[1191], the algorithm was recertified for another five years [1150]. Software implementations of DES were finally allowed to be certified.

Anyone want to guess what will happen in 1998?

Applied Cryptography
Schneier

12.2 DESCRIPTION OF DES

DES is a block cipher; it encrypts data in 64-bit blocks. A 64-bit block of plaintext goes in one end of the algorithm and a 64-bit block of ciphertext comes out the other end. DES is a symmetric algorithm: The same algorithm and key are used for both encryption and decryption (except for minor differences in the key schedule).

The key length is 56 bits. (The key is usually expressed as a 64-bit number, but every eighth bit is used for parity checking and is ignored. These parity bits are the least-significant bits of the key bytes.) The key can be any 56-bit number and can be changed at any time. A handful of numbers are considered weak keys, but they can easily be avoided. All security rests within the key.

At its simplest level, the algorithm is nothing more than a combination of the two basic techniques of encryption: confusion and diffusion. The fundamental building block of DES is a single combination of these techniques (a substitution followed by a permutation) on the text, based on the key. This is known as a **round**. DES has 16 rounds; it applies the same combination of techniques on the plaintext block 16 times (see Figure 12.1).

The algorithm uses only standard arithmetic and logical operations on numbers of 64 bits at most, so it was easily implemented in late 1970s hardware technology. The repetitive nature of the algorithm makes it ideal for use on a special-purpose chip. Initial software implementations were clumsy, but current implementations are better.

Outline of the Algorithm

DES operates on a 64-bit block of plaintext. After an initial permutation, the block is broken into a right half and a left half, each 32 bits long. Then there are 16 rounds of identical operations, called Function f, in which the data are combined with the key. After the sixteenth round, the right and left halves are joined, and a final permutation (the inverse of the initial permutation) finishes off the algorithm.

In each round (see Figure 12.2), the key bits are shifted, and then 48 bits are selected from the 56 bits of the key. The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an XOR, sent through 8 S-boxes producing 32 new bits, and permuted again. These four operations make up Function f. The output of Function f is then combined with the left half via another XOR. The result of these operations becomes the new right half; the old right half becomes the new left half. These operations are repeated 16 times, making 16 rounds of DES.

If B_i is the result of the ith iteration, L_i and R_i are the left and right halves of B_i, K_i is the 48-bit key for round i, and f is the function that does all the substituting and permuting and XORing with the key, then a round looks like:

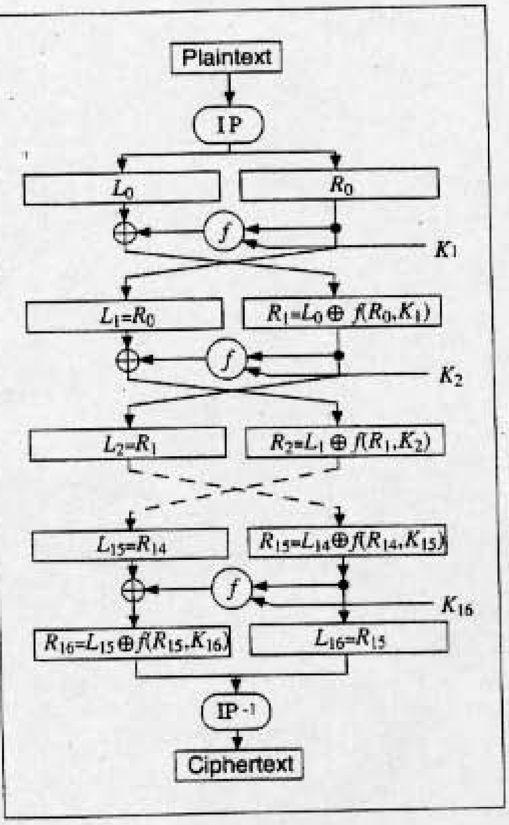
Plaintext
IP
L_0
R_0
f
K_1
$L_1=R_0$
$R_1=L_0 \oplus f(R_0,K_1)$
K_2
$L_2=R_1$
$R_2=L_1 \oplus f(R_1,K_2)$
$L_{15}=R_{14}$
$R_{15}=L_{14} \oplus f(R_{14},K_{15})$
K_{16}
$R_{16}=L_{15} \oplus f(R_{15},K_{16})$
$L_{16}=R_{15}$
IP^{-1}
Ciphertext

Figure 12.1 DES.

