BLOWFISH

-Là 1 thuật toán mật mã khối đối xứng

1. BlockSize: 64-bit

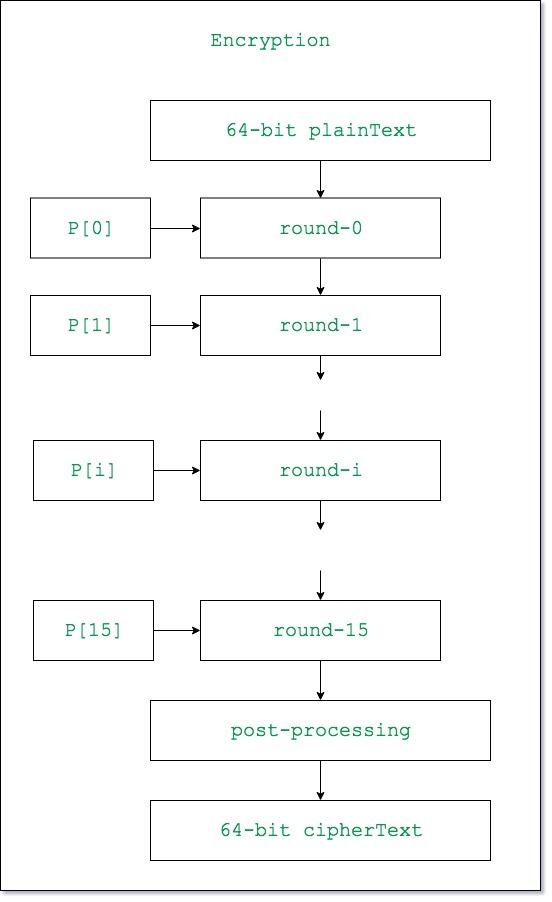
2. keySize: Kích thước biến đổi từ 32 bit đến 448 bit

3. Số lượng khóa con: 18 [P-array]

4. Số vòng: 16

5. Số lượng hộp thay thế: 4 [mỗi hộp có 512 mục, mỗi hộp 32 bit]

-Qúa trình mã hóa



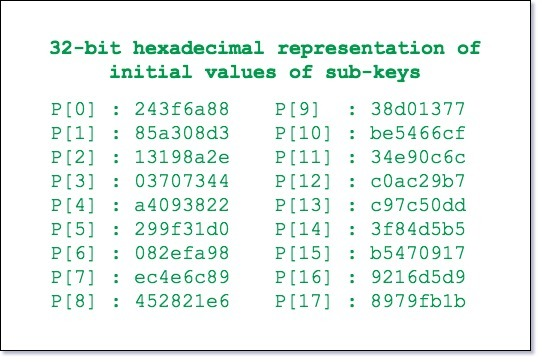
B1: Tạo khóa con

-18 khóa con {P[0] … P[17]}

-đc lưu trữ trong mảng P với mỗi phần tử mảng là 1 mục nhập 32 bit

-Nó đc khởi tạo bằng các chữ số pi(?)

-Biểu diễn của mỗi khóa con:



-Mỗi khóa con đc thay đổi với khóa đầu vào là:

P[0] = P[0] xor 1st 32-bits of input key  
P[1] = P[1] xor 2nd 32-bits of input key  
.  
.  
.  
P[i] = P[i] xor (i+1)th 32-bits of input key   
(roll over to 1st 32-bits depending on the key length)  
.  
.  
.  
P[17] = P[17] xor 18th 32-bits of input key   
(roll over to 1st 32-bits depending on key length)

