A POJECT REPORT ON CRYPTOGRAPHY (STUDY AND ITS APPLICATION)

Submitted in partial fulfillment of the requirements for final semester project for the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

UNDER

DIBRUGARH UNIVERSITY

Submitted by: Bijit Goswami(CS-31/14)

Uddeepta Saikia (CS-24/14)

Arunav Saikia (CS-53/14)

Under the supervision of Mr. Rajib Chakrabartty , Assistant Professor Department of Computer Science and Engineering Jorhat Engineering College.

Candidate's Declaration

We hereby declare that the work presented in this project *"Cryptography (Study and its Application)"* in partial fulfillment of the requirements for final semester project for the degree *Bachelor Of Engineering* in *Computer Science and Engineering*, submitted to the *Department Of Computer Science and Engineering,Jorhat Engineering College*, is an authentic record of our own work carried out for a period from January,2018 till May, 2018, under the supervision of *Mr. Rajib Chakrabartty, Assistant Professor, Jorhat Engineering College, Jorhat.*

The matter embodied in this project has not been submitted by us for the award of any other degree .

This is to certify that the above statement made is correct to the best of my knowledge.

Arunav Saikia(CS-53/14)
Bijit Goswami(CS-31/14)
Uddeepta Saikia(CS-24/14)

Certificate from HoD

Goverment of Assam

Department of Computer Science and Engineering

Jorhat Engineering College

Jorhat-785007

This is to certify that the project entitled "Cryptography (Study and its Application)" was carried out by

Bijit Goswami(CS-31/14)
Uddeepta Saikia (CS-24/14)
Arunav Saikia (CS-53/14)

Students of BE final semester in partial fulfillment of the requirements for final year project for the degree Bachelor Of Engineering in Computer Science and Engineering under Dibrugarh University.

(Dr. Rupam Baruah),
Head of the Department,
Department of Computer Science & Engineering,
Date: Jorhat Engineering College, Jorhat, Assam.

Certificate from External

Goverment of Assam

Department of Computer Science And Engineering

Jorhat Engineering College

Jorhat-785007

This is to certify that the project entitled "Cryptography(Study And Its Application)" was carried out by

Bijit Goswami(CS-31/14)
Uddeepta Saikia (CS-24/14)
Arunav Saikia (CS-53/14)

Students of BE final year in partial fulfillment of the requirements for final semester project for the degree Bachelor Of Engineering in Computer Science And Engineering under Dibrugarh University.

Date: External

Certificate from Guide

Goverment of Assam

Department of Computer Science and Engineering

Jorhat Engineering College

Jorhat-785007

This is to certify that the project entitled "Cryptography(Study and Its Application)" submitted by Bijit Goswami (CS-31/14), Uddeepta Saikia (CS-24/14) and Arunav Saikia (CS-53/14) in partial fulfillment of the requirements for final year project for the dergree of Bachelor Of Engineering in Computer Science and Engineering under Dibrugarh University, has been carried out under my guidance.

The project or any part of it has not been submitted earlier to any other University/Institute for the award of any Degree or Diploma.

Mr. Rajib Chakrabartty,
Assistant Professor,
Department of Computer Science & Engineering,
Date: Jorhat Engineering College, Jorhat, Assam.

Acknowledgement

At the completion of this project and its report, we would like to take this opportunity to express our form of gratitude to those, without whose help this project would have beencompromised.

Deepest form of gratitude goes to out supervisor Mr. Rajib Chakrabartty for his direct and continuous support in terms of ideas and helps just to make sure this project work goes on its track as scheduled. His ideas and valuable advice gave us moral and enlightened us through dark times especially whenever this project comes to halt.

We would also like to thank the entire staff of Computer Science and Engineering Department ,for their continuous support.

Arunav Saikia(CS-53/14)
Uddeepta Saikia(CS-24/14)
Bijit Goswami(CS-31/14)

Abstract

Cryptography deals with the actual securing of digital data. It refers to the design of mechanisms based on mathematical algorithms that provide fundamental information security services.Privacy or secrecy is the fundamental security service provided by cryptography.

The goal of this project is to make a system which encodes a given set of data (so as to protect it from the outside world) by using a key and send it to the receiver ,who can decode the received data using the same or different key. Even if someone in the middle wants to sneak the data that was sent, he will not succeed as the keys are unknown to him.

Here in this project we have laid down a brief description of the task that we did this semester. Firstly we described what cryptography is and the various algorithms needed to implement cryptography. In the later part of this report we described about symmetric and asymmetric cryptosystem. Finally we developed a new algorithm which we named as SSA (Semi Symmetric Asymmetric) algorithm. As the name suggests it is a combination of both symmetric and asymmetric components. Finally we described how the SSA algorithm works and how it was implemented in our project..

Keywords: Encryption, Decryption , Public key , Private key, Hashing , Symmetric and Asymmetric methods.

Contents

Chapter 1

Cryptography —Origin

1.1 Introduction

Human being from ages had two inherent needs: (a) to communicate and share information and (b) to communicate selectively. These two needs gave rise to the art of coding the messages in such a way that only the intended people could have access to the information. Unauthorized people could not extract any information, even if the scrambled messages fell in their hand. The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography. The word 'cryptography' was coined by combining two Greek words, 'Krypto' meaning hidden and 'graphene' meaning writing.

1.2 History of Cryptography

The art of cryptography is considered to be born along with the art of writing. As civilizations evolved, human beings got organized in tribes, groups, and kingdoms. This led to the emergence of ideas such as power, battles, supremacy, and politics. These ideas further fuelled the natural need of people to communicate secretly with selective recipient which in turn ensured the continuous evolution of cryptography as well. The roots of cryptography are found in Roman and Egyptian civilizations.

1.2.1 Hieroglyph – The Oldest Cryptographic Technique

The first known evidence of cryptography can be traced to the use of 'hieroglyph'. Some 4000 years ago, the Egyptians used to communicate by messages written in hieroglyph. This code was the secret known only to the scribes who used to transmit messages on behalf of the kings. One such hieroglyph is shown below.

Fig 1.1: Hieroglyph(Courtesy: tutorialspoint.com)

Later, the scholars moved on to using simple mono-alphabetic substitution ciphers during 500 to 600 BC. This involved replacing alphabets of message with other alphabets with some secret rule. This rule became a key to retrieve the message back from the garbled message. The earlier Roman method of cryptography, popularly known as the Caesar Shift Cipher, relies on shifting the letters of a message by an agreed number (three was a common choice), the recipient of this message would then shift the letters back by the same number and obtain the original message.

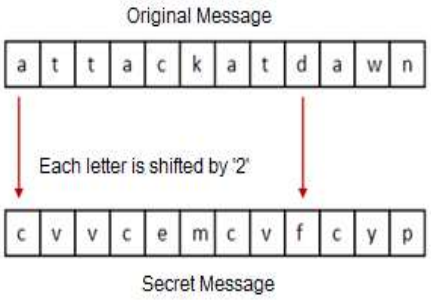


Fig 1.2: Caesar Shift Cipher(Courtesy: tutorialpoint.com)

1.2.2 Steganography

Steganography is similar but adds another dimension to Cryptography. In this method, people not only want to protect the secrecy

of an information by concealing it, but they also want to make sure any unauthorized person gets no evidence that the information even exists. For example, invisible watermarking. In steganography, an unintended recipient or an intruder is unaware of the fact that observed data contains hidden information. In cryptography, an intruder is normally aware that data is being communicated, because they can see the coded/scrambled message.

