# Xây dựng ứng dụng mã hóa sử dụng thư viện OpenSSL

## Giới thiệu

### Giới thiệu

OpenSSL là mã nguồn mở triển khai các hệ mật mã, bao gồm chương trình dòng lệnh trực tiếp sử dụng hàm mật mã và thư viện, cái mà các lập trình viên sẽ sử dụng để phát triển các ứng dụng sử dụng mật mã học.

OpenSSL là một bộ công cụ, thư viện mạnh mẽ, cấp thương mại và đầy đủ tính năng cho các giao thức bảo mật TLS và SSL. Tài liệu về API OpenSSL không có nhiều hướng dẫn về cách sử dụng OpenSSL, vì vậy việc triển khai nó trong các ứng dụng có thể hơi rắc rối đối với người mới bắt đầu. Vậy làm cách nào để có thể triển khai một kết nối an toàn cơ bản bằng OpenSSL hoặc làm quen với mật mã học? Hướng dẫn này sẽ giúp giải quyết vấn đề đó.

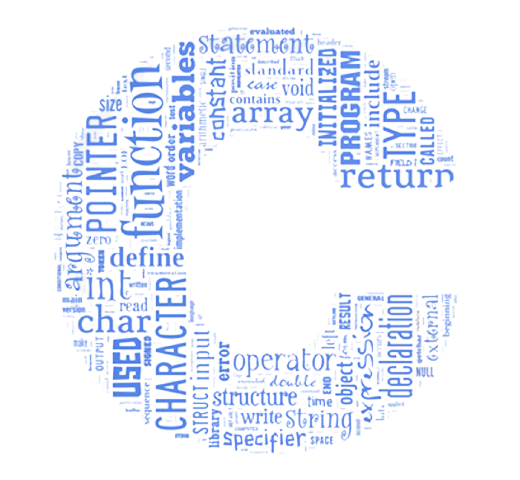
Một phần của vấn đề khi học cách triển khai OpenSSL là thực tế là tài liệu không đầy đủ. Nếu một tài liệu API không đầy đủ thường làm nên khoảng cách giữa nhà phát triển và API. Tuy nhiên đối với OpenSSL vẫn tồn tại và phát triển mạnh mẽ. Tại sao?

OpenSSL là thư viện mở nổi tiếng nhất về kết nối an toàn. Kết quả mà của việc tìm kiếm trên Google cho thư viện SSL sẽ trả về OpenSSL đứng đầu danh sách. OpenSSL được phát hành vào năm 1998, bắt nguồn từ thư viện SSLeay do Eric Young và Tim Hudson phát triển. Các bộ công cụ SSL khác bao gồm GNU TLS, được phân phối theo Giấy phép Công cộng GNU và Dịch vụ An ninh Mạng Mozilla (NSS).

### Nền tảng

Cầu nối tiếp cận OpenSSL API gồm có các nền tảng chính là ngôn ngữ lập trình C và mật mã học.

#### Ngôn ngữ lập trình C



Hình. Logo ngôn ngữ lập trình C.