=>Kết quả là mảng P chứa 18 khóa con đc sử dụng trong toàn bộ quá trình mã hóa

B2: khởi tạo hộp thay thế

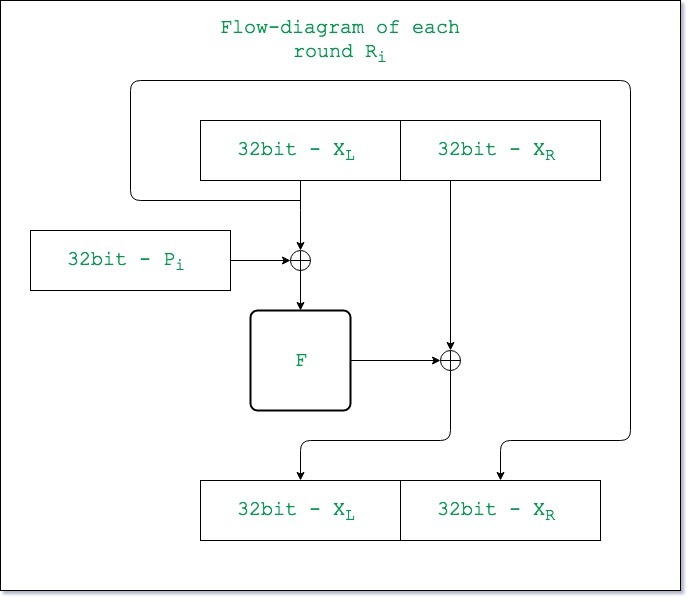
-4 hộp thay thế (S-box) {S[0] … S[4]} có 256 mục trong đó mỗi mục nhập là 32 bit

-Đc khởi tạo bằng các chữ số của pi(?) sau khi khởi tạo mảng P

B3: Mã hóa

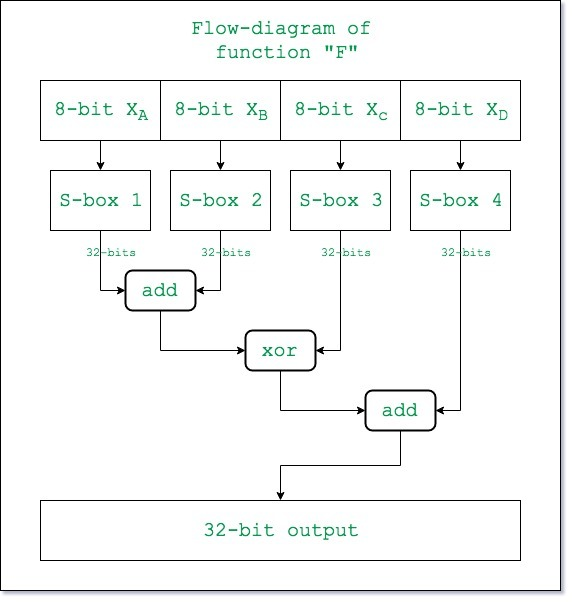
a) Rounds

Mã hóa gồm 16 vòng với mỗi vòng (Ri) lấy đầu vào plaintext (PT) từ vòng trước và khóa con tương ứng (Pi)



(Mô tả: đầu ra đc chia ra làm 2 phần bằng nhau = 32 bit. Phần Bên trái xor với Khóa con Pi, kết quả sẽ được đưa ra làm 32 bit sau của đầu ra và cũng kết quả đó dduc biến đổi bằng hàm F để xor với 32 bit sau của đầu vào. Kết quả xor sẽ đưa làm 32 bit đầu của đầu ra)

-Hàm F

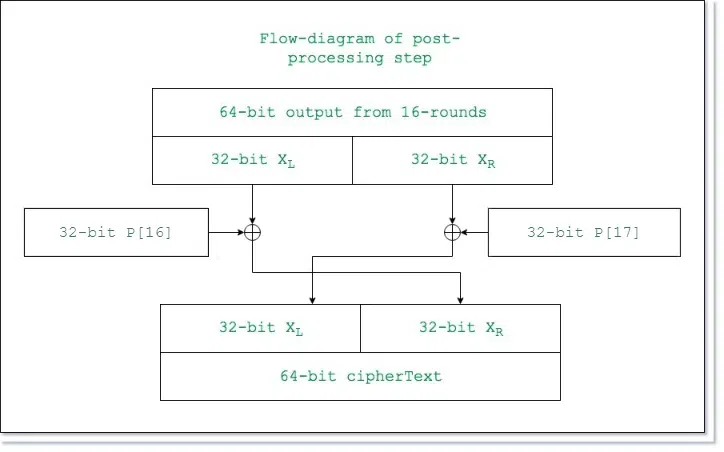


(Mô tả: Đầu vào chia ra làm 4 phần bằng nhau = 8 bit; 2 phần đầu dùng hàm add, kết quả đc xor với phần thứ 3 và cuối cùng là dùng hàm add với phần thứ 4 để đc kết quả đầu ra)