Fig 1.3: Steganography(Courtesy: tutorialpoint.com)

1.3 Evolution of Cryptography

It is during and after the European Renaissance, various Italian and Papal states led the rapid proliferation of cryptographic techniques.Various analysis and attack techniques were researched in this era to break the secret codes.

Improved coding techniques such as Vicegerent Coding came into existence in the 15th century, which offered moving letters in the message with a number of variable places instead of moving them the same number of places. Only after the 19th century, cryptography evolved from the ad hoc approaches to encryption to the more sophisticated art and science of information security.

In the early 20th century, the invention of mechanical and electromechanical machines, such as the Enigma rotor machine, provided more advanced and efficient means of coding the information. During the period of World War II, both cryptography and cryptanalysis became excessively mathematical.

Chapter 2

Modern Symmetric Key Encryption

Digital data is represented in strings of binary digits (bits) unlike alphabets. Modern cryptosystems need to process this binary strings to convert in to another binary string. Based on how these binary strings are processed, a symmetric encryption schemes can be classified in to -

2.1 Block Ciphers

In this scheme, the plain binary text is processed in blocks (groups) of bits at a time; i.e. a block of plain text bits is selected, a series of operations is performed on this block to generate a block of cipher text. bits. The number of bits in a block is fixed. For example, the schemes DES and AES have block sizes of 64 and 128, respectively.

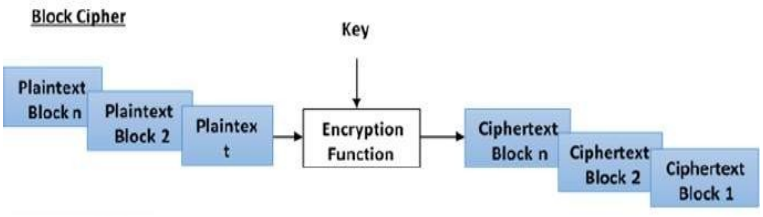


Fig 2.1: Block Ciphers(Courtesy: GoogleImages)

2.2 Stream Ciphers

In this scheme, the plain text is processed one bit at a time i.e. one bit of plain text is taken, and a series of operations is performed on it to

generate one bit of cipher text. Technically, stream ciphers are block ciphers with a block size of one bit.

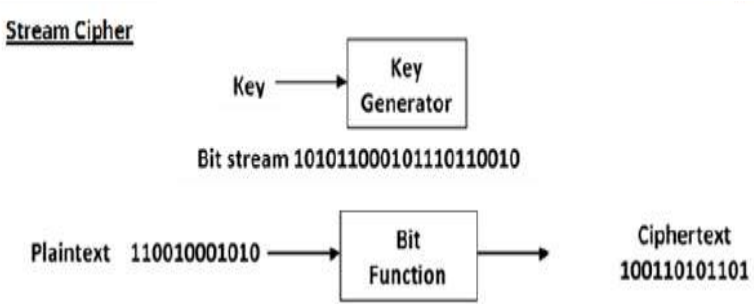


Fig 2.2: Stream Ciphers(Courtesy: GoogleImages)

Chapter 3

Types Of Encryption Algorithms

The generation, modification and transportation of keys have been done by the encryption algorithm. It is also named as cryptographic algorithm. There are many cryptographic algorithms available in the market to encrypt the data. The strength of encryption algorithm heavily relies on the computer system used for the generation of keys.

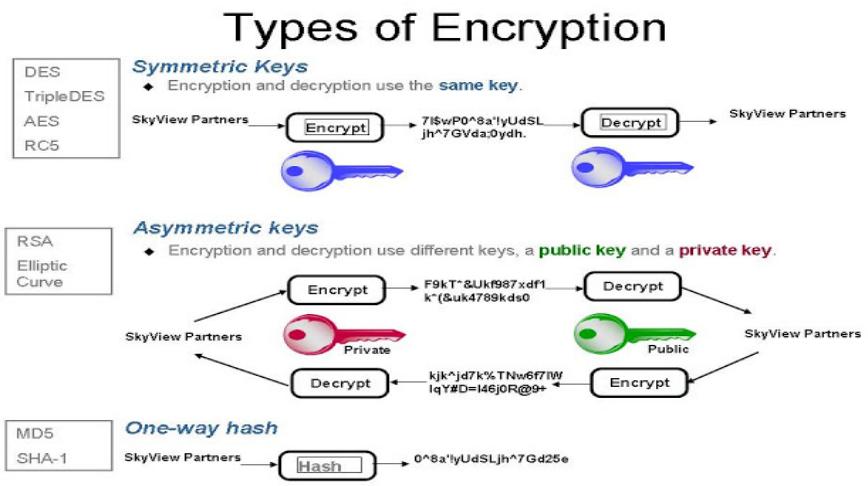


Fig 3.1: Types of Encryption(Courtesy: GoogleImages)

Some important encryption algorithms are discussed here:

3.1 Rivest-Shamir-Adleman (RSA)

RSA is designed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1978. It is one of the best known public key cryptosystems for key exchange or digital signatures or encryption of blocks of data. RSA uses a variable size encryption block and a variable size key. It is an asymmetric (public key) cryptosystem based on number theory, which is a block cipher system. It uses two prime numbers to generate the public and private keys. These two different keys are used for encryption and decryption purpose. Sender encrypts the message using Receiver public key and when the message gets transmit to receiver, then receiver can decrypt it using his own private key.

3.2 Data Encryption Standard (DES)

DES is one of the most widely accepted, publicly available cryptographic systems. It was developed by IBM in the 1970s but was later adopted by the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The Data Encryption Standard (DES) is a block Cipher which is designed to encrypt and decrypt blocks of data consisting of 64 bits by using a 64-bit key. Although the input key for DES is 64 bits long, the actual key used by DES is only 56 bits in length. The least significant (rightmost) bit in each byte is a parity bit, and should be set so that there are always an odd number of 1s in every byte. These parity bits are ignored, so only the seven most significant bits of each byte are used, resulting in a key length of 56 bits. The algorithm goes through 16 iterations that interlace blocks of plain text with values obtained from the key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key are used for decryption.

3.3 Triple DES (3DES)

3DES or the Triple Data Encryption Algorithm (TDEA) was developed to address the obvious flaws in DES without designing a whole new cryptosystem. Data Encryption Standard (DES) uses a 56-bit key and is not deemed sufficient to encrypt sensitive data. 3-DES simply extends the key size of DES by applying the algorithm three times in succession with three different keys. The combined key size is thus 168 bits (3 times 56). TDEA involves using three 64-bit DEA keys (K1,

K2, K3) in Encrypt-Decrypt-Encrypt (EDE) mode, that is, the plain text is encrypted with K1, then decrypted with K2, and then encrypted again with K3 .

