Assignment B4

Title: Implementation of RSA

Problem Statement:

Implement RSA algorithing

Objective:

To understand how RSA algorithmn works.

Outcome:

Students will be able to:

Understand and implement asymmetric encryption using RSA.

Software and Hardware Requirements:

Core 2 Duo /13/i5/i7 64-bit processor OS-Linux 64 bit os.

Editor - gedit / Eclipse

Software - C++ / JAVA / Python

Theory

Principles of Public Key Cryptosystems

1. Public key cryptosystems require the use of two keys- a public key and a private key

2. Public keys are widely known.

3. Private key is kept secret with its owner.

4. The two keys are mathematically related and form a key pair.

5. One key in the key pair cannot be used to derive the other key in the key pair.

6. Any key in the key pair can be used for encryption. The other key then must be used for decryption

7. Sender and receiver both must have their own key pairs.

RSA

RSA, named after its inventors Ron Rivest, Adi Shamir, and Leonard Adleman is an asymmetic key based algorithon RSA, or asymmetric key based algorithms can be used for confidentiality [encryption, decryption], authentication and non-repudiation. RSA, is based on any other finding prime factors for very large numbers. The length of numbers could be around 500 digits.

RSA Key length	Number of digits
1024-bit	309
2048-bit	617
4096-bit	1233

RSA Algorithm

1. Choose two random large prime numbers, p and q

2. Multiply the numbers n= p*q

3. Choose a random integer to be encryption key e such that e

and (p-1)(q-)) are relatively prime.

4. Decryption key is computed as d= e-1 mod ([p-1][q-1])

5. The public key = (n, e)

6. The private key (n.d)

7. For encrypting message M with public key (n. e), you get cipher text. C= Mˆe mod n

8. For decrypting ciphertext with private key (n.d). you get plaintext.

M=Cˆd mod n

Attacks on RSA

1. Brute-force attack

Here the attacker tries to find factors of n by trying out various possibilities.

2. Common medulus

To avoid generating a different modulus n=p*q for each user one may wish to fix n for all the users. It might seem like deriving the decrypting key d is not possible for every encrypting key e by any other user since encrypting key e is a randamily chosen value. But the problem with this approach is that an particular user who knows her pair of e and d can successfully use her own pair to find the factors for common modulus. Once the factors are known since the encrypting key e is known to everyone (because it is public key) the decrypting key d could be found out. Hence, you should not be using common modulus to generate keys for multiple users.

3. Choosing smaller numbers.

The security of the algorithm comes from the fact that factoring large numbers is computationally intensive. Sometimes to improve system performance, smaller numbers can be chosen which can significantly enhance the performance but at the cast of making the algorithm weaker. Hence, you should always choose large numbers to maintain the strength of the algorithm.

4. Man in the middle attack

The attacker could collect all the ciphertext coming out from the user’s system (that is encrypted with her private key) and try to find the private key. The information known to the attacker is the ciphertext and public key.

Conclusion

Successfully understood and implemented RSA algorithm.