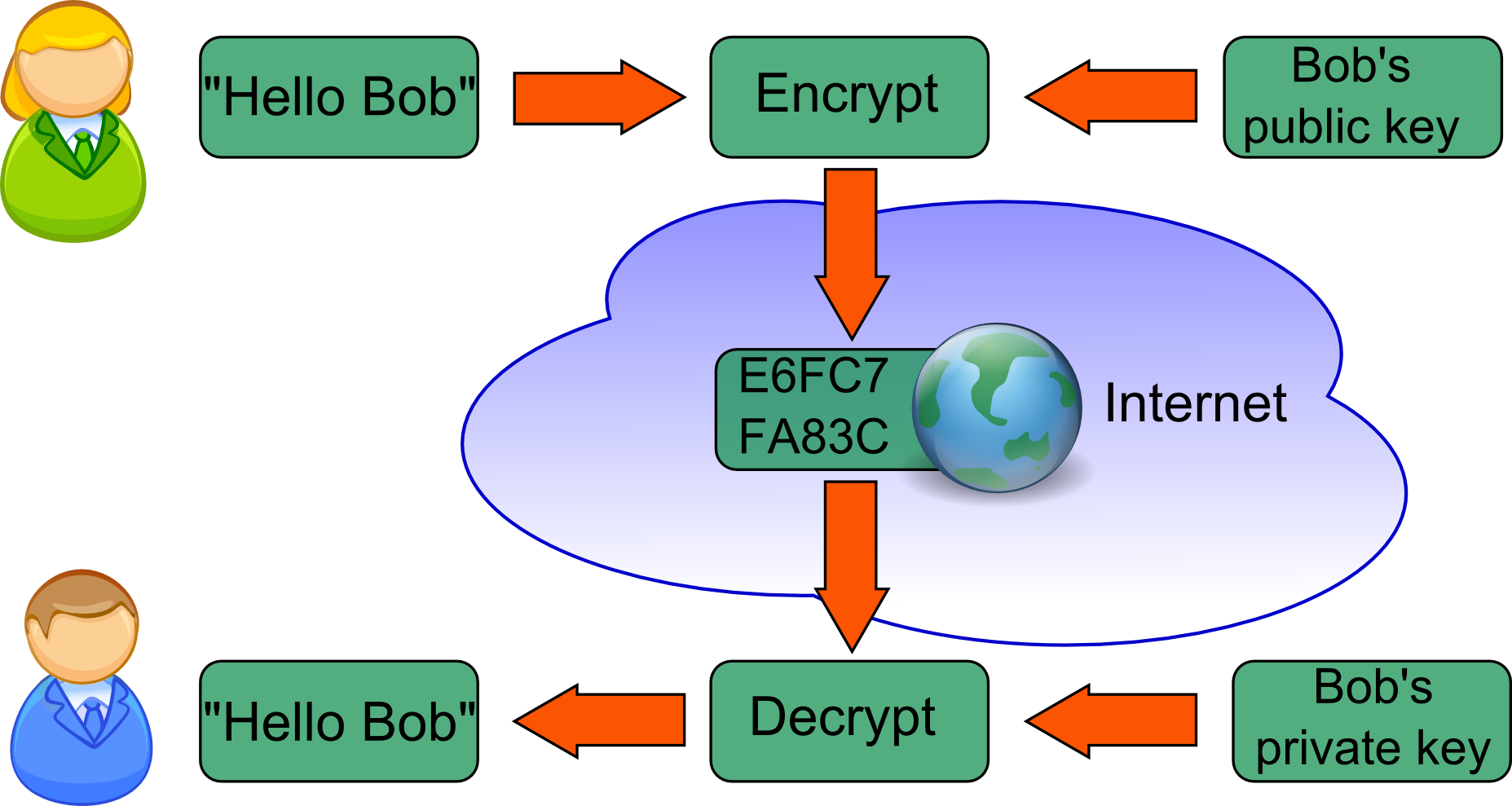
#### Mật mã học

Để tra cứu tài liệu và sử dụng API không bắt buộc có kiến thức với giao thức SSL, vì phần nghiên cứu ngắn gọn về nó sẽ được đưa ra sau.

