1. Introductions Ciphers and Block Ciphers

A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher. A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used.

2. Stream Ciphers

Most of the ciphers we have presented so far are **stream ciphers**; that is, they convert one symbol of plaintext immediately into a symbol of ciphertext. (The exception is the columnar transposition cipher.) The transformation depends only on the symbol, the key, and the control information of the encipherment algorithm. A model of stream enciphering is shown in Figure 1.

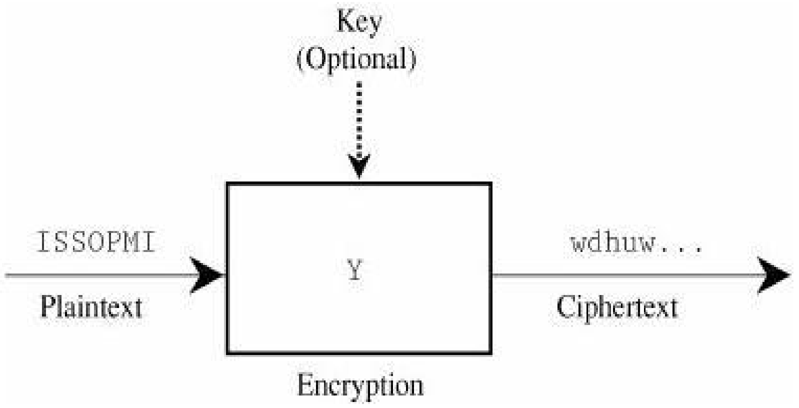


Figure 1: Stream Encryption.

Some kinds of errors, such as skipping a character in the key during encryption, affect the encryption of all future characters. However, such errors can sometimes be recognized during decryption because the plaintext will be properly recovered up to a point, and then all following characters will be wrong. If that is the case, the receiver may be able to recover from the error by dropping a character of the key on the receiving end. Once the receiver has successfully recalibrated the key with the ciphertext, there will be no further effects from this error.

3. Block Ciphers

To address this problem discussed above and make it harder for a cryptanalyst to break the code, block ciphers can be used. A **block cipher** encrypts a group of plaintext symbols as one block. The columnar transposition and other transpositions are examples of block ciphers.

In the columnar transposition, the entire message is translated as one block. The block size need not have any particular relationship to the size of a character.

Block ciphers work on blocks of plaintext and produce blocks of ciphertext, as shown in Figure 2. In the figure, the central box represents an encryption machine: The previous plaintext pair is converted to **po**, the current one being converted is **IH**, and the machine is soon to convert **ES**.

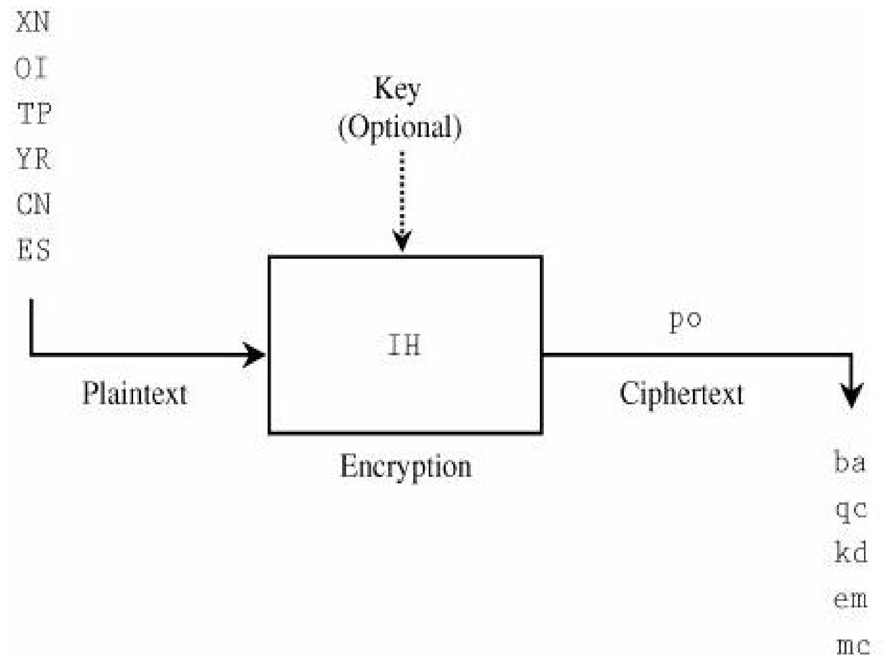


Figure 2. Block Cipher Systems.