+Hàm add là phép công modulo 2^32: (x+y) % (2^32)

b) Xử lý hậu kỳ (post-processing)

-Đầu ta sau 16 vòng đc xử lý như sau;



(Mô tả: Sau 16 vòng, đầu vào vẫn đc chia làm 2 phần = nhau = 32 bit. 32 bit đầu của đầu vào xor với khóa con P[16], kết quả đc làm nửa 32 bit sau của đầu ra. 32 bit sau của đầu vào xor với khóa con P[17], kết quả đc làm nửa 32 bit đầu của đầu ra)

-Chế độ hoạt động: search chat vẽ sơ quy trình mã hóa blowfish với các chế độ hoạt động

# Kỹ thuật giấu tin LSB

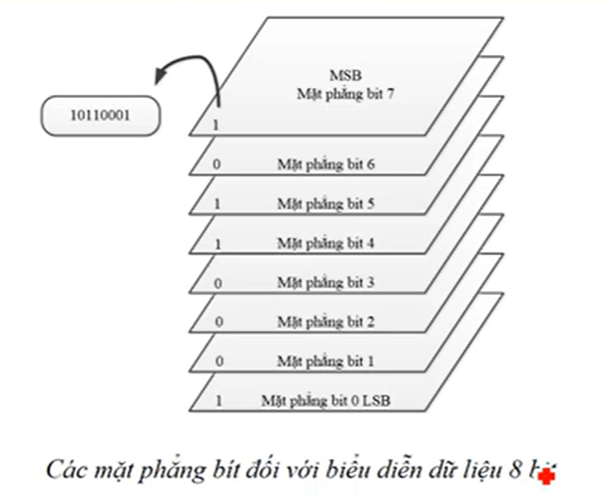
Bit có trọng số thấp nhất (LSB) là bít có ảnh hưởng ít nhất tới việc quyết định màu sắc của mỗi điểm ảnh (Là bit cuối cùng)

