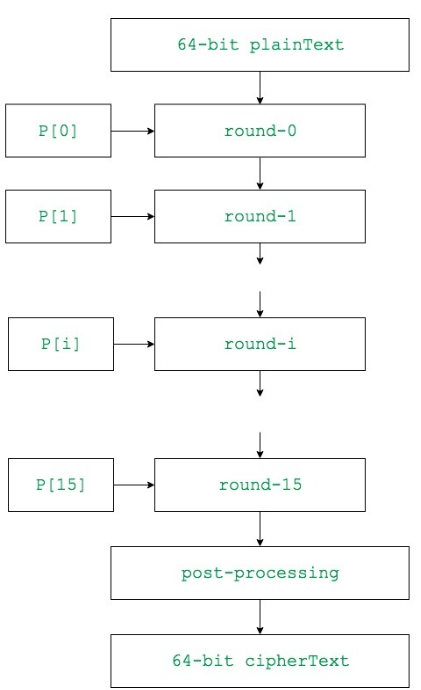
Tóm tắt nội dung thuật toán Blowfish

**Blowfish** là một kỹ thuật mã hóa được thiết kế bởi **Bruce Schneier** vào năm 1993 như là một thay thế cho [Kỹ thuật mã hóa DES](https://translate.googleusercontent.com/translate_c?depth=1&hl=vi&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.geeksforgeeks.org/data-encryption-standard-des-set-1/&usg=ALkJrhhkMDG2p0iFpA6SFjI3F6LrRT4LJw) . Nó nhanh hơn đáng kể so với DES và cung cấp tốc độ mã hóa tốt mà không có [kỹ thuật phân tích mật mã](https://translate.googleusercontent.com/translate_c?depth=1&hl=vi&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=https://www.geeksforgeeks.org/introduction-to-crypto-terminologies/&usg=ALkJrhjdVzgVlZpJAoHaQjdFBgzsl-ZEgg) hiệu quả nào được tìm thấy cho đến nay. Đây là một trong những cyphers khối an toàn đầu tiên không phải chịu bất kỳ bằng sáng chế nào và do đó có sẵn miễn phí cho bất kỳ ai sử dụng.

1. **kích thước khối** : 64 bit
2. **keySize** : Kích thước biến 32-bit đến 448-bit
3. **số lượng khóa con** : 18 [mảng P]
4. **số vòng** : 16
5. **số lượng hộp đăng ký** : 4 [mỗi **hộp** có 512 mục 32 bit]

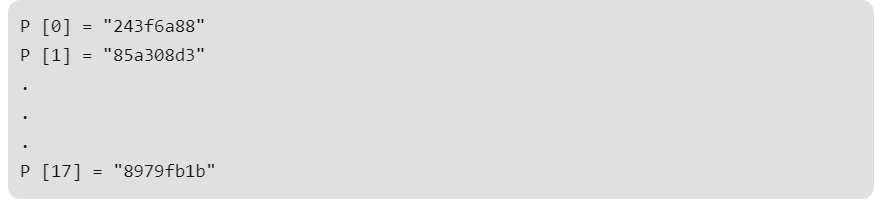
**Thuật toán mã hóa blowfish**

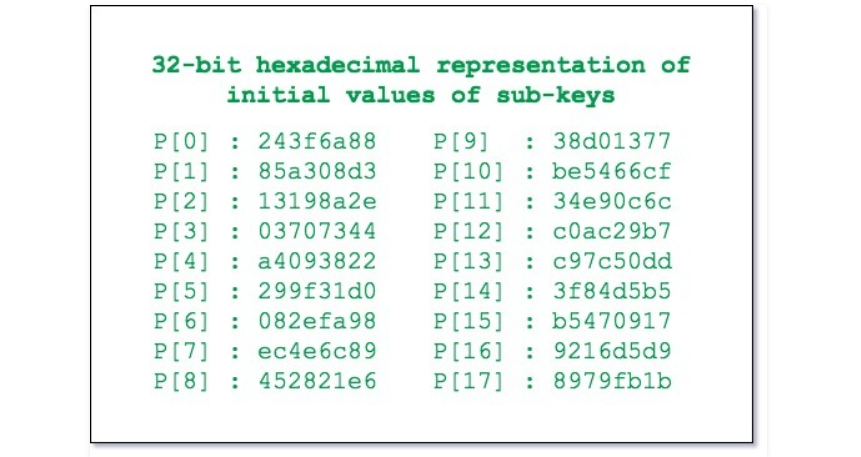
**Toàn bộ quá trình mã hóa có thể được xây dựng như:**



