Standing the Test of Time: The Data Encryption Standard

Susan Landau

Fast and hard, that is all that cryptographers have ever wanted: a system that encrypts quickly but is essentially impossible to break. With their reliance on elementary number theory, public-key systems have captured mathematicians' imagination. Public-key algorithms are too slow to be used for most data transmissions, and instead public-key algorithms are used for establishing a key. Then a private-key system does the encryption. Private-key algorithms are typically faster than public-key ones.

The workhorse private-key algorithm is the Data Encryption Standard (DES), which relies on cryptographic design principles that predate public key. With the exception of RC4 in Web browsers and relatively insecure cable-TV signal encryption, DES is the most widely used public cryptosystem in the world. DES is the cryptographic algorithm used by banks for electronic funds transfer, DES is used for the protection of civilian satellite communications, and a variant of DES is used for UNIX password protection.

Proposed in 1975 and approved in 1977 as a Federal Information Processing Standard,[1] DES was immediately attacked by those who felt that its 56-bit key length was insecure. In spite of such claims, DES remained a strong encryption algorithm until the middle of the 1990s—several times longer than the government had reason to expect. Now, however, DES is past the end of its useful lifetime.

In the summer of 1998 DES's insecurity was definitively demonstrated when a $250,000 computer built by the Electronic Frontier Foundation (EFF) decrypted a DES-encoded message in 56 hours. In January 1999 this was improved to 22 hours through a combination of 100,000 networked PCs and the EFF machine. But until a substitute is found, DES remains a *de facto* standard. The National Institute of Standards and Technology (NIST)—whose predecessor, the National Bureau of Standards, certified DES—is currently seeking a successor to the algorithm. The Advanced Encryption Standard (AES) will work in three key lengths: 128, 192, and 256 bits. Fifteen candidates were submitted in June 1998 (there were actually twenty-one submissions, but six candidates had not fulfilled NIST's requirements). In August 1999 NIST eliminated ten of the fifteen. The agency is scheduled to pick DES's successor in the summer of 2000. The winning algorithm will be one whose security should stand well into the new century.

The publication of DES heralded a new era in cryptography. Academic and industrial researchers had an algorithm available for study that the National Security Agency had certified as secure. This helped develop a community of public cryptographers.

Susan Landau is an associate editor of the Notices *and is senior staff engineer at Sun Microsystems Inc. Her e-mail address is* `susan.landau@east.sun.com`.

[1] *This means the system is approved for sale to the federal government, an important issue for industry.*

When it came time to replace DES, there was a skilled community to take on the task.

In this article I outline DES, the cryptographic principles that underly its design, the algorithm's twenty-year history, and some of the strongest attacks against the algorithm. In a subsequent article I will present the cryptomathematics that evolved over these two decades and the AES effort. My intent is to illuminate the mathematics and politics behind block-structured cryptosystems.

What Is Wanted in a Cryptosystem?

Assume that the unencrypted message, the *plaintext*, is a string of bits. It is to be transformed into an encrypted string, or *ciphertext*, by means of a *cryptographic algorithm* and *key*. So that the recipient can read the message, encryption must be invertible.

Conventional wisdom holds that in order to defy easy decryption, a cryptographic algorithm should produce seeming chaos; that is, ciphertext should look and test random. In theory an eavesdropper should not be able to determine any significant information from an intercepted ciphertext.

One-time pads, whose keys are strings of random bits at least as long as the message itself, achieve this seeming impossibilty. Encryption is simple: if p_i is the i^{th} bit of the plaintext, k_i is the i^{th} bit of the key, and c_i is the i^{th} bit of the ciphertext, then $c_i = p_i \oplus k_i$, where $\oplus$ is exclusive or, often written XOR, and is simply addition modulo 2. Sender and recipient have a copy of the key. One-time pads must be used exactly once; if a key is ever reused, the system becomes highly vulnerable. In the early 1940s the Soviets made just such a mistake. Western intelligence discovered this and exploited it. Study of the messages encoded with the reused keys proved quite fruitful.[2] The constant need to refresh keying material eliminates much of the advantage of one-time pads. If we could efficiently and securely exchange keys, we could almost as easily securely transmit the plaintext, and we would have little need for a cryptosystem.

Broadly speaking, attacks on a cryptosystem fall into two categories: *passive attacks*, in which the adversary monitors the communication channel, and *active* ones, in which the adversary may transmit messages to obtain information (e.g., ciphertext of chosen plaintext). Passive attacks are easier to mount, but yield less. Attackers hope to determine the plaintext from the ciphertext they capture; an even more successful attack will determine the key and thus compromise a whole set of messages. An assumption first codified by Kerckhoffs in the nineteenth century is that the algorithm is known and that the security of the algorithm rests entirely in the secrecy of the key. Cryptographers design their algorithms to resist the following list of increasingly aggressive attacks:

- *ciphertext-only:* The adversary has access to the encrypted communications;
- *known-plaintext:* the adversary has some plaintext and its corresponding ciphertext;
- *chosen-text:* the adversary chooses the plaintext to be encrypted, or the adversary picks the ciphertext to be decrypted (chosen ciphertext), or the adversary chooses the plaintext to be encrypted depending on ciphertext received from previous requests (adaptive chosen plaintext).