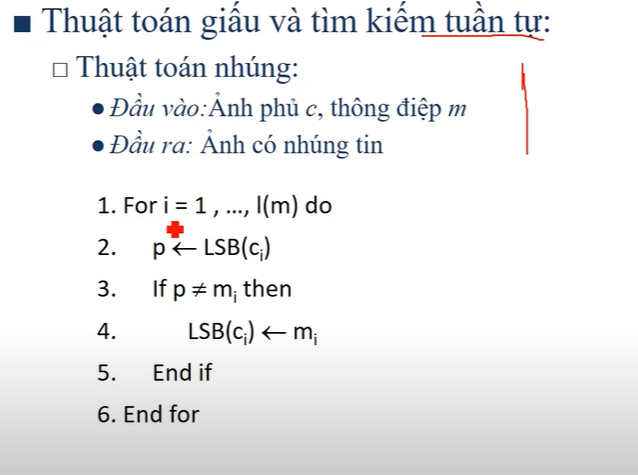
1 LSB: 1 bit cuối

2 LSB: 2 bit cuối

3 LSB: 3 bit cuối (01010**111)**

**=> tăng dung lượng thì chất lượng nhúng giảm đi**





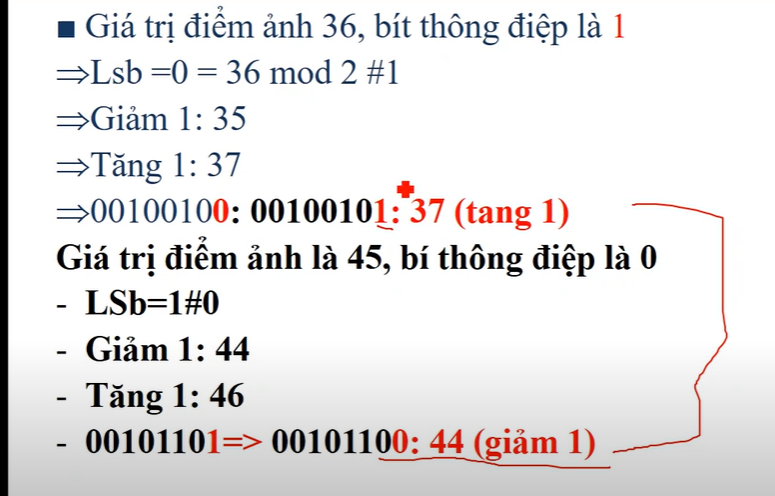
L(m) độ dài của thông điệp < kích thước ảnh phủ

1. Mỗi 1 vị trí bit của thông điệp cần nhúng

2. p = điểm ảnh cần nhúng ( bit có trọng số thấp nhất trong ci)

