Assignment 6: Public Key Cryptography

Aditya Mahajan

Professor Long

Purpose:

Create a set of files that will create keys to view data, similar to ssh keys. One code will be used to encrypt a file and the other shall be used by the receiver to decrypt it. These keys will be created by the keygen code and will use an RSA system to create them.

Structures:

RSA: Implementation of the RSA library which will create random codes for “Keygen”.
Numtheory: The implementation of the number theory functions.
RandState: Implementation of the random state interface used by the “Numtheory” and “RSA” functions.
Encrypt: Implementation of the encryption code, using the rsa key.
Decrypt: Implementation of the decrypt code, using the rsa key.
Keygen: Implementation of the key generation code using the functions created in “RSA”.

Pseudo:
Randstate:

Randstateinit

```
GMP_RandInit_MT(state)
GMP_RandSeed_MT(state , seed)
```

Randstate clear:

```
Gmp_randclear
```

Numtheory:

```
POWER-MOD(a, d, n)
1  v ← 1
2  p ← a
3  while d > 0
4      if ODD(d)
5          v ← (v × p) mod n
6      p ← (p × p) mod n
7      d ← ⌊d/2⌋
8  return v
```

Out = v

Is prime:

```
1  write n − 1 = 2^s r such that r is odd
2  for i ← 1 to k
3      choose random a ∈ {2, 3, ..., n − 2}
4      y = POWER-MOD(a, r, n)
5      if y ≠ 1 and y ≠ n − 1
6          j ← 1
7          while j ≤ s − 1 and y ≠ n − 1
8              y ← POWER-MOD(y, 2, n)
9              if y == 1
10                 return FALSE
11             j ← j + 1
12         if y ≠ n − 1
13             return FALSE
14 return TRUE
```

Make prime:

```
While (!isprime){
    Create random int
    Iter++
    Run through is prime in loop
}
```

GCD:

```
1  while b ≠ 0
2      t ← b
3      b ← a mod b
4      a ← t
5  return a
```

Mod inverse:

```
1  (r, r') ← (n, a)
2  (t, t') ← (0, 1)
3  while r' ≠ 0
4        q ← ⌊r/r'⌋
5        (r, r') ← (r', r − q × r')
6        (t, t') ← (t', t − q × t')
7  if r > 1
8        return no inverse
9  if t < 0
10       t ← t + n
11 return t
```

RSA

Rsa make public:

Make prime p and q
Mul p and q into n
P2 = P - 1
Q2 = Q - 1
p2*q2 = toitent

Write pub:

Print (%x , n)
Print (%x , e)
Print (%x , s)
Print username

Read:

read(1 for n)
Read for e
Read for s
Read for username

Make priv:

P2 = P - 1
Q2 = Q - 1
p2*q2 = toitent
Inverse toitent

Write priv:

Write(%x \n, n)
Write(%x \n, d)

Read priv:

read(%x \n, n)
read(%x \n, d)

Encrypt file:

Init n
Size base in 2
Divide by 8
While read
    Import
    Encrypt

decrypt file:

Init n
Size base in 2
Divide by 8
While read
    Import
    Decrypt

Rsa sign:

Pow_mod

Rsa verify:

Powmod
    If message is correct
        Return true

Return false

Encrypt.c

Check for arguments
    Assign values for verbose

Get signature
Check signature
Encrypt file

Decrypt.c:

- Check for arguments
  - Assign values for verbose
- Read private
- Decrypt problem

Keygen.c:

- Check for arguments
  - Assign values for verbose
- Get username
- Write pub
- Write priv