Experiment 2

Chinmay Parikh A059

Questions

1. Compare and contrast symmetric key encryption and asymmetric key encryption.
2. Explain few of the application of RSA.
3. List advantages and limitations of RSA?
4. What are the more popular values of e in practice? Why?
5. Why decryption using RSA takes more time as compared to encryption?

Answers

1.

Method	DES	RSA
Approach	Symmetric	Asymmetric
Encryption	Faster	Slower
Decryption	Difficult	Slow
Key Distribution	O (Log N)	Easy
Complexity	Moderate	O (N3)
Security	Closed	HIghest
Inherent Vulnerabilities	Weak key usage	Brute Forced and Oracle attack
Vulnerabilities Cause	Confidentially	Weak Implementation

2. A. The problem of secure communication: Suppose that two persons want to communicate with each other, and they want to protect their communication from being overheard by a third party.This could for example be any of the following situations: Alice is deeply in love with Bob, but her parents think she is too young to have a boyfriend. So Sarah needs to send secret messages to Bob that her parents cannot understand, even if they manage to find one of the messages.

   B. Factoring large numbers: One of the ideas behind the RSA cryptography is that it is very hard to factor large integers, even if you use a computer.

You have learnt how to factor small numbers, but how would you find the prime factorization of an integer n with 200 digits? Of course, you could start by checking the primes 2, 3, 5, 7 and so on; to see if any of them divides the large number. However, such a number is so large, that you would become old and probably die before you have checked all primes up to squareroot(n), even if you used a computer.

C. Any message can be expressed in terms of integers – It is easy to transform any message written with letters to a message written in integers. We could use any of the common methods used in computers, such as ASCII, Unicode and so on. In this course, we will simply write 1 instead of A, 2, instead of B, and so on, up to 26 instead of Z. We will also write 0 instead of an empty space. So we could send the numbers 8 9 0 20 8 5 18 5 instead of the message "HI THERE".

D. The RSA Cryptosystem: Suppose that I want to be able to receive secret messages from other people. The fundamental idea is the following. I find a "one-way function", call it E, such that everyone can compute E, but only I can compute the inverse of E. Then anyone can send me a secret message x, by computing E(x), and sending this value to me. Since I am the only one who can compute the inverse of E, I and no-one else can retrieve x.

3. RSA's biggest advantage is that it uses Public Key encryption. This means that your text will be encrypted with someone's Public Key (which everyone knows about). However, only the person it is intended for can read it, by using their private key (which only they know about). Attempting to use the Public Key to decrypt the message would not work. RSA can also be used to "sign" a message, meaning that the recipient can verify that it was sent by the person they think it was sent by.

   Disadvantages: A disadvantage of using public-key cryptography for encryption is speed. There are many secret-key encryption methods that are significantly faster than any currently available public- key encryption method. Nevertheless, public-key cryptography can be used with secret-key cryptography to get the best of both worlds. For encryption, the best solution is to combine public- and secret-key systems in order to get both the security advantages of public-key systems and the speed advantages of secret-key systems. Such a protocol is called a digital envelope.

   Limitations:

   - Using small primes.
   - Using primes that are very close
   - Message is an observable power
   - Two people using the same, receiving the same message.

4. Since e is the public exponent it doesn't need to be selected randomly. In fact, there are a few common values used for e in many public keys (e.g.,

3, 7, 17, 257, and 65537). Choosing a small value makes encryption fast to compute (but it shouldn't be too small to avoid cases where m^e is lower than n or a small multiple of n). The value of d is secret, so must be large and unpredictable.

5. In theory, it doesn't have to be. The encryption and decryption algorithms are essentially identical. Given: d = decryption key e = encryption key n = modulus (product of primes) c = encrypted code group m = plaintext code group Encryption ci = mie (mod n) Decryption mi = cid (mod n) The normal algorithm for raising to a power is iterative, so the time taken depends on the size of the exponent. In most cases, the pair works out with the decryption key being (usually considerably) larger than the encryption key.

Code

```
import math
from fractions import gcd

def keyGen(p,q):
    n = p * q
    euler = (p - 1) * (q - 1)
    ekey = 0
    dkey = 0

    for i in range(2,euler):
        if gcd(i , euler) == 1:
            ekey = i
            break

    i = ekey + 1
    while True:
        if (1 + i *euler) % ekey == 0:
            dkey = (1 + i *euler) / ekey
            break
        i = i + 1
    return ekey, dkey, n, euler

def encrypt(key, message, n):
    CT = []
    for element in message:
        CT.append(pow(ord(element), key) % n)
    return CT

def decrypt(key, message, n):
```

```
    PT = []
    for element in message:
        PT.append(int(pow(element, key) % n))
    return ''.join(map(chr, PT))

def isPrime(n):
    sqrt = int(math.sqrt(n))
    for i in range(2,sqrt):
        if (n % i == 0):
            return False
    return True

def main():
    print "Note : P & Q should be such that 'n' Should exceed 128. If not the program fails.
    p = int(raw_input("Enter 1st Prime 'P' : "))
    q = int(raw_input("Enter 2nd Prime 'Q' : "))
    if (not(isPrime(p) and isPrime(q) and (p != q)and p>2 and q>3)):
        print "Enter Prime Numbers Please"
    else:
        message = raw_input("Enter Message : ")
        e, d, n, euler = keyGen(p, q)
        print "n : " + str(n)
        print "euler : " + str(euler)
        print "e : " + str(e)
        print "d : " + str(d)
        CT = encrypt(e, message, n)
        print "Cipher Text"
        print CT
        print "Decrypted Text"
        print decrypt(d, CT, n)

if __name__ == '__main__':
    main()
```

Output

```
[chinmay@chinmay Desktop]$ python2 rsa.py
Note : P & Q should be such that 'n' Should exceed 128. If not the program fails.
Enter 1st Prime 'P' : 17
Enter 2nd Prime 'Q' : 23
Enter Message : Hello
n : 391
euler : 352
e : 3
d : 587
```

```
Cipher Text
[234, 16, 301, 301, 304]
Decrypted Text
Hello
```