Table 1: Comparing Stream and Block Algorithms

Stream Algorithms	**Block Algorithms**
• Speed of transformation. Because each symbol	• High diffusion. Information from the plain-text is diffused into several

Advantages	is encrypted without regard for any other plaintext symbols, each symbol can be encrypted as soon as it is read. Thus, the time to encrypt a symbol depends only on the encryption algorithm itself, not on the time it takes to receive more plaintext. • Low error propagation. Because each symbol is separately encoded, an error in the encryption process affects only that character.	ciphertext symbols. One ciphertext block may depend on several plaintext letters. Immunity to insertion of symbols. Because blocks of symbols are enciphered, it is impossible to insert a single symbol into one block. The length of the block would then be incorrect, and the decipherment would quickly reveal the insertion.
Disadvantages	• Low diffusion. Each symbol is separately enciphered. Therefore, all the information of that symbol is contained in one symbol of the ciphertext. • Susceptibility to malicious insertions and modifications. Because each symbol is separately enciphered, an active interceptor who has broken the code can splice together pieces of previous messages and transmit a spurious new message that may look authentic.	• Slowness of encryption. The person or machine using a block cipher must wait until an entire block of plaintext symbols has been received before starting the encryption process. • Error propagation. An error will affect the transformation of all other characters in the same block.

3.1 Block Cipher Principles

A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits. There are 2n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular.

Most symmetric block encryption algorithms in current use are based on a structure referred to as a Feistel block cipher. For that reason, it is important to examine the design principles of the Feistel cipher. In order to understand how block ciphers function, a motivation for the Feistel block cipher structure and its implications are given.

Table 2 Reversible Mapping

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Table 3 Irreversible Mapping

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	

In Table 3, a ciphertext of 01 could have been produced by one of two plaintext blocks. So if we limit ourselves to reversible mappings, the number of different transformations is 2n!.

Figure 3 illustrates the logic of a general substitution cipher for n = 4. A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits (confusion). The encryption and decryption mappings can be defined by a tabulation, as shown in Table 4. This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext. Feistel refers to this as the ideal block cipher, because it allows for the maximum number of possible encryption mappings from the plaintext block.