## Tổng quan

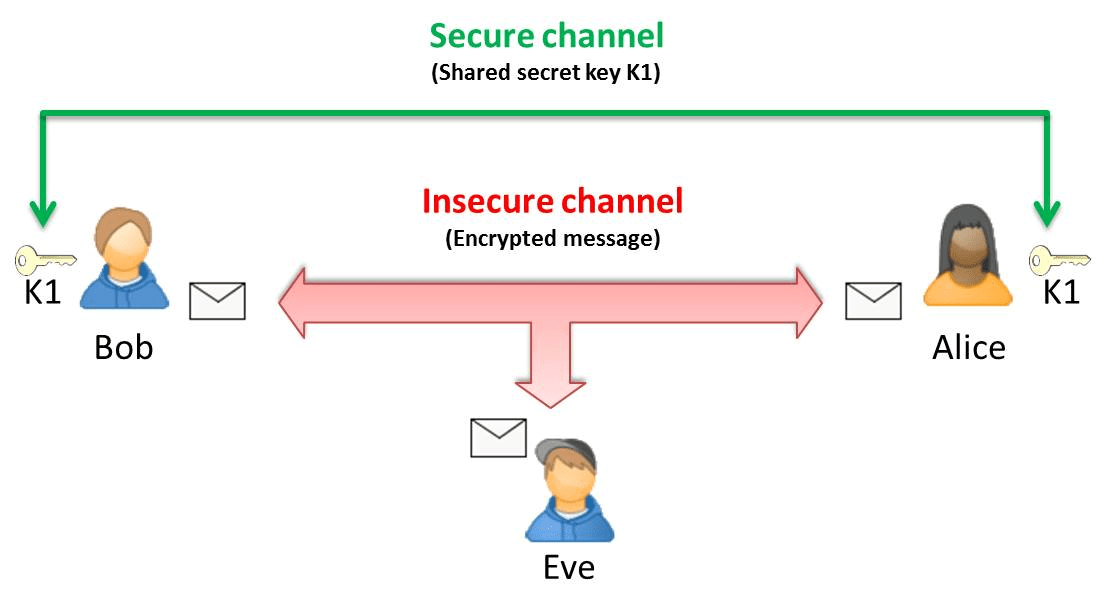
### Khái niệm SSL

SSL là từ viết tắt của Secure Sockets Layer, là tiêu chuẩn đằng sau giao tiếp an toàn trên Internet đã tích hợp mật mã học vào giao thức. Dữ liệu được mã hóa trước khi rời khỏi máy tính và chỉ được giải mã khi nó đến đích. Chứng thực và thuật toán mã hóa hiện diện ở tất cả các hoạt động, trong thực tế cả hai tương hỗ cho nhau và được sử dụng chung thay vì tách riêng.



Hình. Mã hóa dữ liệu trong môi trường Internet.

Về mặt lý thuyết, nếu dữ liệu mã hóa bị chặn hoặc bị nghe trộm trước khi đến được người nhận thì kẻ tấn công gặp khó khăn trong việc giải mã dữ liệu vì không có bất kỳ hệ mã nào tránh khỏi việc bị bẻ khóa bởi phương pháp bạo lực. Nhưng hiệu năng của các hệ thống phần cứng nói riêng và máy tính nói chung ngày càng có tốc độ nhanh hơn, với những tiến bộ mới trong nghiên cứu toán học và phân tích mật mã được tạo ra thì xác suất bẻ khóa các hệ mật mã được sử dụng trong giao thức an toàn SSL đang bắt đầu tăng lên.



Hình. Kẻ tấn công bẻ khóa lấy dữ liệu.

SSL và các các giao thức kết nối an toàn có thể được sử dụng cho bất kỳ loại giao thức nào trên trên không gian mạng, cho dù đó là HTTP, POP3 hay FTP. Giao thức SSL cũng có thể được sử dụng để bảo mật các phiên telnet. Mặc dù mọi kết nối đều có thể được bảo mật bằng SSL, nhưng không nhất thiết phải sử dụng SSL trên mọi loại kết nối, SSL nên được sử dụng nếu kết nối mang thông tin nhạy cảm.

### Khái niệm OpenSSL

OpenSSL là thư viện mã nguồn mở, không chỉ là có SSL nó còn có khả năng chạy thuật toán phức tạp để đào dữ liệu, mã hóa và giải mã các tệp, chứng chỉ số, chữ ký số. OpenSSL API tập hợp rất nhiều các cách giải quyết vấn đề, khả năng mà nó mang lại nhiều hơn những gì mà thư viện chuẩn ngôn ngữ lập C hiện có.

OpenSSL không chỉ là API hỗ trợ trong lập trình mà còn là một công cụ dòng lệnh. Công cụ dòng lệnh có thể làm những điều tương tự như API nhưng đã được tích hợp hỗ trợ cho việc kết nối máy chủ và máy khách. Nhờ vào API nó sẽ cung cấp việc nghiên cứu phát triển ý tưởng dựa trên nên tảng mà nó mang lại.

### Công cụ dòng lệnh

Chương trình OpenSSL là một công cụ dòng lệnh để sử dụng các mật mã khác nhau của thư viện của thư viện mật mã trong OpenSSL. Một số chức năng mà chương trình có thể làm được:

1. Tạo khóa RSA, DH và DSA.
2. Tạo chứng chỉ X.509, CSRs và CRLs.
3. Tính toán bản tóm tắt thông điệp – Message Digests.
4. Mã hóa và giải mã.
5. Xây dựng nhanh mô hình Server – Client thông qua giao thức SSL/TLS.
6. Xử lý S/MIME đã ký và mail được mã hóa.

### Các thuật toán hỗ trợ

#### Mã hóa đối xứng



Hình. Mã hóa đối xứng.