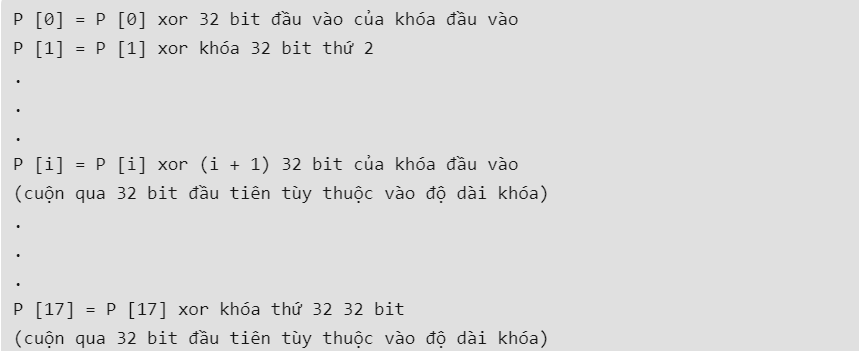
**Bước 1: Tạo các khóa con:**

* 18 khóa con {P [0] Cảng P [17]} là cần thiết trong cả quá trình mã hóa cũng như quá trình giải mã và cùng một khóa con được sử dụng cho cả hai quá trình.
* 18 khóa con này được lưu trữ trong một mảng P với mỗi phần tử mảng là một mục nhập 32 bit.
* Nó được khởi tạo với các chữ số của pi (?).
* Biểu diễn thập lục phân của mỗi khóa con được đưa ra bởi:





* Bây giờ mỗi khóa con được thay đổi liên quan đến khóa đầu vào là:



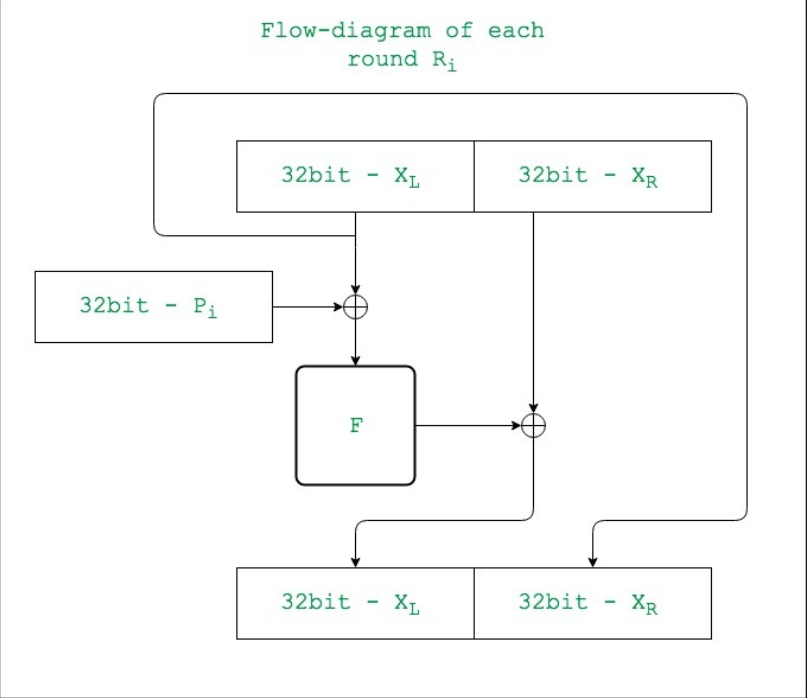
**Mảng P kết quả chứa 18 khóa con được sử dụng trong toàn bộ quá trình mã hóa**

**Bước 2: khởi tạo các hộp thay thế:**

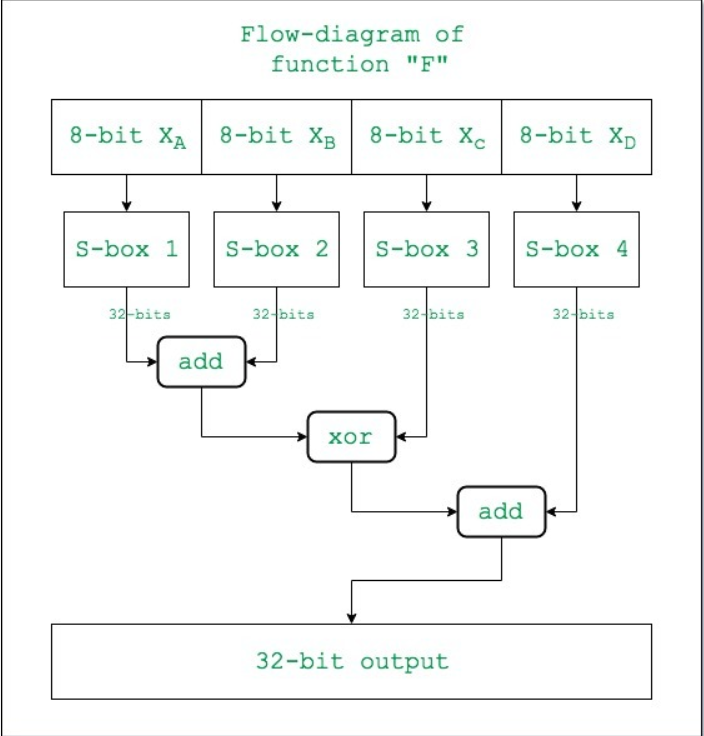
* Cần có 4 hộp thay thế (S-box) {S [0] S 255], 0 & lei & le4} trong đó mỗi mục nhập là 32 bit.
* Nó được khởi tạo với các chữ số của pi (?) Sau khi khởi tạo mảng P. [Bạn có thể tìm thấy các **hộp s** ở đây!](https://translate.googleusercontent.com/translate_c?depth=1&hl=vi&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=https://github.com/Ray784/Blowfish-S-boxes&usg=ALkJrhjLFD2AWEdqR8NfdJ2il1zpz1wTYg)