3. nếu bit có trọng số nhỏ nhất khác bit ở thông điệp

4. thay bit có trọng số nhỏ nhất ở ci = bit của thông điệp



Nếu bằng nhau thì xử lý

47 giảm 1

42 tăng 1

46 giữ nguyên 0

42 giữ 0

46 tăng 1

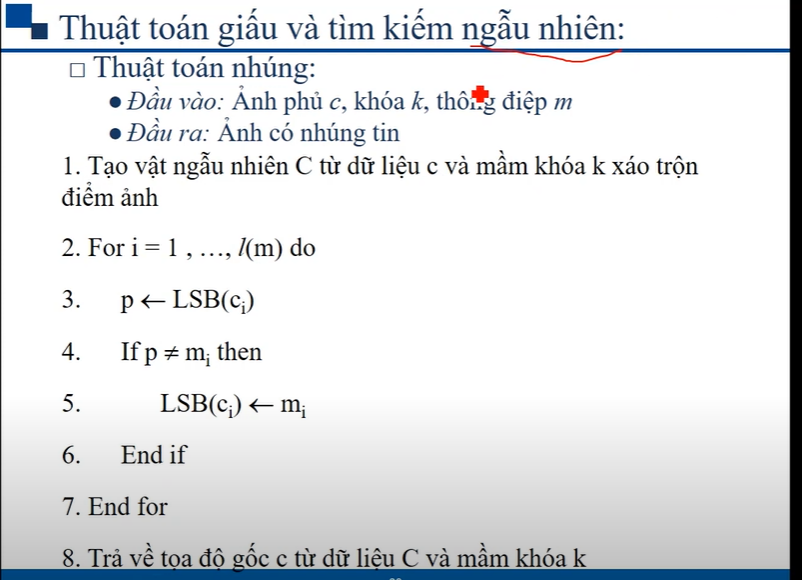
43 giữ 1

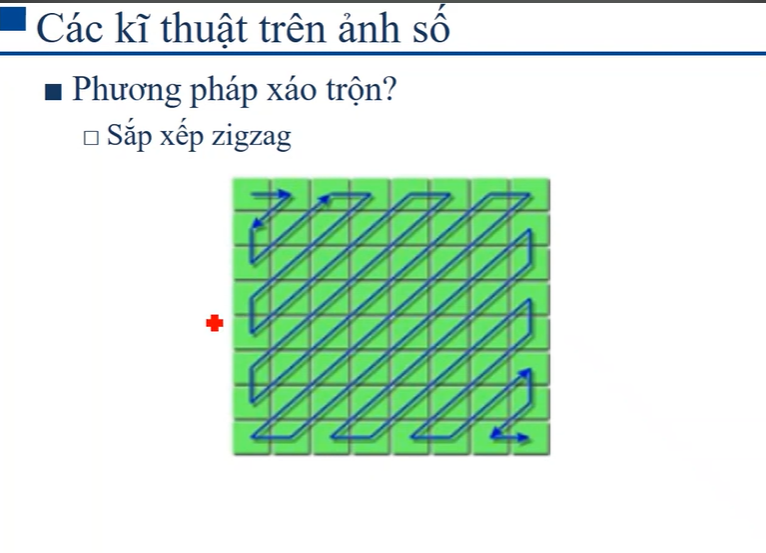
47 giữ 1

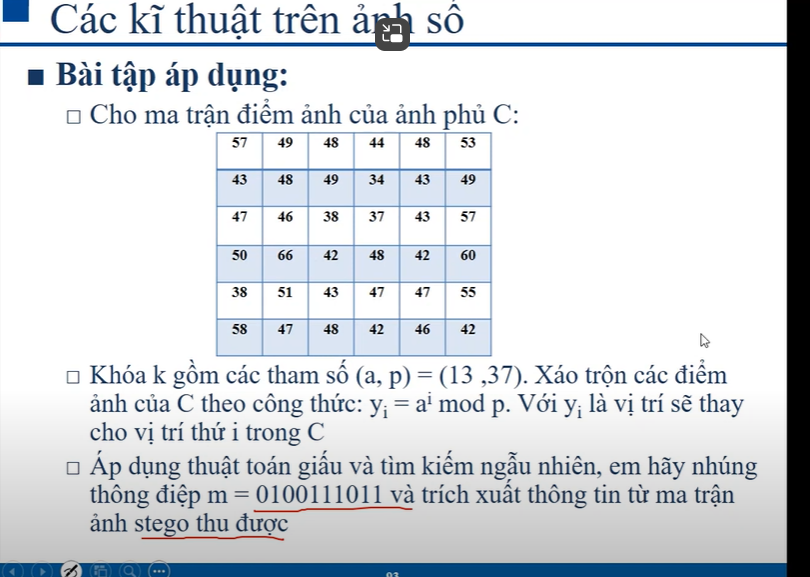
49 giảm 1

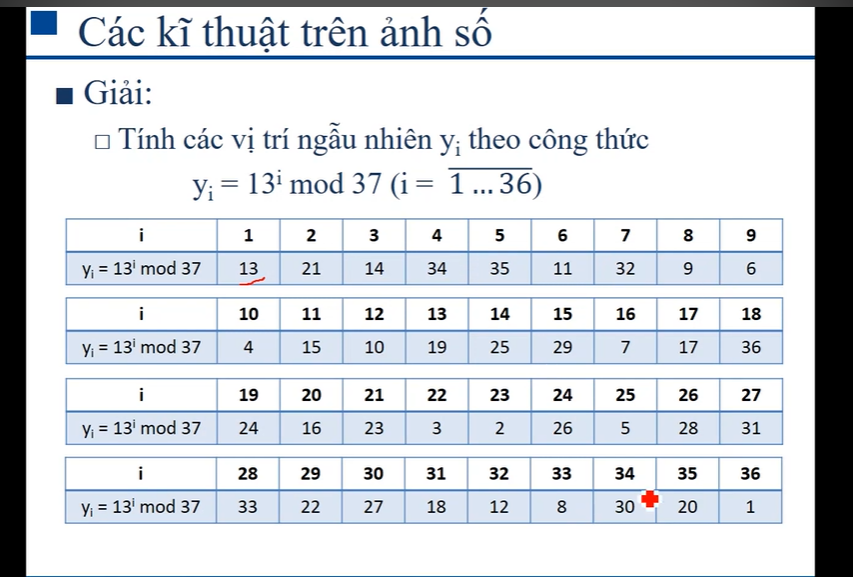
53 giữ 1

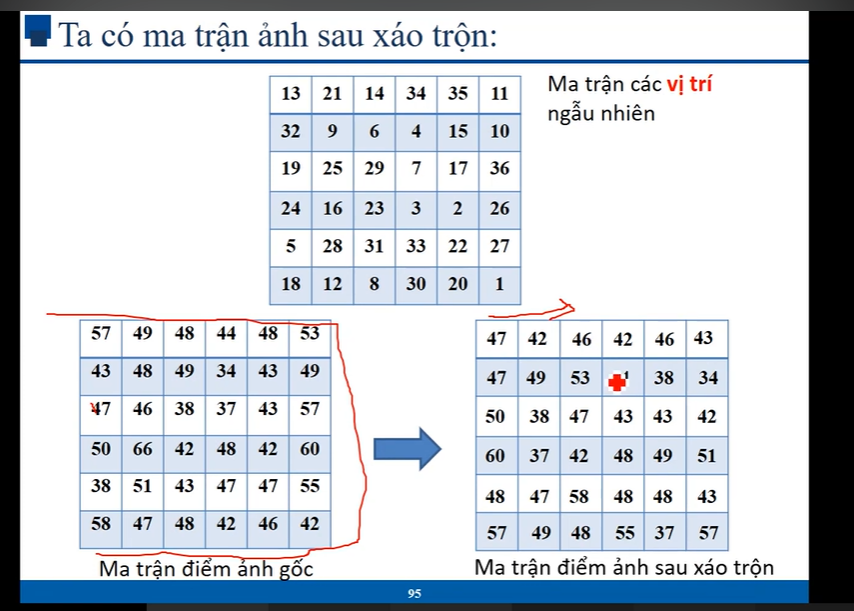
44 tăng 1











47 ở vị trí 13 và y1 =13 => vị trí đầu tiên = 47

Tool openpuff

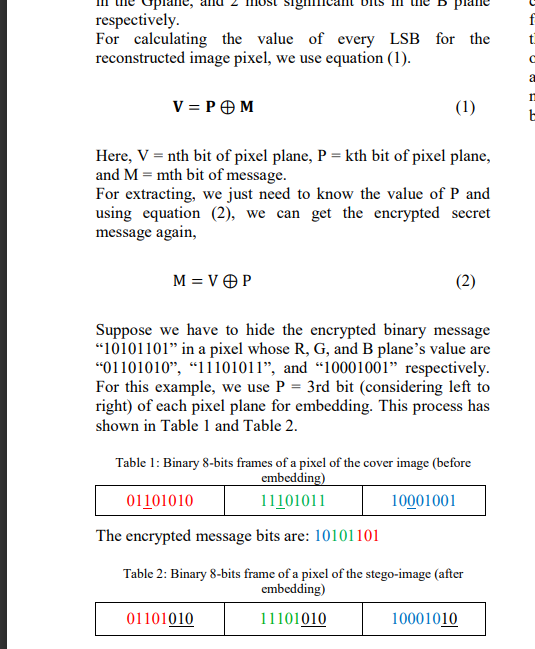
-Mật khẩu dài từ 8-32 ký tự, tỉ lệ khác nhau ít nhất giữa các mật khẩu là 25%, mã hóa file bí mật

Dung lượng nhúng càng nhiều thì tính bền vững, độ trong suốt thấp

Tại sao chọn 2 thuật toán này:

Theo tạp chi quốc tế về khoa học máy tính và an ninh mạng; kết quả thực nghiệm các cuộc tấn công khác nhau vào phương pháp đề xuất

Bằng cách tấn công thống kê sử dụng tấn công Chi-square, không có dữ liệu nhúng nào được phát hiện từ hình ảnh stego. Nhưng trong cuộc tấn công trực quan sử dụng đòn tấn công Enhance LSB mạnh mẽ, kẻ tấn công có thể phát hiện phần dữ liệu bị ẩn nhưng kích thước không chính xác.



Trong quy trình Bit quan trọng nhất được nhúng (E-LSB), tất cả các pixel của hình ảnh RGB được chia thành các khung 8 bit tương ứng cho các mặt phẳng Đỏ (R), Xanh lục (G) và Xanh lam (B). Để áp dụng kỹ thuật này, trước tiên chúng tôi mã hóa tin nhắn bằng phương pháp mã hóa Blowfish. Sau đó, 3, 3 và 2 bit ít quan trọng nhất của khung R, G và B tương ứng của một pixel của hình ảnh sẽ được nhúng cùng với dữ liệu được mã hóa.

V= P xor m

P là vị trí bit ở giá trị nhị phân điểm màu Đỏ

101 xor với số 1 = 010