##### Chế độ mã khối

###### ECB

###### CBC

###### CFB

###### OFB

###### CTR

###### EDE

##### Mã khối

###### DES (Data Encryption Standard)

Mã DES là mã thuộc hệ mã Feistel gồm 16 vòng, ngoài ra DES còn thêm một hoán vị khởi tạo trước khi vào vòng 1 và một hoán vị khởi tạo sau vòng 16. Nó thuộc vào loại mã khối có kích thước khối là 64 bit được biểu diễn theo mã ASCII, mã DES sẽ chỉ làm việc với từng khối 64 bit cho đến khi hoàn tất việc mã hóa hoặc giải mã.

###### 3DES (Triple DES)

Là mã được sinh ra để khắc phục nhược điểm độ an toàn của khóa của mã DES bằng cách tạo ra 3 khóa thay vì tăng chiều dài khóa lên gấp 3 lần, sử dụng ba khóa đó cho ba lần mã hoặc giải mã. Chính vì chiều dài được tăng lên thành 168 bit nên sẽ gây phức tạp hơn nhiều cho việc phá mã.

###### AES (Advanced Encryption Standard)

Mã AES là mã tiêu chuẩn mới của Cục Tiêu Chuẩn Quốc Gia Hoa Kỳ thay thế cho mã DES và còn là kết quả của sự kế thừa bởi khắc phục nhược điểm khóa có yếu tố bảo mật kém bởi độ dài. Giống như DES, mã AES là một mã khối gồm nhiều vòng, mã AES với khóa có kích thước 256 bit được cho là an toàn trong tốc độ phát triển nghành kỹ thuật máy tính ngày nay. Độ an toàn của AES được đặt kỳ vọng nhiều hơn DES và TripleDES và được dự đoán sẽ thay thế vai trò của DES và TripleDES trong tương lai.

Một số đặc điểm chính của AES:

1. Dễ dàng có thể hiểu được các thuật ngữ AES 128/192/256 chính là đầu vào sẽ được chia thành các khối có độ dài 128/192/256 bit.
2. Ngoài ra nó còn cho phép độc lập về mặt kích thước giữa đầu vào và khóa.
3. Số lượng vòng có thể thay đổi từ 10 đến 14 vòng tùy thuộc vào kích thước của khóa.

###### Blowfish

Là thuật toán có độ an toàn cao và dễ thực hiện hơn DES, giống với DES thuật toán này có 16 vòng được xây dựng trên cơ sở sơ đồ Feistel. Mã Blowfish mã khối dữ liệu 64 bit với chiều dài khóa từ 32 đến 128 bit.

###### Camellia

###### CAST

###### Idea

###### RC2

###### RC5

##### Mã dòng

###### RC4

##### Tóm tắt thông điệp

###### MD2

###### MD4

###### MD5

###### SHA-I

###### SHA-224

###### SHA-256

###### SHA-384

###### SHA-512

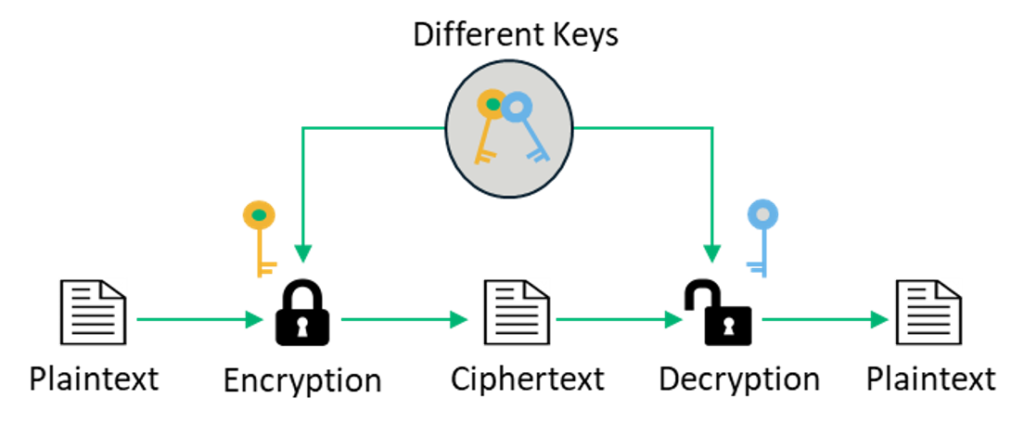
###### Ripemd-160

##### MAC

###### HMAC

###### MDC2

#### Mã hóa bất đối xứng



Hình. Mã hóa bất đối xứng.