**Bước 3: Mã hóa:**

* Chức năng mã hóa bao gồm hai phần:
  1. **Vòng:** Mã hóa bao gồm 16 vòng với mỗi vòng (Ri) lấy đầu vào plainText (PT) từ vòng trước và khóa con tương ứng (Pi). Mô tả của mỗi vòng như sau:

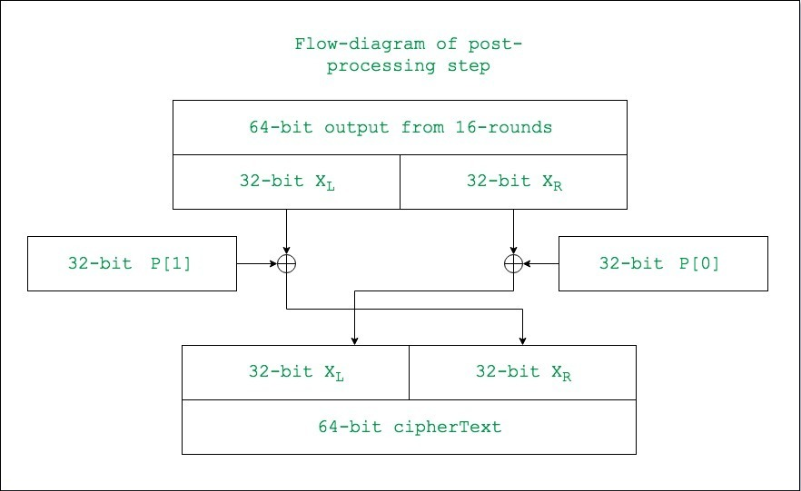


Mô tả về chức năng F F F như sau:



Ở đây, hàm thêm add cộng là modulo 2 ^ 32.

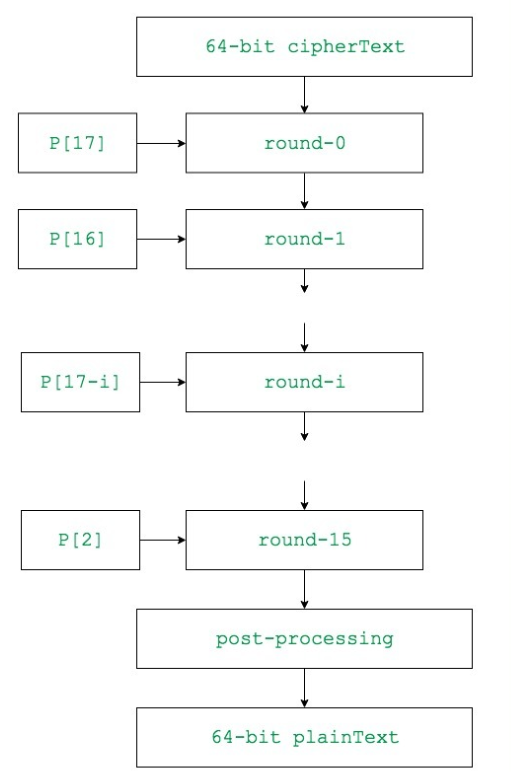
* 1. **Xử lý hậu kỳ:** Đầu ra sau 16 vòng được xử lý như sau:



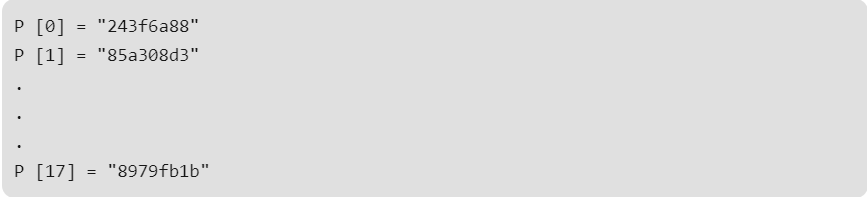
**Giải mã**

Quá trình giải mã tương tự như quá trình mã hóa và các khóa con được sử dụng ngược {P [17] - P [0]}. Toàn bộ quá trình giải mã có thể được xây dựng như sau: Hãy xem từng bước một:

**Bước 1: Tạo các khóa con:**

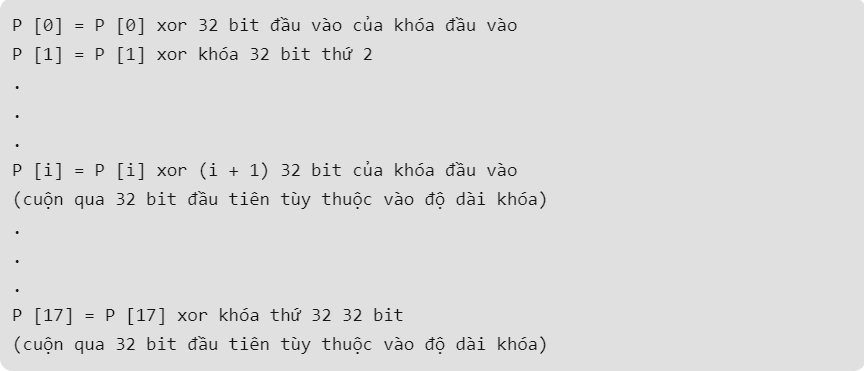


* 18 khóa con {P [0] Cảng P [17]} là cần thiết trong quá trình giải mã.
* 18 khóa con này được lưu trữ trong một mảng P với mỗi phần tử mảng là một mục nhập 32 bit.
* Nó được khởi tạo với các chữ số của pi (?).
* Biểu diễn thập lục phân của mỗi khóa con được đưa ra bởi:



**Lưu ý:** Xem mã hóa cho các giá trị ban đầu của mảng P.

* Bây giờ mỗi khóa con được thay đổi liên quan đến khóa đầu vào là:



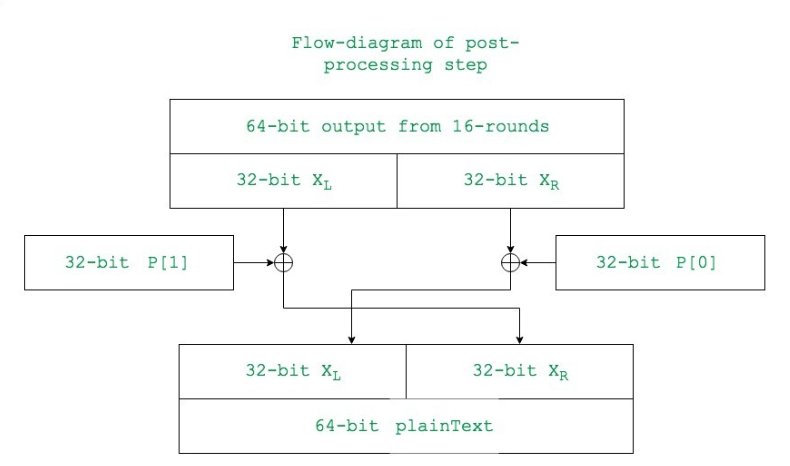
**Mảng P kết quả chứa 18 khóa con được sử dụng trong toàn bộ quá trình mã hóa**

**Bước 2: khởi tạo các hộp thay thế:**

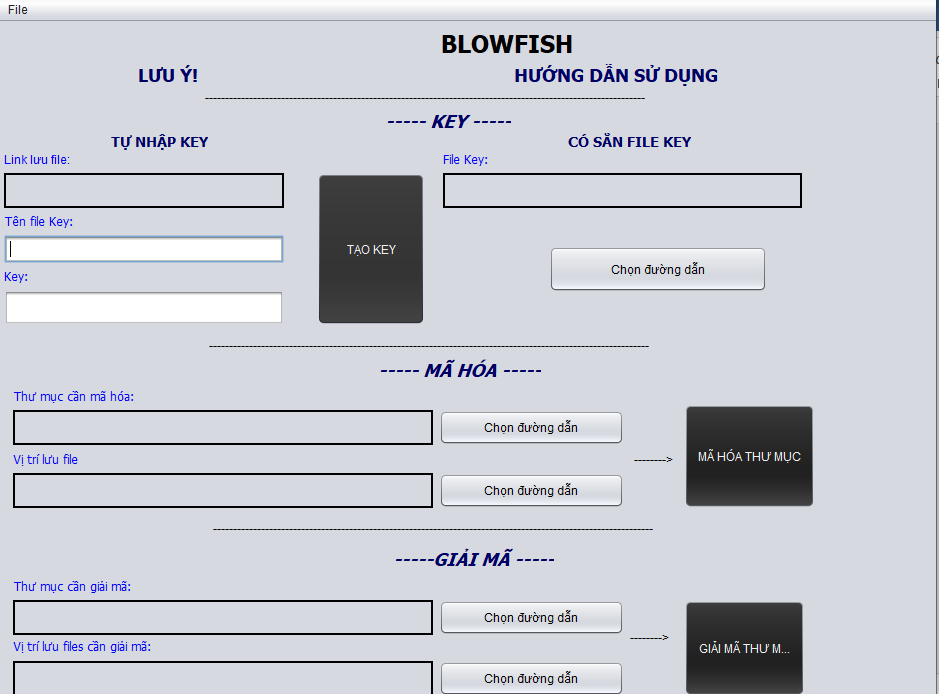
* Cần có 4 hộp thay thế (S-box) {S [0] S 255], 0 & lei & le4} trong đó mỗi mục nhập là 32 bit.
* Nó được khởi tạo với các chữ số của pi (?) Sau khi khởi tạo mảng P. [Bạn có thể tìm thấy các **hộp s** ở đây!](https://translate.googleusercontent.com/translate_c?depth=1&hl=vi&prev=search&pto=aue&rurl=translate.google.com&sl=en&sp=nmt4&u=https://github.com/Ray784/Blowfish-S-boxes&usg=ALkJrhjLFD2AWEdqR8NfdJ2il1zpz1wTYg)

