Assignment 6

Konstantinos Filippou
ics23044

Exercise 1

Let $p = 41$ and $q = 17$ be the primes used in RSA key generation.

- Which of the candidates $e_1 = 5$ and $e_2 = 19$ is a valid RSA public exponent?
- Compute the private key exponent d using the Extended Euclidean Algorithm.

Solution

$$n = pq = 41 \cdot 17 = 697$$

$$\varphi(n) = (p-1)(q-1) = 40 \cdot 16 = 640$$

A valid public exponent e must satisfy $\gcd(e, \varphi(n)) = 1$.

Check $e_1 = 5$:

$$\gcd(5, 640) = 5 \neq 1$$

So e_1 has no inverse modulo $\varphi(n)$ and is **not valid**.

Check $e_2 = 19$:

$$640 = 19 \cdot 33 + 13, \quad 19 = 13 \cdot 1 + 6, \quad 13 = 6 \cdot 2 + 1$$

Thus $\gcd(19, 640) = 1$, so $e_2 = 19$ is **valid**.

We now compute $d \equiv 19^{-1} \pmod{640}$.

From:

$$1 = 13 - 6 \cdot 2$$

and $6 = 19 - 13$:

$$1 = 13 - 2(19 - 13) = 3 \cdot 13 - 2 \cdot 19$$

and $13 = 640 - 19 \cdot 33$:

$$1 = 3(640 - 19 \cdot 33) - 2 \cdot 19 = 3 \cdot 640 - 19(99 + 2) = 3 \cdot 640 - 101 \cdot 19$$

Hence:

$$-101 \cdot 19 \equiv 1 \pmod{640}$$

So:

$$d \equiv -101 \equiv 640 - 101 = 539 \pmod{640}$$

$$\boxed{e = 19, \quad d = 539}$$

Exercise 2

Compute the following modular exponentiations $m^e \bmod n$ by hand using the *square-and-multiply* algorithm:

- $m = 2,\ e = 31,\ n = 101$
- $m = 3,\ e = 97,\ n = 101$

Solution

$2^{31} \bmod 101$

We have $31 = (11111)_2$. Applying square-and-multiply yields:

$$2^{31} \equiv 34 \pmod{101}$$

$$\boxed{2^{31} \bmod 101 = 34}$$

$3^{97} \bmod 101$

We have $97 = (1100001)_2$. Applying square-and-multiply yields:

$$3^{97} \equiv 15 \pmod{101}$$

$$\boxed{3^{97} \bmod 101 = 15}$$

Exercise 3

Using RSA, encrypt and decrypt with the given parameters:

- $p = 3,\ q = 11,\ d = 7,\ m = 4$
- $p = 5,\ q = 11,\ e = 3,\ m = 20$

Solution

Case 1: $p = 3,\ q = 11,\ d = 7,\ m = 4$

$$n = pq = 3 \cdot 11 = 33$$

$$\varphi(n) = (3-1)(11-1) = 2 \cdot 10 = 20$$

We need e such that:

$$ed \equiv 1 \pmod{20}$$

With $d = 7$, we solve:

$$7e \equiv 1 \pmod{20}$$

Using EEA:

$$20 = 7 \cdot 2 + 6, \quad 7 = 6 \cdot 1 + 1$$

Back-substitution:

$$1 = 7 - 6 = 7 - (20 - 7 \cdot 2) = 3 \cdot 7 - 20$$

So:

$$e = 3$$

Encryption:

$$c \equiv m^e \pmod{n} = 4^3 \bmod 33 = 64 \bmod 33 = 31$$

$$\boxed{c = 31}$$

Decryption (given result):

$$\boxed{m = 4}$$

Case 2: $p = 5,\ q = 11,\ e = 3,\ m = 20$

$$n = pq = 5 \cdot 11 = 55, \quad \varphi(n) = (4)(10) = 40$$

Encryption:

$$c \equiv 20^3 \bmod 55 = 8000 \bmod 55$$

$$8000 = 55 \cdot 145 + 25 \Rightarrow c = 25$$

$$\boxed{c = 25}$$

Decryption (given result):

$$\boxed{m = 20}$$

Exercise 4

RSA parameters:

$$p = 31, \quad q = 37, \quad e = 17$$

Decrypt $c = 2$ using the Chinese Remainder Theorem (CRT) and verify by encrypting the plaintext (without CRT).

