Đại học Quốc gia Thành phố Hồ Chí Minh
Trường Đại học Công nghệ Thông tin (UIT)

BÁO CÁO BÀI TẬP

Môn học: NT219

Kỳ báo cáo: Buổi 03 (Session 03)

Tên chủ đề: chủ đề môn học

Ngày báo cáo: 10/04/2023

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT219.N22.ATCL

STT	Họ và tên	MSSV	Email
1	Võ Sỹ Minh	21521146	21521146@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá	Người đóng góp
1	Chậm lại và suy nghĩ 1	100%	
2	Bài tập 01	100%	
3	Bài tập 02	95%	
4	Bài tập 03	100%	
5	Bài tập 04	90%	
6	Bài tập 05	100%	
7	Bài tập 06	100%	
8	Benchmarks	100%	

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

1. **Kịch bản 01/Câu hỏi 01**

2. **Kịch bản 02**
   - Tài nguyên:
   - Mô tả/mục tiêu:
   - Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích)
3. **Kịch bản 03**
   - Tài nguyên:
   - Mô tả/mục tiêu:
   - Các bước thực hiện/ Phương pháp thực hiện (Ảnh chụp màn hình, có giải thích)

Sinh viên đọc kỹ yêu cầu trình bày bên dưới trang này

```
// Generate keys
AutoSeededRandomPool rng;

InvertibleRSAFunction parameters;
parameters.GenerateRandomWithKeySize( rng, 1024 );

RSA::PrivateKey privateKey( parameters );
RSA::PublicKey publicKey( parameters );
```

Chậm lại và suy nghĩ 1: Hàm GenerateRandomWithKeySize đang làm gì, InvertibleRSAFunction có nhiệm vụ gì trong việc tạo khoá?

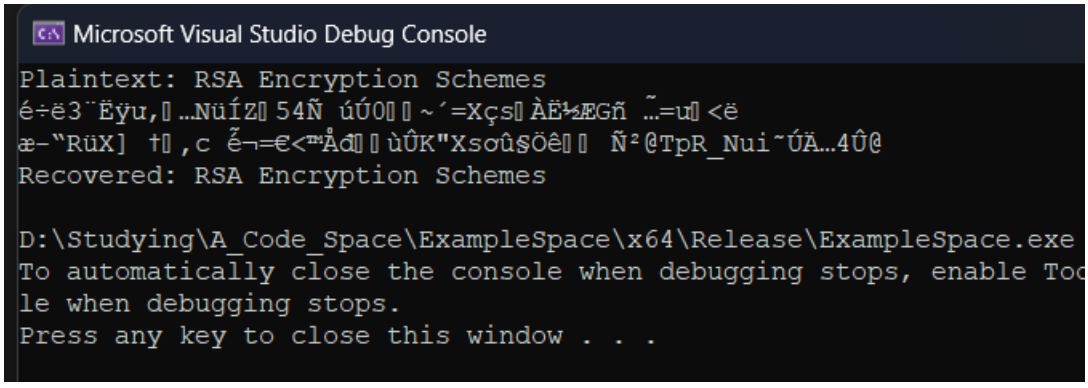
Hàm 'GenerateRandomWithKeySize(rng, keySize)' trong 'InvertibleRSAFunction' có nhiệm vụ tạo ra một cặp khóa RSA, bao gồm khóa public và khóa private, có độ dài là 'keySize' (đơn vị là bit). 'rng' là đối tượng động của lớp bộ sinh số ngẫu nhiên được sử dụng để tạo các giá trị ngẫu nhiên cần thiết cho việc tạo phần tử nguyên tố trong quá trình sinh khóa.

Trong khi 'InvertibleRSAFunction' định nghĩa hàm số để tạo khóa, lớp này không giữ trực tiếp các khóa. Thay vào đó, nó tạo ra các thành phần để tạo khóa public và private dựa trên những thông tin có thể được truyền từ các bộ sinh số ngẫu nhiên.