**Bước 3: Giải mã:**

* Chức năng giải mã cũng bao gồm hai phần:
  1. **Vòng:** Giải mã cũng bao gồm 16 vòng với mỗi vòng (Ri) (như đã giải thích ở trên) lấy đầu vào mã hóa (CT) từ vòng trước và khóa con tương ứng (P [17-i]) (nghĩa là để giải mã các khóa con được sử dụng trong đảo ngược).
  2. **Xử lý hậu kỳ:** Đầu ra sau 16 vòng được xử lý như sau:



Demo mã hóa thuật toán Blowfish

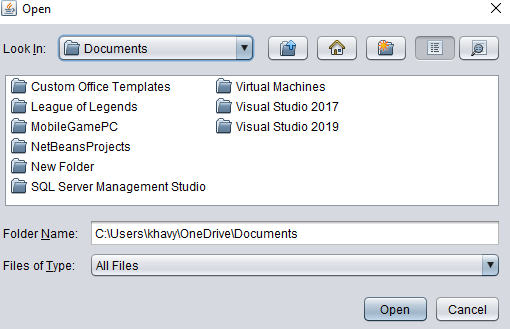


**Bước 1:** chúng ta sẽ tạo key bằng cách nhập tên key và key ví dụ : key.txt , key = “ksfnahgia;hoethafdgaj” rồi chúng ta sẽ bấm nút **tạo key** và chọn thư mục chúng ta lưu key đó.

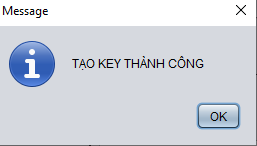


Sau đó nhân TẠO KEY

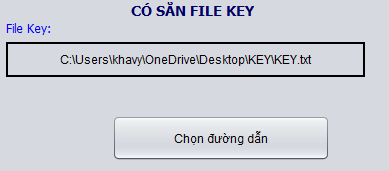
Xuất hiện hộp thoại chọn thư mục để lưu Key