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

The Initial Permutation

The initial permutation occurs before round 1; it transposes the input block as described in Table 12.1. This table, like all the other tables in this chapter, should be read left to right, top to bottom. For example, the initial permutation moves bit 58 of the plaintext to bit position 1, bit 50 to bit position 2, bit 42 to bit position 3, and so forth.

The initial permutation and the corresponding final permutation do not affect DES's security. (As near as anyone can tell, its primary purpose is to make it easier to load plaintext and ciphertext data into a DES chip in byte-sized pieces. Remember that DES predates 16-bit or 32-bit microprocessor busses.) Since this bit-wise permutation is difficult in software (although it is trivial in hardware), many software implementations of DES leave out both the initial and final permutations. While this new algorithm is no less secure than DES, it does not follow the DES standard and should not be called DES.

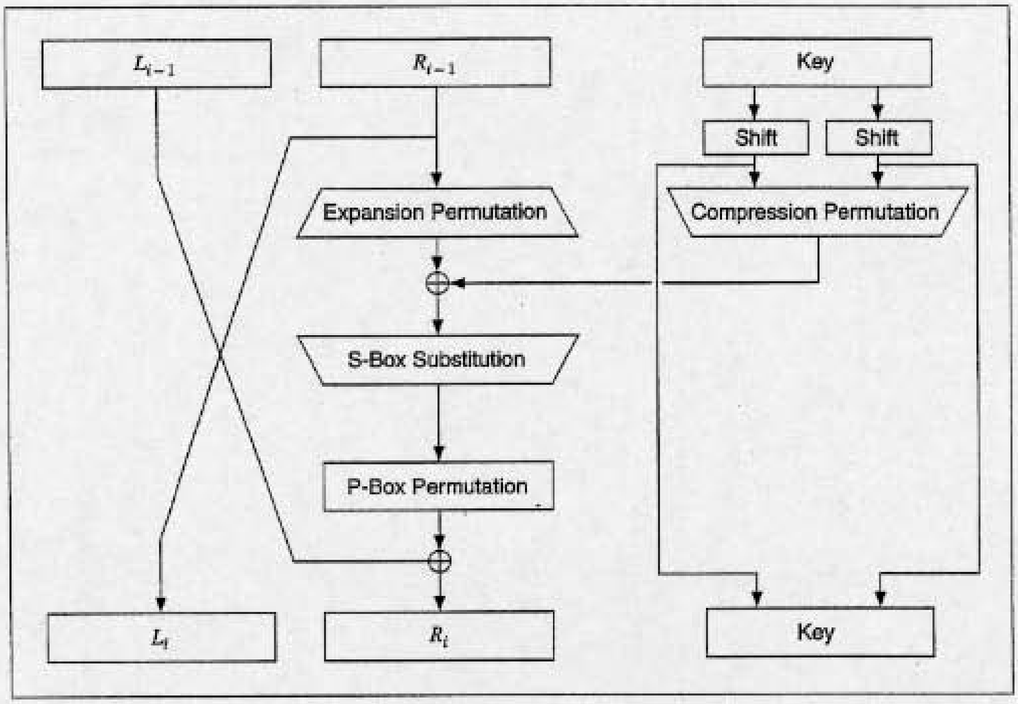

Figure 12.2 One round of DES.

The Key Transformation

Initially, the 64-bit DES key is reduced to a 56-bit key by ignoring every eighth bit. This is described in Table 12.2. These bits can be used as parity check to ensure the key is error-free. After the 56-bit key is extracted, a different 48-bit **subkey** is generated for each of the 16 rounds of DES. These subkeys, K_i, are determined in the following manner.