3.4 Advanced Encryption Standard (AES)

AES is the new encryption standard recommended by NIST to replace DES in 2001. AES algorithm can support any combination of data 128 bits and key length of 128, 192, and 256 bits. The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length. During encryption- decryption process, AES system goes through 10 rounds for 28-bit keys, 12 rounds for 92-bit keys, and 14 rounds for 256-bit keys in order to deliver final cipher-text or to retrieve the original plain-text [19]. AES allows a 128 bit data length that can be divided into four basic operational blocks. These blocks are treated as array of bytes and organized as a matrix of the order of 4×4 that is called the state. For both encryption and decryption, the cipher begins with an AddRoundKey stage. However, before reaching the final round, this output goes though nine main rounds, during each of those rounds four transformations are performed; 1)Sub-bytes, 2) Shift- rows, 3) Mix-columns, 4) Add round Key. In the final 10^{th} round, there is no Mix-column transformation [3, 20]. Figure below shows the overall process. Decryption is the reverse process of encryption and using inverse functions: Inverse Substitute Bytes, Inverse Shift Rows and Inverse Mix Columns.

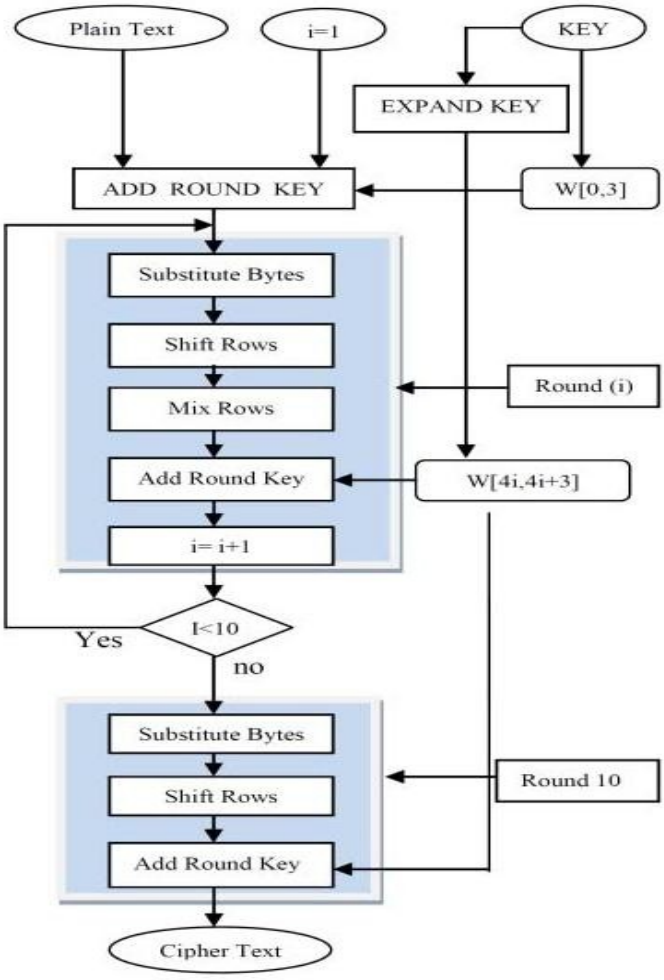


Fig 3.2: AES (Advanced Encryption Standard) process(Courtesy: GoogleImages)

Chapter 4

Introduction to RSA Algorithm

4.1 Introduction

This algorithm is based on the difficulty of factorizing large numbers that have 2 and only 2 factors (Prime numbers). The system works on a public and private key system. The public key is made available to everyone. With this key a user can encrypt data but cannot decrypt it, the only person who can decrypt it is the one who possesses the private key. It is theoretically possible but extremely difficult to generate the private keyfrom the public key, this makes the RSA algorithm a very popular choice in dataencryption.

4.2 Algorithm

First of all, two large distinct prime numbers p and q must be generated. The product of these, we call n is a component of the public key. It must be large enough such that the numbers p and q cannot be extracted from it - 512 bits at least i.e. numbers greater than 10,154 . We then generate the encryption key e which must be co-prime to the number $m = \phi(n) = (p-1)(q-1)$. We then create the decryption key d such that de mod m = 1. We now have both the public and private keys.

4.2.1 Generation of RSA Key Pair

Each person or a party who desires to participate in communication using encryption needs to generate a pair of keys, namely public key and private key. The process followed in the generation of keys is described below:

Generate the RSA modulus (n)

1. Select two large primes, p and q.
2. Calculate $n = p * q$. For strong unbreakable encryption, let n be a large number, typically a minimum of 512 bits.

Find Derived Number (e)

1. Number e must be greater than 1 and less than $(p-1)(q-1)$.
2.There must be no common factor for e and $(p-1)(q-1)$ except for 1. In other words two numbers e and $(p-1)(q-1)$ are co-prime.

Form the public key

1. The pair of numbers (n, e) form the RSA public key and is made public.
2. Interestingly, though n is part of the public key, difficulty in factorizing a large prime number ensures that attacker cannot find in finite time the two primes (p & q) used to obtain n. This is strength of RSA.

Generate the private key

1. Private Key d is calculated from p, q, and e. For given n and e, there is unique number d.
2. Number d is the inverse of e modulo $(p-1)(q-1)$. This means that d is the number less than (p - 1)(q - 1) such that when multiplied by e, it is equal to 1 modulo $(p-1)(q-1)$.
3. This relationship is written mathematically as follows:

$$ed = 1 mod (p-1)(q-1)$$

4.3 Encryption

Once the key pair has been generated, the process of encryption and decryption are relatively straightforward and computationally easy. Interestingly, RSA does not directly operate on strings of bits as in case of symmetric key encryption. It operates on numbers modulo n. Hence, it is necessary to represent the plain-text as a series of numbers less than n. RSA Encryption
1.Suppose the sender wish to send some text message to someone whose public key is (n, e).
2.The sender then represents the plain-text as a series of numbers less than n.
3.To encrypt the first plain-text P, which is a number modulo n. The encryption process is simple mathematical step as:

$$C = P^e \text{ mod n}$$

4.In other words, the cipher-text C is equal to the plain-text P multiplied by itself e times and then reduced modulo n. This means that C is also a number less than n.
5.Returning to our Key Generation example with plain-text P = 10, we get cipher-text C:

$$C = 10^5 \text{ mod } 91$$

4.4 Decryption

1.The decryption process for RSA is also very straightforward. Suppose that the receiver of public-key pair (n, e) has received a cipher-text C.
2.Receiver raises C to the power of his private key d. The result modulo n will be the plain-text P.