Ví dụ, khi chúng ta gọi 'GenerateRandomWithKeySize' để tạo một khối khóa RSA mới, 'InvertibleRSAFunction' sẽ tạo ra hai số nguyên tố ngẫu nhiên đủ lớn, tính tích của hai số nguyên này làm thành phần nằm trong khóa public, cùng với một số nguyên khác làm thành phần nằm trong khóa private. Quá trình này được thực hiện bằng cách sử dụng thuật toán chuẩn để tạo khóa RSA.

Bài tập 1: Sử dụng code mẫu sample_rsa.cpp được cung cấp, chỉnh sửa và mã hoá đoạn plaintext sau: "RSA Encryption Schemes". Kết quả xuất ra màn hình

Sau đây là màn hình và code về mã hoá và giải mã RSA bằng thư viện Crypto++.



```cpp
int main(int argc, char* argv[])
{
    // Generate keys
    AutoSeededRandomPool rng;

    InvertibleRSAFunction parameters;
    parameters.GenerateRandomWithKeySize(rng, 1024);

    RSA::PrivateKey privateKey(parameters);
    RSA::PublicKey publicKey(parameters);

    // Secret to protect
    string plaintext = "RSA Encryption Schemes";

    // Encrypt
    RSAES_OAEP_SHA_Encryptor encryptor(publicKey);

    size_t ecl = encryptor.CiphertextLength(plaintext.size());
    SecByteBlock ciphertext(ecl);

    encryptor.Encrypt(rng, (byte const*)plaintext.data(), plaintext.size(), ciphertext);

    // Decrypt
    RSAES_OAEP_SHA_Decryptor decryptor(privateKey);

    size_t dpl = decryptor.MaxPlaintextLength(ciphertext.size());
    SecByteBlock recovered(dpl);

    DecodingResult result = decryptor.Decrypt(rng, ciphertext, ciphertext.size(), recovered);

    assert(result.isValidCoding);

    recovered.resize(result.messageLength);

    // Verify
    string ciphertextStr((char*)ciphertext.data(), ciphertext.size());
    string recoveredStr((char*)recovered.data(), recovered.size());

    cout << "Plaintext: " << plaintext << endl;
    cout << "Ciphertext: " << ciphertextStr << endl;
    cout << "Recovered: " << recoveredStr << endl;

    return 0;
}
```

Bài tập 2: Sử dụng private key để mã hoá đoạn plaintext trên và giải mã bằng public key. Kết quả xuất ra màn hình.

Sử dụng private key để mã hóa và public key để giải mã là một phương pháp để triển khai chữ kỹ số. Sau đây là ví dụ **Sử dụng private key để mã hoá và và giải mã bằng public key.**

Kết quả cho ra nếu chữ ký sau decryption giống với trước khi encryption thì Signature on message verified. Còn không thì Message verification failed.

```
RSA::PrivateKey privateKey(parameters);
RSA::PublicKey publicKey(parameters);

// Message
string message = "RSA Encryption Schemes";

// Signer object
RSASS<PSS, SHA1>::Signer signer(privateKey);

// Create signature space
size_t length = signer.MaxSignatureLength();
SecByteBlock signature(length);

// Sign message
signer.SignMessage(rng, (const CryptoPP::byte*)message
    message.length(), signature);

// Verifier object
RSASS<PSS, SHA1>::Verifier verifier(publicKey);

// Verify
bool result = verifier.VerifyMessage((const CryptoPP::
    message.length(), signature, signature.size());
```

```
D:\Studying\A_Code_Space\CryptoLAB03\x64\Release\CryptoLAB03.exe
Signature on message verified
RSA Encryption Schemes
Press any key to continue . . .
```

Bài tập 3: Tương tự với quá trình mã hoá, phần này yêu cầu ciphertext được nhập từ file để phục vụ quá trình giải mã. Key được set cố định trong chương trình.

