

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
KHOA CÔNG NGHỆ THÔNG TIN



THỰC HÀNH

NHẬP MÔN MÃ HÓA MẬT MÃ

Đồ án 3: Làm quen với OpenSSL

Sinh viên:
21120107 - Nguyễn Minh Nhật

Giảng viên:
PGS. TS. Nguyễn Đình Thúc
Ths. Nguyễn Văn Quang Huy


Mục lục

1	Khóa RSA trong OpenSSL	2
1.1	Khóa bí mật	2
1.2	Khóa công khai	3
1.3	Thông tin mã nguồn	3
2	Mã hóa khóa công khai RSA trong OpenSSL	4
2.1	Mã hóa	4
2.1.1	Nhắc lại lý thuyết	4
2.1.2	Mã hóa trong OpenSSL	4
2.2	Giải mã	5
2.2.1	Nhắc lại lý thuyết	5
2.2.2	Giải mã trong OpenSSL	5
2.3	Thông tin mã nguồn	6
3	Chữ ký điện tử RSA trong OpenSSL	7
3.1	Tạo chữ ký số	7
3.1.1	Nhắc lại lý thuyết	7
3.1.2	Chữ ký số trong OpenSSL	7
3.2	Xác thực chữ ký	8
3.2.1	Nhắc lại lý thuyết	8
3.2.2	Xác thực chữ ký trong OpenSSL	9
3.3	Thông tin mã nguồn	10
4	Cấu trúc thư mục	11
5	Tham khảo	13

1 Khóa RSA trong OpenSSL

Lý thuyết tham khảo từ tài liệu PKCS [4], ảnh lấy từ bài viết trên cem.me [5].

1.1 Khóa bí mật



RSA PRIVATE KEY

Public-Key Cryptography Standard (PKCS) #1

```

-----BEGIN RSA PRIVATE KEY-----
MIHyAgEAAjEArDZ71puf1AzhF8qbpX1x59EsudjdsShdd7e
bd1JR4MuyRWVcRgUtr2+bzzh4MfPAgMBAAECMarzh9swePUY
F155SpJjd1rn9e9FhR59HAhPODG+sSpDgw2uLh22hepZ/6vy
cnT8uQIZANuJj1Gp9M4b0Tbh1KzJyQn5hvZAa02YjxQIZAM14
9FA406bIRp70Y1DzrA77AitSAygyIYPwCCGm018Ud123t2
b6E1819Y7T1p3kQtAhgUQ1emoZX8SEub18T/Gw9K10JYkKHo
T7ECGQCOTV951KYvpk3/fCfIDZEWHGH1M13r1Y=
-----END RSA PRIVATE KEY-----

```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00:	30	81	F2													
				02	01	00										
10:	50	33	84	5F	2A	6E	95	E2	C7	9F	44	B2	E7	63	76	C4
20:	A1	75	DE	DE	6D	D9	49	47	83	2E	C9	15	95	71	18	14
30:	4E	BD	BE	6F	3C	E1	E0	C7	CF							
				02	03	01	00	01								
40:	0A	F3	1F	DB	30	78	F5	18	16	5E	79	4A	92	63	77	5A
50:	E7	F6	8F	45	85	1E	7D	1C	08	4F	38	31	BE	B1	2A	43
60:	83	00	AE	2E	1D	B6	85	EA	59	FF	AB	F2	72	74	FC	B9
70:	02	19	00	DB	A3	94	6A	7D	33	86	C3	4D	B8	65	2B	32
80:	72	42	7E	61	BD	90	1A	D3	66	23	C5					
				02	19	00	C8	B8								
90:	F4	50	38	D3	A6	C8	46	BA	7B	39	88	83	CE	B0	3B	EE
A0:	D0	22	B5	20	32	83										
				02	18	4C	FC	02	1A	6D	35	F1	47			
B0:	65	67	78	76	6F	A1	22	F3	5F	58	ED	38	A9	DE	44	2D
C0:	02	18	14	43	57	A6	A1	95	FC	48	4B	9B	97	C4	FF	1B
D0:	0F	4A	8B	42	58	9B	A8	68	4F	B1						
				02	19	00	90	4D	5F							
E0:	79	94	A6	2F	A6	4D	FF	7C	27	E2	0D	91	16	1C	68	47
F0:	94	C9	77	AE	56											

