密码学实验 5 实验报告

13061082 陈逸伦

June 12, 2015

引入 RSA-OAEP 的原因

裸 RSA 算法容易导致密文遭受选择密文攻击 (CCA)，如下图

选择密文 CCA 攻击 RSA

1、 利用如下性质：

E(PU,M1)*E(PU,M2)=E(PU,[M1*M2])

选择一个明文，运用目标对象的公钥加密，然后用目标对象的私钥解密取回密文。

2、 解密函数 $C = M^e \bmod n$

1) 计算 $X = (C * 2^e) \bmod n$

2) 将 X 作为选择明文提交，收到解密后的 $Y = X^d \bmod n$

基于以下内容：

X= (C mod n) *(2^e mod n)=(M^e mod n)*(2^e mod n)=(2M)^e mod n

推出 Y=(2M) mod n,从而得出 M,得出原文

图 1: CCA 攻击举例

也已经说明简单地给信息填充伪数据 (填充位) 不足以抵抗这样的攻击，解决这种攻击的办法就是应用称为最优非对称加密填充 (OAEP) 的过程。

RSA-OAEP 算法原理

OAEP 满足以下两个特性:

1. 添加了随机性的元素，可以用来将原有的确定性的加密方案（如传统的 RSA 加密方案）转换成为一种可能性的方案。

2. 防止对于密文的部分解密（或者其它的信息泄露），通过确保对手在不能反转陷门单向转换的情况下，无法复原明文的任何部分。

RSA-OAEP 算法流程

我所完成的是根据老师提供的 PPT 的方法。

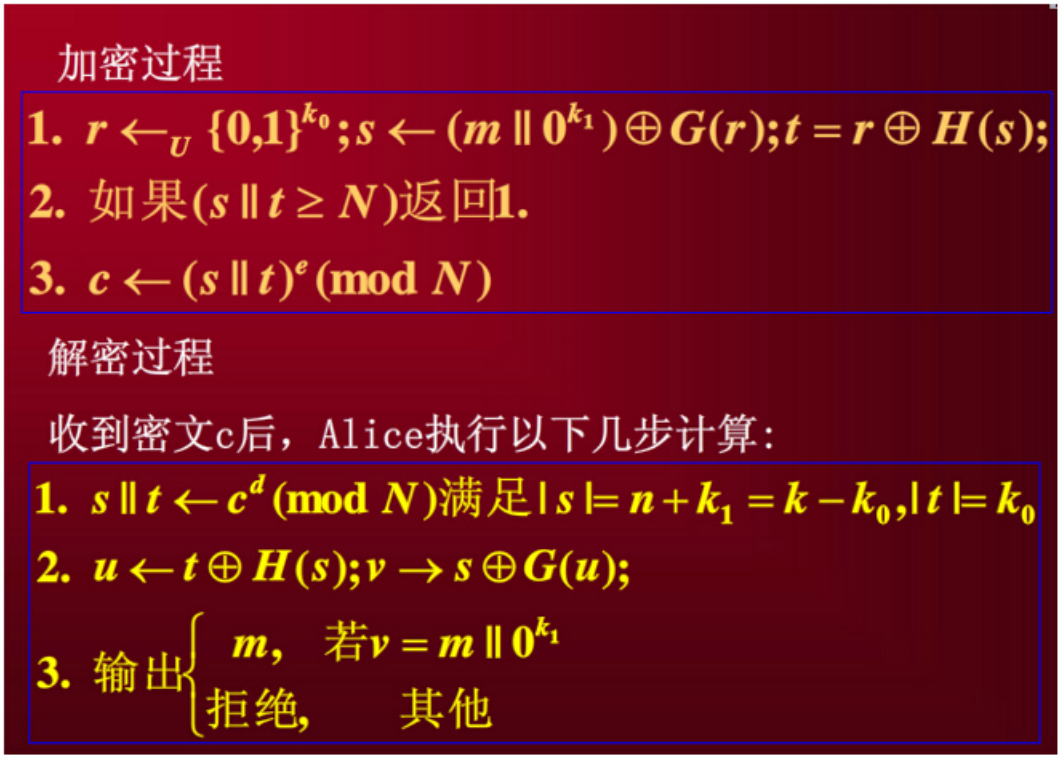

图 2: RSA-OAEP 算法流程

这里有一个小错误，就是明文进行 padding 时应该用"$1[0^{k_1-1}]$" 而不是"0^{k_1}"。若全是 0 则无法分辨出来明文。

代码说明 每次运行 RSA_OAEP.sage 后，会初始化密钥 (随机生成)，后对输入明文进行加密后解密操作。并且将公私钥 e, d, N 保存到文件 RSA_data.txt 中，密文保存在 CipherText.txt 中。

加密函数 RSAEncrypt(PlainText)

解密函数 RSADecrypt(CipherText)