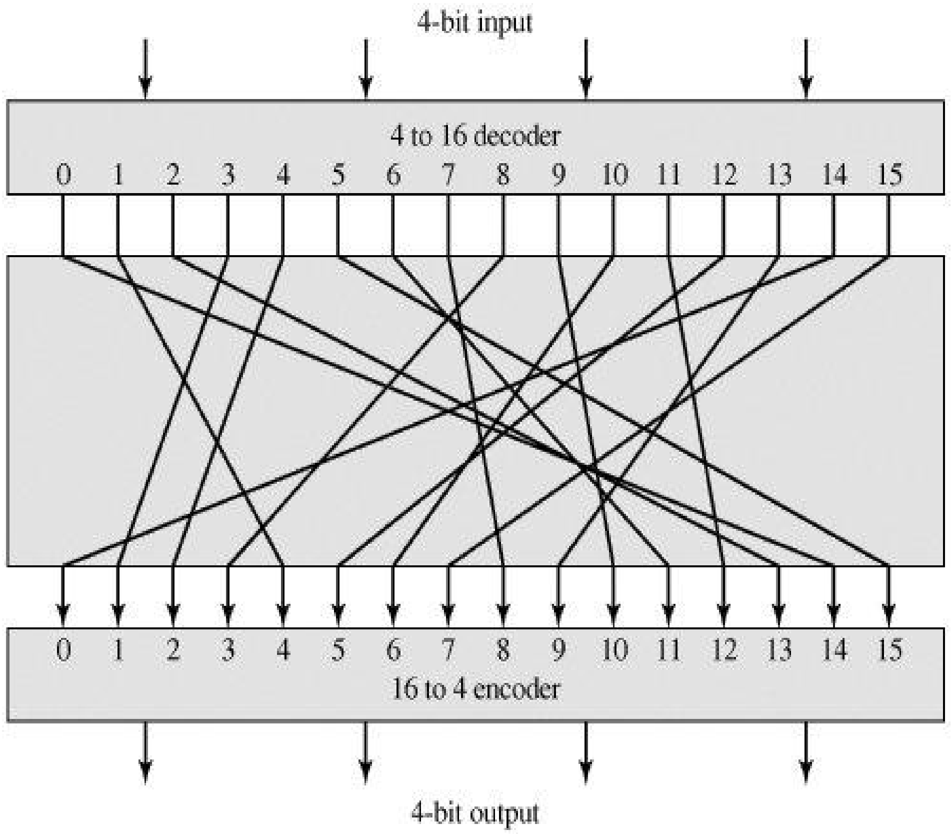


Figure 3: General n-bit-n-bit Block Substitution (shown with n = 4)

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111

0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Table 4: Encryption and Decryption Table for Substitution Cipher of Figure 3.

But there is a practical problem with the ideal block cipher. If a small block size, such as n = 4, is used, then the system is equivalent to a classical substitution cipher. Such systems, as we have seen, are vulnerable to a statistical analysis of the plaintext. This weakness is not inherent in the use of a substitution cipher but rather results from the use of a small block size. If n is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed, then the statistical characteristics of the source plaintext are masked to such an extent that this type of cryptanalysis is infeasible.

An arbitrary reversible substitution cipher (the ideal block cipher) for a large block size is not practical, however, from an implementation and performance point of view. For such a transformation, the mapping itself constitutes the key. Consider again Table 4, which defines one particular reversible mapping from plaintext to ciphertext for n = 4. The mapping can be defined by the entries in the second column, which show the value of the ciphertext for each plaintext block. This, in essence, is the key that determines the specific mapping from among all possible mappings. In this case, using this straightforward method of defining the key, the required key length is (4 bits) x (16 rows) = 64 bits. In general, for an n-bit ideal block cipher, the length of the key defined in this fashion is n x 2n bits. For a 64-bit block, which is a desirable length to thwart statistical attacks, the required key length is 64 x 264 = 270 1021bits.

In considering these difficulties, Feistel points out that what is needed is an approximation to the ideal block cipher system for large n, built up out of components that are easily realizable. But before turning to Feistel's approach, let us make one other observation. We could use the general block substitution cipher but, to make its implementation tractable, confine ourselves to a subset of the possible reversible mappings. For example, suppose we define the mapping in terms of a set of linear equations. In the case of n = 4, we have

