Asymmetric Cryptographic Algorithms

Team Members:

- Srishti Mahantesh Patil – 2GI17CS140
- Shreya Parvati – 2GI17CS125
- Vandana Malapure – 2GI17CS156
- Siddhi Shanbhag - 2GI17CS134

Introduction

Asymmetric cryptography is a branch of cryptography where a secret key can be divided into two parts, a public key and a private key. The public key can be given to anyone, trusted or not, while the private key must be kept secret (just like the key in symmetric cryptography).

Introduction

Asymmetric cryptography has two primary use cases: authentication and confidentiality. Using asymmetric cryptography, messages can be signed with a private key, and then anyone with the public key is able to verify that the message was created by someone possessing the corresponding private key. This can be combined with a proof of identity system to know what entity (person or group) actually owns that private key, providing authentication.

Introduction

Encryption with asymmetric cryptography works in a slightly different way from symmetric encryption. Someone with the public key is able to encrypt a message, providing confidentiality, and then only the person in possession of the private key is able to decrypt it.

Different Asymmetric Algorithms

- RSA
- Ed25519 signing
- X25519 key exchange
- Ed448 signing
- X448 key exchange
- Diffie-Hellman key exchange
- DSA
- Key Serialization

RSA

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and distinct from the decryption key which is kept secret (private). In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the "factoring problem". The acronym RSA is the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who publicly described the algorithm in 1977. Clifford Cocks, an English mathematician working for the British intelligence agency Government Communications Headquarters (GCHQ), had developed an equivalent system in 1973, which was not declassified until 1997

RSA

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e.Â Public KeyÂ andÂ Private Key.Â As the name describes that the Public Key is given to everyone and Private key is kept private.
An example of asymmetric cryptography :
A client (for example browser) sends its public key to the server and requests for some data.
The server encrypts the data using client's public key and sends the encrypted data.
Client receives this data and decrypts it.
Since this is asymmetric, nobody else except browser can decrypt the data even if a third party has public key of browser

RSA

Mechanism behind RSA algorithm:

Generating Public Key :

Select two prime no's. Suppose P = 53 and Q = 59.

Now First part of the Public key : n = P*Q = 3127.

We also need a small exponent say e : But e Must be

An integer and should not be a factor of n. $1 < e < \Phi(n)$ [$\Phi(n)$ is discussed below],

Let us now consider it to be equal to 3.*Our Public Key is made of n and e*

RSA

Generating Private Key :

We need to calculate Φ(n) :

Such that Φ(n) = (P-1)(Q-1) so, Φ(n) = 3016

Now calculate Private Key, d :

d = (k*Φ(n) + 1) / e for some integer k

For k = 2, value of d is 2011.

Now we are ready with our – Public Key (n = 3127 and e = 3) and Private Key(d = 2011)

RSA

Now we will encrypt “HI” :

Convert letters to numbers : H = 8 and I = 9

Thus Encrypted Data c = 89e mod n.

Thus our Encrypted Data comes out to be 1394

Now we will decrypt 1394 :

Decrypted Data = cd mod n.

Thus our Encrypted Data comes out to be 89

8 = H and I = 9 i.e. "HI".

C implementation of RSA

```
#include <stdio.h>

#include <stdlib.h>

#include<string.h>

#include<math.h>

int gcd(int m,int n)

{

   int r=0;

   while(n!=0)

   {

      r=m%n;

      m=n;

      n=r;

   }

   return m;


}
```

C implementation of RSA

```
int main()
{

    int p,q,n,e,d,lambdaN,length,i,j;
    char string[20];
    int message[20],cipher[20];
    printf("\n \nEnter two distinct prime numbers:");
    scanf("%d%d",&p,&q);
    n=p*q;
    lambdaN=(p-1)*(q-1);
    for(i=2;i<lambdaN;i++)
        if(gcd(i,lambdaN)==1)break;
    e=i;
```

C implementation of RSA

```
printf("\nPublic key=(%d,%d)",n,e);
for(i=2;i<lambdaN;i++)
    if((e*i-1)%lambdaN==0)break;
d=i;
printf("\nPrivate key=(%d,%d)\n",n,d);
printf("Enter the message, lower case characters, no space in between:");
scanf("%s",string);
length=strlen(string);
for(i=0;i<length;i++)
{
    message[i]=string[i]-96;
}
```

C implementation of RSA

```
printf("\nAt Sender encrypt message to cipher, cipher=");

    for(i=0;i<length;i++)

    {

     cipher[i]=1;

        for(j=0;j<e;j++)

           cipher[i]=(cipher[i]*message[i])%n;

       printf("\n%c as %d",message[i]+96,cipher[i]);

}

    printf("\nAt Receiver decrypt cipher to message, message=");

    for(i=0;i<length;i++)
```

C implementation of RSA

```
{
      message[i]=1;
        for(j=0;j<d;j++)
            message[i]=(message[i]*cipher[i])%n;



        printf("\n%d as %c",cipher[i],message[i]+96);


    }
    printf("\n");
    return 0;


}
```

C implementation of RSA

OUTPUT:

Enter two distinct prime numbers:11 13

Public key=(143,7)

Private key=(143,103)

Enter the message, lower case characters, no space in between:srishti

At Sender encrypt message to cipher, cipher=

s as 46

r as 138

i as 48

s as 46

h as 57

C implementation of RSA

t as 136

i as 48

At Receiver decrypt cipher to message, message=
46 as s

138 as r

48 as i

46 as s

57 as h

136 as t

48 as i

Thank You