网络安全技术大作业-RSA

杨俊龙　521010910010

2024 年 6 月 18 日

目录

1 简介

本项目为对 RSA 的 C++ 实现，包含了 Textbook-RSA 的实现、CCA2 攻击的实现以及 OAEP 填充的实现。本工作使用了 Gmp 大数运算库来处理大整数运算，使用 Crypto++ 库来进行 AES 加解密和 Hash 函数运算。相关代码和运行方法详见 https://github.com/Flash-sheep/RSA-cpp/tree/main。

2 Task 1

2.1 任务目标

任务一为实现一个 textbook RSA 算法，任务目标为：

1. 生成指定长度的 RSA 密钥对
2. 使用公钥加密明文
3. 使用私钥解密密文

2.2 算法实现

数学函数定义 在 RSA 算法中，需要使用的数学函数有模指运算、逆元运算、素数生成算法（miller rabin 素性校验）、最大公因数计算函数，均在"Crypto.h" 中进行了定义。其中由于 RSA 密钥的长度一般为 1024bit，故需要采用处理大整数的库。在本工作中采用了 GMPXX 库进行大整数运算。算法的具体实现内容不在报告中详述，详见代码部分，具体算法的实现方式均参考网络安全技术课程的 ppt。

RSA 操作封装 在"RSAcipher.h" 中包含了对 RSA 类型以及公钥和私钥类型的定义。在 RSAcipher 类型中，定义了多种加密和解密方法，分别针对单个整数的加密解密、任意长度字符数组的加密解密以及在不同填充方式下的加密和解密。其中针对任意长度字符数组的加密解密，需要对明文和密文进行分组。在本工作中的分组策略如下：对明文按照安全参数 n 的 bit 数进行分组，每个分组在加密后采用 0 填充的方式输出为 2n 位的密文分组；在解密过程中，将密文按照 2n 位进行分组解密，0 填充会在转换过程中被自动忽略。

RSA 操作使用流程 首先使用`RSAcipher rsa(n)`来初始化 rsa，其中 n 为指定的密钥长度。然后可以通过`rsa.genKey()`方法生成随机的 RSA 参数并使用`rsa.printParams()`方法来将参数写入本地文件。或者使用`rsa.loadParams()`方法可以直接从本地加载之前生成的 RSA 参数。接下来使用`rsa.encrypt()`方法来对明文进行加密，该方法获取三个输入，依次为明文、输出进制和填充方式。目前输出进制默认为 16 进制（暂不支

图 1: 任务一结果展示

持其他进制表示），填充方式默认为 0 填充，使用 1 可以进行 OAEP 填充。最后使用`rsa.decrypt()`方法来对密文进行解密，该方法获取三个输入，依次为密文、输入进制和填充方式。参数与加密方法需要保持一致。

2.3 结果展示

在 main() 函数中执行 RSA() 函数可以进行任务一的测试并查看输出结果，如图1所示为理想的输出结果。在运行过程中，与 rsa 相关的参数会写入本地文件中可供查看，末尾的输出 1 代表明文与解密后的明文匹配，解密成功。

3 Task 2

3.1 任务目标

任务二为实现对 textbook RSA 的 CCA2 攻击，具体攻击方式参考Knockel et al. [2018] 第 4 章的实现。在该任务中，需要对如图2所示的攻击场景进行攻击。攻击者需要获取 AES 密钥并解密双方的 WUP 信息。任务目标如下：

1. 设计 WUP 格式、服务端和客户端的通信模型
2. 生成一个历史消息，包含 RSA 加密的 AES 密钥和 AES 加密的请求
3. 发动攻击来获取从历史消息中获取原始的 AES 密钥

3.2 算法实现

服务端客户端通信模型 服务端和客户端的实现在”main.cpp” 中，均包含了使用 Crypto++ 库实现的 AES 加解密函数，服务端包含`setSession`方法，获取客户端提供的加密的 AES 密钥以及加密的 WUP 请求，首先使用私钥解密 AES 密钥，然后使用 AES 密钥解密 WUP 请求，并判断解密的 WUP 是否满足约定好的 WUP 格式（在本工作中，要求一