Sau đó hiện thông báo tạo Key thành công

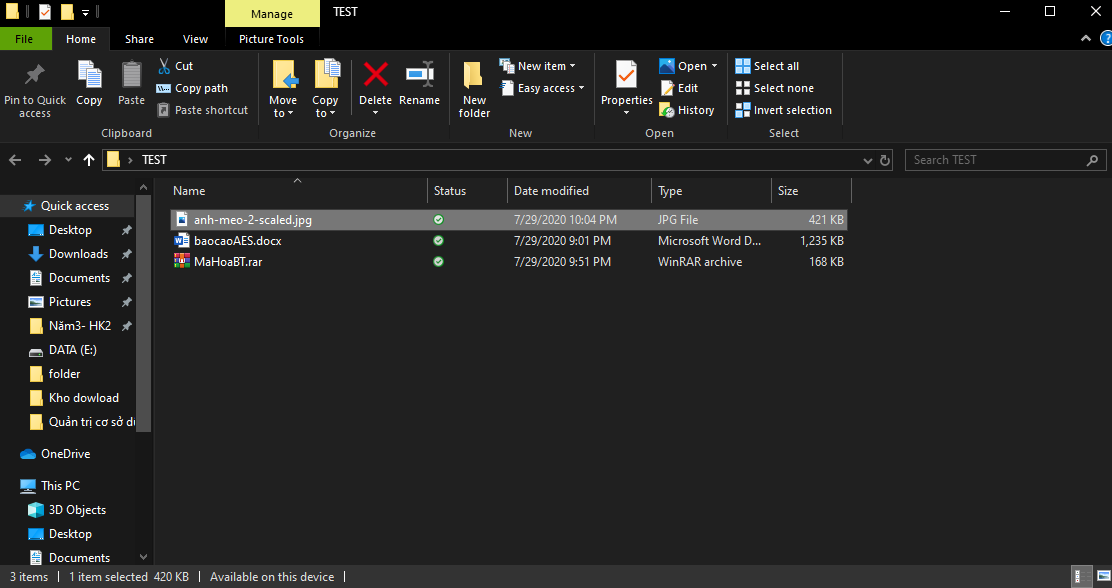


Còn nếu bạn đã tao file chưa Key trước đó thì hãy nhập Link file Key

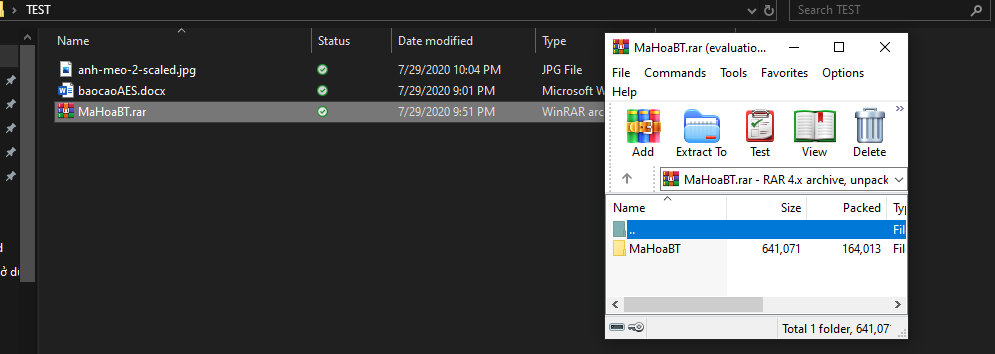


**Bước 2:** Chọn thư mục chúng ta cần mã hóa bằng cách nhấn nút **Thư mục mã hóa** và chọn nơi ta lưu **thư mục đã mã hóa.** Sau đó nhận nút **Mã hóa thư mục** để mã hóa thư mục ta chọn

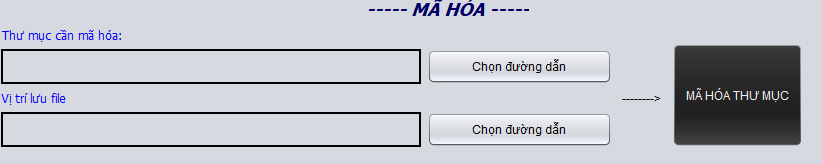
* + Tạo thư mục chứa các file chưa được mã hóa bất kì

****

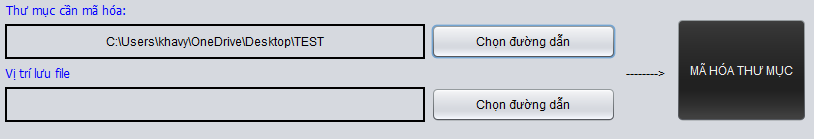
Vẫn mở được cách file vì chưa mã hóa

****

**Giao diện mã hóa**

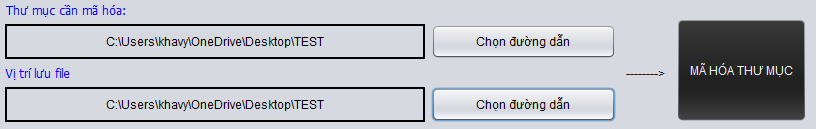
****

**Chọn đường dẫn thư mục cần mã hóa**

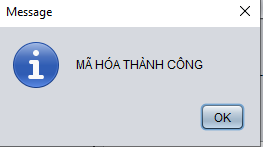
****

Sau đó chọn vị trí lưu file , ta có thể lưu bất kì thư mục nào cũng được , **lưu file phải vào trong 1 thư mục riêng chứa toàn bộ file mã hóa đó, không được lưu riêng lẻ từng file ngoài DESKTOP** ( TOÀN BỘ FILE PHẢI ĐƯỢC LƯU 1 THƯ MỤC NHẤT ĐỊNH ĐỂ THUẬN TIỆN CHO VIỆC GIẢI MÃ)

