Meadows [1311] has pointed out that under certain conditions it is possible to confuse the tag with data. Li and Wang [1174] examine the underlying reasons that a protocol is vulnerable to this attack.

12.1.5 Summary

Despite the use of sophisticated cryptosystems and random keys, cipher systems may provide inadequate security if not used carefully. The protocols directing how these cipher systems are used, and the ancillary information that the protocols add to messages and sessions, overcome these problems. This emphasizes that ciphers and codes are not enough. The methods, or protocols, for their use also affect the security of systems.

12.2 Stream and Block Ciphers

Some ciphers divide a message into a sequence of parts, or blocks, and encipher each block with the same key.

> **Definition 12–1.** Let E be an encryption algorithm, and let $E_k(b)$ be the encryption of message b with key k. Let a message $m = b_1 b_2 \ldots$, where each b_i is of a fixed length. Then a *block cipher* is a cipher for which $E_k(m) = E_k(b_1)E_k(b_2)\ldots$.

EXAMPLE: The AES is a block cipher. It breaks the message into 128-bit blocks and uses the same key to encipher each block.

Other ciphers use a nonrepeating stream of key elements to encipher characters of a message.

> **Definition 12–2.** Let E be an encryption algorithm, and let $E_k(b)$ be the encryption of message b with key k. Let a message $m = b_1 b_2 \ldots$, where each b_i is of a fixed length, and let $k = k_1 k_2 \ldots$ be the bits in k. Then a *stream cipher* is a cipher for which $E_k(m) = E_{k_1}(b_1)E_{k_2}(b_2)\ldots$.

If the key stream k of a stream cipher repeats itself, it is a *periodic cipher*.

EXAMPLE: The Vigenère cipher (see Section 10.2.2.1) is a stream cipher. Take b_i to be a character of the message and k_i to be a character of the key. This cipher is periodic, because the key is of finite length, and should the key be shorter than the message, the key is repeated.

The one-time pad (see Section 10.2.2.2) is also a stream cipher but is not periodic, because the key stream never repeats.

12.2.1 Stream Ciphers

The one-time pad is a cipher that can be proven secure (see Section 10.2.2.2, "One-Time Pad"). Bit-oriented ciphers implement the one-time pad by exclusive-or'ing each bit of the key with one bit of the message. For example, if the message is 00101 and the key is 10010, the ciphertext is $0 \oplus 1 \parallel 0 \oplus 0 \parallel 1 \oplus 0 \parallel 0 \oplus 1 \parallel 1 \oplus 0$ or 10111. But how can one generate a random, infinitely long key?

12.2.1.1 Synchronous Stream Ciphers

To simulate a random, infinitely long key, synchronous stream ciphers generate bits from a source other than the message itself. The simplest such cipher extracts bits from a register to use as the key. The contents of the register change on the basis of the current contents of the register.

> **Definition 12–3.** An *n-stage linear feedback shift register* (LFSR) consists of an n-bit *register* $r = r_0 \ldots r_{n-1}$ and an n-bit *tap sequence* $t = t_0 \ldots t_{n-1}$. To obtain a key bit, r_{n-1} is used, the register is shifted one bit to the right, and the new bit $r_0 t_0 \oplus \ldots \oplus r_{n-1} t_{n-1}$ is inserted.

EXAMPLE: Let the tap sequence for a four-stage LFSR be 1001, and let the initial value of the register be 0010. The key bits extracted, and the values in the register, are