Chosen-text attacks are largely used to simplify analysis of cryptosystems, but because of such devices as "smart cards" (credit card-sized objects equipped with a small processor), such attacks can occur in practice.

If an algorithm uses a k-bit key, the measure of security is how close the algorithm is to being 2^k-secure, that is, whether there are methods for breaking the system that are significantly better than a brute-force search of the entire key space. Sometimes an algorithm's weakness is readily apparent; such was the case for "Magenta", German Telecom's submission to the AES competition. The "key scheduling" (the order in which key bits are fed to the algorithm) was poorly designed, and this insecurity was discovered by rival cryptographers during the first public meeting to discuss the AES candidates.

Frequently, weaknesses may take years to discover. With DES, one strong form of attack, "differential cryptanalysis", had apparently been known to the algorithm's designers, but "linear cryptanalysis", discovered by Mitsuru Matsui [5] eighteen years after DES was proposed as a Federal Information Processing Standard, seems to be new. DES was indeed at least theoretically vulnerable to this type of attack. Designing secure cryptosystems is a mixture of a few well-known principles, some theorems, and, at least at present, some magic.

Block Cipher Designs

The simplest techniques for encrypting a block of symbols are substitution and permutation. *Substitution* replaces a symbol by another; *permutation* moves the symbols of a block around. Neither simple substitution nor simple permutation work very well by themselves. Frequency analysis, using the relative commonness of letters, pairs, triples, etc., is a strong tool against both.[3] Any message of reasonable length that is encrypted via a substitution or permutation function can be quickly deciphered using this technique; a trained

[2]*Details may be found at* `http://www.nsa.gov:8080/docs/venona/`.

[3]*In English, for example, the letter "e" appears 13% of the time in text, with "t,r,n,i,o,a,s" being the next most frequent letters. Similarly, there are data on the frequency of various letters appearing at the beginning and end of words, etc. Blanks (spaces) can be ignored.*

cryptanalyst can break a simple substitution cipher given only 25 characters of ciphertext.

Nonetheless, substitution and permutation form the backbone of modern cryptosystems. Fifty years ago Claude Shannon observed that the fundamental techniques for encryption are confusion—obscuring the relationship between the plaintext and the ciphertext—and diffusion—spreading the change throughout the ciphertext. Substitution is the simplest type of confusion, and permutation is the the simplest method of diffusion.

Cryptanalysis can be viewed as approximation theory; given ciphertext, determine the plaintext by an approximation process. Seen this way, linear functions of the input and key are poor design choices; such functions can be easily solved. Thus nonlinear functions form the basis of cryptographic design. But cryptographic functions must be invertible and fast-to-compute, and they should have small key size and memory requirements; consequently linear functions nonetheless play an essential role. A proper combination of simple operations such as $\oplus$, substitution, and permutation, produces a cryptosystem whose strength is greater than the sum of its parts.

The Data Encryption Standard (DES)

The three operations—XOR, substitution, and permutation—are all that is behind DES, which is an *iterated block cipher*, a cryptosystem on a block of symbols that sequentially repeats an internal function, called a *round*. It is customary at present to encrypt data using a primitive that operates on a block of symbols of moderate size. Although there are noniterative block ciphers (e.g., the public-key algorithm RSA), iteration is a natural way to proceed because that yields an algorithm with a small set of instructions, an important issue for hardware.

Some kind of self-invertibility is also valuable. This enables one object (a chip, a piece of software) to both encrypt and decrypt. *Feistel ciphers*, in which the $2t$-bit input is split into t-bit halves L_0, R_0 and mapped after r rounds to L_r, R_r, succinctly accomplish this. In the i^{th} round, the right half of the previous round becomes the new left half,

$$L_i := R_{i-1},$$

while the new right half, R_i, is the XOR of the previous left half and a function of a round subkey, K_i, and the previous right half:

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

An easy computation allows us to invert, obtaining L_{i-1} and R_{i-1} from L_i and R_i:

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

regardless of the round function f used. Decryption is the algorithm run in reverse, with subkeys used in the opposite order. In order to make decryption a genuine inverse of encryption, the final round of a Feistel cipher switches the ciphertext to (R_r, L_r). Put another way, in decryption the swap is done at the beginning of each round. DES is a 16-round Feistel cipher.

In 1965, when computers were clunky mainframes and the networked world was more science fiction than scientific fact, Congress charged the National Bureau of Standards (NBS) with developing computer standards for civilian use. In the early 1970s, the National Security Agency (NSA) and NBS realized that civilians needed to be able to secure their "sensitive but unclassified" data. Though NSA would have been the usual agency to build such a cryptosystem, the agency was reluctant to create an algorithm for public use. There was concern that work in cracking an NSA-designed algorithm might in turn enable attacks on other NSA-designed systems.

So NBS issued a public solicitation for a cryptosystem. IBM responded. Originally IBM's proposal was to have been a 16-round Feistel cipher[4] with a 64-bit key, but the company modified the submission to work with a 56-bit key. There have been persistent rumors that NSA had pressed for the shorter key length. But IBM claimed the truth was more mundane: IBM engineers had insisted on parity bits for register-to-register transfer of key data, thus decreasing the key length from 64 to 56 bits, with 8 bits for parity. The new algorithm became the Data Encryption Standard (DES).