Solution

$$n = pq = 31 \cdot 37 = 1147$$

$$\varphi(n) = (p-1)(q-1) = 30 \cdot 36 = 1080$$

We compute $d \equiv e^{-1} \pmod{1080}$.
EEA:

$$1080 = 17 \cdot 63 + 9, \quad 17 = 9 \cdot 1 + 8, \quad 9 = 8 \cdot 1 + 1$$

Back-substitution:

$$1 = 9 - 8, \quad 8 = 17 - 9 \Rightarrow 1 = 9 - (17 - 9) = 2 \cdot 9 - 17$$

$$9 = 1080 - 17 \cdot 63 \Rightarrow 1 = 2(1080 - 17 \cdot 63) - 17 = 2 \cdot 1080 - 17 \cdot 127$$

Thus:

$$d \equiv -127 \equiv 1080 - 127 = 953 \pmod{1080}$$

$$\boxed{d = 953}$$

CRT exponents:

$$d_p = d \bmod (p-1) = 953 \bmod 30 = 23$$

$$d_q = d \bmod (q-1) = 953 \bmod 36 = 17$$

Compute:

$$m_p = c^{d_p} \bmod p = 2^{23} \bmod 31 = 8$$

$$m_q = c^{d_q} \bmod q = 2^{17} \bmod 37 = 18$$

Compute inverses:

$$t_p \equiv q^{-1} \pmod{p} \Rightarrow 37 \equiv 6 \pmod{31}$$

Inverse of 6 mod 31 is 26 (since $6 \cdot 26 = 156 \equiv 1 \pmod{31}$), so:

$$t_p = 26$$

$$t_q \equiv p^{-1} \pmod{q} \Rightarrow 31^{-1} \pmod{37} = 6$$

Combine:

$$m \equiv q t_p m_p + p t_q m_q \pmod{n}$$

$$m \equiv 37 \cdot 26 \cdot 8 + 31 \cdot 6 \cdot 18 \pmod{1147}$$

$$= 7696 + 3348 = 11044$$

$$11044 \bmod 1147 = 721$$

$$\boxed{m = 721}$$

Verification (no CRT)

Compute:

$$c \equiv m^e \pmod{n} = 721^{17} \bmod 1147$$

This evaluates to:

$$\boxed{c = 2}$$

so the decryption is correct.

Exercise 5

Implement the RSA Activity (define and call the functions as described).

Solution

Implementation shown via provided code:

Exercise 6

Design a simple RSA-based protocol that allows Alice and Bob to agree on a shared secret session key over an insecure channel. Who determines the session key: Alice, Bob, or both?

Solution

Let Bob have RSA public key (n, e) and private key d.

1. Alice generates a random session key K such that $K < n$.
2. Alice encrypts it with Bob's public key:

   $$C \equiv K^e \pmod{n}$$

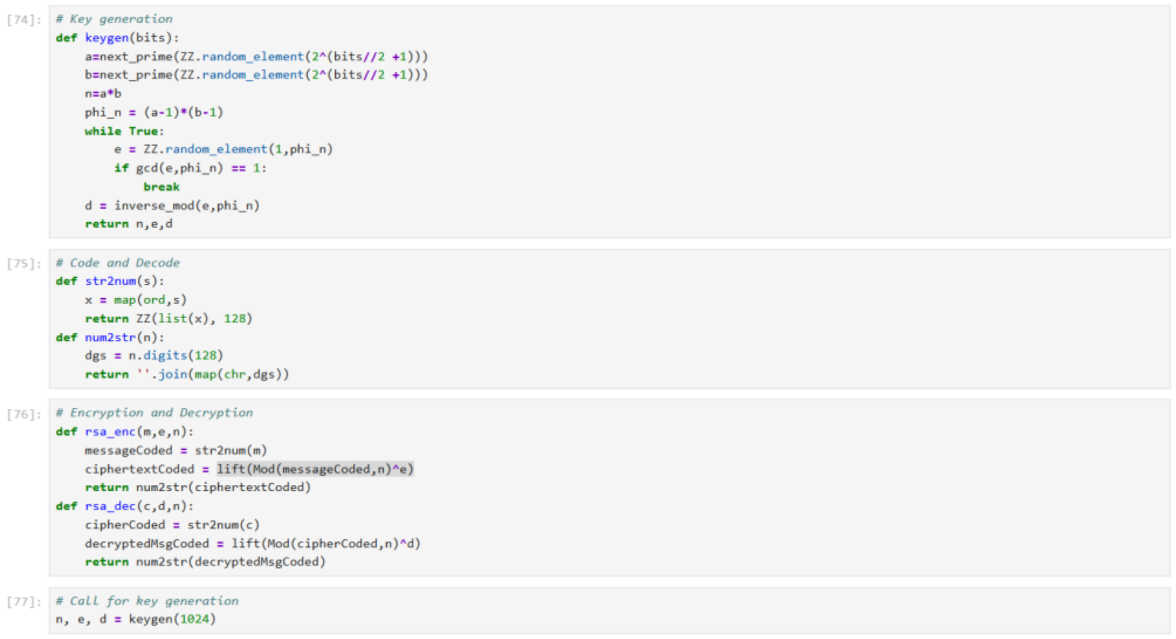
   and sends C to Bob.
3. Bob decrypts using his private key:

   $$K \equiv C^d \pmod{n}$$

Now both share the same K. In this protocol, **Alice determines** the session key and Bob only receives it.

