**CS6353/HW4- RSA Encryption &Decryption Submitted by: Prosunjit Biswas**

**12/03/2013**

The following diagram shows key size (in Kilo Bits) on the X axis and time (in microsecond) in Y axis for encrypting & Decrypting 256 Byte value with RSA Cryptosystem. In following cases, encryption is done with public key and decryption is done with private key.

**Fig 1: Key size (X-Axis in Bits) vs encryption-decryption time (Y-Axis in sec) of a single run**

**Fig 2: Encryption & Decryption time of 100 random messages with same key size (1024 bit). Y axis represent required time & X axis represent random messages**

**Fig 3: Average Encryption-Decryption time of 100 random Messages over different key size. X axis represent key size in bits and Y axis represent time in second.**

**Observation:**

1. The time required to encrypt and decrypt a string with different keysizes vary significantly in the experiment.
2. I have implementedhw4 in python language with Pycrypto Library [1] (default with python installation) which does not allow RSA encryption with private key and RSA decryption with public key. Following is an excerpt from [2]

***“You can encrypt something with the private key as the private key contains the information required to make the public key, but it would be unusual to do so, as normally the person encrypting the data does*not *have the private key”***

**References:**

[1] <https://pypi.python.org/pypi/pycrypto>

[2] <http://stackoverflow.com/questions/9893080/pycrypto-decrypt-only-with-public-key-in-file-no-privatepublic-key>

***Appendix 1: Python code for HW4***

|  |  |
| --- | --- |
|  | **from** Crypto.PublicKey **import** RSA  **from** Crypto.Cipher **import** PKCS1\_OAEP  **import** datetime**,** time  **import** random  **def** **generate\_RSA**(keysize**=**1024):      rsa **=** RSA**.**generate(keysize)      publicKey **=** rsa**.**publickey()**.**exportKey()      privateKey **=** rsa**.**exportKey()  **return** publicKey, privateKey  **def** **createMessage**(size**=**256):  **return** ''**.**join(chr(random**.**randint(97,122)) **for** i **in** range(0,255))  **def** **encryptNDecryptLongMessage** (pub\_key,priv\_key, msg**=**None):  **if** msg **==** None:          msg **=** createMessage()      chunks **=** msg**.**\_\_len\_\_()      chunk\_size **=** 50      splitedStr **=** [ msg[i:i**+**chunk\_size] **for** i **in** range(0, chunks, chunk\_size) ]      stime **=** datetime**.**datetime**.**now()  **for** msg **in** splitedStr :          encrypted\_msg **=** encrypt(pub\_key,msg)          decrypted\_msg **=** decrypt(priv\_key,encrypted\_msg)  *#print encrypted\_msg, decrypted\_msg*      end\_time **=** datetime**.**datetime**.**now()  **return** stime, end\_time  **def** **encrypt**(key, msg):      publicKey **=** RSA**.**importKey(key)      cipher **=** PKCS1\_OAEP**.**new(publicKey)      cipher\_text **=** cipher**.**encrypt(msg)  **return** cipher\_text  **def** **decrypt** (key, encypted\_msg**=**None):      privKey **=** RSA**.**importKey(key)      cipher **=** PKCS1\_OAEP**.**new(privKey)      message **=** cipher**.**decrypt(encypted\_msg)  **return** message  **def** **hw4**():      keysize **=** [ i**\***1024 **for** i **in** range(1,20) ]  **for** size **in** keysize:          (pub\_key, priv\_key) **=** generate\_RSA(size)          (s1\_time,end1\_time) **=** encryptNDecryptLongMessage(pub\_key,priv\_key)  *#(s2\_time,end2\_time) = encryptNDecryptLongMessage(priv\_key,pub\_key)*  **print** size, end1\_time **-** s1\_time  **def** **timestamp**(date):  **return** time**.**mktime(date**.**timetuple())  **def** **diffKeySize100MsgTest**():      keysize **=** [ i**\***1024 **for** i **in** range(1,10) ]  **for** size **in** keysize:        avg**=**0  **for** i **in** range(1,101):          (pub\_key, priv\_key) **=** generate\_RSA(size)          (s1\_time,end1\_time) **=** encryptNDecryptLongMessage(pub\_key,priv\_key)          avg **=** (avg **\*** (i **-**1) **+** (timestamp (end1\_time) **-** timestamp (s1\_time) ))**/**i  *#(s2\_time,end2\_time) = encryptNDecryptLongMessage(priv\_key,pub\_key)*  **print** size, avg    **def** **diffKeySingleMessage**():     keysize **=** [ 1024 **for** i **in** range(1,101) ]  **for** size **in** keysize:          (pub\_key, priv\_key) **=** generate\_RSA(size)          (s1\_time,end1\_time) **=** encryptNDecryptLongMessage(pub\_key,priv\_key)  *#req\_time = (timestamp (end1\_time) - timestamp (s1\_time) )*  *#(s2\_time,end2\_time) = encryptNDecryptLongMessage(priv\_key,pub\_key)*  **print** size, end1\_time **-** s1\_time    *#diffKeySingleMessage()*  diffKeySize100MsgTest() |