y1= k11x1 + k12x2 + k13x3 + k14x4

y2= k21x1 + k22x2 + k23x3 + k24x4

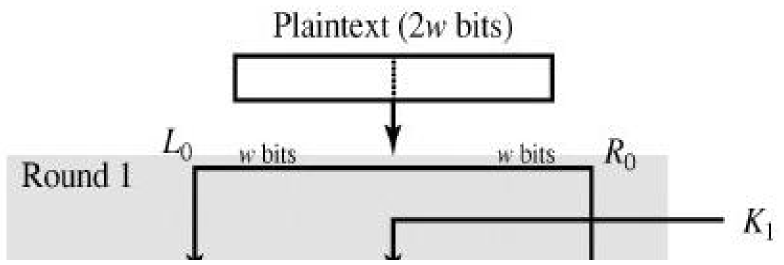
y3= k31x1 + k32x2 + k33x3 + k34x4

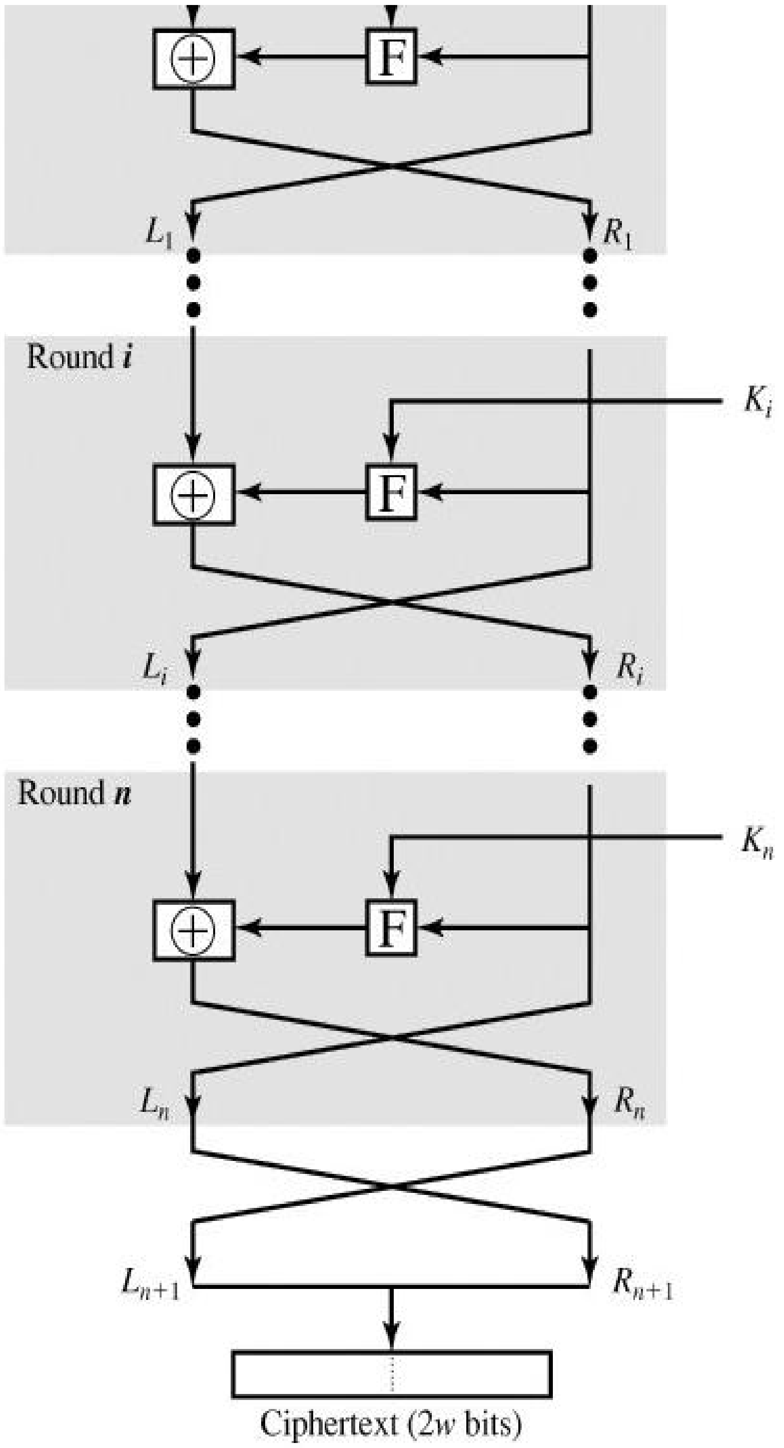
y4= k41x1 + k42x2 + k43x3 + k44x4

where the xi are the four binary digits of the plaintext block, the yi are the four binary digits of the ciphertext block, the kij are the binary coefficients, and arithmetic is mod 2. The key size is just n2, in this case 16 bits. The danger with this kind of formulation is that it may be vulnerable to cryptanalysis by an attacker that is aware of the structure of the algorithm.