current register	*key*	*new bit*	*new register*
0010	0	$01 \oplus 00 \oplus 10 \oplus 01 = 0 \oplus 0 \oplus 0 \oplus 0 = 0$	0001
0001	1	$01 \oplus 00 \oplus 00 \oplus 11 = 0 \oplus 0 \oplus 0 \oplus 1 = 1$	1000
1000	0	$11 \oplus 00 \oplus 00 \oplus 01 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$	1100
1100	0	$11 \oplus 10 \oplus 00 \oplus 01 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$	1110
1110	0	$11 \oplus 10 \oplus 10 \oplus 01 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$	1111
1111	1	$11 \oplus 10 \oplus 10 \oplus 11 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$	0111
0111	1	$01 \oplus 10 \oplus 10 \oplus 11 = 0 \oplus 0 \oplus 0 \oplus 1 = 1$	1011
1011	1	$11 \oplus 00 \oplus 10 \oplus 11 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$	0101
0101	1	$01 \oplus 10 \oplus 00 \oplus 11 = 0 \oplus 0 \oplus 0 \oplus 1 = 1$	1010
1010	0	$11 \oplus 00 \oplus 10 \oplus 01 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$	1101
1101	1	$11 \oplus 10 \oplus 00 \oplus 11 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$	0110
0110	0	$01 \oplus 10 \oplus 10 \oplus 01 = 0 \oplus 0 \oplus 0 \oplus 0 = 0$	0011
0011	1	$01 \oplus 00 \oplus 10 \oplus 11 = 0 \oplus 0 \oplus 0 \oplus 1 = 1$	1001
1001	1	$11 \oplus 00 \oplus 00 \oplus 11 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$	0100
0100	0	$01 \oplus 10 \oplus 00 \oplus 01 = 0 \oplus 0 \oplus 0 \oplus 0 = 0$	0010
0010	0	$01 \oplus 00 \oplus 10 \oplus 01 = 0 \oplus 0 \oplus 0 \oplus 0 = 0$	0001

and the cycle repeats. The key stream that this LFSR produces has a period of 15 and is 010001111010110.

The LFSR method is an attempt to simulate a one-time pad by generating a long key sequence from a little information. As with any such attempt, if the key is shorter than the message, breaking part of the ciphertext gives the cryptanalyst information about other parts of the ciphertext. For an LFSR, a known plaintext attack can reveal parts of the key sequence. If the known plaintext is of length $2n$, the tap sequence for an n-stage LFSR can be determined completely.

Nonlinear feedback shift registers do not use tap sequences; instead, the new bit is a function of the current register bits.

> **Definition 12–4.** An *n-stage nonlinear feedback shift register* (NLFSR) consists of an n-bit register $r = r_0 \ldots r_{n-1}$. To obtain a key bit, r_{n-1} is used, the register is shifted one bit to the right, and the new bit is set to $f(r_0, \ldots, r_{n-1})$, where f is any function of n inputs.

EXAMPLE: Let the function f for a four-stage NLFSR be $f(r_0, r_1, r_2, r_3) = (r_0 \text{ and } r_2) \text{ or } r_3$, and let the initial value of the register be 1100. The key bits extracted, and the values in the register, are

current register	*key*	*new bit*	*new register*
1100	0	$f(1, 1, 0, 0) = (1 \text{ and } 0) \text{ or } 0 = 0$	0110
0110	0	$f(0, 1, 1, 0) = (0 \text{ and } 1) \text{ or } 0 = 0$	0011
0011	1	$f(0, 0, 1, 1) = (0 \text{ and } 1) \text{ or } 1 = 1$	1001
1001	1	$f(1, 0, 0, 1) = (1 \text{ and } 0) \text{ or } 0 = 0$	0100
0100	0	$f(0, 1, 0, 0) = (0 \text{ and } 0) \text{ or } 0 = 0$	0010
0010	0	$f(0, 0, 1, 0) = (0 \text{ and } 1) \text{ or } 0 = 0$	0001
0001	1	$f(0, 0, 0, 1) = (0 \text{ and } 0) \text{ or } 1 = 1$	1000
1000	0	$f(1, 0, 0, 0) = (1 \text{ and } 0) \text{ or } 0 = 0$	0100
0100	0	$f(0, 1, 0, 0) = (0 \text{ and } 1) \text{ or } 0 = 0$	0010
0010	0	$f(0, 0, 1, 0) = (0 \text{ and } 1) \text{ or } 0 = 0$	0001

and the cycle repeats. The key stream that this NLFSR produces has a period of 4 (with an initial nonrepeating sequence of length 4) and is $0011\overline{0010}\ldots$ (the overstruck part repeats indefinitely).

NLFSRs are not common because there is no body of theory about how to build NLFSRs with long periods. By contrast, it is known how to design n-stage LFSRs with a period of $2^n - 1$, and that period is maximal.