In encrypting ordinary text, DES begins by grouping the text into 64-bit blocks. DES then performs a number of operations on each block. Throughout the message, a single key of 56 bits determines how the transformation of the blocks is to be carried out. DES iterates sixteen identical rounds of mixing; each round of DES uses a 48-bit subkey.

DES begins with an intitial permutation $\mathbb{P}$ and ends with its inverse, $\mathbb{P}^{-1}$. These permutations are of minor crytographic significance but form part of the official algorithm. Without these permutations the algorithm is not DES.

The selection of subkeys, or "key schedule", begins by splitting the 56-bit key into two 28-bit halves and rotating each half one or two bits (one bit in rounds 1, 2, 9, and 16; two bits otherwise). The two halves are put back together, and then 48 particular bits are chosen and put in order as follows:

[4]*Horst Feistel, whose career had included building cryptographic "Identification-friend-or-foe" systems for the Air Force, was part of the IBM design team.*

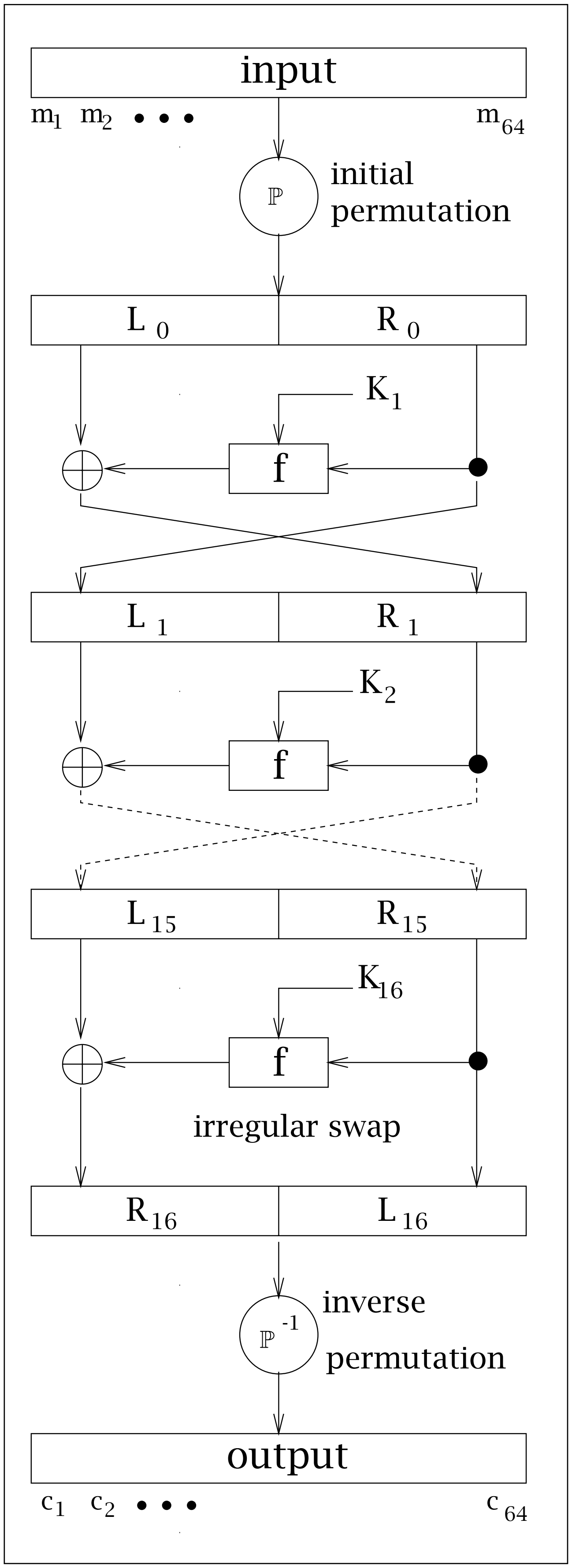

Figure 1. The Data Encryption Standard.

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 rotation ensures that a different subset of key bits is used for each of the sixteen rounds of DES.

DES is a standard Feistel construction:

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

where

$$f(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i)),$$

with the operations E (expansion), S (S-box lookup), and P (permutation) defined below.

In order to have one bit of input affect more than one bit of output, the right half of the data is expanded to 48 bits. This is done by the expansion operation E, which, beginning with bit 32 and cycling back to the beginning, uses all the bits in order, repeating every fourth and fifth bits. Figure 2 shows this.

Because of E, every bit of the output of a DES encryption depends on every bit of the plaintext and every bit of the key. Indeed, this is true after five rounds of the 16-round cipher.

Now the round subkey and the expanded right "half" of the data are XORed together. The result is passed through the "S-boxes". Each S-box takes input of six bits and outputs four bits. The outputs are concatenated, and as there are eight S-boxes there are 32 bits of output.