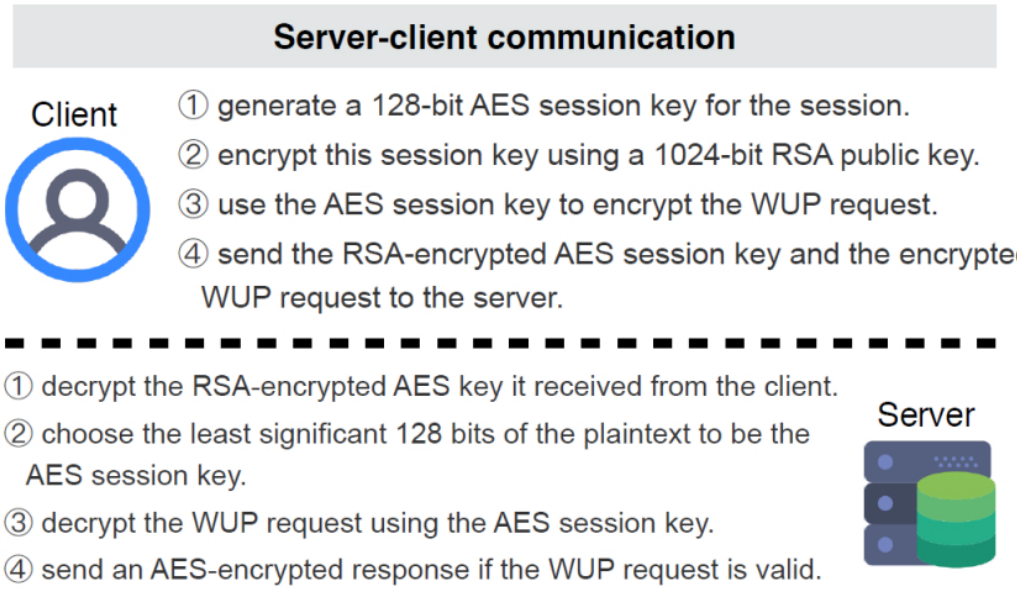

图 2: CCA2 攻击场景

个合格的 WUP 请求的开头为”Hello”。)，并返回一个验证信息。对于客户端而言，能够生成加密的 AES 密钥和使用 AES 密钥加密的 WUP 报文。

CCA2 攻击 对于攻击者而言，能获取的信息包含：RSA 公钥、RSA 加密的 AES 密钥、AES 加密的 WUP 请求。具体的攻击方式参考了Knockel et al. [2018] 中的方法，从 0 到 127 位依次对 AES 密钥进行攻击，最后输出结果。算法的实现细节和原理不作详述。

3.3 结果展示

运行”main.cpp” 中的 CCA2() 函数即可测试 CCA2 攻击流程，如图3为初始化阶段和服务端和客户端的通信阶段，生成历史信息文件，如图4中，攻击者获取了历史信息，并依次对 AES 密钥的比特进行攻击，最后成功输出了正确的密钥，并解密出了客户端此前发送的 WUP 请求”Hello Server”。

4 Task 3

4.1 任务目标

任务三为实现 OAEP 填充的 RSA 算法，并利用 OAEP 填充来抵御任务二中的 CCA2 攻击。

```
AES_Key.txt cleared successfully.
WUP_Requst.txt cleared successfully.
AES_Encrypted_WUP.txt cleared successfully.
History_messages.txt cleared successfully.
RSAcipher created
Private key and public key created
-----initiate server client and attacker-------
Server initiated
Client initiated
Client initiated
-----communication between client and server-------
AES_Key.txtwritten
WUP_Requst.txtwritten
AES_Encrypted_WUP.txtwritten
History_messages.txtwritten
History_messages.txtwritten
Has connection to server
-----attacker begin attacking using CCA2-------
attacker has rsa encrypted aes key and aes encrypted wup
0123456789012345
trying bit 0
128
bit 0 is 1
trying bit 1
128
```

图 3: 任务二攻击流程 1

```
128
bit 120 is 0
trying bit 121
128
bit 121 is 0
trying bit 122
128
bit 122 is 0
trying bit 123
128
bit 123 is 0
trying bit 124
128
bit 124 is 1
trying bit 125
128
bit 125 is 1
trying bit 126
128
bit 126 is 0
trying bit 127
128
bit 127 is 0
guess key is 0123456789012345
Hello Server
```