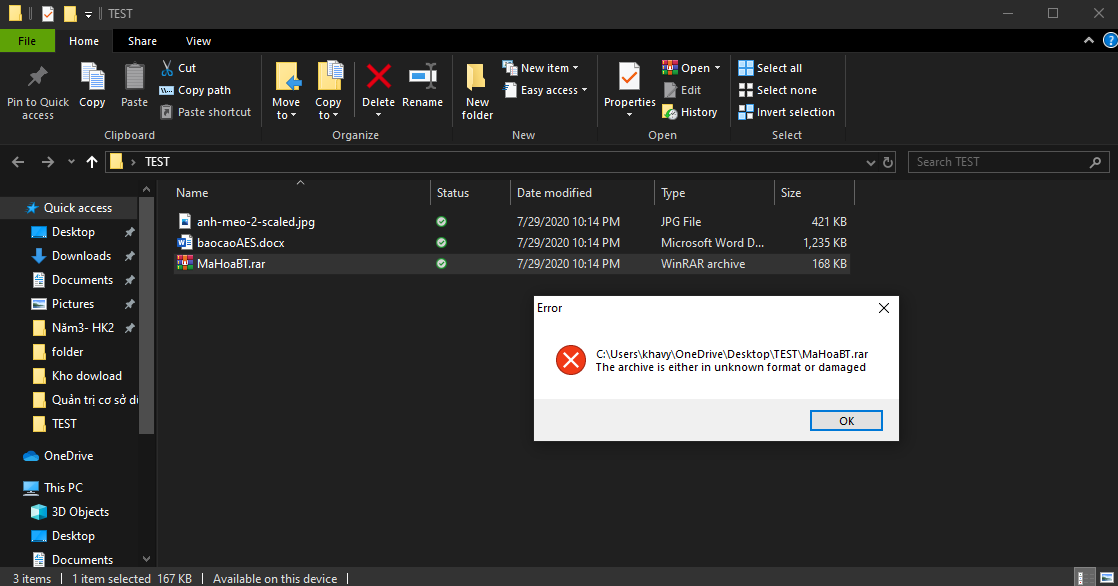
Ta có thể mã hóa đè lên thư mục hiện hành



Hiển thị thông báo



Khi mã hóa xong , ta không mở được File



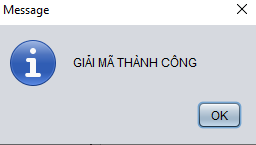
**Bước 3:** Chọn thư mục chúng ta cần giải mã bằng cách nhấn nút **Thư mục giải mã** và chọn nơi ta lưu **thư mục đã giải mã.** Sau đó nhận nút **giải mã thư mục** để mã hóa thư mục ta chọn. ( lưu ý là lúc ta mã hóa nó sẽ trả về chúng ta là file nên bạn cần bỏ nó vào thư mục thì mới giải má đc)



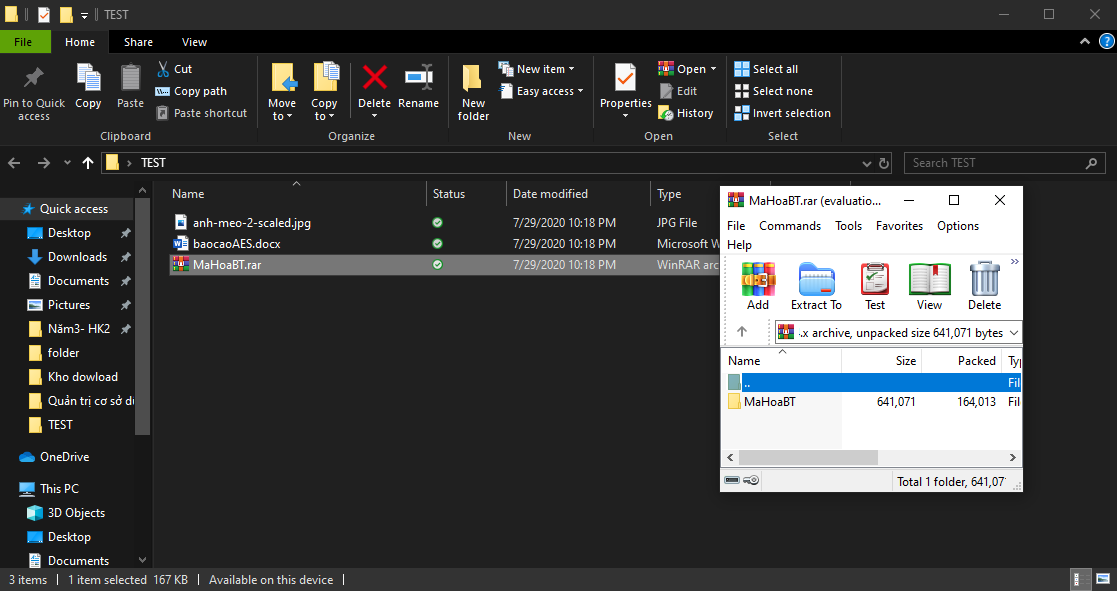
Làm tương tự như mã hóa để giải mã



Và hiện thông báo

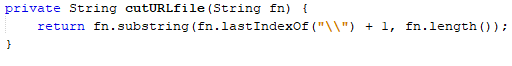


Bây giờ bạn có thể đọc tất cả file trong thư mục đó được rồi.