A second technique for eliminating linearity is called *output feedback mode*. Let E be an encryption function. Define k as a cryptographic key, and define r as a register. To obtain a bit for the key, compute $E_k(r)$ and put that value into the register. The rightmost bit of the result is exclusive-or'ed with one bit of the message. The process is repeated until the message is enciphered. The key k and the initial value in r are the keys for this method. This method differs from the NLFSR in that the register is never shifted. It is repeatedly enciphered.

A variant of output feedback mode is called the *counter method*. Instead of using a register r, simply use a counter that is incremented or otherwise transformed for every encipherment, so that the value of the counter is unique for each encryption. The initial value of the counter replaces r as part of the key. This method enables one to generate the ith bit of the key without generating the bits $0, \ldots, i-1$. If the initial counter value is i_0 and the value is incremented for each encryption, set the register to $i + i_0$. By way of contrast, in output feedback mode, one must generate all the preceding key bits.

12.2.1.2 Self-Synchronous Stream Ciphers

Self-synchronous ciphers obtain the key from the message itself. The simplest self-synchronous cipher is called an *autokey* cipher and uses the message itself for the key.

EXAMPLE: The following is an autokey version of the Vigenère cipher, with the key drawn from the plaintext:

key	XTHEBOYHASTHEBA
plaintext	THEBOYHASTHEBAG
ciphertext	QALFPNFHSLALFCT

Contrast this with the example on page 295. The key there is "VIG" and the resulting ciphertext contains a three-character repetition.

The problem with this cipher is the selection of the key. Unlike a one-time pad, any statistical regularities in the plaintext show up in the key. For example, the last two letters of the ciphertext associated with the plaintext word "THE" are always "AL," because "H" is enciphered with the key letter "T" and "E" is enciphered with the key letter "H." Furthermore, if the analyst can guess any letter of the plaintext, she can determine all successive plaintext letters.

An alternative is to use the ciphertext as the key stream. A good cipher will produce pseudorandom ciphertext, which approximates a random one-time pad better than a message with nonrandom characteristics (such as a meaningful English sentence).

EXAMPLE: The following is an autokey version of the Vigenère cipher, with the key drawn from the ciphertext:

key	XQXBCQOVVNGNRTT
plaintext	THEBOYHASTHECAT
ciphertext	QXBCQOVVNGNRTTM

This eliminates the repetition ("ALF") in the preceding example.

This type of autokey cipher is weak, because plaintext can be deduced from the ciphertext. For example, consider the first two characters of the ciphertext, "QX." The "X" is the ciphertext resulting from enciphering some letter with the key "Q." Deciphering, the unknown letter is "H." Continuing in this fashion, the analyst can reconstruct all of the plaintext except for the first letter.

A variant of the autokey method, *cipher feedback mode*, uses a shift register. Let E be an encipherment function. Define k as a cryptographic key and r as a register. To obtain a bit for the key, compute $E_k(r)$. The rightmost bit of the result is exclusive-or'ed with one bit of the message, and the other bits of the result are discarded. The resulting ciphertext is fed back into the leftmost bit of the register, which is right shifted one bit. (See Figure 12–1.)

Cipher feedback mode has a *self-healing property*. If a bit is corrupted in transmission of the ciphertext, the next n bits will be deciphered incorrectly. But after n uncorrupted bits have been received, the shift register will be reinitialized to the value used for encipherment and the ciphertext will decipher properly from that point on.

As in the counter method, one can decipher parts of messages enciphered in cipher feedback mode without deciphering the entire message. Let the shift register contain n bits. The analyst obtains the previous n bits of ciphertext. This is the value in the shift register before the bit under consideration was enciphered. The decipherment can then continue from that bit on.

12.2.2 Block Ciphers

Block ciphers encipher and decipher multiple bits at once using the same key. Errors in transmitting one block generally do not affect other blocks, but as each block is enciphered independently, using the same key, identical plaintext blocks produce identical ciphertext blocks. This allows the analyst to search for data by determining what the encipherment of a specific plaintext block is.