$$PlaintextP = C^d \text{ mod n}$$

3.Returning again to our numerical example, the cipher-text C = 82 would get decrypted to number 10 using private key 29:

$$Plaintext = 82^{29} \text{ mod } 91 = 10$$

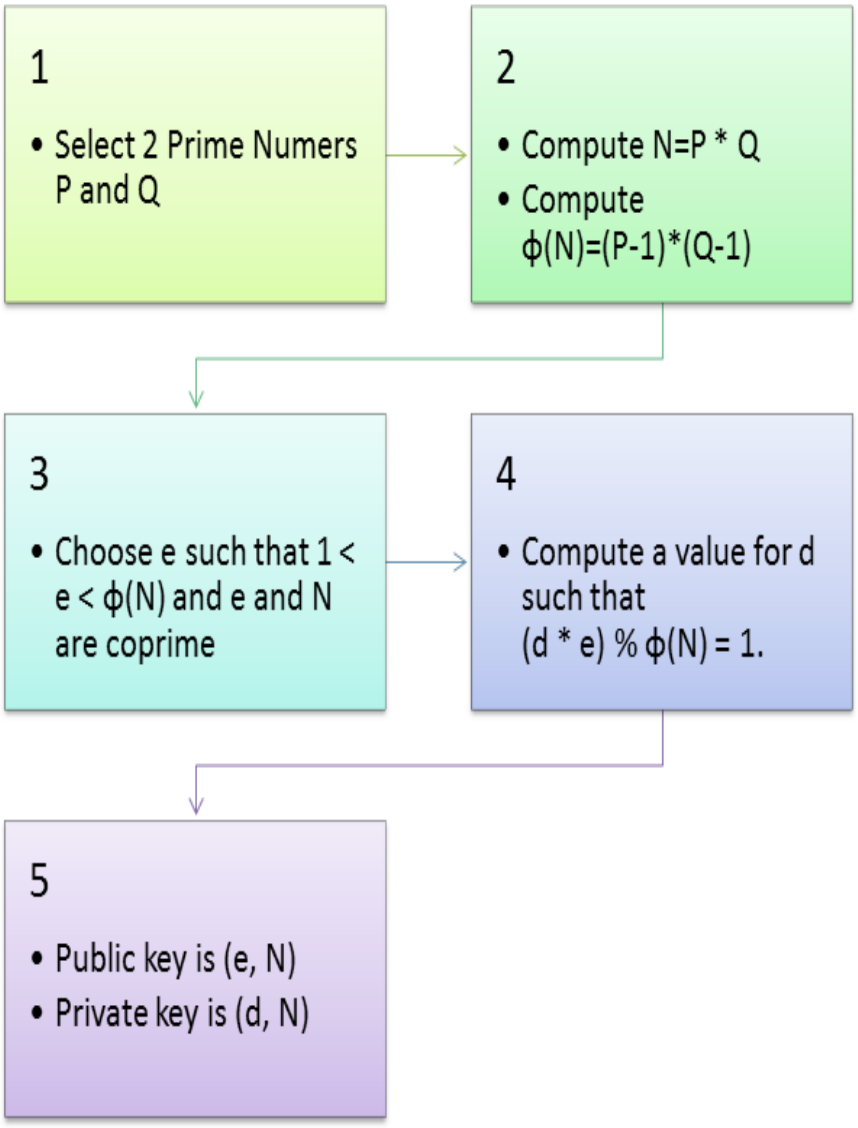


Figure 4.1: Flow chart of RSA Algorithm(Courtesy: GoogleImages).

4.5 How secure is RSA?

The RSA algorithm is indeed among the strongest, but can it withstand anything? Certainly nothing can withstand the test of time. In fact, no encryption technique is even perfectly secure from an attack by a realistic cryptanalyst. Methods such as brute-force are simple but lengthy and may crack a message, but not likely an entire encryption scheme. We must also consider a probabilistic approach, meaning there's always a chance some one may get the "one key out of a million". So far, we don't know how to prove whether an encryption scheme is unbreakable. If we cannot prove it, we will at least see if someone can break the code. This is how the NBS standard and RSA were essentially certified. Despite years of attempts, no one has been known to crack either algorithm. Such a resistance to attack makes RSA secure in practice.

4.6 Conclusion

RSA is a strong encryption algorithm that has stood a partial test of time. RSA implements a public-key cryptosystem that allows secure communications and "digital signatures", and its security rests in part on the difficulty of factoring large numbers. The authors urged anyone to attempt to break their code, whether by factorization techniques or otherwise, and nobody to date seems to have succeeded. This has in effect certified RSA, and will continue to assure its security for as long as it stands the test of time against such break-ins. RSA is slower than certain other symmetric cryptosystems. RSA is, in fact, commonly used to securely transmit the keys for another less secure, but faster algorithm.

A very major threat to RSA would be a solution to the Riemann hypothesis. Thus a solution has neither been proven to exist nor to not exist. Development on the Riemann hypothesis is currently relatively stagnant. However, if a solution were found, prime numbers would be too easy to find, and RSA would fall apart. Undoubtedly, much more sophisticated algorithms than RSA will continue to be developed as mathematicians discover more in the fields of number theory and cryptanalysis.

Chapter 5

Cryptography Hash Functions

A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length.

Values returned by a hash function are called message digest or simply hash values. The following picture illustrated hash function.

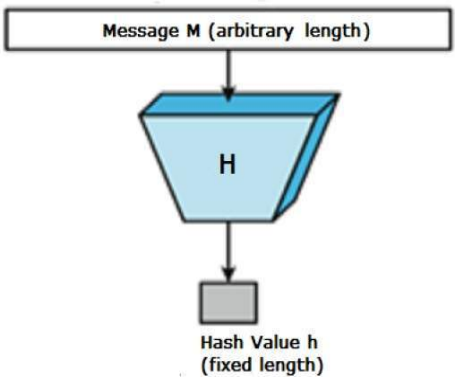


Fig 5.1: Hash Function(Courtesy : tutorialspoint.com)

5.1 Features Of Hash Functions

The typical features of hash functions are-

1. Fixed Length Output (Hash Value):
→Hash function converts data of arbitrary length to a fixed length. This process is often referred to as hashing the data.

→In general, the hash is much smaller than the input data, hence hash functions are sometimes called compression functions.
→Since a hash is a smaller representation of a larger data, it is also referred to as a digest.
→Hash function with n bit output is referred to as an n-bit hash function. Popular hash functions generate values between 160 and 512 bits.

2.Efficiency of Operation:
→Generally for any hash function h with input x, computation of h(x) is a fast operation.
→Computationally hash functions are much faster than a symmetric encryption.

5.2 Properties Of Hash Functions

→ Pre-Image Resistance: This property means that it should be computationally hard to reverse a hash function. In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.

This property protects against an attacker who only has a hash value and is trying to find the input.

→Second Pre-Image Resistance: This property means given an input and its hash, it should be hard to find a different input with the same hash. In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).

This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.

→Collision Resistance: This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function. In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).

Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.

This property makes it very difficult for an attacker to find two in-

put values with the same hash.

5.3 Popular Hash Functions

Some of the popular hash functions are -

5.3.1 Message Digest (MD)

MD5 was most popular and widely used hash function for quite some years.

→The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It was adopted as Internet Standard RFC 1321. It is a 128-bit hash function.

→MD5 digests have been widely used in the software world to provide assurance about integrity of transferred file. For example, file servers often provide a pre-computed MD5 checksum for the files, so that a user can compare the checksum of the downloaded file to it.

→In 2004, collisions were found in MD5. An analytical attack was reported to be successful only in an hour by using computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

5.3.2 Secure Hash Function (SHA)

Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from same family, there are structurally different.

→The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0.

→SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.

→In 2005, a method was found for uncovering collisions for SHA-1 within practical time frame making long-term employability of SHA-1 doubtful.

→SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512 depending up on number of bits in their hash value. No successful attacks have yet been reported on SHA-2 hash function.

→Though SHA-2 is a strong hash function. Though significantly different, its basic design is still follows design of SHA-1. Hence, NIST called for new competitive hash function designs.

→In October 2012, the NIST chose the Keccak algorithm as the new SHA-3 standard. Keccak offers many benefits, such as efficient performance and good resistance for attacks.

5.3.3 RIPEMD

The RIPEMD is an acronym for RACE Integrity Primitives Evaluation Message Digest. This set of hash functions was designed by open research community and generally known as a family of European hash functions.

→The set includes RIPEND, RIPEMD-128, and RIPEMD-160. There also exist 256, and 320-bit versions of this algorithm.

→Original RIPEMD (128 bit) is based upon the design principles used in MD4 and found to provide questionable security. RIPEMD 128-bit version came as a quick fix replacement to overcome vulnerabilities on the original RIPEMD.

→RIPEMD-160 is an improved version and the most widely used version in the family. The 256 and 320-bit versions reduce the chance of accidental collision, but do not have higher levels of security as compared to RIPEMD-128 and RIPEMD-160 respectively.

5.3.4 Whirlpool

This is a 512-bit hash function.

→It is derived from the modified version of Advanced Encryption Standard (AES). One of the designer was Vincent Rijmen, a co-creator of the AES.

→Three versions of Whirlpool have been released; namely WHIRLPOOL-0, WHIRLPOOL-T, and WHIRLPOOL.

5.4 Application Of Hash Functions

There are two direct applications of hash function based on its cryptographic properties.

5.4.1 Password Storage

Hash functions provide protection to password storage.

→Instead of storing password in clear, mostly all login processes store the hash values of passwords in the file.

→The Password file consists of a table of pairs which are in the form (user id, h(P)).

→The process of log on is depicted in the following illustration -

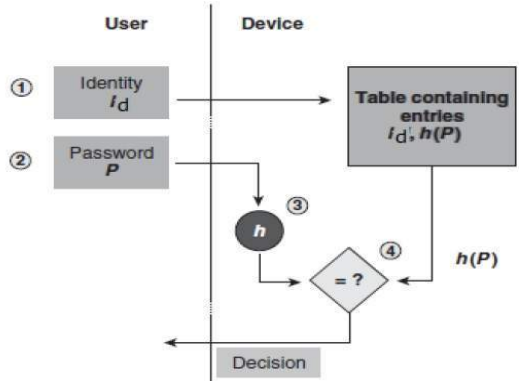


Fig 5.2: Process of Log On(Courtesy: tutorialspoint.com)

→An intruder can only see the hashes of passwords, even if he accessed the password. He can neither log on using hash nor can he derive the password from hash value since hash function possesses the property

of pre-image resistance.

5.4.2 Data Integrity Check

Data integrity check is a most common application of the hash functions. It is used to generate the checksums on data files. This application provides assurance to the user about correctness of the data.
The process is depicted in the following illustration -

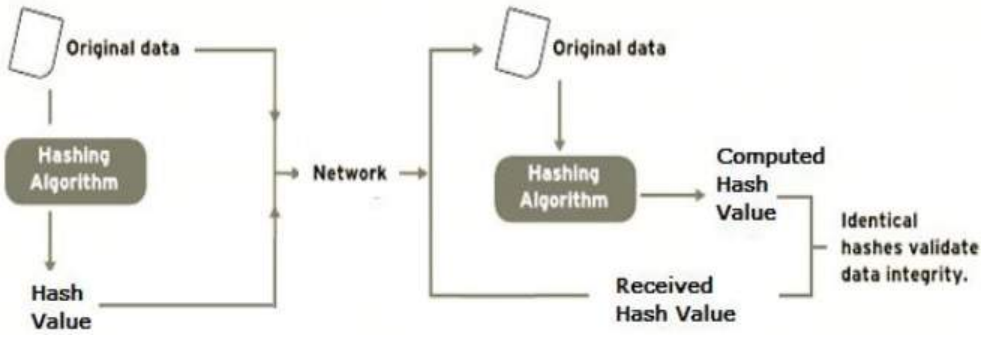


Fig 5.3: Process of Data Integrity Check(Courtesy: tutorialspoint.com)

Chapter 6

Digital Signature

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

6.1 Model Of Digital Signature And Its Explanation

The model of digital signature scheme is depicted in the following illustration -

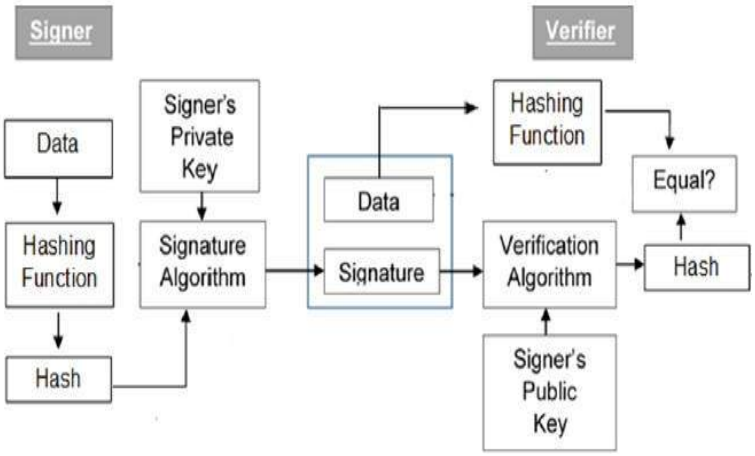


Fig 6.1: Model of Digital Signature(Courtesy: tutorialspoint.com).

The following points explain the entire process in detail -

→Each person adopting this scheme has a public-private key pair.
→Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
→Signer feeds data to the hash function and generates hash of data.
→Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
→Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
→Verifier also runs same hash function on received data to generate hash value.
→For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
→Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence signing a hash is more efficient than signing the entire data.

6.2 Importance Of Digital Signature

→Message Authentication :When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.

→Data Integrity: In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.

→Non-Repudiation : Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future

By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely - Privacy, Authentication, Integrity, and Non-repudiation.

6.3 Encryption With Digital Signature

In many digital communications, it is desirable to exchange an encrypted messages than plain-text to achieve confidentiality. In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver.

This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of mes-

sage authentication and non-repudiation.

This can archived by combining digital signatures with encryption scheme. Let us briefly discuss how to achieve this requirement. There are two possibilities, “sign-then- encrypt” and “encrypt-then-sign”.

However, the cryptosystem based on sign-then-encrypt can be exploited by receiver to spoof identity of sender and sent that data to third party. Hence, this method is not preferred. The process of encrypt-then-sign is more reliable and widely adopted. This is depicted in the following illustration -

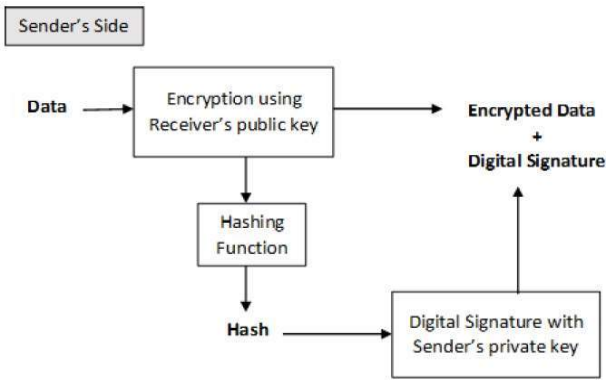


Fig 6.2: Encryption with Digital Signature(Courtesy: tutorialspoint.com)

The receiver after receiving the encrypted data and signature on it, first verifies the signature using sender’s public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.