The S-boxes are the source of DES's complexity. One way to view the S-boxes is as having "inputs" b_2, b_3, b_4, b_5 and "instructions" b_1 and b_6. There are two possible values for each of b_1, b_6, and two possible values for each of b_2, b_3, b_4, b_5. The S-box may be written as a 4×16 table; for example, here is the "table" for S_5:

$$\begin{pmatrix} 2 & 12 & 4 & 1 & 7 & 10 & 11 & 6 & 8 & 5 & 3 & 15 & 13 & 0 & 14 & 9 \\ 14 & 11 & 2 & 12 & 4 & 7 & 13 & 1 & 5 & 0 & 15 & 10 & 3 & 9 & 8 & 6 \\ 4 & 2 & 1 & 11 & 10 & 13 & 7 & 8 & 15 & 9 & 12 & 5 & 6 & 3 & 0 & 14 \\ 11 & 8 & 12 & 7 & 1 & 14 & 2 & 13 & 6 & 15 & 0 & 9 & 10 & 4 & 5 & 3 \end{pmatrix}$$

The bits b_1, b_6 determine the row, while the bits b_2, b_3, b_4, b_5 determine the column; the output is the entry in the intersection. Note that each possible four-bit entry $0, \ldots, 15$ appears in each row of the S-box output. This is true for all rows of DES S-boxes. I will return to the S-boxes later.

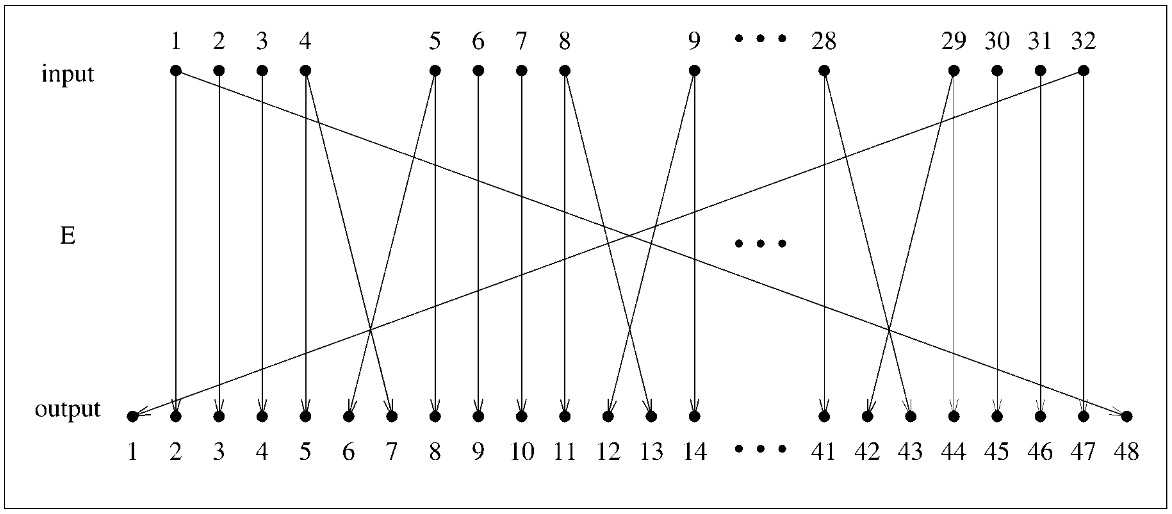

Figure 2. The Expansion Operation ($\mathcal{E}$).

P, a specific 32-bit permutation on the output of the S-boxes, completes the round function. It carries $1, \ldots, 32$ into the following list:

$$\begin{array}{rrrrrrrr} 16, & 7, & 20, & 21, & 29, & 12, & 28, & 17, \\ 1, & 15, & 23, & 26, & 5, & 18, & 31, & 10, \\ 2, & 8, & 24, & 14, & 32, & 27, & 3, & 9, \\ 19, & 13, & 30, & 6, & 22, & 11, & 4, & 25. \end{array}$$

P ensures that the output from one round of DES affects the input to multiple S-boxes in the next round. After completing the 16 rounds, DES ends with a final exchange of the left and right halves (and then $\mathbb{P}^{-1}$).

Some Observations about DES

The simplicity of DES gives rise to some not entirely desirable properties. One is complementation. Let $\overline{X}$ denote the bitwise complement of X. If C is the DES encryption of plaintext $\mathcal{P}$ with key $\mathcal{K}$, then $\overline{\mathcal{P}}$ is the DES encryption of $\overline{\mathcal{P}}$ with key $\overline{\mathcal{K}}$. Although in some cases complementation can simplify DES cryptanalysis by essentially cutting the search space in half, in general this property does not cause a serious weakness in the algorithm.

DES permutations do not form a group. Of course, the set generated by DES permutations does form a group. This group has at least 10^{2499} elements [6]. Lack of group structure is a DES strength; if DES were a group, multiple encryption by different keys, $(\mathcal{E}_{\mathcal{K}_k} \ldots (\mathcal{E}_{\mathcal{K}_1}(\mathcal{P}) \ldots))$, would not be any stronger than single encryption. Instead, it appears to be.

Surprisingly, double encryption, encryption twice by two different keys, $\mathcal{E}_{\mathcal{K}_2}(\mathcal{E}_{\mathcal{K}_1}(\mathcal{P}))$, is actually no stronger than single encryption. This is because of "meet-in-the-middle" attacks. Given a plaintext-ciphertext pair, an adversary computes all 2^{56} possible encipherings of the plaintext, $\mathcal{E}_{\mathcal{K}_i}(\mathcal{P})$, and indexes these. The adversary then computes all possible decipherings of the ciphertext, $\mathcal{E}_{\mathcal{K}_j}^{-1}(C)$, and compares these against the list of encrypted plaintexts. If there is a match, $\mathcal{E}_{\mathcal{K}_i}(\mathcal{P}) = \mathcal{E}_{\mathcal{K}_j}(C)$, then $\mathcal{K}_i, \mathcal{K}_j$ is a possible encryption pair. This pair of keys is checked against another plaintext-ciphertext pair to see whether the key pair is correct. The process continues until the correct encryption pair is found. The time to perform this computation is not much more than the time to break a single DES encryption.

