جامـــــعة المـــــستقبل
AL MUSTAQBAL UNIVERSITY

قســـم الامـــــن الـــــــــسيبرانـــــــي

Department of Cyber Security

Subject:

Blowfish

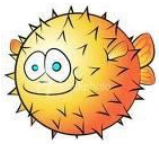
Class:

Second

Lecturer:

Asst. lect. Mustafa Ameer Awadh

Lecture: (6)



Blowfish

Blowfish is optimized for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than DES when implemented on 32-bit microprocessors with large data caches, such as the Pentium and the PowerPC. Blowfish is not suitable for applications, such as packet switching, with frequent key changes, or as a one-way hash function. Its large memory requirement makes it infeasible for smart card applications.

1. Fast. Blowfish encrypts data on 32-bit microprocessors at a rate of 26 clock cycles per byte.
2. Compact. Blowfish can run in less than 5K of memory.
3. Simple. Blowfish uses only simple operations: addition, XORs, and table lookups on 32-bit operands. Its design is easy to analyze which makes it resistant to implementation errors.
4. Variably Secure. Blowfish's key length is variable and can be as long as 448 bits.

Description of Blowfish

Blowfish is a 64-bit block cipher with a variable-length key. The algorithm consists of two parts:

key expansion and data encryption. Key expansion converts a key of up to 448 bits into several subkey arrays totaling 4168 bytes.

Data encryption consists of a simple function iterated 16 times. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are additions and XORs on 32 bit words. The only additional operations are four indexed array data lookups per round.

Blowfish uses a large number of subkeys. These keys must be precomputed before any data encryption or decryption.

The P-array consists of 18 32-bit subkeys:

$$P_1, P_2,..., P_{18}$$

Four 32-bit S-boxes have 256 entries each:

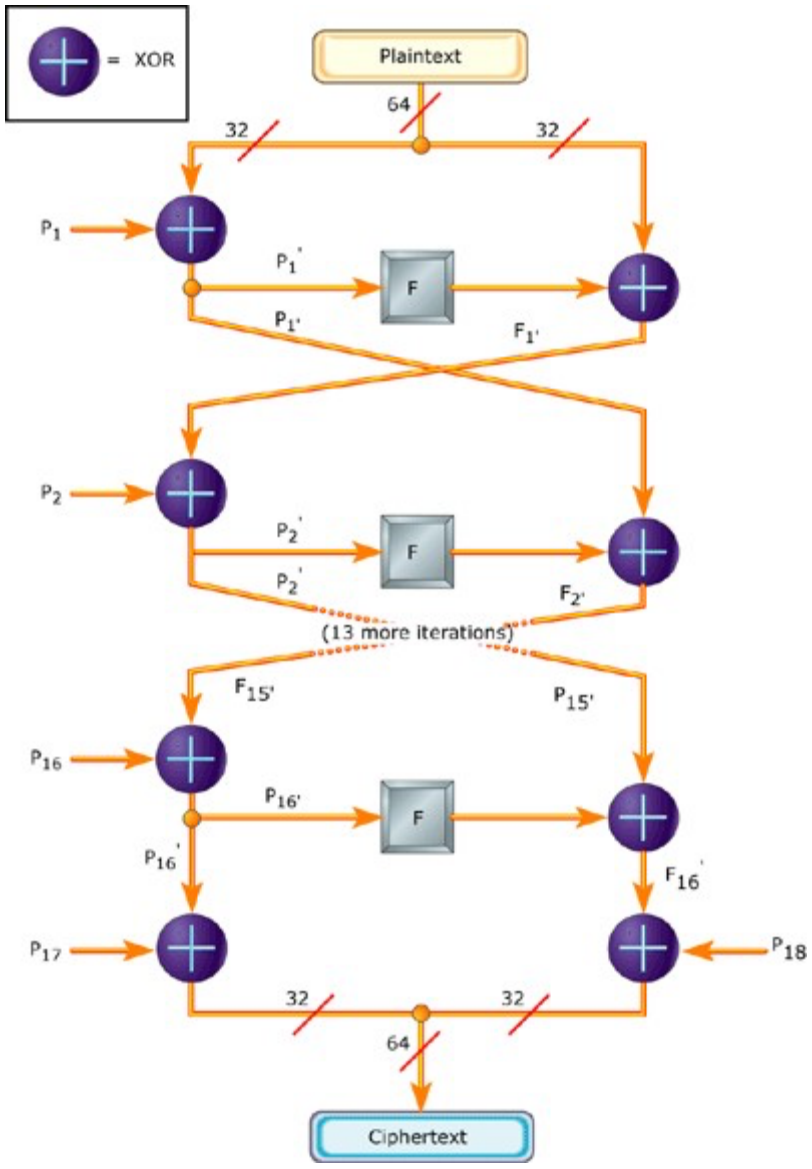
$$S1,0, S1,1,..., S1,255$$

$$S2,0, S2,1,..., S2,255$$

$$S3,0, S3,1,..., S3,255$$

$$S4,0, S4,1,..., S4,255$$

The exact method used to calculate these subkeys will be described later in this section.



Blowfish is a Feistel network (see Section 14.10) consisting of 16 rounds. The input is a 64-bit data element, x. To encrypt:

Divide x into two 32-bit halves:

x_L, x_R **For *i* = 1 to 16**:

$x_L = x_L \oplus P_i$

$x_R = F(x_L) \oplus x_R$

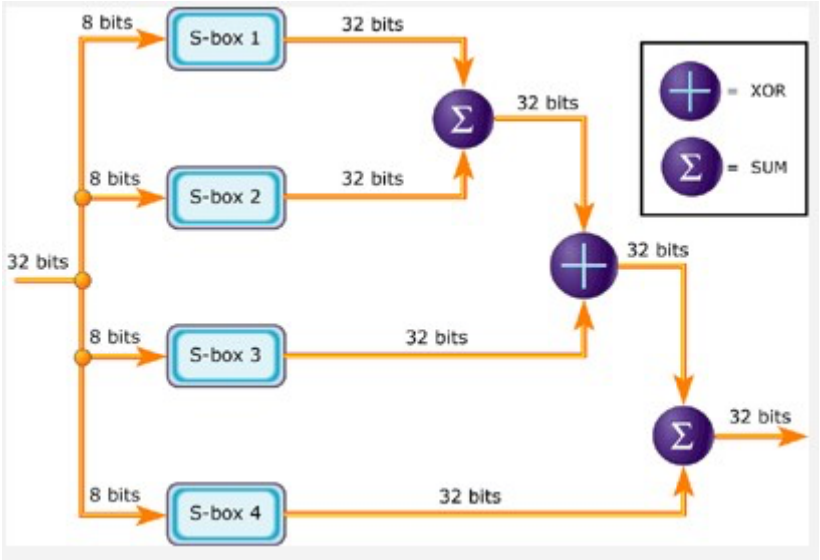
Swap x_L and x_R

Swap x_L and x_R(Undo the last swap.)

$x_R = x_R \oplus P_{17}$

$x_L = x_L \oplus P_{18}$

Recombine x_L and x_R



Function F

Function F is as follows (see Figure 14.3):

Divide x_L into four eight-bit quarters:

a, b, c, and *d* $F(x_L) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \oplus S_{3,c}) + S_{4,d} \bmod 2^{32}$

Decryption is exactly the same as encryption, except that P_1, P_2,..., P_{18} are used in the reverse order. Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all subkeys are stored in cache.

The subkeys are calculated using the Blowfish algorithm. The exact method follows.

(1) Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of p.

(2) XOR P_1 with the first 32 bits of the key, XOR P_2 with the second 32-bits of the key, and so on for all bits of the key (up to P_{18}). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits.

(3) Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).

(4) Replace P_1 and P_2 with the output of step (3).

(5) Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.

(6) Replace P_3 and P_4 with the output of step (5).

(7) Continue the process, replacing all elements of the P-array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys—there's no need to execute this derivation process multiple times. *NOTE:*

P=18 32-bit to byte (div 8)

P=18*4=72

S=256 32-bit to byte (div 8)

S=256*4(byte)*4(number s-box)=4096

The total= 4168

In each round we replace 2 p then 8 round to replace all p

Then

521(number Round)* 8(Number P-)=4168 byte as subkey