Chapter 7

Implementation

7.1 Socket programming in C

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

State diagram for server and client model:

We assume that, there is a server and only one client, where client always sends messages to the server. And after successfully receiving the message server sends a feedback to the client.

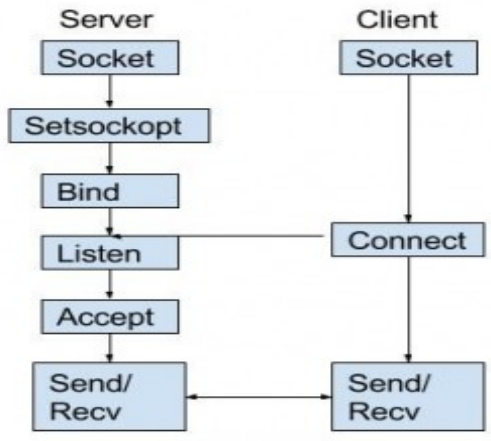


Fig 7.1: FlowChart for Socket Programming(Courtesy: GoogleImage).

7.2 The steps involved in establishing a TCP socket on the server side:

→Create a socket with the socket() function.

```
int sockfd, newsockfd, portno;
socklen_t clilen;
char buffer[256];
struct sockaddr_in serv_addr, cli_addr;
int n,opt=1;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

The socket function returns a non-negative integer number, similar to a file descriptor, that we define socket descriptor or -1 on error.

→Bind the socket to an address using the bind() function.

```
if (bind(sockfd, (struct sockaddr *) &serv_addr,
            sizeof(serv_addr)) < 0)
            error("ERROR on binding");
```

The bind() assigns a local protocol address to a socket. With the Internet protocols, the address is the combination of an IPv4 or IPv6 address (32-bit or 128-bit) address along with a 16 bit TCP port number.

bind() returns 0 if it succeeds, -1 on error.

→Listen for connections with the listen() function.

```
listen(sockfd,2);
```

→The listen() function converts an unconnected socket into a passive socket, indicating that the kernel should accept incoming connection requests directed to this socket.

The function listen() return 0 if it succeeds, -1 on error.

→Accept a connection with the accept() function system call. This call typically blocks until a client connects with the server.

```
clilen = sizeof(cli_addr);
newsockfd = accept(sockfd,
            (struct sockaddr *) &cli_addr,
            &clilen);
if (newsockfd < 0)
     error("ERROR on accept");
```

The accept() is used to retrieve a connect request and convert that into a request.

→Send and receive data by means of send() and receive().

→Since a socket endpoint is represented as a file descriptor, we can use read and write to communicate with a socket as long as it is connected and thus the rev() function is similar to read().

```
bzero(buffer,256);
n = read(newsockfd,buffer,255);

if (n < 0)
 error("ERROR reading from socket");
```

```
n = write(newsockfd,"server has  succesfully received your message",45);
if (n < 0) error("ERROR writing to socket");
```

7.3 The steps involved in establishing a TCP socket on the client side:

→Create a socket using the socket() function.

```
int sockfd, portno, n;
struct sockaddr_in serv_addr;
struct hostent *server;

char buffer[256];
char store[255];
portno = 7000;
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
    error("ERROR opening socket");
```

The socket function returns a non-negative integer number, similar to a file descriptor, that we define socket descriptor or -1 on error.

→Connect the socket to the address of the server using the connect() function.

```
if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
error("ERROR connecting");
```

The connect() function is used by a TCP client to establish a connection with a TCP server. The function returns 0 if the it succeeds in establishing a connection.

→Send and receive data by means of the read() and write() functions.

Since a socket endpoint is represented as a file descriptor, we can use read and write to communicate with a socket as long as it is connected and thus the rev() function is similar to read().

```
n = write(sockfd,store,strlen(store));
if (n < 0)
     error("ERROR writing to socket");
bzero(store,256);
n = read(sockfd,store,255);
if (n < 0)
error("ERROR reading from socket");
printf("%s\n",store);
close(sockfd);
return 0;
```

All of the above functions are in the header file #include <says/socket>.

The normal close() function is used in both the server side and client side programme to close a socket and terminate a TCP socket. It returns 0 if it succeeds, -1 on error. It is n the header file #include <unionist>.

7.4 Diagrammatic representation of the function calls:

The sequence of function calls for the client and a server participating in a TCP connection is represented as-

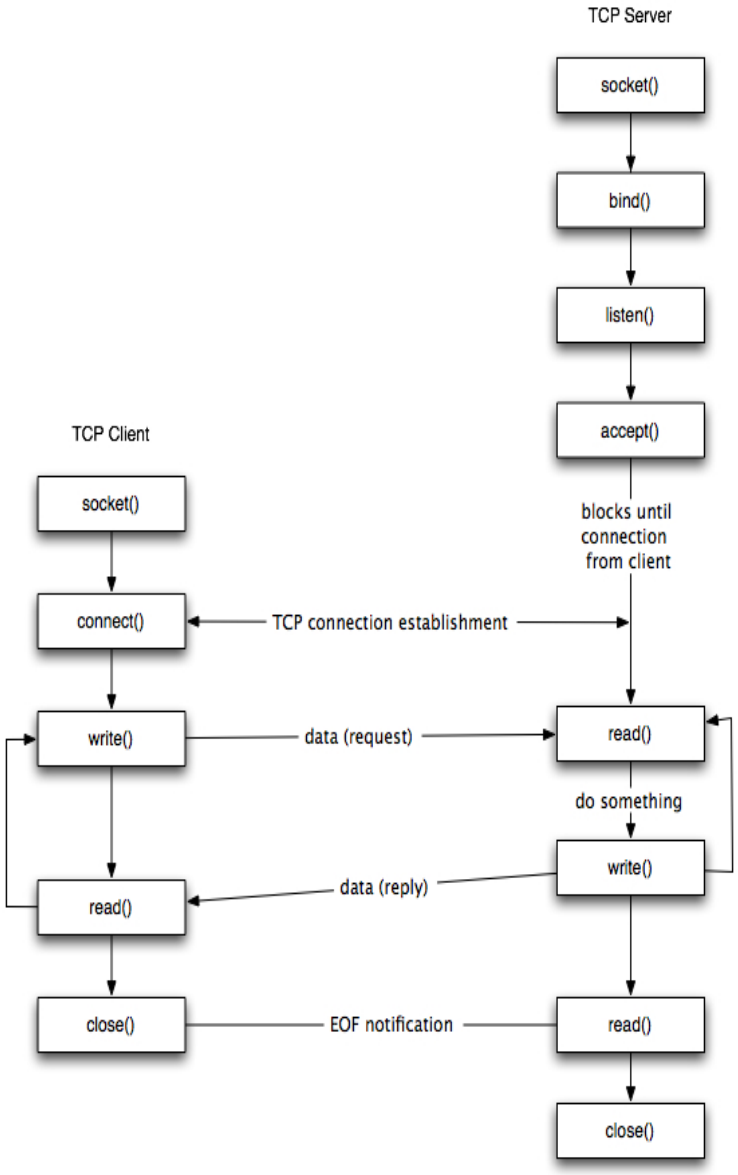


Fig 9.2: Flow Chart of Client-Server Socket Connection(Courtesy: GoogleImages)

7.5 RSA Algorithm

This algorithm is based on the difficulty of factorizing large numbers that have 2 and only 2 factors (Prime numbers). The system works on a public and private key system. The public key is made available to everyone. With this key a user can encrypt data but cannot decrypt it,

the only person who can decrypt it is the one who possesses the private key.

7.6 Generation of keys

→Public key