##### DH

##### DSA

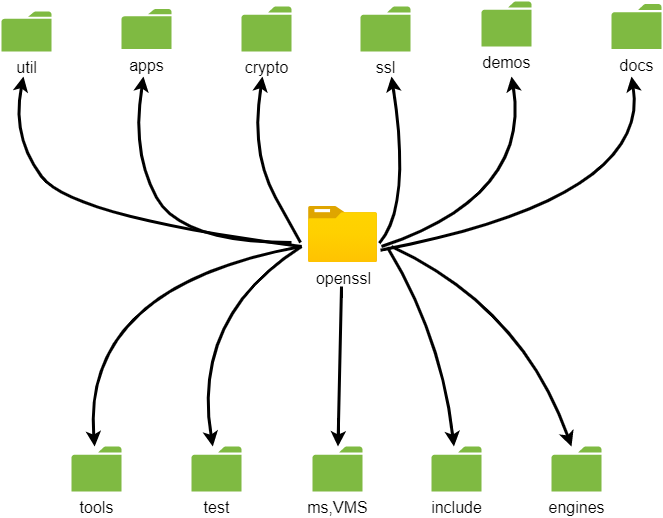
##### RSA

## Kiến trúc

### Kiến trúc thư mục openssl

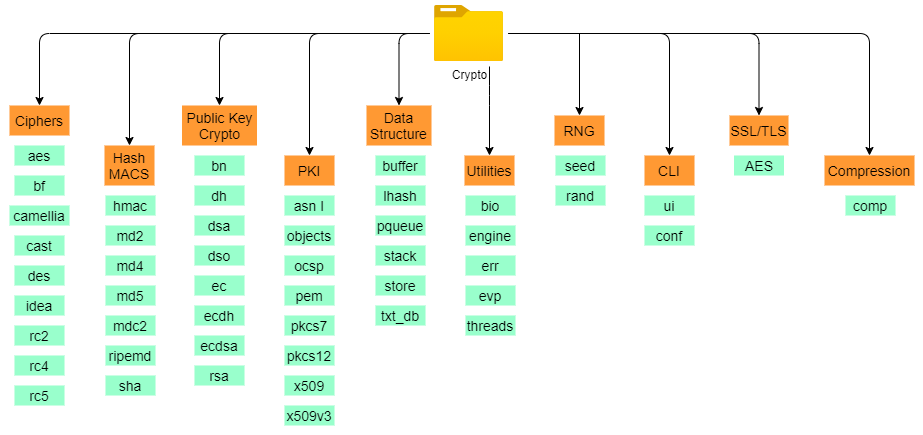
Thư viện phân mục rất rõ ràng và mạch lạc bao gồm 7 mục chính:

1. Apps: nơi chứa công cụ dòng lệnh OpenSSL, tức chương trình mà phần gới thiệu đã đề cập.
2. Crypto: nơi chứa thư viện mật mã.
3. Ssl: nơi chứa thư viện SSL dùng trong an toàn liên lạc ở tầng Session trong mô hình OSI.
4. Demos: nơi chứa những bản demo được thể hiện ở dạng mã nguồn, là thành phẩm của việc lắp ráp và sử dụng api tạo nên một sản phẩm nhỏ.
5. Docs: nơi chứa tài liệu tham khảo về các api.
6. Engines: nơi chứa cung cấp driver hỗ trợ sử dụng tài nguyên phần cứng trong việc tăng tốc hiệu năng.
7. Include: nơi chứa tệp tin header \*.h.
8. VMS, ms: nơi chứa chỉ định nền tảng, tên sẽ được thay đổi tùy theo hệ điều hành được sử dụng. Ví dụ: MACOS nếu sử dụng hệ điều hành của Apple, ...
9. Test: nơi chứa những bài test, chủ yếu là các thuật toán được hỗ trợ.
10. Util: Per shell được tạo ra bởi mã C.



Hình. Kiến trúc thư mục openssl.

### Kiến trúc thư mục crypto



Hình. Kiến trúc trong thư mục openssl/crypto.