Sử dụng file .key để lưu trữ cố định các key và sử dụng các hàm Load để sử dụng

File ciphertext và cặp key sử dụng được đính kèm theo trong folder báo cáo

```
void Load(const string& filename, BufferedTransformation& bt)
{
    FileSource file(filename.c_str(), true /*pumpAll*/);
    file.TransferTo(bt);
    bt.MessageEnd();
}

void LoadPrivateKey(const string& filename, RSA::PrivateKey& key)
{
    ByteQueue queue;
    Load(filename, queue);
    key.Load(queue);
}

void LoadPublicKey(const string& filename, RSA::PublicKey& key)
{
    ByteQueue queue;
    Load(filename, queue);
    key.Load(queue);
}
```

```
AutoSeededRandomPool rng;

// Load key from files
RSA::PrivateKey privateKey;
LoadPrivateKey("rsa-private.key", privateKey);

// Decryption
string cipherHex, cipher, recovered; // ciphertext
cipherHex.clear();
InputCiphertextFromFile(cipherHex);

/* Decrypt */

    // Hex decode the input cipher
cipher.clear();
StringSource(cipherHex, true,
    new HexDecoder(new StringSink(cipher)));

RSAES_OAEP_SHA_Decryptor d(privateKey);
recovered.clear();
StringSource(cipher, true,
    new PK_DecryptorFilter(rng, d,
        new StringSink(recovered)
    ) // PK_EncryptorFilter
); // StringSource

wcout << "Recover text: " << ConvertStringtoWstring
```

```
D:\Studying\A_Code_Space\CryptoLAB03\x64\Releas...
Recover text: Mê mẩn mùa hoa bung nở đẹp như tranh vẽ ở
 Pakistan. Mùa xuân ở thung lũng Hunza ở Pakistan luôn
làm người ta mê mẩn bởi sắc anh đào tuyệt đẹp, bao quan
h là những đỉnh núi tuyết sừng sững giữa trời xanh.

Press any key to continue . . .
```

Bài tập 4: plaintext hỗ trợ đầu vào bao gồm các kí tự thuộc UTF-16

```
// Generate keys
AutoSeededRandomPool rng;

// Load key from files
RSA::PublicKey publicKey;
LoadPublicKey("rsa-public.key", publicKey);

// Input Plaintext
string plain, cipher;
InputPlaintextFromFile(plain);
string encoded = "";

for (int i = 0; i < 1000; i++)
{
    // Setup publicKey for Encryption
    RSAES_OAEP_SHA_Encryptor e(publicKey); //
    cipher.clear();
    // Create a pipelining for encryption
    StringSource(plain, true,
        new PK_EncryptorFilter(rng, e,
            new StringSink(cipher)
        ) // PK_EncryptorFilter
    ); // StringSource

    // Write cipher as hexa code
    encoded.clear();
    StringSource(cipher, true,
```

```
D:\Studying\A_Code_Space\CryptoLAB03\x64\Release\CryptoLAB03.exe
Plaintext:Rộn ràng Tết cổ truyền của đồng bào Khmer tại Bạc Liêu. Tết cổ truyền Chôl-chnăm
-thmây tiếp thêm động lực để bà con Khmer hướng đến những điều tốt đẹp trong năm mới, góp
phần xây dựng quê hương Bạc Liêu ngày càng phát triển.
* Ciphertext: 04D9A4E2F2497FD1CA13542770070125A53FA2DF7787FE1FF083C0B553E3478C0777D421DF6B
EFD516A8220AC42A10CDDD27D1C9778C20A504A868F1FEF2E284B88D1FE2AD95ED6CDF63154FF1FE206807E0AC
AFAD0E5411C21E823AF4AF51E85537B23A0D3380C5C7EAD89EC4C78204979A594B46DC6317454418B1DFDDCFD2
088A7B88EA80647D94C5B294D0BDF14B26072C0C24C2C0AD7089B030C7CFFE7797109173A67927E2C0DFBEC3E8
66BD410CD813DF8BA3C0C0C003B731F38354F42329194D50D6789B54CAC8AF1783A76F2C942069C29882D4825E
94422F2E23035552CB79C799AC0F20FC67919C5D6EBEB900696F38D9D9C9A856F9403346DB553CC0762C27BDA9
931077E85BA61FB4FC28B5852F8E586D73EEBD5535667C4E284B480AA65AD8DD2719C0275CF005A45B41D14EE5
73745C2B94655DFF97A46466713025A14FF5BE3C2D043C60C4E3A421253BD269D79946EE26F7198D1091D91112
D416B7AB9E02CEFDEBD7E45D5C0A6F1EAEC2F0A275F5217E6F4E6AA32A60F2
Press any key to continue . . .
```

```cpp
/* convert string to wstring */
wstring ConvertStringtoWstring(const std::string& str)
{
    // Declare an object `converterX` of the `wstring_convert` class
    using convert_typeX = codecvt_utf8<wchar_t>;
    wstring_convert<convert_typeX, wchar_t> converterX;
    // Return the result of the conversion from a byte string to a wide string.
    return converterX.from_bytes(str);
}

/* convert wstring to string */
string ConvertWstringToString(const std::wstring& wstr)
{
    // Declare an object `converterX` of the `wstring_convert` class
    using convert_typeX = codecvt_utf8<wchar_t>;
    wstring_convert<convert_typeX, wchar_t> converterX;
    // Return the result of the conversion from a wide-character encoding to a byte string.
    return converterX.to_bytes(wstr);
}

void InputPlaintextFromFile(string& plain)
{
    FileSource file("plain-text.txt", true, new StringSink(plain));
    wcout << L"Plaintext:" << ConvertStringtoWstring(plain) << endl;
}
```