```
long long int p,q;
p=3;                    //FIRST PRIME NUMBER DECLARED..
q=43;                   //SECOND PRIME NUMBER DECLARED..

long long int pub = p*q;    //THIS IS THE PUBLIC KEY WHICH IS KNOWN BY ALL..
```

- Compute N=P * Q
- Compute ϕ(N)=(P-1)*(Q-1)

Fig 9.3: Public Key Evaluation.

→Private Key

```
long long int n=(p-1)*(q-1);  // NEEDED FOR EVALUATING PRIVATE KEY

long long int e;              // E NEEDED FOR CALCULATING PRIVATE KEY..

for(long long int i=2;i<n;i++)
      {
        if((n%i)!=0)
           {
             e=i;
               break;         // E FOUND... TERMINATE FROM LOOP..
           }

      }
printf("E is=== %lli\n",e);

 long long int k;

 long long int pr,private;

 for(k=0;pr!=0;k++)
 {
   pr=(1+k*n)%e;              //CALCULATE pr SUCH THAT WHEN pr=0, TERNIMATE.
 }
printf("k is == %lld\n",--k);

  private=(1+k*n)/e;          //PRIVATE KEY CALCULATED..WHICH IS KNOWN ONLY TO THE USER..
```

- Choose e such that 1 < e < ϕ(N) and e and N are coprime →
- Compute a value for d such that (d * e) % ϕ(N) = 1.

Fig 9.4: Private Key Evaluation.

The generated key pairs are-
→Public Key(e,N)
→Private key(d,N)

7.7 Encryption and Decryption

After the keys are generated we are to take the input data and encrypt it using the public key. While, the decryption is done using the private key.

→Key Encryption

```
k1=getw(f1);

printf("key1 is ==%d\n",k1);
result=k1;
printf("result is ==%d\n",result);
int power = 1;
for(i=1;i<=e;i++) //pow(result,e)
{

power=power*result;

}
power= power % pub;



fclose(f1);
f1=fopen("encryptedkeyfile1.txt","w");
printf("power ==%d\n",power);
putw(power,f1);
```

Here “char x” contains the character to be encrypted , “int pub” has the public key and “int e” holds the value of e found during the generation of keys. Finally “char b” is the encrypted form of x.

→Key Decryption

```
int g = k;
printf("THE VALUE OF G IS ==%d\n",g);
int res=1;
g = g % pub ;

while(pri>0)
{
   if(pri & 1)
      {
         res=(res*g)%pub;
      }
   pri=pri/2;