242 Bytes [RSA Private Key]

1 Byte [version]
0

49 Bytes [modulus]
26505978354099432385854240949
82001802591172809983985557600
65597336270782727025227749976
35806320016501911976396507087

3 Bytes [public exponent, e]
65537

48 Bytes [private exponent, d]
18853138196977636259190170749
02777291880528165276399258207
97998336732348017565500790977
9932043027924692388684299449

25 Bytes [prime 1, p]
53855322025246274766204372722
83925975453806997483574141893

25 Bytes [prime 2, q]
49217008379736307618029248664
01218884538432719849865097859

24 Bytes [exponent 1, d mod (p-1)]
18876521693729224371797315193
19377810735148092515604644909

24 Bytes [exponent 2, d mod (q-1)]
49684869225069107710282971323
2379635816956389131737911217

25 Bytes [coefficient, q⁻¹ mod p]
35382805823555797330395780560
71861297596450213620182658646

ASN.1 Types
xx Bytes

02 xx Integer
30 xx Sequence

ASN.1 starts with a
Type ID followed
by a length

Ex: Type 7 0
Length 3
07 03 ...

Hình 1: Tiêu chuẩn khóa bí mật

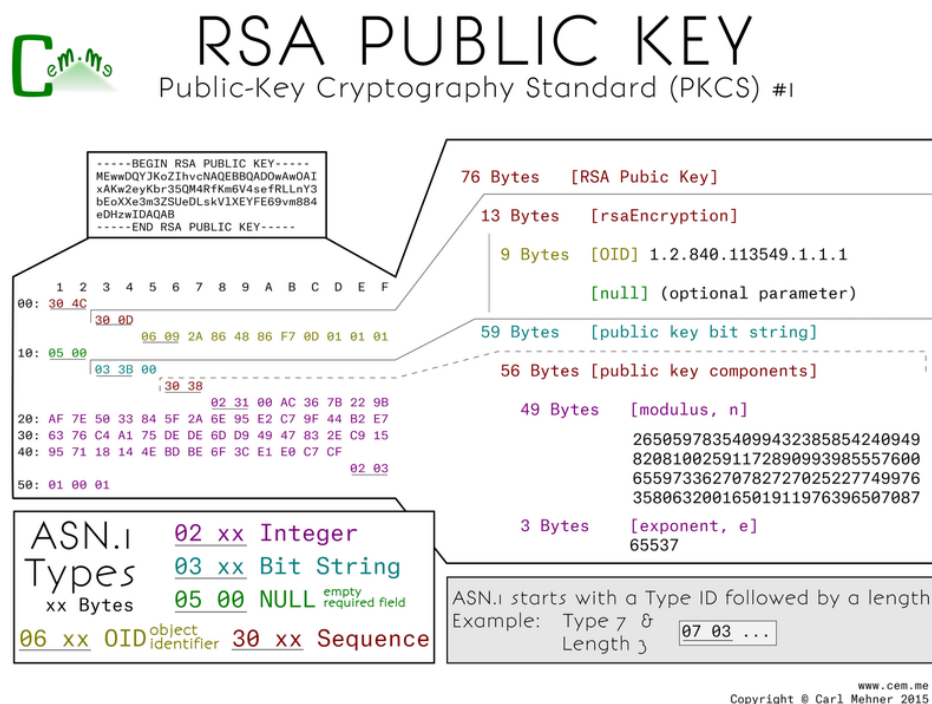
Trong đó:

- **version:** để tương thích với các sửa đổi sau này của tài liệu. Phiên bản này của tài liệu là 0, còn khi sử dụng nhiều số nguyên tố (**otherPrimeInfos**) thì là 1.
- **modulus:** là mô-đun RSA n .
- **publicExponent:** là số mũ - khóa công khai e của RSA.
- **privateExponent:** là số mũ - khóa bí mật d của RSA.
- **prime1:** số nguyên tố p của n .
- **prime2:** số nguyên tố q của n .
- **exponent1** = $d \bmod (p - 1)$.
- **exponent2** = $d \bmod (q - 1)$.
- **coefficient:** là hệ số CRT $q^{-1} \bmod p$.

