CS 4823: Homework #9

Due on April 6, 2018

Christopher Tse

Problem 1

Suppose a RSA public key is:

n = 125953175678398351570149977764211035679420156938429586850000579961775054888014711050952194404928504160243324417202380464659083542772305519159214463831847643286738542961736012
1

e = 65537

(a) What is the cipher text if you encrypt your student ID number using the textbook RSA algorithm?

Solution

Using Sage:

```
n = 1259531756783983515701[...] # Too long to show
e = 65537
m = 12971666

R = Integers(n)
print R(m)**e
```

The output we get is 99009315352242465786715311574012216462850619187795282901162931332508387277509348357744985081467310536849937550865181130385731925177677382233084021205368085629126390713138219

(b) Explain why the textbook RSA is not safe for encrypting student ID numbers. Is the attack ciphertext only, known plaintext, chosen plaintext, or chosen ciphertext? How can you improve the security of the textbook RSA?

Solution

This is not a safe method since this is using the public key for encryption. Since the public key is made known to the attacker, they can use it to encrypt their own numbers to make it a **known plaintext** attack.

To increase the security, we can do several things. The first measure we can take is to choose a large enough value of e such that m^e is never strictly smaller than the RSA modulus. We can also use techniques such as padding the plaintext with randomized values.

Problem 2

Examine the certificates of your browser, and find the RSA public key n and e (in decimal) for https://www.google.com.

Solution

Using OpenSSL we can obtain the public key:

```
openssl s_client -connet www.google.com:443 | openssl x509 -text -pubkey
```


-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApBZZku2BwanibJycWr3R
3gXK7D1f+js+YgU9VRSJ5rBfGpQs479pAUWGf4GDSvhWpBs++guBQuW7/QUuz1/G
+AmJZspNXl5use14uxWDROu88YTBgwDoPoNAwR9RKk8JWkDroLCVnOHf7NGnrUU+
umuuV/3EWj/zaBAi3mnhVjE41pjPjc/sA4LLyOo2SSdD3sgJy/iizTHtN1cjeCo+
QDqvQ/rvCYqvhJuQDYS9J8uMZli+zyO2C+qN62rDDUVtxVGWZLS0chZTbS4qSIbZ
QMz6ssXOAQd8PrsCj/xzOIyoPndDsW8bw3QmYCTMOsHgzR4RQbRfwHIXlgJWsraF
5QIDAQAB
-----END PUBLIC KEY-----

Then using the PyCrypto package, we can decode the n and e values from the public key:

```python
from Crypto.PublicKey import RSA
key_encoded='''-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCdZGziIrJOlRomzh7M9qzo4ibw
QmwORcVDI0dsfUICLUVRdUN+MJ8ELd55NKsfYy4dZodWX7AmdNO2zm1Gk5V5i2Vw
GVWE2O5u7DhtRe85W1oR9WTsMact5wuqU6okJd2GKrEGotgd9iuAJm90N6TDeDZ4
KHEvVEE1yTyvrxQgkwIDAQAB
-----END PUBLIC KEY-----'''


pubkey = RSA.importKey(key_encoded)
print(pubkey.n)
print(pubkey.e)
```

We find that the n and e are as follows:

n = 110524622184298189406696366981362867320131527048683492811128204661745388510505145389459518039217549444918405620726988722254633562452576638635488354260221598432448974859895979017211032905988949400704082939941050902513120244660937339078367607684436944094809985731012813959774525636937965082155868293686780764307

e = 65537