3.2 Feistel Cipher Structure

Figure 4 depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K. The plaintext block is divided into two halves, L0 and R0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs Li-1 and Ri-1, derived from the previous round, as well as a subkey Ki, derived from the overall K. In general, the subkeys Ki are different from K and from each other.





L_1
R_1
Round *i*
K_i
F
L_i
R_i
Round *n*
K_n
F
L_n
R_n
L_{n+1}
R_{n+1}
Ciphertext (2*w* bits)

Figure 4: Classical Feistel Network

All rounds have the same structure. A substitution is performed on the left half of the data. This is done by applying a *round function* F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey Ki. Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data.

4. The Data Encryption Standard

Following the limitations found in the Feistel cipher, a more improved cipher algorithm called Data Encryption Algorithm was developed. This resulted in the Data Encryption Standard (DES). The DES principle of operation is similar to that of the Feistel cipher.

DES is the most widely used encryption scheme. It was adopted in 1977 by the National Bureau of Standards, now the US National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The algorithm itself is referred to as the Data Encryption Algorithm (DEA). For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption.

The DES algorithm is a careful and complex combination of two fundamental building blocks of encryption: substitution and transposition. The algorithm derives its strength from repeated application of these two techniques, one on top of the other, for a total of 16 cycles. The sheer complexity of tracing a single bit through 16 iterations of substitutions and transpositions has so far stopped researchers in the public from identifying more than a handful of general properties of the algorithm.

The algorithm begins by encrypting the plaintext as blocks of 64 bits. The key is 64 bits long, but in fact it can be any 56-bit number. (The extra 8 bits are often used as check digits and do not affect encryption in normal implementations.) The user can change the key at will any time there is uncertainty about the security of the old key.

The algorithm leverages the two techniques to conceal information: confusion and diffusion. That is, the algorithm accomplishes two things: ensuring that the output bits have no obvious relationship to the input bits and spreading the effect of one plaintext bit to other bits in the ciphertext. Substitution provides the confusion, and transposition provides the diffusion. In general, plaintext is affected by a series of cycles of a substitution then a permutation. The iterative substitutions and permutations are performed as outlined in Figure 5.

