TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

ĐỒ ÁN 03 – LÀM QUEN VỚI OPENSSL

Họ tên sinh viên: Ngũ Duy Tính

MSSV: 21120572

Lớp: 21_22

Giảng viên: Nguyễn Văn Quang Huy

1. Tìm hiểu về khóa RSA của OpenSLL

<priv.pem>

Định dạng: PEM (Base64 encode).

Thành phần:

-----BEGIN PRIVATE KEY-----: Dòng bắt đầu của dữ liệu khóa bí mật.

Dữ liệu khóa bí mật: Chuỗi Base64 được mã hóa theo ASN.1 DER-TLV.

-----END PRIVATE KEY-----: Dòng kết thúc của dữ liệu khóa bí mật.

<pub.pem>

Định dạng: PEM (Base64 encode).

Thành phần:

-----BEGIN PUBLIC KEY-----: Dòng bắt đầu của dữ liệu khóa công khai.

Dữ liệu khóa công khai: Chuỗi Base64 được mã hóa theo ASN.1 DER-TLV.

-----END PUBLIC KEY-----: Dòng kết thúc của dữ liệu khóa công khai.

Dữ liệu là chuỗi Base64, vì vậy ta cần decode về dạng hex string để dễ dàng sử lý.

Dữ liệu được mã hóa theo ASN.1 DER-TLV. ASN.1 DER-TLV là một chuẩn mã hóa dữ liệu được sử dụng để định dạng và truyền tải thông tin có cấu trúc trong mạng và bảo mật thông tin, sử dụng cấu trúc TLV và quy tắc mã hóa DER.

Dữ liệu được chia thành các trường TLV. Mỗi trường TLV bao gồm các phần sau:

Tag: Được xác định loại dữ liệu và cấu trúc của nó.

Một số Tag indicating thông thường:

01: BOOLEAN

02: INT

03: BIT STRING, bytes đầu tiên thường là ‘unused bits’

04: OCTET STRING

05: NULL

06: OBJECT

30: SEQUENCE

Nếu Tag indicating là 03, 04, 30 thì trong trường TLV đó có thể chứa thêm các trường TLV khác.

Length: Xác định độ dài của của giá trị dữ liệu tính từ offset bắt đầu value.

Trong hệ mã RSA length thường nằm từ [1..0xFFFF] nên ta có:

Mọi độ dài từ 0 đến 0x7F đều được mã hóa thành một byte trong 00..7F;

Mọi độ dài cao hơn lên tới 0xFF đều được mã hóa dưới dạng tiền tố 81 và một byte;

Mọi độ dài cao hơn lên tới 0xFFFF đều được mã hóa dưới dạng tiền tố 82 và hai byte;

Mọi độ dài cao hơn lên tới 0xFFFFFF đều được mã hóa dưới dạng tiền tố 83 và ba byte;

Quy tắc tiếp tục áp dụng cho độ dài cao hơn, nhưng một số tiêu chuẩn (bao gồm ISO/IEC 7816-4:2013, phụ lục E.2) loại trừ rõ ràng những điều này.

Value: Giá trị thực tế của phần dữ liệu, có thể chứa các trường TLV khác.

Cấu trúc phần dữ liệu khóa file <priv.pem>

0 SEQUENCE{

4 INT

7 SEQUENCE{

9 OBJECT: rsaEncryption

}

20 NULL

22 OCTET STRING{

SEQUENCE{

Version

Modulus

Public exponent

Private exponent

Secret prime p

Secret prime q

dp: d mod (p – 1)

dq: d mod (q – 1)

$q_{inv} = q^{-1} \bmod p$

}

}

}

Cấu trúc phần dữ liệu khóa file <pub.pem>

0 SEQUENCE{

4 SEQUENCE{

6 OBJECT: rsaEncryption

}

17 NULL

19 BIT STRING{

Unused bits

SEQUENCE{

Modulus

Public exponent

}

}

Với mỗi giá trị khóa số nguyên được đọc thì bit trái nhất chính là bit dấu.

Các tham số được đọc và ghi và file <privKey.txt> và <pubKey.txt> dưới dạng dãy hexa (big-edian).

2. Tìm hiểu về mã hóa công khai RSA của OpenSSL

a. Mã hóa bằng khóa công khai.