Triple-DES encryption does not fall to meet-in-the-middle attacks. Triple-DES can also be implemented using just two keys, but this DES variation has been shown to be about 2^{56}-bit secure, rather than the 2^{108}-bit security one might expect. In recent years, triple-DES has become popular.

If all DES subkeys are equal, then $\mathcal{E}_{\mathcal{K}} = \mathcal{E}_{\mathcal{K}}^{-1}$. Any key that satisfies this condition is a "weak key", and there are four of them. "Semi-weak keys" are those pairs of keys $\mathcal{K}_i, \mathcal{K}_j$ such that $\mathcal{E}_{\mathcal{K}_i}(\mathcal{E}_{\mathcal{K}_j}(\mathcal{P})) = \mathcal{P}$. DES has six pairs of semi-weak keys. Finally, there are the "possibly weak keys", which generate only four subkeys, used four times each in DES. There are 48 possibly weak keys. As all DES weak, semi-weak, and possibly weak keys are known, they can be avoided and so present no problem to the security of the algorithm.

Attacks on DES

DES's selection was quickly followed by protests. Some researchers objected to the algorithm's small key space. The inventors of public-key cryptography, Whitfield Diffie and Martin Hellman, claimed that a \$20 million machine with a million specially designed VLSI chips, each capable of searching one key per microsecond and working in parallel, could break a DES-encoded message in about a day. David Chaum and Jan-Hendrik Evertse effectively used a meet-in-the-middle attack to break a four-round version of DES 2^{19} times faster than exhaustive search. Their technique did not extend past seven rounds.

None of these attacks posed serious threats to DES. Then two Israeli and one Japanese researcher poked harder into the innards of DES and discovered anomolies that led to the first attacks that were theoretically substantially better than exhaustive search. In 1990, looking at the XOR of plaintexts and ciphertexts, Eli Biham and Adi Shamir discovered a "differential-cryptanalysis" attack on DES that required examining only 2^{47} texts—a large number to be sure, but fewer than the 2^{56} that would be required by exhaustive search. Several years later Mitsuru Matsui, examining sums of plaintext and ciphertext bits, discovered relations that, in the aggregate, revealed information about sums of key bits. Matsui's "linear-cryptanalysis" attack on DES required studying 2^{43} encrypted texts—again, a large number, but again fewer than 2^{56}.

Ironically, neither differential nor linear cryptanalysis broke DES. Faster, cheaper chips did. Yet the Biham-Shamir and Matsui attacks are extremely important. These attacks work against *any* block-structured system, and so all block-structured cryptosystems must be designed to be secure against differential and linear cryptanalyis. Indeed, if one looks at the AES finalists, the choice of operations and the number of rounds of each of the candidates were frequently determined by differential and linear cryptanalysis. In what follows I describe the three serious attacks against DES—differential cryptanalysis, linear cryptanalysis, and the EFF DES Cracker—in the order in which they occurred.

Differential Cryptanalysis

As researchers studied DES, oddities about the S-boxes began to surface. Although the S-boxes have balanced output (each possible output appears four times, once in each "row"), there were subtle imbalances. In particular, the output of differences of inputs has an uneven distribution. Biham and Shamir exploited this in their 1990 differential cryptanalysis attack on DES.

Consider S-box 5 on page 344, and denote it by S_5. I use data from the difference distribution table in [1] for S_5; the interested reader can calculate this material directly from the description of S_5 given earlier. Notation is in hexadecimal, using characters $0_x, 1_x, \ldots, 9_x, A_x, B_x, \ldots, F_x$.

For each S-box I can create a "difference distribution table", a table of the distribution of all input XOR (there are 64 of these) and output XOR (there are 16 of these) pairs. The entries in the table are the number of possible pairs of a particular input difference and a particular output difference. Although the entries in the difference distribution table average 4, there is wide variance. This variance is exploited by differential cryptanalysis.

Suppose that the input difference of X and X^* to S_5, namely, $\Delta X = X \oplus X^*$, is 27_x. In this case the output difference has no chance of being $2_x, 4_x$, or F_x; $\frac{2}{64}$ chance of being $1_x, 3_x, 6_x, 7_x, D_x$, or E_x; $\frac{4}{64}$ chance for each of $8_x, 9_x$, or B_x; $\frac{8}{64}$ chance for each of A_x or C_x; and $\frac{12}{64}$ chance of being 0_x or 5_x. If I measure ΔX before and after processing by S_5, I would have a probability distribution on the key bits input to S_5 (or in other words, a probability distribution on the bits of K_5).