## Các thành phần mật mã

### Mã hóa bất đối xứng

#### Hàm PEM (Privacy Enhanced Mail)

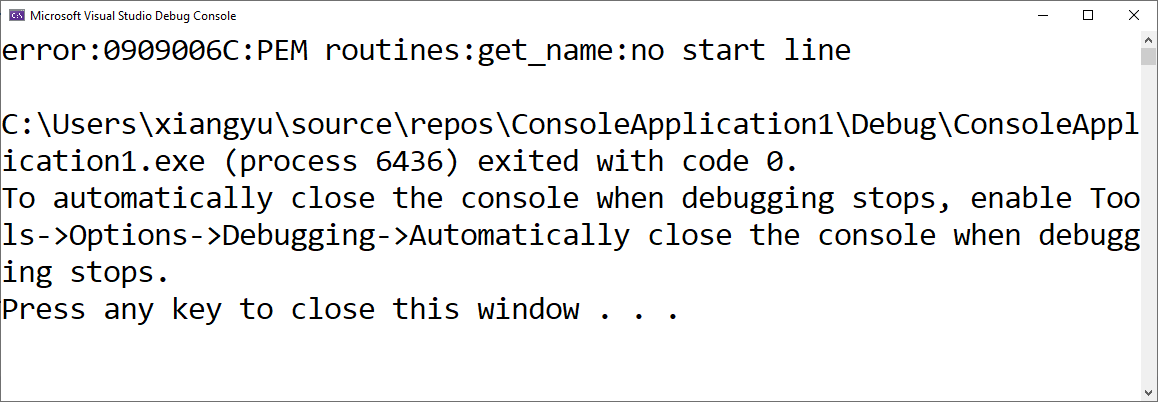
PEM là một định dạng cho việc encode các đối tượng mật mã, được viết tắt là Privacy Enhanced Mail. Ban đầu PEM được thiết kế cho PGP, nhưng nó đã trở thành một chuẩn định dạng để lưu trữ bản mã nhị phân trên ổ cứng. Đây là một định dạng tiêu chuẩn được sử dụng cho các khóa công khai và khóa riêng, chứng chỉ số, v.v ... Tệp tin có định dạng mở rộng \*.pem chính là PEM.

Thư viện OpenSSL cung cấp các hàm, các cấu trúc dữ liệu để chuyển đổi từ đối tượng PEM sang OpenSSL và ngược lại. Ví dụ để đọc khóa RSA được lưu trong tệp tin định dạng PEM.



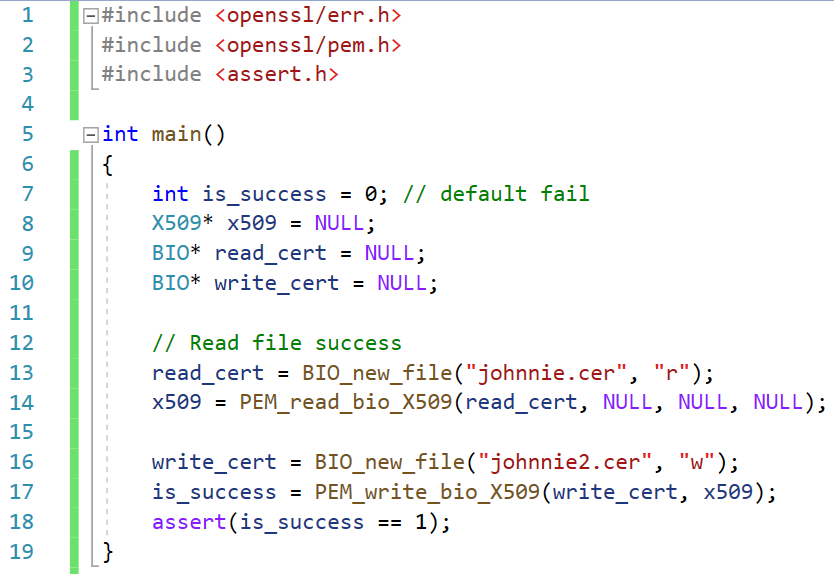
Hình. Đọc khóa bí mật từ tệp tin.

Mục đích của mã nguồn sẽ đọc khóa bí mật từ tệp tin có định dạng PEM và chuyển đổi sang đối tượng quản lý RSA trong OpenSSL thông qua con trỏ FILE đang hỗ trợ chế độ đọc. Nếu tệp tin không đúng định dạng hoặc không tồn tại, sử dụng cơ chế thông báo lỗi của thư viện để có được thông tin lỗi chi tiết.

Hình. Kết quả khi đọc sai định dạng PEM.