Exercise 7

Suppose in RSA an adversary discovers a non-zero message m that is *not* relatively prime to $n = pq$. Prove the adversary can factor n and thus break RSA. If m is chosen uniformly at random, what is the probability that m is not relatively prime to n?



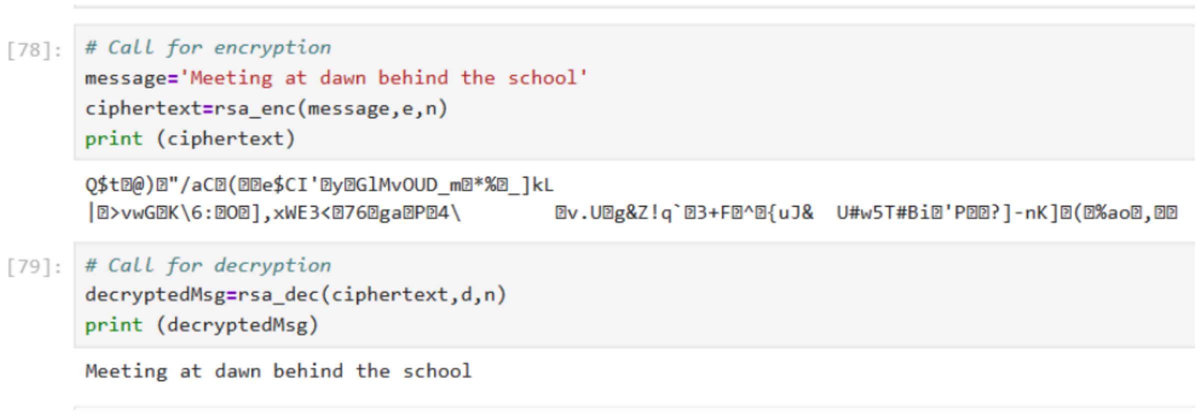
```
[74]: # Key generation
      def keygen(bits):
          a=next_prime(ZZ.random_element(2^(bits//2 +1)))
          b=next_prime(ZZ.random_element(2^(bits//2 +1)))
          n=a*b
          phi_n = (a-1)*(b-1)
          while True:
              e = ZZ.random_element(1,phi_n)
              if gcd(e,phi_n) == 1:
                  break
          d = inverse_mod(e,phi_n)
          return n,e,d

[75]: # Code and Decode
      def str2num(s):
          x = map(ord,s)
          return ZZ(list(x), 128)
      def num2str(n):
          dgs = n.digits(128)
          return ''.join(map(chr,dgs))

[76]: # Encryption and Decryption
      def rsa_enc(m,e,n):
          messageCoded = str2num(m)
          ciphertextCoded = lift(Mod(messageCoded,n)^e)
          return num2str(ciphertextCoded)
      def rsa_dec(c,d,n):
          cipherCoded = str2num(c)
          decryptedMsgCoded = lift(Mod(cipherCoded,n)^d)
          return num2str(decryptedMsgCoded)

[77]: # Call for key generation
      n, e, d = keygen(1024)
```

Figure 1:



```
[78]: # Call for encryption
      message='Meeting at dawn behind the school'
      ciphertext=rsa_enc(message,e,n)
      print (ciphertext)

      Q$t⍰@)⍰"/aC⍰(⍰⍰e$CI'⍰y⍰GlMvOUD_m⍰*%⍰_]kL
      |⍰>vwG⍰K\6:⍰O⍰],xWE3<⍰76⍰ga⍰P⍰4\        ⍰v.U⍰g&Z!q`⍰3+F⍰^⍰{uJ&  U#w5T#Bi⍰'P⍰⍰?]-nK]⍰(⍰%ao⍰,⍰⍰

[79]: # Call for decryption
      decryptedMsg=rsa_dec(ciphertext,d,n)
      print (decryptedMsg)

      Meeting at dawn behind the school
```

Figure 2:

Solution

If:

$$g = \gcd(m, n), \quad 1 < g < n,$$

then g is a non-trivial divisor of n. Hence:

$$n = g \cdot \frac{n}{g},$$

and both factors are integers greater than 1. Therefore n is factored.

Once p and q are known, the adversary can compute:

$$\varphi(n) = (p-1)(q-1),$$

and then recover d from e by solving:

$$ed \equiv 1 \pmod{\varphi(n)},$$

thus fully breaking RSA.

Probability

Among $\{1, 2, \ldots, n-1\}$, the values not coprime to $n = pq$ are those divisible by p or by q.

Count multiples of p:

$$\left\lfloor \frac{n-1}{p} \right\rfloor = q - 1$$

Count multiples of q:

$$\left\lfloor \frac{n-1}{q} \right\rfloor = p - 1$$

Total:

$$(q-1) + (p-1) = p + q - 2$$

Thus:

$$\Pr(\gcd(m, n) \neq 1) = \frac{p + q - 2}{n - 1}.$$

For large primes, this probability is extremely small.