First, the 56-bit key is divided into two 28-bit halves. Then, the halves are circularly shifted left by either one or two bits, depending on the round. This shift is given in Table 12.3.

Table 12.1
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

Table 12.2
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

After being shifted, 48 out of the 56 bits are selected. Because this operation permutes the order of the bits as well as selects a subset of bits, it is called a **compression permutation**. This operation provides a subset of 48 bits. Table 12.4 defines the compression permutation (also called the permuted choice). For example, the bit in position 33 of the shifted key moves to position 35 of the output, and the bit in position 18 of the shifted key is ignored.

Because of the shifting, a different subset of key bits is used in each subkey. Each bit is used in approximately 14 of the 16 subkeys, although not all bits are used exactly the same number of times.

The Expansion Permutation

This operation expands the right half of the data, R_i, from 32 bits to 48 bits. Because this operation changes the order of the bits as well as repeating certain bits, it is known as an **expansion permutation**. This operation has two purposes: It makes the right half the same size as the key for the XOR operation and it provides a longer result that can be compressed during the substitution operation. However, neither of those is its main cryptographic purpose. By allowing one bit to affect two substitutions, the dependency of the output bits on the input bits spreads faster. This is called an **avalanche effect**. DES is designed to reach the condition of having every bit of the ciphertext depend on every bit of the plaintext and every bit of the key as quickly as possible.

Figure 12.3 defines the expansion permutation. This is sometimes called the **E-box**. For each 4-bit input block, the first and fourth bits each represent two bits of the output block, while the second and third bits each represent one bit of the output block. Table 12.5 shows which output positions correspond to which input positions. For example, the bit in position 3 of the input block moves to position 4 of the output block, and the bit in position 21 of the input block moves to positions 30 and 32 of the output block.

Although the output block is larger than the input block, each input block generates a unique output block.

Table 12.3
Number of Key Bits Shifted per Round

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table 12.4
Compression Permutation

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

The S-Box Substitution

After the compressed key is XORed with the expanded block, the 48-bit result moves to a substitution operation. The substitutions are performed by eight **substitution boxes**, or **S-boxes**. Each S-box has a 6-bit input and a 4-bit output, and there are eight different S-boxes. (The total memory requirement for the eight DES S-boxes is 256 bytes.) The 48 bits are divided into eight 6-bit sub-blocks. Each separate block is operated on by a separate S-box: The first block is operated on by S-box 1, the second block is operated on by S-box 2, and so on. See Figure 12.4.

Each S-box is a table of 4 rows and 16 columns. Each entry in the box is a 4-bit number. The 6 input bits of the S-box specify under which row and column number to look for the output. Table 12.6 shows all eight S-boxes.

The input bits specify an entry in the S-box in a very particular manner. Consider an S-box input of 6 bits, labeled b_1, b_2, b_3, b_4, b_5, and b_6. Bits b_1 and b_6 are combined to form a 2-bit number, from 0 to 3, which corresponds to a row in the table. The middle 4 bits, b_2 through b_5, are combined to form a 4-bit number, from 0 to 15, which corresponds to a column in the table.

For example, assume that the input to the sixth S-box (i.e., bits 31 through 36 of the XOR function) is 110011. The first and last bits combine to form 11, which cor-