测试样例

```
sage: load("RSA_OAEP.sage")
/***  RSA-OAEP  ***/
Plain Text (hex): 123456789abcdef
>>> Init Keys
Random P :  8f4ecda6bd04dcc809657c9e90df2efb8132a66e6222f6
5ee964aaaa3069d4677714a9c70354d864f8724f6a2ef99186a8e114c8
3bc2efb79865be8b6a2b3f9336de77d748048180aaa2d33ba6c34bb54d
d4d2d32dde51dcee7b3ad1b16cfbd6e5b96a4d6c9d2e038991d06592e7
a546088e0772a92fd59950faf5a4dafe981b
Random Q :  b99cd8e911ad4a1f0aed7f6491e6e49601dbf277f779c4
9f59a2865cf01777716e89b39c2e14fc3363869bad691a9a4f924c10fa
2512274b44deb10d69349752211aa83be57fb17cccf14323dd479db32c
a96760abbda42f1034bba92f20960a4b67088d54f166f7814a4d7d84d3
8e76fc7056e6484259849a04f80821af793d
Mod N :  67e7c00fc656cc2870bbfaf8c8468b9496b61a1d94418140a
7c878d9d5fea2e67883d379b56deab24d8c4339cddb4c3db004e359867
6684cbeb2bfab57206e4eb1e48dec53736df0d7ca4b4342a2d21aa2be2
0497b6de7493194c21028db0a24aa8e544199ec05dfce85bab9b1a3cbe
560e1b9164b36bbebdee9164ef21fde8d8f3fbe12ee6e189b1f03330ff
395572887a1737636edd9b9e2ed6eb3def2a3bad12a6b42b66ad7e73ef
d779e9004907e36985f6ef298e4baf1c01a3da1c7dabf0519137ecfe09
da5b07aa0cab0c1ab00319fe7be62eb772df5c563cd431d4b85a477f6a
da0ad5986714b0548c567dbbd569ffd4498d7d7bb2f8ee247fa04016f
Cipher Key :  0x32c60f9c0897c29a5e4c37bdf4323719b3dc2f360c
4023308e11eb07394e62caf2d3ff0ac7a6bdc7952c2785679ab3c4b93c
71431c546fd81923dcfd8c63241be9da7eb828185912bee1d1e7f80278
186bd3f531785524264efa6456545feecec70ae0f961ccf7d601d94412
dc3743e820e031e7a6949c99a2fd468b2e16d85802c40262ace64fa5fc
746b824e6b8b7f939c57b79b11668beaba4508848998f87aee071b7be8
6c4d4ea5380c4796bf52967b9eed1f911d7427c96955ee6dfa199327de
e72c0b33a2553e49aa72d87062192b54aaa75e729498e9e4b5f4e18b75
4835877d74515c3e64be7042dc2fdf20c140eab115e277c3c24154a493
1fd029L
Public Key :  490f6b99ac8ff2cb77ec4ecb4902d9675146954f9513
9a04239625db83b9c2747138d2b7ebc925b61bf1c74d9c3643b825f4fb
```

```
3bd7292cff8716ca64d0a9e3d2d60c0515edf9fd3b3e1a6fd6845dd281
5eb73138fbefd6fea2fc9290f96c0812dcd4f65c26d227abb640abd85c
8b862a3e5e513f8788f0b7b923692a964ad784bb0744f25f926fc101b1
268cfa4862640976bb9def1f2a93bbcaaee497132f40f64c0010c383c2
ed68bc6b3463b864073c87f3ca49c0abe0a3e701135223acb7d47df4c4
61794df71f06e6ec7dbf55e04cd8ff6556902c8fe3b55aa9d24910abd1
c244c05e73b6a4650abab58227a0a2ecbc926b7448f207c6cc3d4ec35c
7f99
k : 7ff
k0 : 64
k1 : 64
---------- Initialize Finished ----------
>>> Encrypt
Cipher Text : 32e3770ce6cc4ad56313c0f0630678f2febd198ce1f
278a778c4887c58bea84ce95883323d675ca84f216cf49a019fd22f2b8
69b4e1c274d3524312450209b08a27232f1e435c8c74561e1501332018
c31239eccd76d6acafc627d8555613fc597bf2973df16ae181f6bc1d81
a6d6f4c8c0cd08b1103116b3a63e0837de296a869d7e3a32b54cfccba1
6563996f643471c9df61e2756185fe352fd7f2c9a4d694b6f85cc7c620
3d6f990236e5d1f192d546d12c8c988f45180bd32aeb9411263ab652a1
f5b3cdb82c6fc399a417221f808d99161c9583d0ab95c7554291c61bf0
e4fe1531b585874883c37b76e19ca9dbaca365687813de25af890ce7cf
a8029
>>> Decrypt
Plain Text : 123456789abcdef
```