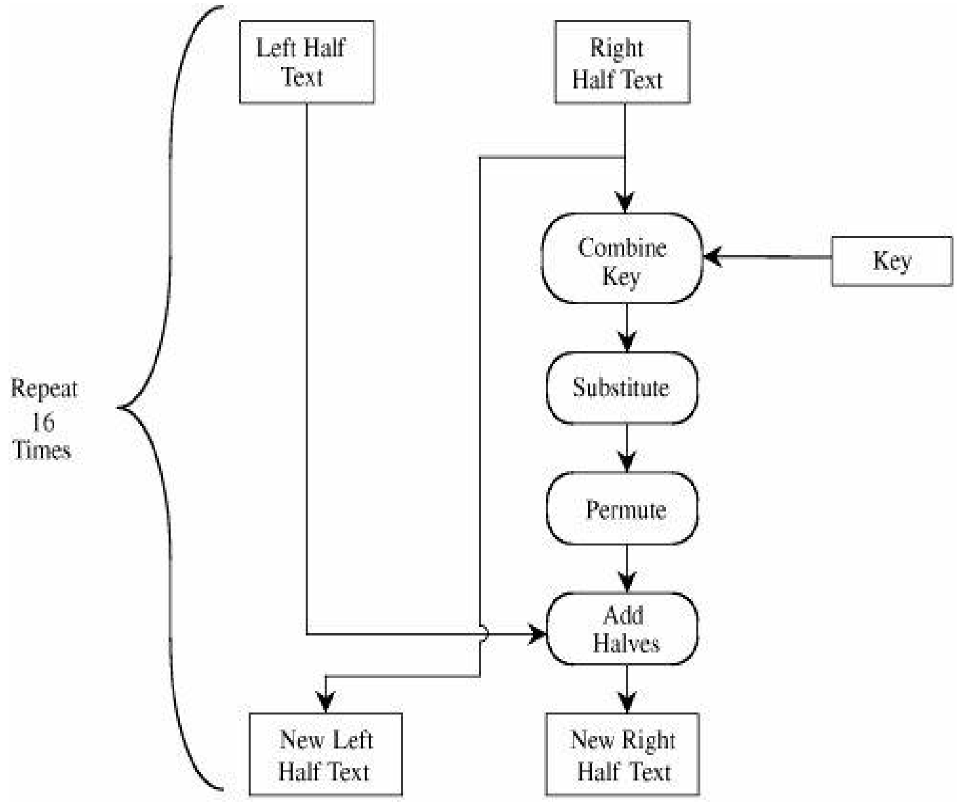


Figure 5: Cycles of Substitution and Permutation

Animation: Illustrate the Data Encryption

4.1 Double and Triple DES

The DES algorithm is fixed for a 56-bit key. Given the potential vulnerability of DES to a brute-force attack, there has been considerable interest in finding an alternative. One approach is to design a completely new algorithm. Another alternative, which would preserve the existing investment in software and equipment, is to use multiple encryption with DES and multiple keys. We examine the simplest example of this second alternative. We then look at the widely accepted triple Double DES and Triple DES (3DES) approach.

4.1.1 Double DES

To address the DES mistrust, some researchers suggested using a double encryption for greater secrecy. The double encryption works in the following way. Take two keys, k1 and k2, and perform two encryptions, one on top of the other: E(k2, E(k1,m)). In theory, this approach should multiply the difficulty of breaking the encryption, just as two locks are harder to pick than one.

Unfortunately, that assumption is false. In the literature it has been shown that two encryptions are no better than one. The basis of their argument is that the cryptanalyst works plaintext and ciphertext toward each other. The analyst needs two pairs of plaintext (call them P1 and P2) and corresponding ciphertext, C1 and C2, but not the keys used to encrypt them. The analyst computes and saves P1 encrypted under each possible key. The analyst then tries decrypting C1 with a single key and lookingfor a match in the saved Ps. A match is a possible pair of double keys, so the analyst checks the match with P2 and C2. Computing all the Ps takes 256 steps, but working backward from C1 takes only the same amount of time, for a total of 2 * 256 or 257, equivalent to a 57-bit key. Thus, the double encryption only doubles the work for the attacker. As we soon see, some 56-bit DES keys have been derived in just days; two times days is still days, when the hope was to get months if not years for the effort of the second encryption.

4.1.2 Triple DES

However, a simple trick does indeed enhance the security of DES. Using three keys adds significant strength.

The so-called **triple DES** procedure is C = E(k3, E(k2, E(k1,m))). That is, you encrypt with one key, decrypt with the second, and encrypt with a third. This process gives a strength equivalent to a 112-bit key (because the double DES attack defeats the strength of one of the three keys).

A minor variation of triple DES, which some people also confusingly call triple DES, is C = E(k1, D(k2, E(k1,m))). That is, you encrypt with one key, decrypt with the second, and encrypt with the first again. This version requires only two keys.

(The second decrypt step also makes this process work for single encryptions with one key: The decryption cancels the first encryption, so the net result is one encryption.) This approach is subject to another tricky attack, so its strength is rated at only about 80 bits.

In summary, ordinary DES has a key space of 56 bits, double DES is scarcely better, but two-key triple DES gives an effective length of 80 bits, and three-key triple DES gives a strength of 112 bits. Now, over three decades after the development of DES, a 56-bit key is inadequate for any serious confidentiality, but 80- and 112-bit effective key sizes provide reasonable security.

Animation: Illustrate the Data Encryption Standard

References

1. Stallings, William - Cryptography And Network Security 4Th Ed - Prentice Hall - (2005)
2. Security in Computing, Fourth Edition By Charles P. Pfleeger. Pfleeger Consulting Group, Shari Lawrence Pfleeger - RAND Corporation

Back to Top