Ở trường hợp ví dụ hàm **PEM\_read\_RSAPrivateKey** như trên thì 3 đối số sau dùng được đưa vào là NULL. Đối với các đối số thứ hai, cho phép chỉ định không gian lưu trữ đối tượng quản lý khóa. Đối số thứ ba và thứ tư chỉ định một hàm gọi lại (callback) trong trường hợp cần mật khẩu để mở tệp tin PEM bị mã hóa. Giá trị NULL sẽ thông báo cho thư viện sử dụng hàm gọi lại (callback) mặc định và không cần mật khẩu để giải mã tệp tin PEM.

Tệp tin header PEM có hỗ trợ chuyển đổi ngược lại từ kiểu dữ liệu quản lý của OpenSSL sang dữ liệu trên ổ cứng. Ví dụ: lưu chứng chỉ X.509 dưới dạng tệp tin.



Hình. Ghi đối tượng quản lý X509 vào tệp tin.

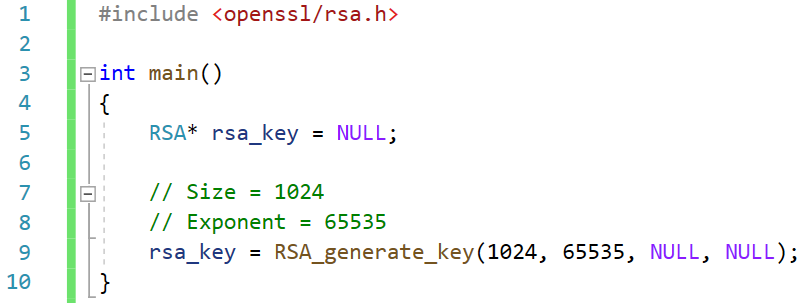
Hàm **PEM\_write\_bio\_X509** sẽ trả về giá trị 1 nếu như ghi tệp tin thành công, ngược lại là giá trị 0 nếu thất bại. Hàm assert có chức năng gây crash chương trình, là một cách thông báo/debug cực kỳ rõ ràng dành cho lập trình viên.

#### Mã hóa khóa công khai

##### Sinh khóa

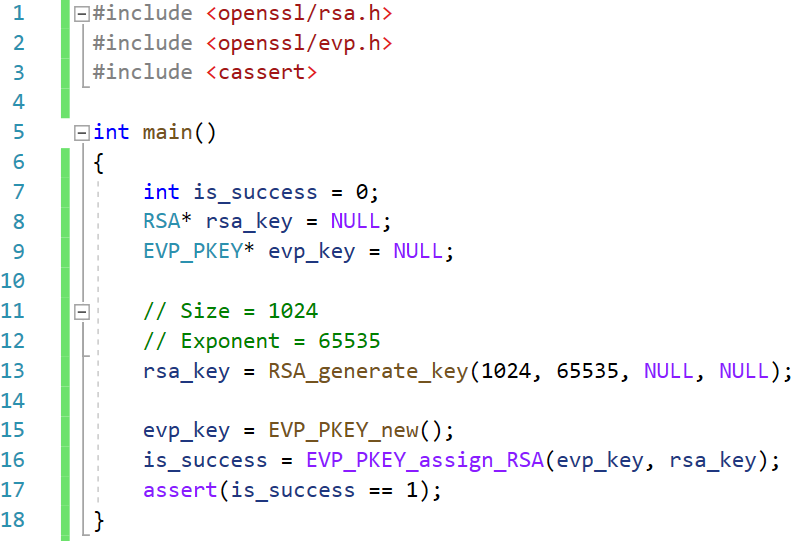
Thư viện OpenSSL sử dụng 2 loại cấu trúc khóa cho mã hóa khóa công khai. Đầu tiên, nó sử dụng cấu trúc hàm để lưu trữ khóa cho thuật toán cụ thể nào đó, ví dụ RSA hoặc DSA. Sau đó, nó sử dụng cấu trúc EVP\_PKEY để lưu trữ khóa để sử dụng cho bất kỳ thuật toán nào.

Trong cả hai trường hợp, OpenSSL không tạo riêng cấu trúc để phân biệt giữa khóa bí mật và khóa công khai. Ở ví dụ trên sử dụng cấu trúc dữ liệu RSA để lưu trữ khóa, đặt tên gợi nhớ là khóa riêng tư để sử dụng cho các hàm sau đó. Ví dụ sau đây tạo khóa bí mật RSA có độ dài 1024 bit.



Hình. Tạo khóa RSA.

Ở trong ngữ cảnh high interface, thay vì sử dụng RSA như low interface, EVP cung cấp một cấu trúc riêng để lưu trữ khóa công khai tên là EVP\_PKEY sử dụng cho mã hóa khóa công khai. Ví dụ: tạo khóa thông qua low interface.