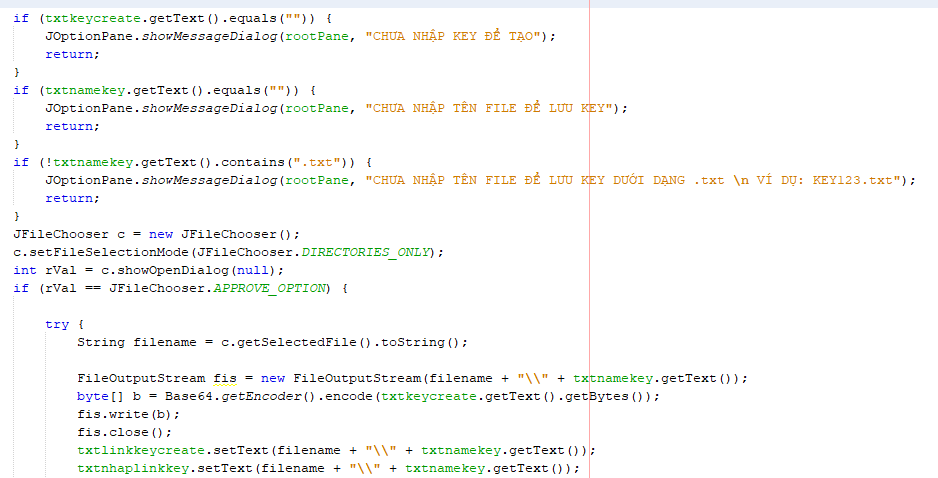


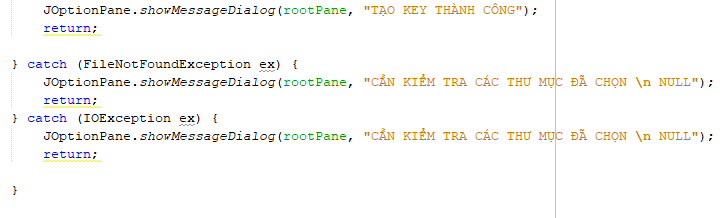
CODE

Hàm cắt chuỗi lấy tên File, chuối tạo vào có dạng String File.getabsPath(0;

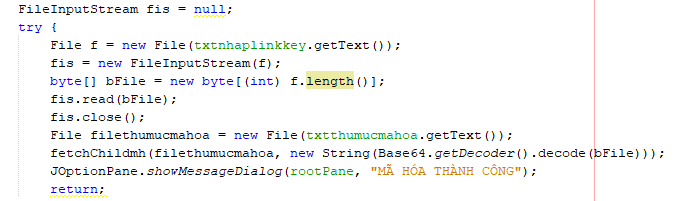


Các ràng buộc trong quá trình tạo KEY và thuật toán tạo Key

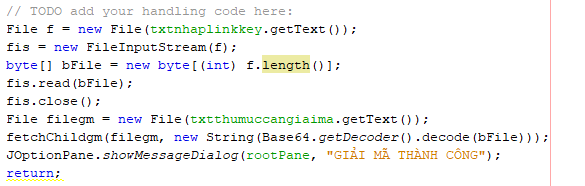




Khi tiến hành mã hóa, thì ta đọc Key từ FILE

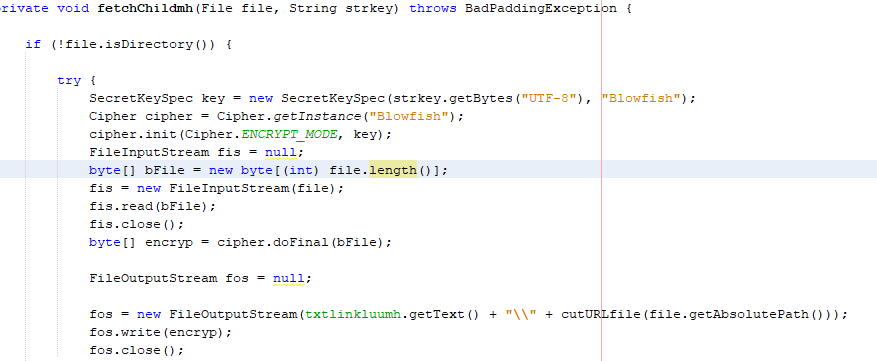


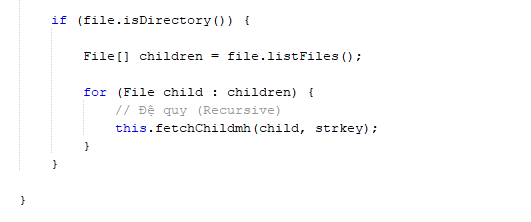
Giaỉ mã tương tự



Ta gọi đệ quy để duyệt hết tất cả file trong thu mục

Nếu là file thì tiến hành mã hóa





Giaỉ mã cũng tương tự vậy, duyệt từng File để giải mã