   g=(g*g)%pub;
}
return res;
}
```

Here “char n” contains the character to be decrypted , “int pub” has the public key and “int x” is the private key. Finally “char s” is the

decrypted form of n.

7.8 SSA(Semi Symmetric Asymmetric) Algorithm

The SSA(Semi Symmetric Asymmetric) algorithm is a new data cryptotechinque used for secure transfer of messages from one system to another. This application comprises one main tecnique called cryptography.SSA(Semi Symmetric Asymmetric) algorithm is an algorithm which uses both asymmetric and symmetric encryption and decryption methods.

Encryption:Here in the first part of the algorithm we use symmetric encryption to encrypt the data and in the second part we use asymmetric encryption to encrypt the keys used in the first part.

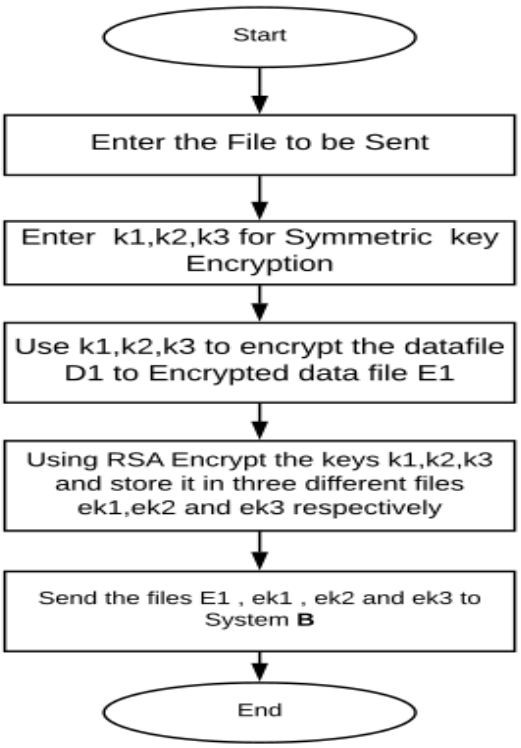
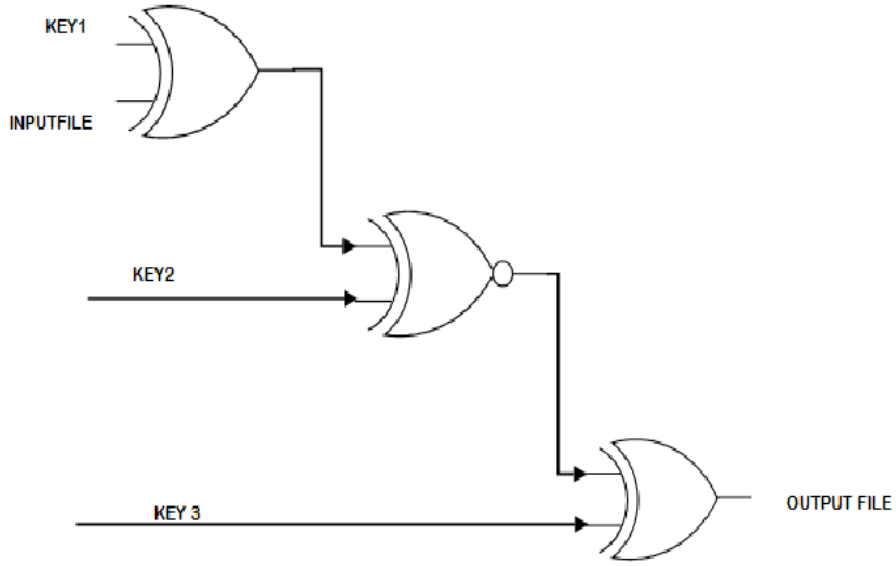


Fig 9.6: Flow Chart for Encryption System.



The diagram represented here shows how the different data has been encrypted using different keys. This was the first phase of the SSA Algorithm.

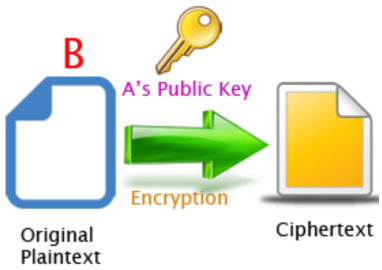


Fig 9.7: Encryption Block Diagram(Courtesy: GoogleImages).

The diagram shown above shows how the key files are encrypted using the public key pair .This was the second part of the SSA algorithm.

Decryption:Here in the first part we use asymmetric decryption method to get the keys using the private key pair and in the second part we use symmetric encryption to get the original file.

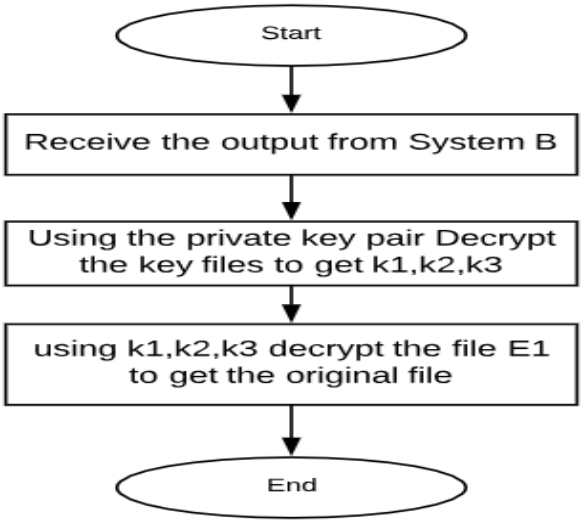


Fig 9.8: Flow Chart of Decryption System.

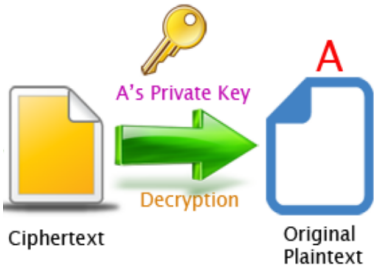
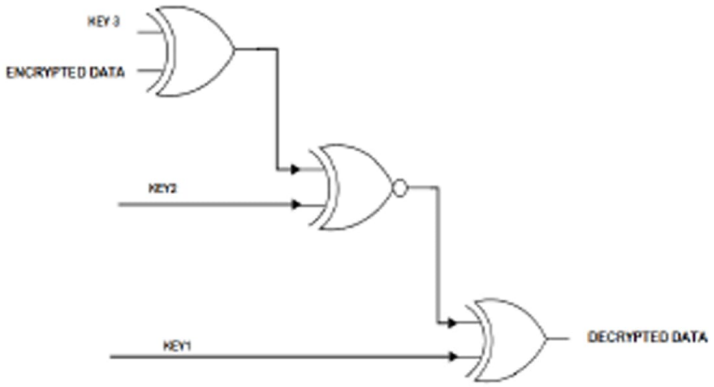


Fig 9.9: Decryption Block Diagram(Courtesy: GoogleImages).

This is the first part of the algorithm . Here the keys are retrieved using the private key pair.



This is the second part of the algorithm. Here we use those keys to decrypt the data.

7.9 Progrees of the Report

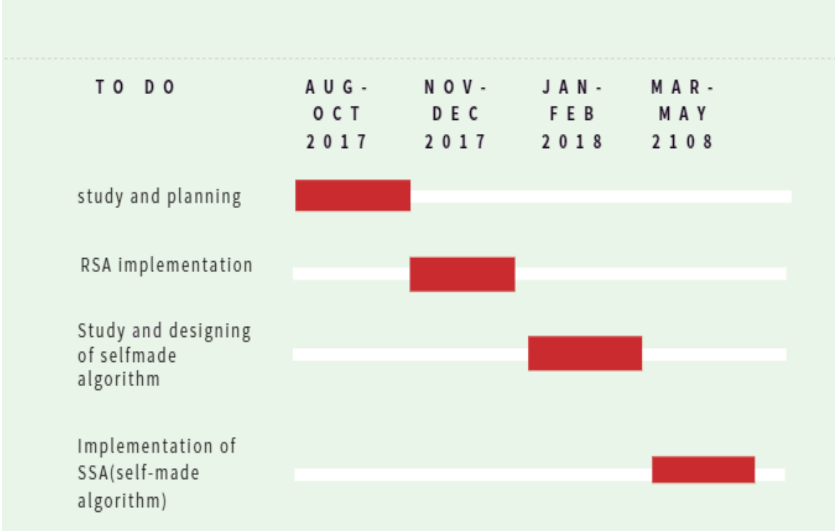


Fig 9.8: GANTT CHART for progress.

Chapter 8

Conclusion

8.1 Result

In the present academic session, we have successfully completed our project on cryptography. Here we developed a new algorithm ,named as SSA(Semi Symmetric Asymmetric Algorithm). This SSA algorithm is a combination of both symmetric and asymmetric algorithm. We used socket programming to demonstrate this. The time requirement in achieving this was more than approximated as cryptography happens to be a completely new field we have tried to experiment on and we started off with only a faint idea of the concepts involved. Therefore, the project period involved grasping basic concepts and developing the software accordingly.

8.2 Project and Report analysis

The project is all about Information Security. In the 7th semester we studied about various topics of Information security and finally we decided to choose the implementation of RSA Algorithm (Cryptography). As Cryptography was completely a new topic for us, so it took a lot time in study stage.

In the 8^{th} semester, we have decided to design a new algorithm. For this we studied the Symmetric Cryptography in detail. And finally we have ended up with a new self designed algorithm which we named as Semi Symmetric Asymmetric (SSA) algorithm. The concept of this algorithm is completely new and was implemented with the help of RSA algorithm. The algorithm is tested successfully and was well implemented using a chatting interface. Coding was done in Linux (Fedora)

platform. During the implementation coding efficiency has been taken care of.

8.3 Responsibilities in carrying out the project work

While carrying out the project work, the responsibilities were being shared equally among all the group members. Firstly, as cryptography was a new topic for us at that time, so all the group members had to do a deep study on the topic before starting the project. All the members are responsible for the overall integrity and coherence of the programme, and develops and maintains the programme environment. Finally, the group members have ensure timely and effective communication within the project.

8.4 Plagiarism Checking

Through Plagiarism Checker we have ensure that we have not stolen and publish another author's "language, thoughts, ideas, or expressions" and the representation of our own original work. The above figure shows the uniqueness of our project that was checked in the website *plagiarisma.net*

8.5 Future work

1.Analyse the Time and Space Complexity.
2.Try to reduce the Loop Holes.
3.Add features for better key availability

Bibliography

Websites:

→https://www.google.com
→https://www.wikipedia.org
→https://www.tutorialspoint.com
→https://www.geeksforgeeks.org
→https://www.youtube.com
→https://www.cs.dartmouth.edu
→https://www.slideshare.net
→https://www.cs.rpi.edu
→https://www.plagiarisma.net

Books:

→Data Communications and Networking.
→Beginning Linux Programing.
→Cryptography and Network Security.