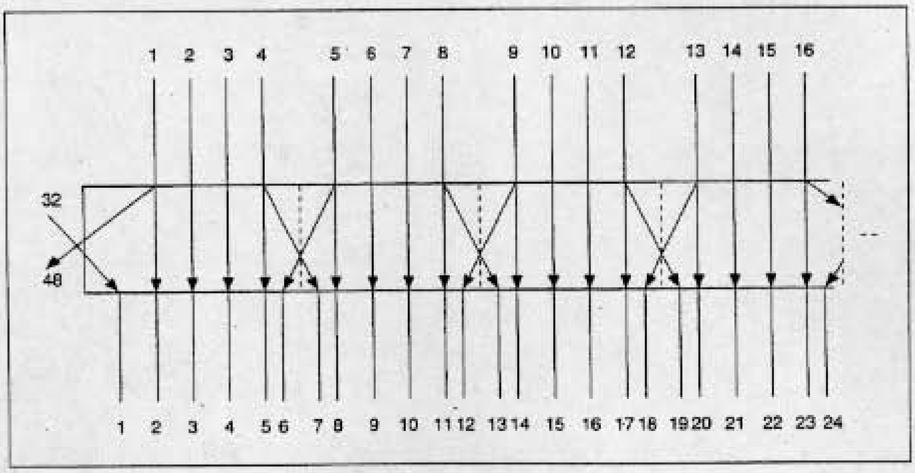

Figure 12.3 Expansion permutation.

Table 12.5
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

responds to row 3 of the sixth S-box. The middle 4 bits combine to form 1001, which corresponds to the column 9 of the same S-box. The entry under row 3, column 9 of S-box 6 is 14. (Remember to count rows and columns from 0 and not from 1.) The value 1110 is substituted for 110011.

It is, of course, far easier to implement the S-boxes in software as 64-entry arrays. It takes some rearranging of the entries to do this, but that's not hard. (Don't just change the indexing without rearranging the entries. The S-boxes are designed very carefully.) However, this way of describing the S-boxes helps visualize how they work. Each S-box can be viewed as a substitution function on a 4-bit entry: b_2 through b_5 go in, and a 4-bit result comes out. Bits b_1 and b_6 come from neighboring blocks; they select one out of four substitution functions available in the particular S-box.

The S-box substitution is the critical step in DES. The algorithm's other operations are linear and easy to analyze. The S-boxes are nonlinear and, more than anything else, give DES its security.

The result of this substitution phase is eight 4-bit blocks which are recombined into a single 32-bit block. This block moves to the next step: the P-box permutation.

The P-Box Permutation

The 32-bit output of the S-box substitution is permuted according to a **P-box**. This permutation maps each input bit to an output position; no bits are used twice and no bits are ignored. This is called a **straight permutation** or just a permutation. Table 12.7 shows the position to which each bit moves. For example, bit 21 moves to bit 4, while bit 4 moves to bit 31.

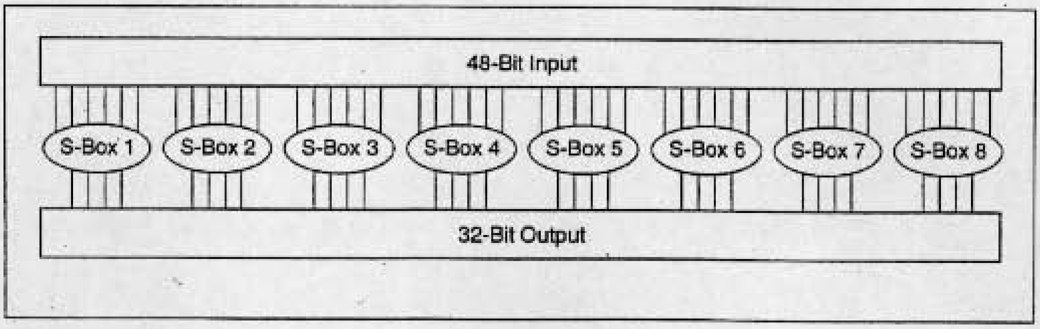

Figure 12.4 S-box substitution.

Table 12.6
S-Boxes

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

Table 12.7
P-Box Permutation

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

Finally, the result of the P-box permutation is XORed with the left half of the initial 64-bit block. Then the left and right halves are switched and another round begins.

The Final Permutation

The final permutation is the inverse of the initial permutation and is described in Table 12.8. Note that the left and right halves are not exchanged after the last round of DES; instead the concatenated block $R_{16}L_{16}$ is used as the input to the final permutation. There's nothing going on here; exchanging the halves and shifting around the permutation would yield exactly the same result. This is so that the algorithm can be used to both encrypt and decrypt.

Decrypting DES

After all the substitutions, permutations, XORs, and shifting around, you might think that the decryption algorithm is completely different and just as confusing as the encryption algorithm. On the contrary, the various operations were chosen to produce a very useful property: The same algorithm works for both encryption and decryption.

With DES it is possible to use the same function to encrypt or decrypt a block. The only difference is that the keys must be used in the reverse order. That is, if the encryption keys for each round are $K_1, K_2, K_3, \ldots, K_{16}$, then the decryption keys are $K_{16}, K_{15}, K_{14}, \ldots, K_1$. The algorithm that generates the key used for each round is circular as well. The key shift is a right shift and the number of positions shifted is 0,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1.

Modes of DES

FIPS PUB 81 specifies four modes of operation: ECB, CBC, OFB, and CFB (see Chapter 9) [1143]. The ANSI banking standards specify ECB and CBC for encryption, and CBC and n-bit CFB for authentication [52].

In the software world, certification is usually not an issue. Because of its simplicity, ECB is most often used in off-the-shelf commercial software products, although

Table 12.8
Final Permutation

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

it is the most vulnerable to attack. CBC is used occasionally, even though it is just slightly more complicated than ECB and provides much more security.

Hardware and Software Implementations of DES

Much has been written on efficient hardware and software implementations of the algorithm [997,81,533,534,437,738,1573,176,271,1572]. At this writing, the recordholder for the fastest DES chip is a prototype developed at Digital Equipment Corporation [512]. It supports ECB and CBC modes and is based on a GaAs gate array of 50,000 transistors. Data can be encrypted and decrypted at a rate of 1 gigabit per second, which translates to 16.8 million blocks per second. This is impressive. Table 12.9 gives the specifications for some commercial DES chips. Seeming discrepancies between clock speed and data rate are due to pipelining within the chip; a chip might have multiple DES engines working in parallel.

The most impressive DES chip is VLSI's 6868 (formerly called "Gatekeeper"). Not only can it perform DES encryption in only 8 clock cycles (prototypes in the lab can do it in 4 clock cycles), but it can also do ECB triple-DES in 25 clock cycles, and OFB or CBC triple-DES in 35 clock cycles. This sounds impossible to me, too, but I assure you it works.

A software implementation of DES on an IBM 3090 mainframe can perform 32,000 DES encryptions per second. Most microcomputers are slower, but impressive nonetheless. Table 12.10 [603,793] gives actual results and estimates for various Intel and Motorola microprocessors.

12.3 SECURITY OF DES

People have long questioned the security of DES [458]. There has been much speculation on the key length, number of iterations, and design of the S-boxes. The S-boxes were particularly mysterious—all those constants, without any apparent reason as to why or what they're for. Although IBM claimed that the inner workings were the result of 17 man-years of intensive cryptanalysis, some people feared that the NSA embedded a trapdoor into the algorithm so they would have an easy means of decrypting messages.

The U.S. Senate Select Committee on Intelligence, with full top-secret clearances, investigated the matter in 1978. The findings of the committee are classified, but an unclassified summary of those findings exonerated the NSA from any improper involvement in the algorithm's design [1552]. "It was said to have convinced IBM that a shorter key was adequate, to have indirectly assisted in the development of the S-box structures and to have certified that the final DES algorithm was, to the best of their knowledge, free of any statistical or mathematical weaknesses" [435]. However, since the government never made the details of the investigation public, many people remained unconvinced.

Tuchman and Meyer, two of the IBM cryptographers who designed DES, said the NSA did not alter the design [841]:

> Their basic approach was to look for strong substitution, permutation, and key scheduling functions. . . . IBM has classified the notes containing the selection