- **otherPrimeInfos**: chứa thông tin cho các số nguyên tố bổ sung r_3, \dots, r_u theo thứ tự. Nếu **version** = 1.

Lưu ý: trong các phiên bản mới của *OpenSSL* thường bỏ qua thông số này.

1.2 Khóa công khai



Hình 2: Tiêu chuẩn khóa công khai

Trong đó:

- **modulus**: là mô-đun RSA n .
- **publicExponent**: là số mũ - khóa công khai e của RSA.

1.3 Thông tin mã nguồn

- Ngôn ngữ sử dụng: Python.
- Thư viện cần cài đặt:

```
pip install pycryptodome
```

- Cách thức chạy:

```
cd Source
python pkcs.py
```

- Link video demo: [demo Câu 1](#)

2 Mã hóa khóa công khai RSA trong OpenSSL

Lý thuyết tham khảo từ tài liệu PKCS [4].

2.1 Mã hóa

2.1.1 Nhắc lại lý thuyết

Input	Ouput
(n, e) khóa công khai RSA	c số nguyên đại diện cho mã hóa, từ 0 đến $n - 1$
m số nguyên đại diện cho thông điệp, từ 0 đến $n - 1$	

Các bước thực hiện:

1. Kiểm tra số nguyên đại diện bản mã m có thuộc từ 0 đến $n - 1$.
2. Tính $c = m^e \bmod n$.
3. Trả ra c .

2.1.2 Mã hóa trong OpenSSL

Tham khảo theo open source từ openssl [3], cụ thể là hàm `rsa_ossl_public_encrypt()` trong file `rsa_ossl.c` [1].

1. Kiểm tra kích thước mô đun n và số mũ - khóa công khai e .
2. Chuẩn bị dữ liệu và thực hiện padding (**PKCS1** hoặc **OAEP**).
3. Chuyển đổi dữ liệu và padding thành **BIGNUM**.
4. Kiểm tra kích thước dữ liệu sau chuyển đổi.
5. Cache mô-đun nếu được yêu cầu.
6. Thực hiện phép lũy thừa: $ret = f^e \bmod n$.

Trong đó:

- f là thông điệp sau khi thêm padding.
 - ret là kết quả phép lũy thừa.
7. Chuyển đổi **BIGNUM** thành dữ liệu đầu ra.
 8. Giải phóng bộ nhớ và trả về bản mã.

Lưu ý: đối với câu lệnh:

```
openssl pkeyutl -in <plain> -out <cipher> -inkey <pub.pem> -pubin -encrypt
```

- padding mặc định là *PKCS1v1.5*.

2.2 Giải mã

2.2.1 Nhắc lại lý thuyết

Input	Ouput
khóa bí mật RSA, gồm: K - cặp (n, d) - bộ các số $(p, q, dP, dQ, qInv)$ c - số nguyên đại diện bản mã, từ 0 đến $n - 1$	m - số nguyên đại diện cho thông điệp, từ 0 đến $n - 1$

Các bước thực hiện:

1. Kiểm tra bản mã c có thuộc từ 0 đến $n - 1$.
2. Tính m :
 - Cách tính theo (n, d) : $m = c^d \bmod n$.
 - Cách tính theo $(p, q, dP, dQ, qInv)$:
 - (a) Tính $m_1 = c^{dP} \bmod p$ và $m_2 = c^{dQ} \bmod q$
 - (b) Tính $h = (m_1 - m_2) * qInv \bmod p$.
 - (c) Tính $m = m_2 + q * h$.
3. Trả ra m .

Lưu ý: ở đây bỏ qua vấn đề RSA cho nhiều số nguyên tố.

2.2.2 Giải mã trong OpenSSL

Tham khảo theo open source từ openssl [3], cụ thể là hàm `rsa_ossl_private_decrypt()` trong file `rsa_ossl.c` [1].

1. Khởi tạo biến f và ret , đây là hai biến **BIGNUM** được sử dụng trong quá trình tính toán.
2. Xử lý padding, điều chỉnh nếu cần.
3. Chuyển đổi dữ liệu đầu vào (from) thành một biến BIGNUM f .
4. Cache modulus nếu cần tối ưu hiệu suất.
5. Thực hiện blinding nếu được cho phép (để ngăn chặn Side-Chanel).
6. Thực hiện mã hóa RSA, chia làm 2 hướng sử dụng:
 - (n, e) nếu thiếu dữ liệu cho giải mã nhanh.
 - $(p, q, dP, dQ, qInv)$ giải mã nhanh.
7. Chuyển đổi unblinding nếu blinding được sử dụng.
8. Tính Key Derivation Key (KDK) nếu sử dụng padding PKCS1.
9. Chuyển đổi kết quả thành dữ liệu đã giải mã.
10. Áp dụng padding (PKCS1 hoặc OAEP) và kiểm tra tính toàn vẹn.

11. Xử lý kết quả và gán dữ liệu đã giải mã.

12. Xóa bộ nhớ và trả về kết quả.

Lưu ý: đối với câu lệnh:

```
openssl pkeyutl -in <cipher> -out <plain> -inkey <priv.pem> -decrypt
```

- padding mặc định là *PKCS1v1.5*.

2.3 Thông tin mã nguồn

- Ngôn ngữ sử dụng: Python.
- Thư viện cần cài đặt:

```
pip install cryptography
```

- Cách thức chạy:

```
cd Source  
python encode.py  
python decode.py
```

- Link video demo: [demo Câu 2](#)

3 Chữ ký điện tử RSA trong OpenSSL

Lý thuyết tham khảo từ tài liệu PKCS [4], ảnh lấy từ bài viết trên whitehat.vn [7]. Code được thực hiện dựa trên thuật toán tại [6].

3.1 Tạo chữ ký số

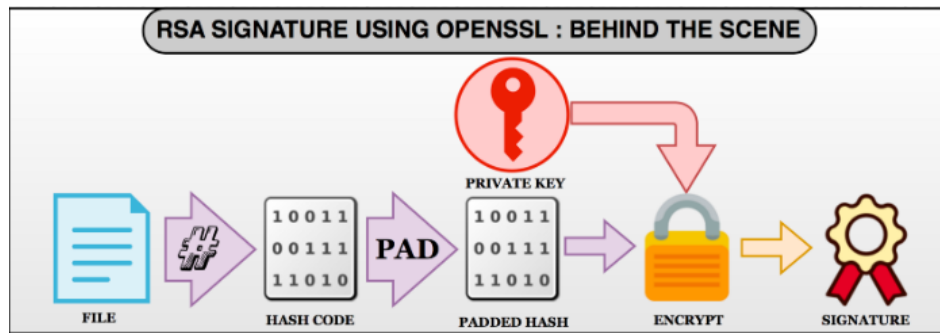
3.1.1 Nhắc lại lý thuyết

Input	Output
khóa bí mật RSA, gồm: K - cặp (n, d) - bộ các số $(p, q, dP, dQ, qInv)$ m số nguyên đại diện cho thông điệp, từ 0 đến $n - 1$	s số nguyên đại diện cho chữ ký số, từ 0 đến $n - 1$

Các bước thực hiện:

1. Kiểm tra thông điệp m có thuộc từ 0 đến $n - 1$.
2. Tính s :
 - Cách tính theo (n, d) : $s = m^d \bmod n$.
 - Cách tính theo $(p, q, dP, dQ, qInv)$:
 - (a) Tính $s_1 = m^{dP} \bmod p$ và $s_2 = m^{dQ} \bmod q$
 - (b) Tính $h = (s_1 - s_2) * qInv \bmod p$.
 - (c) Tính $s = s_2 + q * h$.
3. Trả ra s .

3.1.2 Chữ ký số trong OpenSSL



Hình 3: Quy trình ký trong OpenSSL

Tham khảo theo open source từ openssl [3], cụ thể là hàm `RSA_sign()` trong file `rsa_sign.c` [2].