图 4: 任务二攻击流程 2

4.2 算法实现

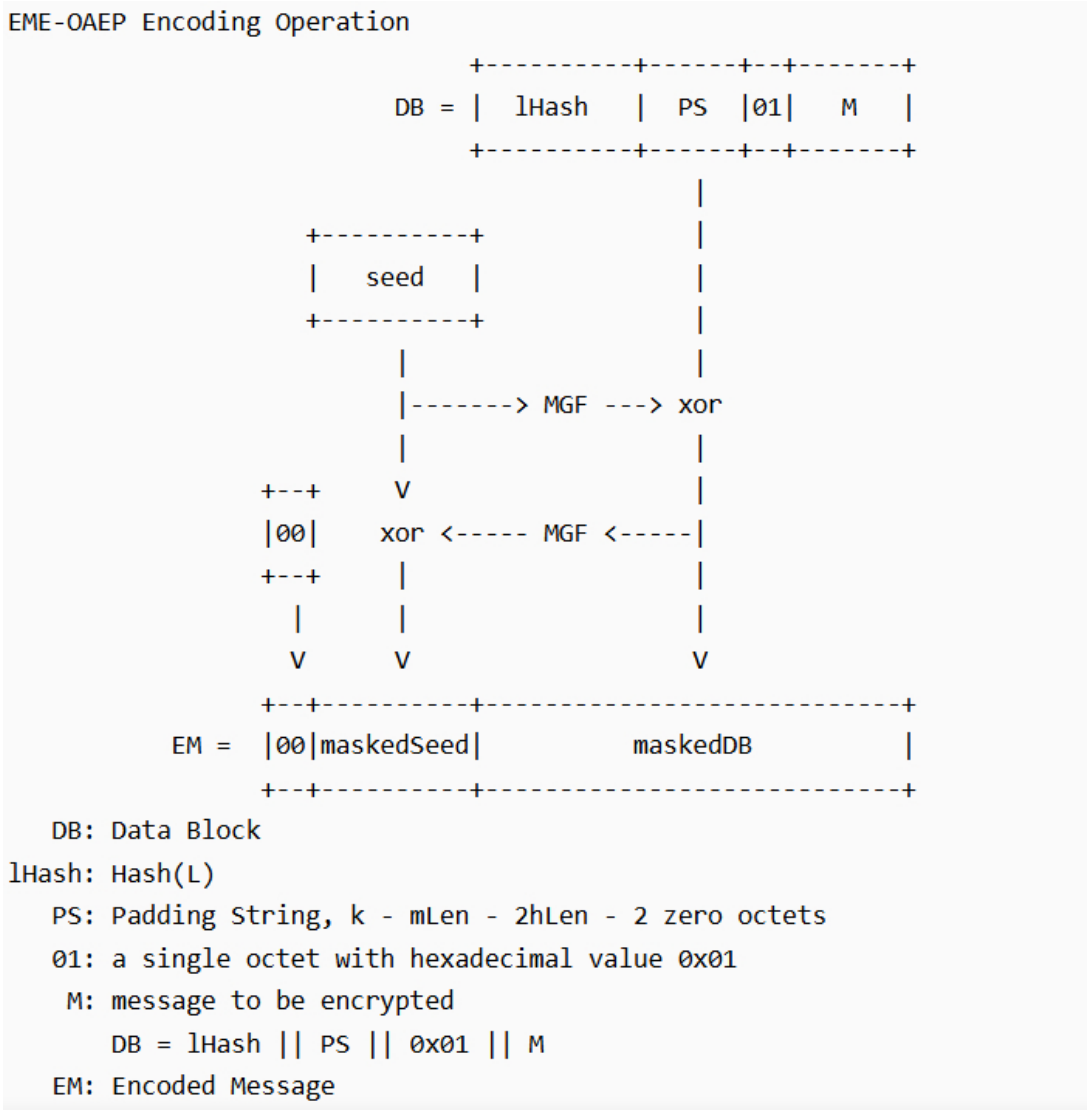

图 5: OAEP 填充

OAEP 编码 如图5所示为本工作采用的 OAEP 填充方式，具体流程如下：

1. 选定一个字符串，计算其哈希值为 lHash
2. 对明文进行分组，明文消息长度最长为 k-2hLen-2 字节，其中 k 为编码后的总消息长度，一般为密钥长度的 2 倍。
3. 构造 DB=lHash+PS+01+M，其中 PS 为 0 填充，DB 总长度为 k-1-hLen
4. 生成随机种子 seed，并对其用 MGF 函数进行扩展然后与 DB 异或得到 maskedDB
5. 对 DB 使用 MGF 函数压缩然后与 seed 异或得到 maskedSeed

6. 生成编码 EM=00+maskedSeed+maskedDB

具体实现详见”Crypto.cpp” 中的`oaepEncode()`函数。

OAEP 解码 OAEP 解码的流程如下：

1. 分割 EM 消息，基于约定好的 hLen 对消息进行分割得到 maskedSeed 和 maskedDB
2. 使用 maskedDB 和 maskedSeed 反推得到 seed 和 DB
3. 剥离 DB 头部的 lHash，然后根据填充字符串和结束标记 0x01 找到 M 的起始位置从而解码。