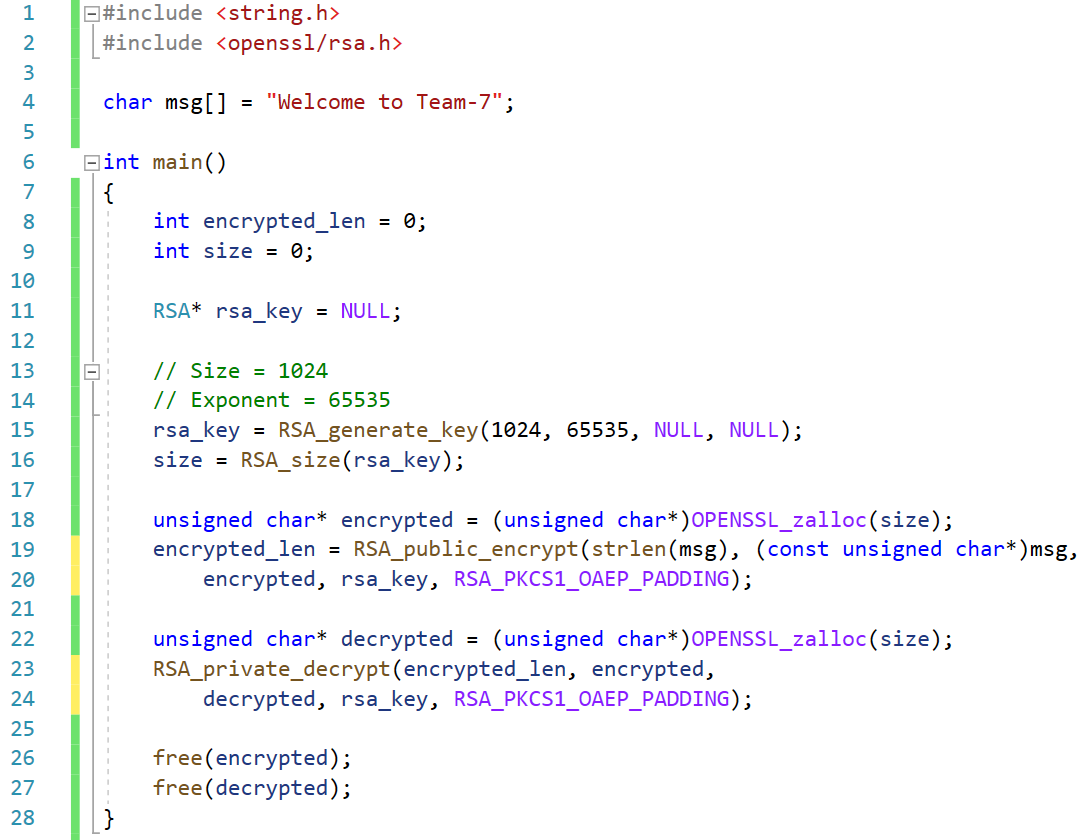


Hình. Tạo khóa theo cấu trúc EVP thông hàm gán.

Đoạn code trên đã phân tách riêng biệt khóa công khai trong cặp khóa sang cho biến evp\_key lưu trữ.

##### Mã hóa và giải mã

Mã hóa khóa công khai ở trường hợp này sử dụng low interface dễ dàng cho việc tiếp cận. Theo trình tự mã hóa và giải mã, hàm **RSA\_public\_encrypt** được sử dụng để mã bản tin, hàm **RSA\_private\_decrypt** để giải mã.



Hình. Mã hóa và giải mã khóa công khai.

Ở đối số thứ nhất và thứ hai, hàm **RSA\_public\_encrypt** yêu cầu cần được cung cấp chiều dài và thông tin bản rõ, đối số thứ ba là nơi sẽ chứa bản mã sau khi mã hóa bằng khóa công khai được lưu trong cấu trúc của biến rsa\_key, đối số cuối cùng cung cấp cho thuật toán RSA sử dụng thuật toán padding PKCS#1, thông tin trả về sẽ là chiều dài bản mã.

Nhờ vào thông tin chiều dài bản mã được mã hóa, hàm **RSA\_private\_decrypt** sử dụng khóa bí mật được lưu ở bên trong cấu trúc rsa\_key để giải mã và cho ra bản rõ.