More formally, consider a pair of inputs to DES, X, X^*. The round function is the composition of E, $\oplus K$, the S-box transformation, and P. Observe that:

$$\begin{aligned} E(X) \oplus E(X^*) &= E(X \oplus X^*) \\ (X \oplus K) \oplus (X^* \oplus K) &= X \oplus X^* \\ P(X) \oplus P(X^*) &= P(X \oplus X^*). \end{aligned}$$

Furthermore, the output XOR of f is linear in the function that connects the rounds. If (Y, Y^*) is a pair of 32-bit strings, then

$$(X \oplus Y) \oplus (X^* \oplus Y^*) = (X \oplus X^*) \oplus (Y \oplus Y^*).$$

But the S-boxes are nonlinear, and $\mathrm{DES}(X \oplus X^*) \neq \mathrm{DES}(X) \oplus \mathrm{DES}(X^*)$.

Biham and Shamir discovered *characteristics* to help them push the knowledge gained from the XORs through the rounds. Informally, characteristics are differences in plaintext pairs that have a high probability of causing certain differences in ciphertext pairs. A trivial characteristic is input ΔX=0= output ΔX (that is, begin and end with the same string). This occurs, of course, with probability 1. A more interesting one-round characteristic has 0 as the input difference to seven S-boxes, while the input to the remaining S-box is nonzero and is chosen to maximize the probability the input ΔX may cause in the output. (Since several of the input bits to this remaining box also affect two neighboring S-boxes, these must be zero.) One high-probability way to do this is:

$$\begin{aligned} S_1 : C_x \to E_x \ &\text{ with probability } \frac{14}{64} \\ S_2, \ldots, S_8 : 00_x \to 0_x \ &\text{ with probability 1.} \end{aligned}$$

One can now concatenate the above two one-round characteristics to get a two-round characteristic that has probability $\frac{14}{64}$. Indeed, one can put these together to have a 3-round characteristic with probability $(\frac{14}{64})^2 \simeq .05$ ([1], p. 26); see Figure 3.

An *iterative characteristic* is one that can be concatenated with itself. Biham and Shamir developed what they believe is an optimal set. Their differential cryptanalysis attack on DES is:

1. Pick an appropriate difference ΔX.
2. Create an appropriate number of plaintext pairs with this ΔX, encrypt with DES, and store the ciphertext pairs.

3. For each pair, from the plaintext ΔX and the ciphertext pair, determine the expected output difference of as many S-boxes in the last round as possible.
4. For each possible key value, count the number of pairs that result with the expected output change using the value in the last DES round.
5. The right key value is the one suggested by all the key pairs.

Biham and Shamir found a 13-round characteristic that requires encryption of *only* 2^{47} chosen plaintexts. This finds 48 bits of the key used in round 16 and then determines the 8 other bits by exhaustive search (a relatively fast process in this case). The total time is essentially bounded by the time taken to do the 2^{47} encryptions.

Biham and Shamir experimented with various modified versions of DES. Without the permutation P, the algorithm lacks sufficiently quick diffusion from the S-boxes. Reordering the S-boxes leads to higher iterative characteristics that could be exploited by a differential-cryptanalysis attack. Almost every variation on DES that Biham and Shamir tried resulted in a weaker algorithm.

IBM said this was no accident. After the Biham-Shamir attack, Don Coppersmith, an IBM researcher who worked on the DES design, revealed the criteria used in the S-box design two decades earlier.

1. Each S-box should have six bits of input and four bits of output. (In 1974 this was the largest size S-box that could be accommodated if DES were to fit on a single chip.)
2. No output bit of an S-box should be too close to a linear function of the input bits. (The S-boxes are the only nonlinear part of DES. Their nonlinearity is the algorithm's strength.)
3. Each "row" of an S-box should contain all possible outputs. (This randomizes the output.)
4. If two inputs to an S-box differ in exactly one bit, their outputs should differ in at least two bits.
5. If two inputs to an S-box differ exactly in the middle two bits, their outputs must differ by at least two bits. (Criteria (4) and (5) provide some diffusion.)
6. If two inputs to an S-box differ in their first two bits and agree on their last two, the two outputs must differ.
7. For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference.

Call an S-box "active" if not all input differences to the box are zero. The S-boxes were designed to increase the number of active boxes. This maxim, along with a simplifying assumption that S-box events are statistically independent, ensures that with n active S-boxes, the probability of a particular pattern holding through n boxes is $1/4^n$.