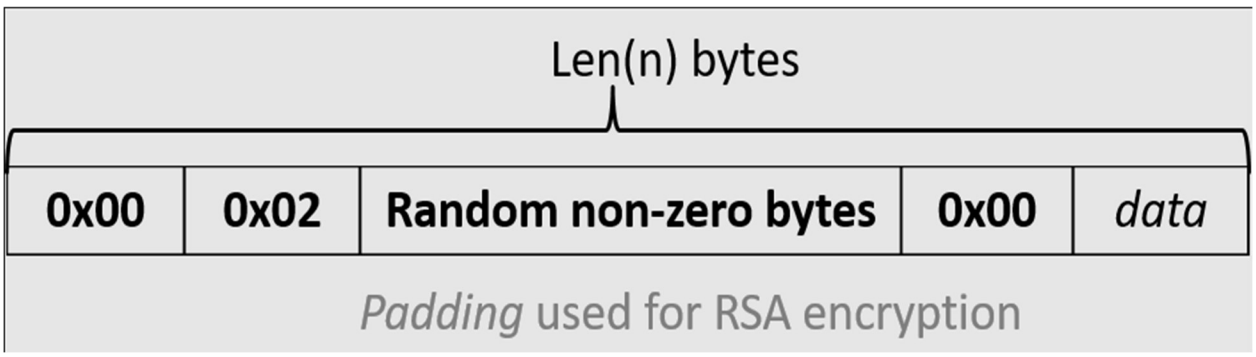
Dùng các hàm được định nghĩa sẵn để đọc khóa công khai.

Sau khi thiết lập ngữ cảnh, có được khóa công khai và nội dung cần mã hóa thì sẽ tiến hành mã hóa.

Tùy vào từng loại padding mà quá trình thêm padding khác nhau:

RSA_PKCS1_PADDING:

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



EM = 0x00 0x02 || Random non-zero bytes || 0x00 || data

Kích thước của data cần mã hóa phải không lớn hơn len(n) – 11.

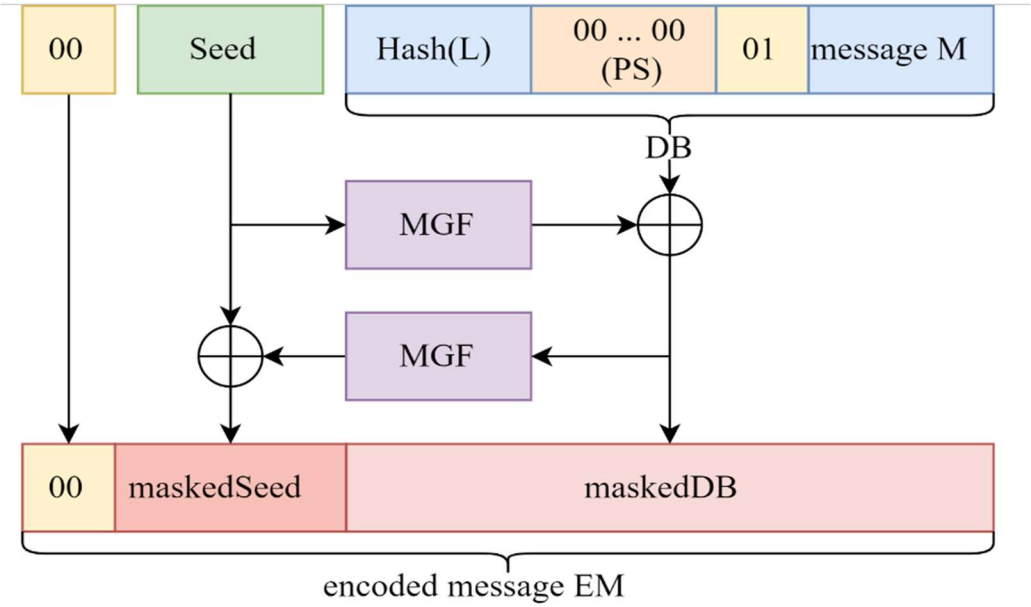
EM có kích thước bằng len(n) bytes.

Radombytes có kích thước bằng len(n) – 3 – len(data).

Thêm bytes 0x00 để đảm bảo rằng EM nhỏ hơn len(n) bytes.

RSA_PKCS1_OAEP_PADDING:

$ openssl pkeyutl -in <plain> -out <cipher> -inkey <pub.pem> -pubin -encrypt -pkeyopt\
rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:digest



EM = 0x00 || maskedSeed || maskedDB

Kích thước của data cần mã hóa không lớn hơn len(n) - 42

EM có kích thước bằng len(em) bytes.

Kiểm tra các điều kiện cần thiết trước khi thêm padding:

len(data) phải nhỏ hơn len(em) -2*len(digest) -1.

len(em) phải lớn hơn 2*len(digest) -1.

DB = Hash(P) || PS || 01 ||data

P: encoding parameter.

PS: dãy 0x00 bytes với kích thước len(em) – 2*len(digest) – 2

Thực hiện tạo EM:

Tiến hành băm P: Hash(P)

Tạo PS có kích thước len(em)-len(data) -2*len(digest) -1.

Seed là dãy random bytes có kích thước len(digest) bytes.

dbMask = MGF(mgfSeed, len(n) -len(digest) – 1, digest)

maskedDB = DB XOR dbMask

mgfSeed = MGF(maskedDB, len(digest), digest)

maskedSeed = mgfSeed XOR mgfSeed

EM = 00 || maskedSeed || maskedDB

Bytes đầu tiên là 0x00 là bắt buộc để cho EM có kích thước nhỏ hơn len(n).

RSA_NO_PADDING:

$ openssl pkeyutl -in <cipher> -out <plain> -inkey <pub.pem> -pubin -encrypt -pkeyopt\
rsa_padding_mode:none

Kiểm tra các điều kiện trước khi thêm padding:

len(data) phải bằng len(n).

EM = data

Sau khi có EM, chuyển EM thành EM_int để thực hiện mã hóa.

So sánh EM_int với modulus: Nếu EM_int >= modulus thì má hóa thất bại.

Tiến hành mã hóa $c\_int = EM\_int^e \ mod \ n$

Chuyển c_int về bytes và thêm 0x00 byte vào đầu c nếu len(c) < len(n).

Tiến hành hũy các đối tượng đã tạo.

Ghi c vào file và kết thúc.

b. Giải mã bằng khóa bí mật.

Dùng các hàm được định nghĩa để dọc khóa mật.

Sau khi thiết lập ngữ cảnh, có được khóa bí mật và nội dung cần giải mã thì tiến hành giải mã.

Nội dung cần giải mã là c, ta chuyển c thành c_int và giải mã:

$$m\_int = c^d \ mod \ n$$

Sau khi có m_int, tiến hành chuyển m_int thành m là dãy bytes.

Tùy từng vào loại padding mà quá trình kiểm tra padding khác nhau:

RSA_PKCS1_PADDING:

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

Kiểm tra các điều kiện trước khi tiến hành kiểm tra padding:

len(n), len(data) > 0

len(data) < len(n)

Sau khi thỏa các điều kiện tiến hành kiểm tra padding.

Sao chép dữ liệu từ ‘m’ sang ‘em’ một cách an toàn.

Tìm 0-byte trong em nếu không tìm thấy thì kiểm tra không đúng và kết thúc.

Sau khi có được vị trí 0-byte thì tiến hành sao chép data vào biến trả về-‘to’ nhằm hạn chế việc rò rỉ thông tin qua các kênh phụ thuộc thời gian.

RSA_PADDING_OAEP:

$ openssl pkeyutl -in <plain> -out <cipher> -inkey <pub.pem> -pubin -encrypt -pkeyopt\
rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:digest

Kiểm tra các điều kiện để tiến hành kiểm tra padding:

len(n), len(data) > 0

len(n) > len(data), len(n) > 2*len(digest) + 2

Sau khi thỏa các điều kiện tiến hành kiểm tra.

Sao chép dữ liệu từ ‘m’ sang ‘em’ một cách an toàn.

Tìm DB bằng các bước sau:

seedMask = MGF(maskedDB, len(digest))

Seed = maskedSeed XOR seedMask

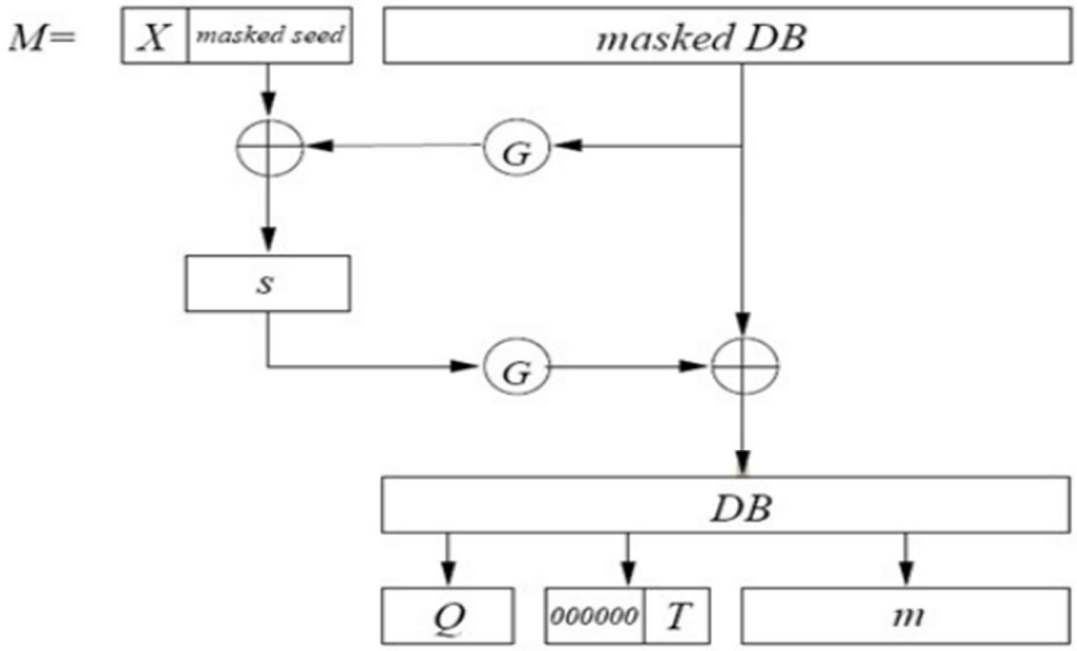
dbMask = MGF(maskedSeed, len(n) - len(digest) - 1)

DB = maskedDB XOR dbMask

DB = Hash(P) || PS || 0x01 || data

Tìm 1-byte trong DB nếu không tìm thấy thì kiểm tra không đúng và kết thúc.

Sau khi có được vị trí 1-byte thì tiến hành sao chép data vào biến trả về-'to' nhằm hạn chế việc rò rỉ thông tin qua các kênh phụ thuộc thời gian.



RSA_NO_PADDING:

$ openssl pkeyutl -in <cipher> -out <plain> -inkey <priv.pem> -decrypt -pkeyopt \
rsa_padding_mode:none

Sao chép data vào biến trả về-'to'.

Sau khi quá trình giải mã hoàn thành, tiến hành ghi ra file.

3. Tìm hiểu về chữ ký điện tử RSA của OpenSSL

a. Ký bằng khóa bí mật

Dùng các hàm được định nghĩa để dọc khóa mật.

Sau khi thiết lập ngữ cảnh, có được khóa bí mật và nội dung vần ký thì tiến hành ký.

Khi không sử dụng thuật toán băm:digest thì data có kích thước tối đa là 64 bytes. Khi sử dụng thuật toán băm, kích thước của nội dung cần ký không giới hạn.

Tùy vào từng loại padding mà quá trình thêm padding sẽ khác nhau.

RSA_PKCS1_PADDING:

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

$ openssl digest -sha256 -sign <priv.pem> -out <sign> <mess>

EM = 0x00 0x01 || 0xFF…FF || 0x00 || data

Data = HashAlgID || Hash(M)

Kiểm tra điều kiện để tiển hành thêm padding:

len(data) < len(n) - 11

Sau khi thỏa điều kiện thì tiến hành thêm padding.

Thực hiện tạo EM.

Thêm 0x00 0x01 vào đầu EM.

Thêm dãy có it nhất 8 bytes 0xFF, có kích thước len(n) – 3 – len(data).

Thêm 0x00 bytes kế tiếp để kết thúc dãy padding.

Thêm data vào cuối dãy EM.

RSA_PSS_PADDING

$ openssl dgst -sha256 -sign <priv.pem> -sigopt rsa_padding_mode:pss -sigopt\ rsa_pss_saltlen:-1 -out <sign> <mess>

EM = maskedDB || H || bc

Kiểm tra các điều kiện trước khi tiến hành tạo EM.

Xác định kích thước muối:

-1: sLen = hLen

-2: Kích thước tối đa

-3: Kích thước tối đa, nhưng dùng trong ký

-4: Kích thước nhỏ nhất. min(hLen, maximum length)

-N: Không xác định

Tạo giá trị salt ngẫu nhiên.

Tính H = Hash(0x00… || Hash(data) || salt)

DB = 0x00… || salt, có độ dài len(EM) – hlen - 1

maskH = MGF(H, len(DB), digest)

maskedDB = DB XOR maskH

Thêm maskedDB, H, bc vào đúng thứ tự trong EM để tạo EM.

Sau khi có EM, tiến hành chuyển EM thành EM_int để mã hóa bằng khóa bí mật.

So sánh EM_int với modulus để đảm bảo rằng EM_int < modulus.

Tiến hành mã hóa: $sign\_int = EM^d \ mod \ n$

Chuyển sign_int về bytes và thêm 0x00 byte vào đầu sign nếu len(sign) < len(n).

Tiến hành hũy các đối tượng đã tạo.

Ghi sign vào file và kết thúc.

b. Xác nhận chữ ký bằng khóa công khai.

Dùng các hàm có sẵn để đọc khóa công khai.

Sau khi thiết lập ngữ cảnh, có được nội dung cần được xác nhận thì tiến hành xác nhận.

Nội dung cần xác nhận là sign, chuyển sign thành sign_int và tiến hành giải mã:

$$messint = sign\ int^e mod\ n$$

Chuyển mess_int thành mess để bắt đầu quá trình xác thực.

Tùy vào từng loại padding mà có quá trình khôi phục dữ liệu đã ký khác nhau:

RSA_PKCS1_PADDING

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

00 || 01 || PS || 00 || Data

Kiểm tra các điều kiện cần thiết trước khi kiểm tra padding.

len(n) > 11

Bytes đầu tiến là 0x00 nếu len(n) bằng len(data) hoặc bằng 0x01 nếu len(n) khác len(data) + 1.

Sau khi thỏa các điều kiện tiến hành duyệt qua toàn bộ padding data, nếu không tìm thấy byte 0x00 thì dừng và kết thúc.

Kiểm tra các điều kiện trước khi trả về data.

RSA_PSS_PADDING

$ openssl dgst -sha256 -verify <pub.pem> -sigopt rsa_padding_mode:pss -sigopt\ rsa_pss_saltlen:-1 -signature <sign> <mess>

EM = maskedDB || H || bc

Kiểm tra PSS padding:

Byte cuối là bc.

Từ EM rút ra các giá trị maskedDB có kích thước len(EM) -hLen-1, H.

maskH = MGF1(H, len(maskedDB), digest)

DB = maskedDB XOR maskH

DB = 00…01 || salt

Xóa padding của DB để xác định salt, đồng thời kiểm tra có 0x01 byte trong DB không.

Tính toán giá trị băm H’ = Hash(0x00 * 8 || Hash(data) || salt)

Kiểm tra xem H’ có bằng H không để trả về kết quả xác nhận.

Sau khi có được dữ liệu khôi phục thì đem so sánh với thông điệp đã cho. Nếu khớp thì xác nhận thành công,ngược lại xác nhận thất bại.(Thông điệp đã cho khi so sánh có thể bị băm)

4. Hướng dẫn sử dụng chương trình.

Thông tin mã nguồn

Ngôn ngữ lập trình python.

Các thư viện: base64, sys, cryptography,math, secrets, hashlib.

Chương trình sẽ tự động đọc cả 2 khóa nên cần tạo khóa trước khi bắt đầu.

Chương trình chỉ mới hổ trợ md:sha256 và mgf1:sha256. Nên mặc định khi gọi các hàm có sử dụng hàm băm, chương trình sẽ sử dụng thuật toán bắm sha256.

Demo

Tạo khóa bí mật và khóa công khai bằng OpenSSL:

$ openssl genpkey -out <priv.pem> -algorithm RSA -pkeyopt rsa_keygen_bits:<numbits>

$ openssl pkey -in <priv.pem> -out <pub.pem> -pubout

In khóa chương trình python đọc được:

$ python main.py -print priv

$ python main.py -print pub

**Lưu ý: cấu trúc được in ra giống với cấu trúc khóa trong SEQUENCE chứ khóa đã nói ở phần 1.*

Mã hóa bằng khóa công khai, padding mode: pkcs1 type2

$ python main.py -in <plain> -out <cipher> -encrypt

Kiểm tra lại bằng OpenSSL:

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

Mã hóa bằng khóa công khai, padding mode: oaep, md: sha256

$ python main.py -in <plain> -out <cipher> -encrypt -pkeyopt rsa_padding_mode: oaep

Kiểm tra lại bằng OpenSSL:

$ openssl pkeyutl -decrypt -in <cipher> -inkey <priv.pem> -out <plain.dec> -pkeyopt \
rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:sha256

Mã hóa bằng khóa công khai, padding mode: none

$ python main.py -in <plain> -out <cipher> -encrypt -pkeyopt rsa_padding_mode: none

Kiểm tra lại bằng OpenSSL:

$ openssl pkeyutl -decrypt -in <cipher> -inkey <priv.pem> -out <plain.dec> -pkeyopt \
rsa_padding_mode:none

Giải mã bằng khóa bí mật, padding mode: pkcs1 type2

Tạo bản mã bằng OpenSSL và giải mã bằng python:

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

$ python main.py -in <cipher> -out <plain.dec> -decrypt

Giải mã bằng khóa mật, padding mode: oaep, md: sha256

Tạo bản mã bằng OpenSSL và giải mã bằng python:

$ openssl pkeyutl -in <plain> -out <cipher> -inkey <pub.pem> -pubin -encrypt -pkeyopt\
rsa_padding_mode:oaep -pkeyopt rsa_oaep_md:digest

$ python main.py -in <cipher> -out <plain.dec> -decrypt -pkeyopt rsa_padding_mode:\
oaep

Giải mã bằng khóa bí mật, padding mode: none, cần cung cấp bản plain có kích thước len(n)

$ openssl pkeyutl -in <plain> -out <cipher> -inkey <pub.pem> -pubin -encrypt -pkeyopt\
rsa_padding_mode:none

$ python main.py -in <cipher> -out <plain.dec> -decrypt -pkeyopt rsa_padding_mode:\
none

Ký bằng khóa bí mật, padding mode: pkcs1 type1

$ python main.py -in <mess> -out <sign> -sign

Kiểm tra lại bằng OpenSSl:

$ openssl dgst -sha256 -verify <pub.pem> -signature <sign> <mess>

Ký bằng khóa bí mật, padding mode: pss, md: sha256, mgf1: sha256

$ python main.py -in <mess> -out <sign> -sign -pkeyopt rsa_padding_mode: pss

Kiểm tra bằng OpneSSL

$ openssl dgst -sha256 -verify <pub.pem> -sigopt rsa_padding_mode:pss -sigopt \
rsa_pss_saltlen:-1 -signature <sign> <plain>

Xác thực bằng khóa công khai, padding mode: pkcs1 type1

Tạo chữ ký bằng OpenSSL và xác thực bằng python:

$ openssl dgst -sha256 -sign <priv.pem> -signature <sign> <mess>

$ python main.py -in <sign> -out <mess> -verify

// Hàm sẽ lấy file out <mess> làm thông điệp cần xác thực.

Xác thực bằng khóa công khai, padding mode: pss, md: sha256, mgf1: sha256

Tạo chữ ký bằng OpenSSL và xác thực bằng python:

$ openssl dgst -sha256 -sign <priv.pem> -sigopt rsa_padding_mode:pss -sigopt \ rsa_pss_saltlen:-1 -out <sign> <mess>

$ python main.py -in <sign> -out <mess> -verify -pkeyopt rsa_padding_mode: pss

// Hàm sẽ lấy filem out <mess> làm thông điệp cần xác thực.

5. Nguồn tham khảo.

https://www.openssl.org

https://en.wikipedia.org

https://crypto.stackexchange.com

https://datatracker.ietf.org