具体实现详见”Crypto.cpp” 中的`oaepDecode()`函数。

MGF 实现 MGF 实现为对一个四字节的字符串与原始字符串拼接进行哈希，得到输出结果。如果输出结果长度不足，则对四字节字符串加 1 然后重复操作；若输出结果长度超出，则对字符串进行裁剪得到最终结果。具体实现详见”Crypto.cpp” 中的`generateMGF1()`函数。

4.3 结果展示

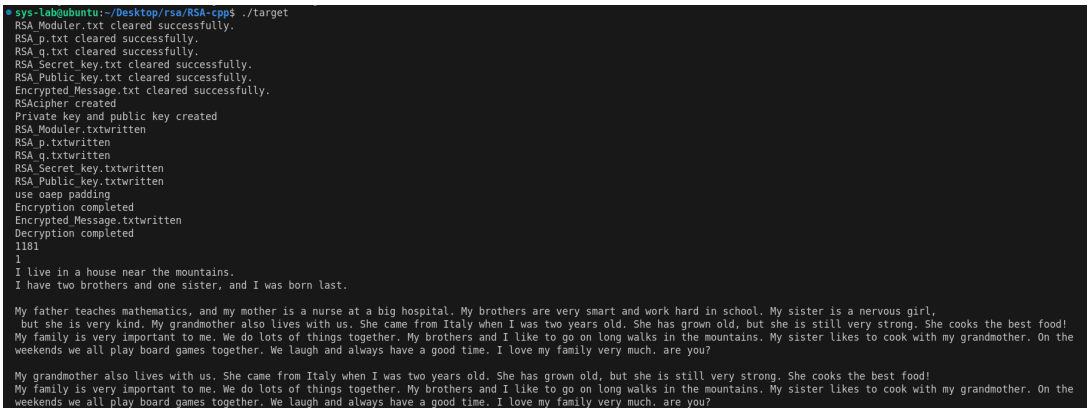

图 6: RSA-OAEP 执行效果

运行”main.cpp” 中的 RSA_OAEP() 函数即可测试使用 OAEP 填充的 RSA 加解密效果。如图6为使用 rsa-oaep 进行加解密的效果展示。

运行”main.cpp” 中的 CCA2_OAEP() 函数可以测试使用 CCA2 攻击方式对 RSA-OAEP 的攻击效果。如图7为初始化阶段，与任务二相同。如图8阶段为攻击者尝试每一位进行攻击，可以看到由于 OAEP 填充的存在，攻击者的攻击策略失效，无法计算出 AES 密钥。

```
sys-lab@ubuntu:~/Desktop/rsa$ ./target
AES_Key.txt cleared successfully.
WUP_Requst.txt cleared successfully.
AES_Encrypted_WUP.txt cleared successfully.
History_messages.txt cleared successfully.
RSAcipher created
Private key and public key created
-----initiate server client and attacker-------
Server initiated
Client initiated
Client initiated
-----communication between client and server-------
generate hash
AES_Key.txtwritten
WUP_Requst.txtwritten
AES_Encrypted_WUP.txtwritten
History_messages.txtwritten
History_messages.txtwritten
Has connection to server
-----attacker begin attacking using CCA2-------
attacker has rsa encrypted aes key and aes encrypted wup
0123456789012345
trying bit 0
```

图 7: 任务三攻击流程 1

```
the message is not properly decoded
bit 122 is 0
trying bit 123
128
bit 123 is 0
trying bit 124
128
bit 124 is 0
trying bit 125
128
the message is not properly decoded
bit 125 is 0
trying bit 126
128
the message is not properly decoded
bit 126 is 0
trying bit 127
128
bit 127 is 0
guess key is
terminate called after throwing an instance of 'CryptoPP::InvalidKeyLength
  what():  AES/CBC: 0 is not a valid key length
Aborted (core dumped)
```

图 8: 任务三攻击流程 2

4.4 结果讨论

RSA-OAEP 填充相比于 textbook RSA，一方面通过随机种子引入了随机数，从而使得 RSA 加密变成了不确定算法，这增加了其抵御 CPA 攻击的能力。另一方面通过 MGF 函数和异或的方式，将明文的消息糅杂了起来，从而避免攻击者可以对密文进行加工从而获取明文的部分信息来实施攻击。在 RSA-OAEP 场景下，想要获取明文的任意信息，由于 MGF 内部哈希函数的单向性，无法通过密文的性质去推理明文，从而该填充方式可以抵御 CCA2 攻击。

5 Bibliography

J. Knockel, T. Ristenpart, and J. Crandall. When textbook rsa is used to protect the privacy of hundreds of millions of users, 2018.