Bài tập 5: Đầu vào plaintext được nhập thủ công vào chương trình

```cpp
// Load key from files
RSA::PublicKey publicKey;
LoadPublicKey("rsa-public.key", publicKey);

// Input Plaintext
string plain, cipher;
wstring wplain;
wcout << "Input Plaintext: ";
fflush(stdin);
getline(wcin, wplain);
plain = ConvertWstringToString(wplain);
string encoded = "";


// Setup publicKey for Encryption
RSAES_OAEP_SHA_Encryptor e(publicKey); // RSAES
cipher.clear();
// Create a pipelining for encryption
StringSource(plain, true,
    new PK_EncryptorFilter(rng, e,
        new StringSink(cipher)
    ) // PK_EncryptorFilter
); // StringSource
```

```
D:\Studying\A_Code_Space\CryptoLAB03\x64\Release\Crypto...
Input Plaintext: Đầu vào plain text được nhập thủ công vào ch
ương trình.
* Ciphertext: 48D142811DC5068CD99272C271C16499F47EF35D34DBC87
8A1017ECC5DFE020153CD620A109E71AF4CB8A7191207F978E0EACD916AF6
3BA6972A9E18BEB53901C0EC53D6B7C83C6C67A139A3AF470CA4D2F7E45A5
A39B982C4309F4F50550C58421866DFFC2485DDAA560DAFF3398CB18115EB
87E1DB781DC5F63C770018AE6F71F5EC98B3AA50D664989091204995A290A
CCBF11849AFD0CE673B890073F08C9E5D865D2AEF2E7B0F8BB18617E4FAE5
3E34A4C924C6A4354D7A2B0A5E6861B5576B52219B83210F0BE81100D1E4E
B3C12033A393974B36F61D56D9420FBCC19DC9B90035C64295F705B625BA1
3B4D0D676790DD875BFC953D2D055FE13273EDC1F83E22228B549F949F9EC
A827607979A84BC68A8B30FD4CEDC679B6784FA533E938D99BBF1643C97E7
EA415E78D1B7F598ADCA07D4ED2DCB77B5C3749FA567B8195CD91840997ED
5B03D7EC28AF7DEB8328D3DAE22626F2043093DFE2899BBBBB9B7F04922AD
373ED62A2F3242235FE4277CAAFB37354958B9D62394CB4ACD
Press any key to continue . . .
```

Kiểm thử: sử dụng Bài tập 3 để kiểm tra

```
C:\Users\minhs\OneDrive\Máy tính\BT3_CipherTextFromFile\CipherTextFromFile.exe
Recover text: Đầu vào plain text được nhập thủ công vào chương trình.
Press any key to continue . . .
```