1. Kiểm tra có sử dụng hàm `rsa->meth->rsa_sign`. Nếu có thì sử dụng trực tiếp hàm.

2. Xét trường hợp loại chữ ký **NID_md5_sha1** (dành cho TLS 1.1 và các phiên bản trước) sử dụng:
 - Kết hợp **MD5** và **SHA-1**. Không thêm DigestInfo.
 - Ngược lại, sử dụng hàm `encode_pkcs1` để chuẩn bị dữ liệu đã được băm, thêm lớp bọc DigestInfo.
3. Kiểm tra xem dữ liệu chuẩn bị có quá lớn so với kích thước của khóa RSA hay không. Nếu có, đánh dấu lỗi.
4. Mã hóa dữ liệu sử dụng hàm `RSA_private_encrypt` để thực hiện mã hóa dữ liệu chuẩn bị bằng khóa bí mật RSA.
5. Kiểm tra kết quả của quá trình mã hóa.
6. Gán kích thước chữ ký và trả về kết quả thành công.
7. Xử lý lỗi và giải phóng bộ nhớ nếu cần.

Lưu ý: đối với câu lệnh:

```
openssl pkeyutl -in <mess> -out <sign> -inkey <priv.pem> -sign
```

- padding mặc định là *PKCS1v1.5*.

3.2 Xác thực chữ ký

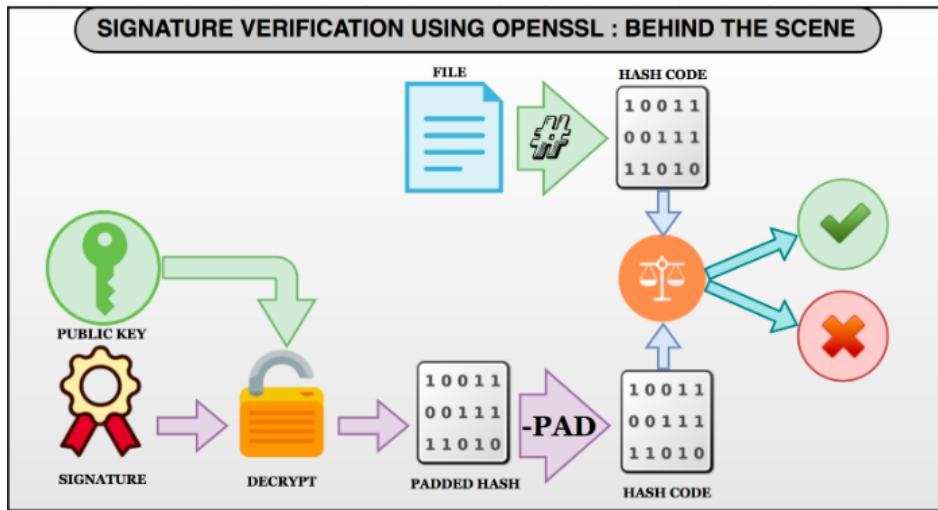
3.2.1 Nhắc lại lý thuyết

Input	Output
(n, e) khóa công khai RSA	m số nguyên đại diện cho thông điệp, từ 0 đến $n - 1$
s số nguyên đại diện cho chữ ký số, từ 0 đến $n - 1$	

Các bước thực hiện:

1. Kiểm tra chữ ký số s có thuộc từ 0 đến $n - 1$.
2. Tính $m = s^e \bmod n$.
3. Trả ra m .

3.2.2 Xác thực chữ ký trong OpenSSL



Hình 4: Quy trình xác thực trong OpenSSL

Tham khảo theo open source từ openssl [3], cụ thể là hàm `RSA_sign()` trong file `rsa_sign.c` [2].

1. Kiểm tra độ dài chữ ký có khớp với kích thước của khóa RSA hay không. Nếu không khớp, đánh dấu lỗi và trả về 0.
2. Khôi phục dữ liệu đã được mã hóa sử dụng `RSA_public_decrypt` để giải mã chữ ký bằng khóa công khai RSA. Kết quả sau khi giải mã được lưu trong `decrypt_buf`.
3. Xử lý các trường hợp đặc biệt:
 - Trường hợp sử dụng `NID_md5_sha1` (đối với TLS 1.1).
 - Trường hợp sử dụng `NID_md5`.
 - Ngược lại, sử dụng hàm `encode_pkcs1` để xây dựng encoded digest và so sánh với `decrypt_buf`. Nếu không khớp, đánh dấu lỗi.
4. Gán dữ liệu đã khôi phục (nếu có).
5. Trả về kết quả xác minh chữ ký (1 nếu thành công, 0 nếu thất bại).
6. Xử lý lỗi và giải phóng bộ nhớ nếu cần.

Lưu ý: đối với câu lệnh:

```
openssl pkeyutl -in <mess> -sigfile <sign> -inkey <pub.pem> -pubin -verify
```

- padding mặc định là `PKCS1v1.5`.

3.3 Thông tin mã nguồn

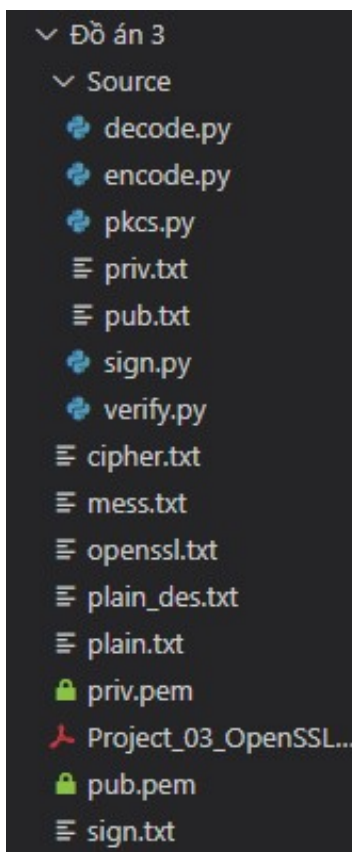
- Ngôn ngữ sử dụng: Python.

- Cách thức chạy:

```
cd Source  
python sign.py  
python verify.py
```

- Link video demo: [demo Câu 3](#)

4 Cấu trúc thư mục



Hình 5: Cấu trúc thư mục đồ án 3

Trong đó:

- **Câu 1:** `pkcs.py`
 - Đầu vào là cặp khóa `priv.pem` và `pub.pem`.
 - Đầu ra là chi tiết cặp khóa `priv.txt` và `pub.txt` (nằm cùng thư mục `Source`).
- **Câu 2:**

Mã hóa: `encode.py`

 - Đầu vào là khóa công khai `pub.pem` và bản rõ `plain.txt`.
 - Đầu ra là bản mã `cipher.txt`.

Giải mã: `decode.py`

 - Đầu vào là khóa bí mật `priv.pem` và bản mã `cipher.txt`.
 - Đầu ra là bản rõ `plain_des.txt`.
- **Câu 3:**

Ký: `sign.py`

- Đầu vào là khóa bí mật `priv.pem` và thông điệp `mess.txt`.
- Đầu ra là chữ ký số `sign.txt`.

Xác thực: `verify.py`

- Đầu vào là khóa công khai `pub.pem`, thông điệp cần kiểm tra `mess.txt` và chữ ký xác thực `sign.txt`.
- Đầu ra là kết quả xác thực.

5 Tham khảo

Tài liệu

- [1] *Code RSA trong OpenSSL*. https://github.com/openssl/openssl/blob/master/crypto/rsa/rsa_oss1.c. Dec. 2023.
- [2] *Code Sign RSA trong OpenSSL*. https://github.com/openssl/openssl/blob/master/crypto/rsa/rsa_sign.c. Dec. 2023.
- [3] *Github OpenSSL*. <https://github.com/openssl/openssl/>. Dec. 2023.
- [4] Burt Kaliski Jakob Jonsson. *Public-Key Cryptography Standards (PKCS) 1: RSA Cryptography Specifications Version 2.1*. <https://datatracker.ietf.org/doc/rfc3447/>. Dec. 2023.
- [5] Carl Mehner. *Certificate Binary Posters (Part One)*. <https://www.cem.me/20141221-cert-binaries.html>. Dec. 2023.
- [6] *PKCS 1: RSA Cryptography Specifications Version 2.2*. <https://www.rfc-editor.org/rfc/rfc8017#section-9.2>. Dec. 2023.
- [7] *RSA chữ ký và xác minh bằng Openssl*. <https://whitehat.vn/threads/rsa-chu-ky-va-xac-minh-bang-openssl.11606/>. Dec. 2023.