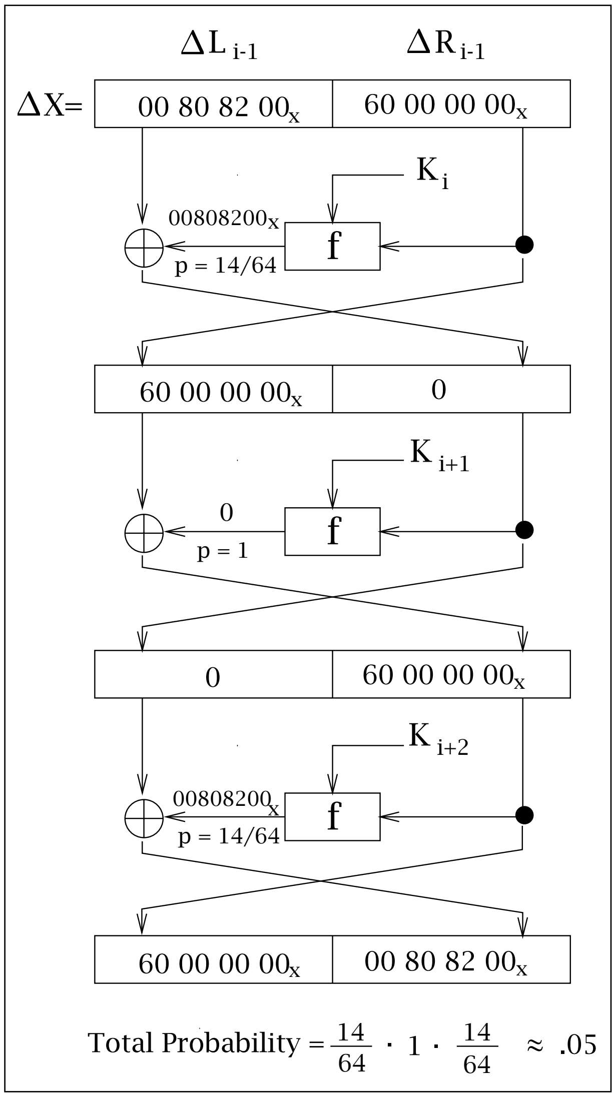

Figure 3. A Three-Round Characteristic.

Coppersmith commented that a better criterion than (2) would have been:

2′. No linear combination of output bits of an S-box should be too close to a linear function of the input bits.

While neither (2) nor (2′) could be perfectly achieved, (2′) would have increased DES's ability to resist differential cryptanalysis. So would larger S-boxes, but these were not possible in the technology of the time. There were also criteria to promote further randomization by the permutation P.

Linear Cryptanalysis

If DES was designed with differential cryptanalysis in mind, it seems clear the algorithm's developers had not anticipated Mitsuru Matsui's linear-cryptanalysis attack. Like many good insights, Matsui's idea was startingly simple. Cryptanalysts cannot expect that the ciphertext will be a linear function of plaintext and key bits (or equivalently, that some key bits are a linear function of the plaintext and ciphertext bits), but some of the bits were not too far off from some linear function.

Assume that input to DES will be random; then the mixing effect of $\mathbb{P}$ and $\mathbb{P}^{-1}$ can be ignored in performing this analysis. Let $B[i]$ denote the i^{th} bit of an array B of any length, and define

$$B[i_1, i_2, \dots, i_k] = B[i_1] \oplus B[i_2] \oplus \cdots \oplus B[i_k].$$

Let $\mathcal{P}$ be the 64-bit DES data after the initial permutation, C be the 64-bit DES data just before the final permutation, and $\mathcal{K}$ the key. Then find a set of bit positions $i_1, i_2 \dots, i_a, j_1, j_2, \dots, j_b, k_1, k_2, k_c$ such that for random plaintext and ciphertext the equation

$$\mathcal{P}[i_1, i_2, \dots, i_a] \oplus C[j_1, j_2, \dots, j_b] = \mathcal{K}[k_1, k_2, \dots, k_c]$$

holds with probability $p \neq 1/2$. (The farther p is from 1/2, the better.)

One computes the left-hand side for many plaintext-ciphertext pairs, then guesses the value for the right-hand side that occurs most often. This gives one bit of information about the key. If one does this computation for more than $|p - 1/2|^{-2}$ pairs, the chance of a wrong guess is small. Chaining single-round expressions together, one obtains an effective linear expression for DES.

The issue is to determine on which set of key bits to work. One looks at the correlations between the input and output bits in the various S-boxes, and different S-boxes work better than others. Matsui observed, for example, that the parity of the first, second, third, fourth, and sixth input bits of the third S-box agrees with the parity of all of the output bits 38 of 64 times. The second input bit of S_5 agrees with the XOR of all four output bits of S_5 with probability $\frac{12}{64} = 0.19$ (an observation first made by Shamir). Specifically, from the E expansion and the P permutation of the round function, one finds for a fixed key and random input X that the following holds with probability .19:

$$X[17] \oplus f[X,K][3,8,16,25] = K[26],$$

and the following holds with probability $1 - 0.19 = .81$:

$$X[15] \oplus f[X,K][3,8,16,25] = K[26] \oplus 1.$$

Here $f(X,K)$ is the DES round function.

For three rounds of DES, Matsui discovered that:

$$L_0[3,8,14,25] \oplus R_0[17] \oplus R_3[3,8,14,25] \oplus L_3[17] = K_1[26] \oplus K_3[26]$$

with probability .695. Linear cryptanalysis works by chaining together such relations. How many plaintexts must be examined? Matsui showed:

Theorem. *Let N be the number of given random plaintexts, and let p be, as above, the probability that*

$$\mathcal{P}[i_1, i_2, \dots, i_a] \oplus C[j_1, j_2, \dots, j_b] = \mathcal{K}[k_1, k_2, \dots, k_c].$$

If $|p - \frac{1}{2}|$ is sufficiently small, the success rate for the algorithm above is

$$\int_{-2\sqrt{N}\,|\frac{p-1}{2}|}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx.$$

Corollary. *With the same assumptions as above, the success rate of the algorithm is dependent only on $\sqrt{N}\,|\frac{p-1}{2}|$.*

Matsui calculated the following table:

N	$\frac{1}{4}\lvert\frac{p-1}{2}\rvert^{-2}$	$\frac{1}{2}\lvert\frac{p-1}{2}\rvert^{-2}$	$\lvert\frac{p-1}{2}\rvert^{-2}$
Success Rate	84.1%	92.1%	97.7%

As with differential cryptanalysis, the issue is how to join one-round characteristics into a longer chain.

Theorem. *Let $X_i, 1 \le i \le n$, be independent random variables whose values are 0 with probability p_i or 1 with probability $1 - p_i$. Then the probability that $X_1 \oplus X_2 \oplus \dots \oplus X_n$ equals 0 is*

$$\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2}\right).$$

Matsui determined the best linear approximate expressions for DES going up to 20 rounds (recall that DES is 16 rounds). Then he used a combination of reduced-rounds linear approximation and exhaustive search to find the key. His plaintext-ciphertext attack requires, on average, 2^{43} known plaintexts. With a network of twelve workstations in 1994, linear cryptanalysis broke a DES-encoded message in fifty days.

Breaking DES

Despite these successes, differential and linear cryptanalysis attacks are largely theoretical. The attack model is a problem. Consider Table 1, taken from [6], p. 259. In practice it is considerably easier to do an exhaustive search with one known plaintext-ciphertext pair and 2^{55} DES operations than it is to perform linear cryptanalysis requiring 2^{43} known plaintext-ciphertext pairs. Similarly, exhaustive search beats differential cryptanalysis on 16-round DES. Differential and linear cryptanalysis do pose threats to block-structured algorithms, and an attack that is successful even .01% of the time is potentially devastating.

In 1993 Michael Wiener updated the exhaustive-search machine to then-current technology. The

Attack Method	Data Complexity		Storage Complexity	Processing Complexity
	Known	Chosen		
Exhaustive Precomputation	—	1	2^{56}	1 (table lookup)
Exhaustive Search	1	—	negligible	2^{55}
Linear Cryptanalysis	2^{43}	—	for texts	2^{43}
Differential Cryptanalysis	—	2^{47}	for texts	2^{47}
Differential Cryptanalysis	2^{55}	—	for texts	2^{55}

Table 1.

result was a one-million-dollar machine using 57,000 DES chips and a "pipelined" architecture—one that admits sufficient parallelism so as to constantly feed data and perform computations through all components simultaneously. Wiener estimated that the machine could break a DES-encrypted message in three and a half hours [9]. Wiener's proposal raised interest in many circles.

In July 1998, using custom-designed chips and a personal computer, the Electronic Frontier Foundation built "DES Cracker". Costing less than \$250,000 and taking less than a year to build, DES Cracker broke a DES-encoded message in fifty-six hours. There was some luck in the process; the key was found after only a quarter of the key space was searched rather than the expected half. DES Cracker was built using 1,536 chips; these searched 88 billion keys per second. There was nothing terribly novel about the decryption machine except that it was built. The machine scales: if EFF spends another \$250,000 and links the resulting machines together, it would have a DES "Double-Cracker" that could decode DES-encrypted messages in half the time.

An Advanced Encryption Standard

Now even the U.S. Government had to agree that DES was insecure.[5] The National Institute for Standards and Technology had opened a competition for a DES replacement, the Advanced Encryption Standard (AES), a block-structured algorithm with variable-length keys of 128-, 192-, and 256-bits. Candidates were submitted in June 1998. Though the winner would be a U.S. Federal Information Processing Standard, the AES competition was open to foreign submissions. There were public evaluation meetings. (It was assumed that NSA was also doing some private vetting.) Fifteen candidates passed the initial criteria; after one year of evaluation, there are now five finalists. I will discuss these in a subsequent article.

[5] *Though a 1996 National Research Council report had urged that export controls on cryptography be immediately lifted to include the export of 56-bit DES, this change occurred only after DES Cracker had definitively established that DES was easily breakable. Cryptography controls are controversial (see [3] for a discussion).*

Acknowledgments

I am greatly indebted to Whitfield Diffie, Eli Biham, Germano Caronni, Neil Immerman, and Andrew Odlyzko, whose comments on an earlier version of this paper substantially improved it.

References

[1] Eli Biham and Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard,* Springer-Verlag, 1993.

[2] Don Coppersmith, The Data Encryption Standard (DES) and its strength against attacks, *IBM J. Res. Dev.*, **38** (1994), 243–250.

[3] Whitfield Diffie and Susan Landau, *Privacy on the Line: The Politics of Wiretapping and Encryption*, MIT Press, 1998.

[4] Electronic Frontier Foundation, *Cracking DES: Secrets of Encryption Research, Wiretap Politics, and Chip Design*, O'Reilly and Associates, 1998.

[5] Mitsuru Matsui, Linear Cryptanalysis method for DES cipher, *Eurocrypt '93*, Springer-Verlag, 1994, pp. 386–397.

[6] Alfred Menezes, Paul van Oorschot, and Scott Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[7] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, second edition, Wiley, New York, 1996.

[8] United States Department of Commerce, National Bureau of Standards, *Data Encryption Standard*, Federal Information Processing Standards (FIPS) Publication no. 46, January 15, 1977.

[9] Michael Wiener, Efficient DES Key Search, Technical Report TR-244, School of Computer Science, Carleton University, Ottawa, 1994; presented at the 1993 Crypto Rump Session.