Bài tập 6: Key được load lên từ file. Giá trị key >= 3072 bits.

```
void Load(const string& filename, BufferedTransformation& bt)
{
    FileSource file(filename.c_str(), true /*pumpAll*/);
    file.TransferTo(bt);
    bt.MessageEnd();
}

void LoadPrivateKey(const string& filename, RSA::PrivateKey& key)
{
    ByteQueue queue;
    Load(filename, queue);
    key.Load(queue);
}

void LoadPublicKey(const string& filename, RSA::PublicKey& key)
{
    ByteQueue queue;
    Load(filename, queue);
    key.Load(queue);
}
```

Sử dụng các hàm Load để lấy Keys từ file.

Bài tập luyện tập 1: Đánh giá hiệu năng của thuật toán RSA

1. Trường hợp 1: Dữ liệu dạng utf-16

2. Trường hợp 2: Dữ liệu lớn hơn 100 MB

```
// Message
string message = "";
ifstream file;
file.open("100MB.txt", ios::out | ios::app | ios::binary);
if (file.is_open())
{
    string line = "";
    while (getline(file, line))
    {
        message += line + "\n";
    }
    file.close();
}

// utf-16 message
wstring wmessage = L"RSA mật mã học LAB03D";
string utf16_message = wstring_to_string(wmessage);

// Signer object
RSASS<PSS, SHA1>::Signer signer(privateKey);

// Create signature space
size_t length = signer.MaxSignatureLength();
SecByteBlock signature(length);
```

```
Select Microsoft Visual Studio Debug Console
Signature on message verified
UTF-16 case's performance:
8437.5 Cycles per byte
0.305176 MiB per second
Signature on message verified
100MB file input case's performance:
3.41056 Cycles per byte
754.984 MiB per second

D:\Studying\A_Code_Space\CryptoLAB03\x64\Release\CryptoLAB03.exe
To automatically close the console when debugging stops, enable
le when debugging stops.
Press any key to close this window . . .
```

- Với trường hợp UTF-16, thuật toán RSA đã thực hiện trên mỗi byte dữ liệu trong khoảng 8437.5 chu kỳ tính toán, và tốc độ truyền dữ liệu đạt 0.305176 MiB mỗi giây.
- Với trường hợp100MB file, thuật toán RSA đã thực hiện trên mỗi byte dữ liệu trong khoảng 3.41056 chu kỳ tính toán, và tốc độ truyền dữ liệu đạt 754.984 MiB mỗi giây.

Ta thấy RSA có nhược điểm về hiệu suất, đặc biệt là khi được sử dụng để mã hóa các tập tin lớn hoặc trong các ứng dụng yêu cầu tốc độ xử lý cao.

YÊU CẦU CHUNG

– Sinh viên tìm hiểu và thực hành theo hướng dẫn.

– Nộp báo cáo kết quả chi tiết những việc (**Report**) bạn đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả (nếu có); giải thích cho quan sát (nếu có).

– Sinh viên báo cáo kết quả thực hiện và nộp bài.

Báo cáo:

– File .PDF. Tập trung vào nội dung, không mô tả lý thuyết.

– Nội dung trình bày bằng Font chữ Times New Romans/ hoặc font chữ của mẫu báo cáo này (UTM Neo Sans Intel/UTM Viet Sach)– cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.

– Đặt tên theo định dạng: [Mã lớp]-SessionX_GroupY. (trong đó X là Thứ tự buổi Thực hành, Y là số thứ tự Nhóm Thực hành/Tên Cá nhân đã đăng ký với GV).

*Ví dụ: [*NT101.K11.ANTT*]-Session1_Group3.*

– Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.

– Không đặt tên đúng định dạng – yêu cầu, sẽ **KHÔNG** chấm điểm.

– Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Đánh giá: Sinh viên hiểu và tự thực hiện. Khuyến khích:

– Chuẩn bị tốt.

– Có nội dung mở rộng, ứng dụng trong kịch bản/câu hỏi phức tạp hơn, có đóng góp xây dựng.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT