**USC UPSTATE**

**CSCI 455: Computer Security**

**Spring 2019**

**Lab 5**

**RSA** can also be used for message authentication. Its security assumptions are based on complexity theory: computing the product of two prime numbers is easy (polynomial time), but there is no efficient algorithm for factoring them back (so far, all factorization methods are in the non-polynomial class).

The keys for the RSA algorithm are generated the following way:

1. Choose two different large random prime numbers ***p*** and ***q***
2. Calculate ***n = pq***
   * ***n*** is the modulus for the public key and the private keys
3. Calculate the totient: ***phi =(p-1)(q-1)***.
4. Choose an integer ***e*** such that 1 < ***e*** < ***phi***, and ***e*** is coprime to ***phi***
   * ***e*** is released as the public key exponent
5. Compute ***d*** to satisfy the congruence relation ***d\*e ≡ 1 mod phi (the remainder of d\*e / phi is 1)***
   * ***d*** is kept as the private key exponent

The public key is made of the modulus ***n*** and the public (or encryption) exponent ***e***.

The private key is made of the modulus ***n*** and the private (or decryption) exponent ***d*** which must be kept secret.

Encrypting a message: ***c = m ^ e (mod n)***

Decrypting a message: ***m = c ^ d (mod n)***

Example:

p = 29, q = 31

n = p \* q = 29 \* 31 = 899

phi = (p -1) \* (q – 1) = (29 – 1) \* (31 – 1) = 840

e = 11

d \* e ≡ 1 mod phi => (d \* 11) / phi will give us a remainder of one.

(611 \* 11) = 6721 and 6721 / 840 = 8 with remainder 1 => d = 611

C = M^e mod n

C = 119^11 mod 899 = 595

M = C^d mod n

M = 595^611 mod 899 = 119

**Problem 1**: RSA Digital Signature (Signing a message)

The signing algorithm is: Signature S = M^d mod n.

In order to sign the message, you need to obtain secret key d (private key). Use the private/public key pairs generated in Lab 4 problem 1.

1. Sign the message as follows and return the signature in hex string.

M = “I owe you $2000.”

1. Make a slight change to the message M, such as changing $2000 to $3000, and sign the modified message. Compare both signatures (in hex string) and describe what you observe.

Answer: The hex string is completely different between the two messages when the original messaged is changed.

Note: Some useful functions.

1. Convert string into hex. *''.join(hex(ord(x))[2:] for x in 'I owe you $2000.')*
2. Convert hex into integer. int('7368616b6564', 16)
3. Convert integer into hex. hex(65537)[2:]
4. Convert hex into string. bytes.fromhex('7368616b6564').decode()

**Problem 2**: RSA Digital Signature (Verifying a Signature)

Bob receives a message M = "Launch a missile." from Alice, with her signature S. We know that Alice’s public key is (e, n). The verification algorithm is to compute S^e mod n and then compare the result with the message M. The public key and signature (hexadecimal) are listed in the following:

M = Launch a missle.

S = 643D6F34902D9C7EC90CB0B2BCA36C47FA37165C0005CAB026C0542CBDB6802F

e = 010001 (this hex value equals to decimal 65537)

n = AE1CD4DC432798D933779FBD46C6E1247F0CF1233595113AA51B450F18116115

1. Please verify whether the signature is indeed Alice’s or not.

Answer: Yes, the signature in Alice’s

1. Suppose that the signature is corrupted, such that the last byte of the signature changes from 2F to 3F, i.e, there is only one bit of change. Please repeat this task and describe what will happen to the verification process.

Answer: The verification process will end in an error. Since the signature has been modified, the verifying of the signature wasn’t possible.