 and then switch these around to get the p.t. L_0R_0.

(d) **Note:** Decryption is same process as encryption, with same set of keys - thus both sender and receiver can use the same machines, with the receiver reversing the ciphertext halves.

3. The function f used in the simplification and in DES are made up of **three** different components:

   (a) The expander (also called E-box in DES). In DES: input 32 bits, output 48 bits.

   - **Example:** For our example, the input is 6 bits; the output is 8 bits.
     Here's the one we will use:



   - **i.** Use the expander on 010101.

   (b) The next part are the S-boxes: DES uses 8 - each with 4 rows and 6 columns. In DES, inputs are 6 bits, outputs are 4 bits.

   - **Example:** We will use two S-boxes for our example:

$$S_1 = \left| \begin{array}{cccccccc} 010 & 111 & 101 & 000 & 110 & 001 & 011 & 100 \\ 111 & 000 & 110 & 001 & 101 & 010 & 100 & 011 \end{array} \right|$$

$$S_2 = \left| \begin{array}{cccccccc} 000 & 110 & 011 & 100 & 010 & 111 & 101 & 001 \\ 110 & 001 & 101 & 100 & 011 & 111 & 000 & 010 \end{array} \right|$$

     The input to an S-box is 4 bits; the output is 3 bits.
   - To use the S-boxes: If have 4 bit input, the first bit tells us which row to use: 0=top,1=bottom. The last 3 bits are a binary number that represents the column, with 000=1st col, 001=2nd column,...111=8th column.
   - **i.** Suppose our four bit input to S_1 is 1000. What is the output?
     ii. Suppose our four bit input to S_2 is 0101. What is the output?

   (c) The last part is the key K (64 bits in DES).

   - **Example:** The key K is 9 bits here
   - Need round i key: K_i - found by using 8 bits of K, starting at ith. So this is shifting K $i-1$ units to the left.
   - **iv.** Suppose our key is $K = 111000111$. What is K_1? What is K_6?

   (d) How do we put these together to get $f(R_{i-1}, K_i)$?

   - **Example:** Note that R_{i-1} is 6 bits, K_i is 8 bits.
   - i. Put R_{i-1} into E-box to expand to 8 bits.
     ii. XOR this result with K_i: 8 bits
     iii. Send first 4 bits to S_1, last 4 bits to S_2.
     iv. Outputs from the S-boxes - 3 bits each. Put these together (next to each other) to get 6 bits. **This** is $f(R_{i-1}, K_i)$!

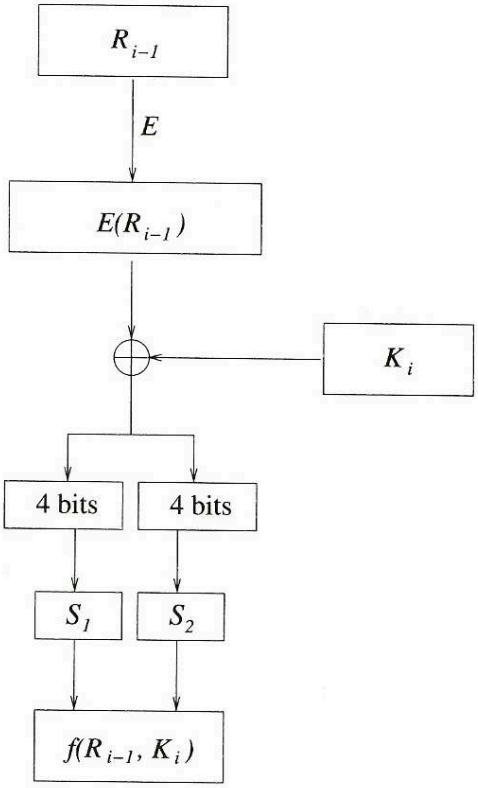


Figure 4.3: The Function $f(R_{i-1}, K_i)$.

4. **Example:** Perform the first round of the Fiestel network, using the plaintext message

$$m = 100110101101$$

and key

$$K = 101100001.$$

§4.4 DES

1. Input into DES: blocks of plaintext of 64 bits. The key has 56 bits, but is expressed as 64-bits. The 8th, 16th, 24th, ... bits are parity bits - arranged so that each block of 8 bits has an odd number of 1's (built in error detection!) The output from DES: 64 bit ciphertext.

2. **Example:** If key k starts 0000001 1001011 0100100 ... what is the key that we use?

3. Algorithm for DES Encryption:

   (a) Permute the bits of the plaintext message m (64 bits). This is done by a fixed initial permutation (IP) and outputs $IP(m) = m_0$. Now split m_0 into the left and right halves, each of 32 bits: $m_0 = L_0R_0$

   (b) We now complete 16 rounds of the following:
   For $1 \leq i \leq 16$, let
   $$L_i = R_{i-1}, R_i = L_{i-1} \oplus f(R_{i-1}, K_i),$$
   where the key K is used to obtain K_i, which has 48 bits. The function f will be described below as will the process to find K_i from K. After 16 rounds, we now have $L_{16}R_{16}$.

   (c) Switch left and right: $R_{16}L_{16}$. Now apply the inverse of the initial permutation (IP^{-1}): Ciphertext is
   $$c = IP^{-1}(R_{16}L_{16})$$

4. Algorithm for DES Decryption:

   (a) Same as for encryption, but the keys are used in the reverse order. In other words, the order of the keys used is $K_{16}, ...K_1$. Note: in step (c) above, we switch the left and right, so we don't have to start with a switch in the decryption step - it is built in.

5. The DES algorithm is represented graphically on the next page.

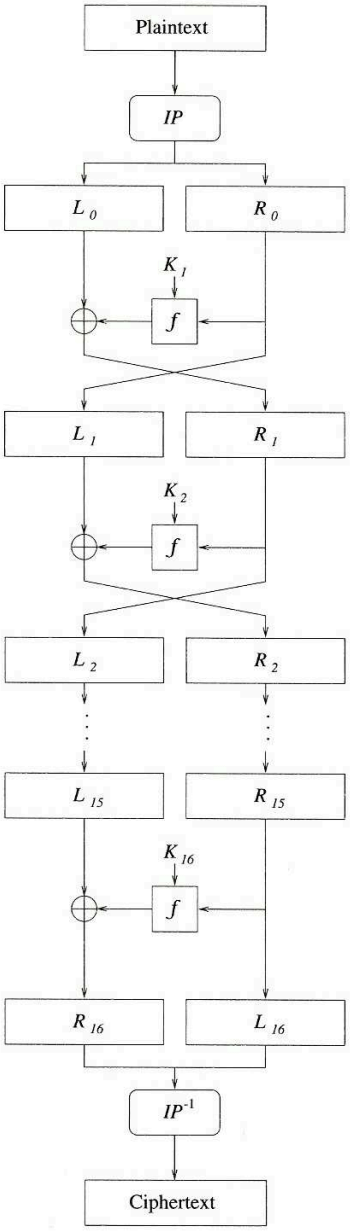


Figure 4.4: The DES Algorithm.

6. IP: The Initial Permutation is described by the Initial Permutation Table:

Initial Permutation															
58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

How do we find $IP(m)$? The 58th bit of m becomes first bit of m_0, the 50th bit of m becomes the second bit of m_0,...62nd bit of m becomes seventeenth bit of m_0...

7. The function $f(R, K_i)$:

Recall that the ith round key K_i has 48 bits and that $R = R_{i-1}$ has 32 bits.

(a) Expand R to $E(R)$ using the Expansion Permutation, or E-box table:

Expansion Permutation											
32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

This works just like it did in section 4.2: the first bit of $E(R)$ is the 32nd bit of R, ... This E-box has 48 entries, so $E(R)$ has 48 bits. (So input is 32 bits; output is 48 bits.)

Why do this? The pattern isn't very complicated. But by allowing one bit of text to affect more than one bit of the expanded version, we get an avalanche effect: changes in relatively few bits of the plaintext should cause many bits of the ciphertext to change. Thus as we go through all 16 rounds of DES, changing one bit should cause more and more to change.

(b) XOR the output from the E-box with the key K_i, and then split up into blocks of 6 bits each:

$$E(R) \oplus K_i : \ 48 \text{ bits} = B_1 B_2 ... B_8,$$

where each B_j has 6 bits.

(c) Now use the Substitution boxes or S-boxes on the output from $E(R) \oplus K_i$ (see next page). There are 8 S-boxes, each of which takes a 6-bit input and produces a 4-bit output. We feed B_1 into the S-box S_1, B_2 into the S-box S_2, etc. Since there will be 8 outputs of 4 bits each, we stick them back together to get a 32-bit total output.

Now, how do we use the S-boxes? Each of the 8 S-boxes has a table with 4 rows and 6 columns. Each entry in the table is a 4-bit number, (i.e. in the range 0-15), which written in binary will be the output of the S-box. The 6-bit input to the S-box specifies the row and column:

Let the six bits be $b_1, b_2, ...b_6$. Then row (expressed in binary notation) is $b_1 b_6$ and the column=$b_2 b_3 b_4 b_5$.

(d) **Example:** Suppose $B_4 = 111010$. What output would we get here?

S-box 1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-box 2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-box 3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-box 4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-box 5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-box 6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-box 7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-box 8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

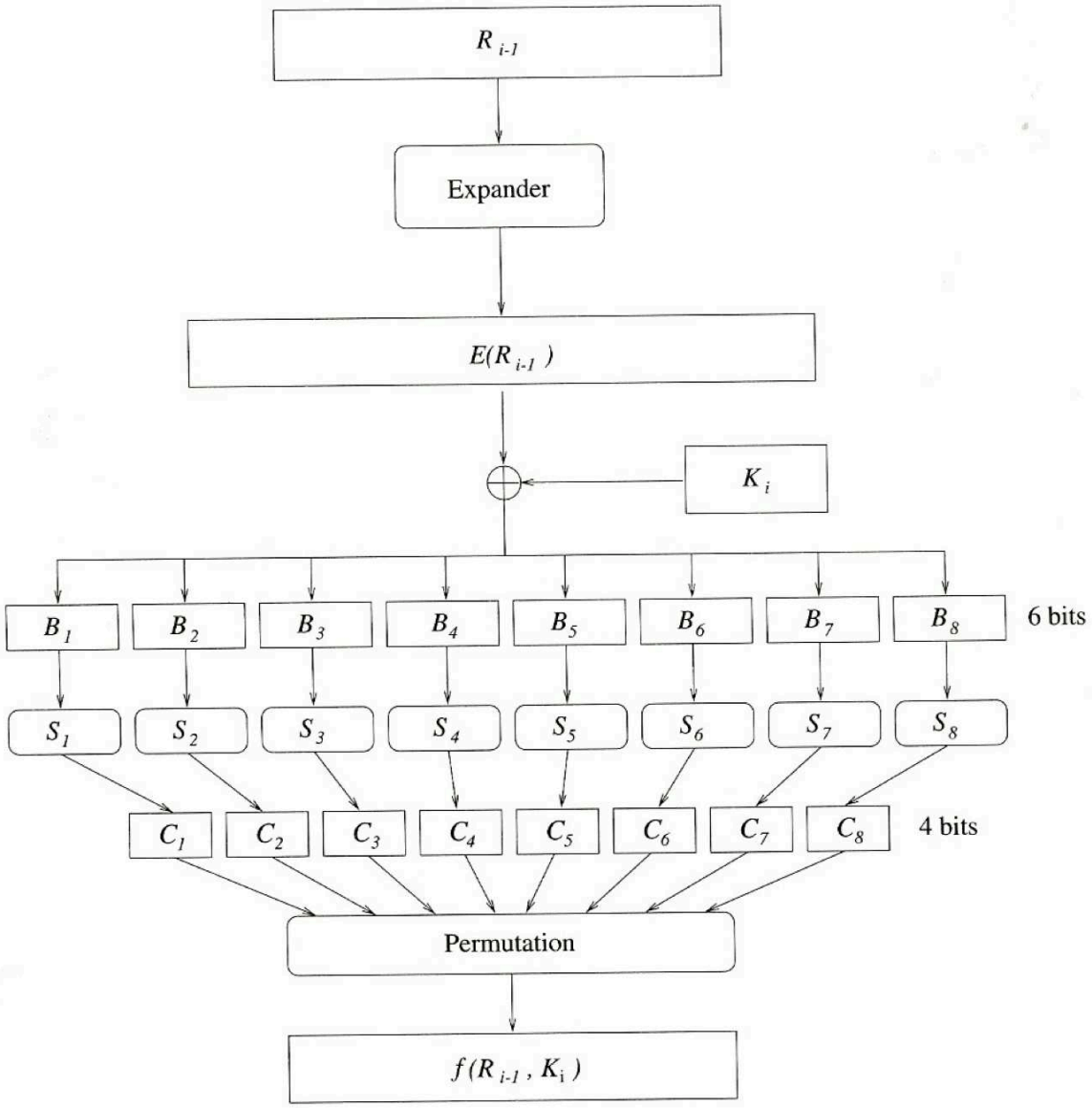
In this way we get 8 4-bit outputs, $C_1, ...C_8$. We put these together into a 32-bit string: $C_1C_2...C_8$.

(e) The string $C_1C_2...C_8$ is (genuinely) permuted according to the following table (also known as the P-box):

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

The resulting 32-bit string is $f(R, K_j)$.

The following is a graphical representation of the function $f(R_{i-1}, K_i)$:



8. How do we find the keys $K_1, ...K_{16}$? Each round uses a different 48-bit subkey of the 56-bit key, which is reduced from the original 64-bit key. The description of the details of how to use the key is called key scheduling.

   Start with 64-bit key K.

   (a) The 64-bit key has every eighth bit removed, and is rearranged according to the following key permutation:

Key Permutation													
57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

   Write the result as C_0D_0, where both C_0, D_0 have 28 bits.

   (b) For $1 \leq i \leq 16$, let $C_i = LS_i(C_{i-1})$ and $D_i = LS_i(D_{i-1})$, where LS_i means shift the input one or two places to the left (with wrap-around) according to the following table:

Number of Key Bits Shifted per Round															
Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

   We now have a shifted batch of 56 bits, C_iD_i.

   (c) 48 bits are chosen from the 56-bit string C_iD_i according to the following table. The formula is the same for each round's compression permutation. The output is K_i.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

9. See criteria on page 129.

10. What is the current status of DES? What is the current status of triple DES? What is the current status of AES?

Current Status of DES

1. The Data Encryption Standard was once a predominant symmetric-key algorithm for the encryption of electronic data.

2. On 17 March 1975, the proposed DES was published in the Federal Register. Public comments were requested, and in the following year two open workshops were held to discuss the proposed standard. There was some criticism from various parties, including from public-key cryptography pioneers Martin Hellman and Whitfield Diffie, citing a shortened key length and the mysterious "S-boxes" as evidence of improper interference from the NSA. The suspicion was that the algorithm had been covertly weakened by the intelligence agency so that they but no-one else could easily read encrypted messages. Alan Konheim (one of the designers of DES) commented, "We sent the S-boxes off to Washington. They came back and were all different." The United States Senate Select Committee on Intelligence reviewed the NSA's actions to determine whether there had been any improper involvement. In the unclassified summary of their findings, published in 1978, the Committee wrote:

   In the development of DES, NSA convinced IBM that a reduced key size was sufficient; indirectly assisted in the development of the S-box structures; and certified that the final DES algorithm was, to the best of their knowledge, free from any statistical or mathematical weakness.

   However, it also found that

   NSA did not tamper with the design of the algorithm in any way. IBM invented and designed the algorithm, made all pertinent decisions regarding it, and concurred that the agreed upon key size was more than adequate for all commercial applications for which the DES was intended.

3. DES is now considered to be insecure for many applications. This is chiefly due to the 56-bit key size being too small; in January, 1999, distributed.net and the Electronic Frontier Foundation collaborated to publicly break a DES key in 22 hours and 15 minutes (see chronology). There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are infeasible to mount in practice. The algorithm is believed to be practically secure in the form of Triple DES, although there are theoretical attacks. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES). Furthermore, DES has been withdrawn as a standard by the National Institute of Standards and Technology (formerly the National Bureau of Standards).

4. Triple DES: Has a level of security equivalent to a 112-bit key. One way to implement: Choose three keys K_1, K_2, K_3 and perform $E_{K_1}(E_{K_2}(E_{K_3}(m)))$.

5. AES (Advanced Encryption Standard): In 1997 the National Institute of Standards and Technology put out a call for candidates to replace DES (recall that part of the criteria for using DES is that it comes up for renewal every 5 years). The new algorithm was to allow key sizes of 128, 192, and 256 bits and operate on blocks of 128 input bits. Five finalists were chosen and Rijndael (Joan Daemen and Vincent Rijmen) was winner.