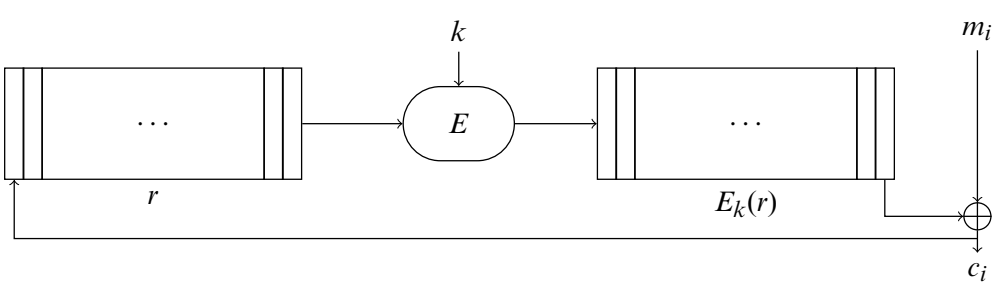

Figure 12–1 Diagram of cipher feedback mode. The register *r* is enciphered with key *k* and algorithm *E*. The rightmost bit of the result is exclusive-or'ed with one bit of the plaintext m_i to produce the ciphertext bit c_i. The register *r* is right-shifted one bit, and c_i is fed back into the leftmost bit of *r*.

To prevent this type of attack, some information related to the block's position is inserted into the plaintext block before it is enciphered. The information can be bits from the preceding ciphertext block [658] or a sequence number [1037]. The disadvantage is that the effective block size is reduced, because fewer message bits are present in a block.

Cipher block chaining does not require the extra information to occupy bit spaces, so every bit in the block is part of the message. The CBC mode is an iterative mode in which a block of ciphertext depends not only on its input but also on the preceding ciphertext block. Before a plaintext block is enciphered, that block is exclusive-or'ed with the preceding ciphertext block. In addition to the key, this technique requires an *initialization vector* with which to exclusive-or the initial plaintext block. Taking E_k to be the encipherment algorithm with key k, and I to be the initialization vector, the cipher block chaining technique is

$$c_0 = E_k(m_0 \oplus I)$$
$$c_i = E_k(m_i \oplus c_{i-1}) \text{ for } i > 0$$

Figure 12–2 shows this mode, and Figure 12–3 visually compares the effect of enciphering an image without and with cipher block chaining.

Like cipher feedback mode, CBC mode has the *self-healing property*. If one block of ciphertext is altered, the error propagates for at most two blocks. Figure 12–4 shows how a corrupted block affects others.

12.2.2.1 Multiple Encryption

Other approaches involve multiple encryption. Using two keys k and k' of length n to encipher a message as $c = E_{k'}(E_k(m))$ looks attractive because it has an effective key length of $2n$, whereas the keys to E are of length n. However, Merkle and Hellman [1325] have shown that this encryption technique can be broken using 2^{n+1} encryptions, rather than the expected 2^{2n} (see Exercise 3).

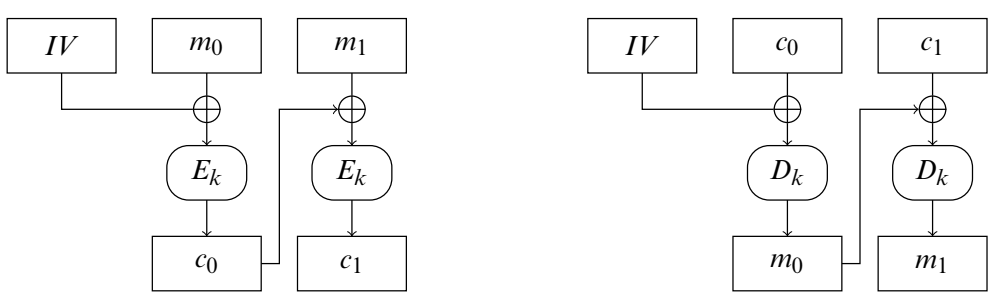

Figure 12–2 Cipher block chaining mode. The left diagram shows encipherment; each ciphertext is "fed back" into the cipher stream. The right diagram shows decipherment.