Women University Press
Jayoti Publication Desk
ISBN No. : 978-81-950200-0-3

Probabilistic Encryption & Decryption for Message Decoding

- JV'n Nishu Sharma
- JV'n Bhawana Verma
- JV'n Swarnima

JAYOTI VIDYAPEETH WOMEN'S UNIVERSITY, JAIPUR
UGC Approved Under 2(f) & 12(b) | NAAC Accredited | Recognized by Statutory Councils

Printed by :
JAYOTI PUBLICATION DESK

Published by :
Women University Press
Jayoti Vidyapeeth Women's University, Jaipur

Faculty of Education & Methodology

Title: Probabilistic Encryption & Decryption for Message Decoding
Author Name: Ms. Nishu Sharma
Ms. Bhawana Verma
Ms. Swarnima

Published By: Women University Press

Publisher's Address: Jayoti Vidyapeeth Women's University, Jaipur
Vedant Gyan Valley,
Village-Jharna, Mahala Jobner Link Road, NH-8
Jaipur Ajmer Express Way,
Jaipur-303122, Rajasthan (India)

Printer's Detail: Jayoti Publication Desk

Edition Detail:

ISBN: 978-81-950200-0-3

Index

Chapter - 1

INTRODUCTION

1.1 Introduction

In the current scenario of the world, the technologies have advanced so much that most of the people prefer working with the internet as the main medium to transfer data or messages from one location to another in the world. There are various possible ways to transmit data through internet: via e-mails, chats, etc. The data transition is made very fast, simple, efficient and accurate via the internet.

However, the main problems while sending the data over the internet is security threat it faces i.e. the personal or confidential data can be stolen or hacked in many ways. Therefore it becomes important to consider the data security, as it is one of the most important factors that need attention in the process of data transferring.

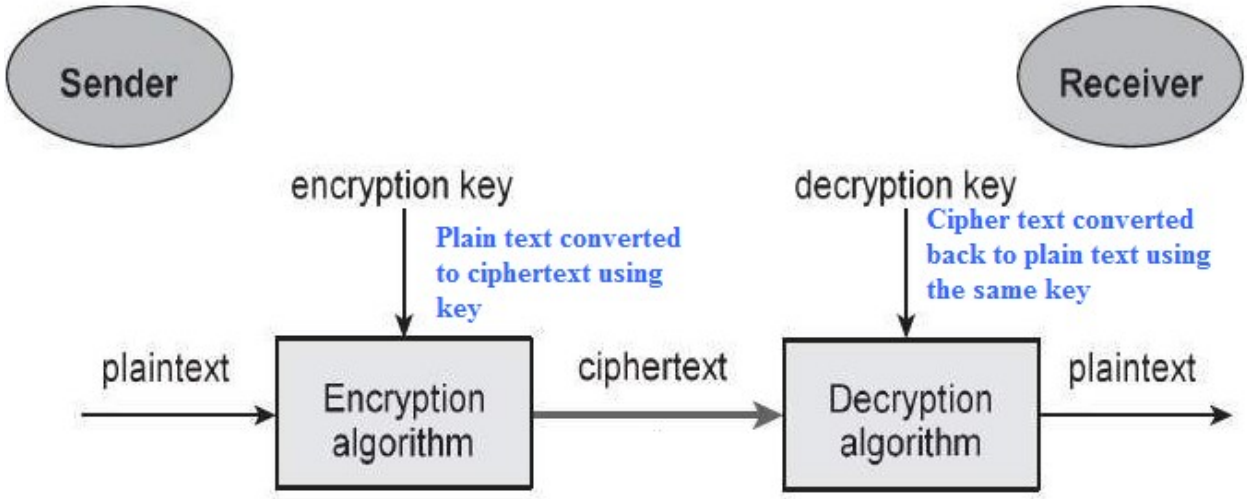
Data security means protecting the data from unauthorized users or hackers and providing high security to stop or prevent data modification. This area of data security has gained much more attention over the last few years of period of time due to the heavy increase in data transfer rate through the internet.

With the introduction and revolution in data security in communications, one more change that affected security is the introduction of distributed applications or systems which requires carrying of data between different type of users and a set of computers. Network security measures are needed to measure data protection during their transmission. The mechanisms which are used to meet the requirements of secure data transfer like authentication and confidentiality are observed are quite complex. Security mechanisms usually involves different type of algorithms or protocols for encryption & decryption purposes and for generation of different keys and subkeys to be mapped with plain text to generated cipher text. It means that participants of transmission will have some piece of secret information (Key), which will be used for protecting data from unauthorized users.

To work with security services of any company or organization, the policy makes needs to work in a systematic way. They need to make proper policy for all the aspects related to data security. Basically all the aspects might be divided in three different categories. These categories are security attack, security services and security mechanism.

In the study of security attacks, one tries to identify the different possible modes by which hackers of intruders might enter in the system for unauthorized use of resources. The security services are the ways to counter security attacks. There might be different security services for different organizations or different type of organizations. The security services choosen by one policymaker might be different than another one. There might be one or more security mechanism to provide the service.

Different type of mechanism provides different services specified. There is no single mechanism which can be applied on all specified services. The general form of mechanism that supports all type of information security is cryptographic technique. Encryption of information or data is the one of the most common way of providing data security. The general model for encryption can be represented by following figure.:



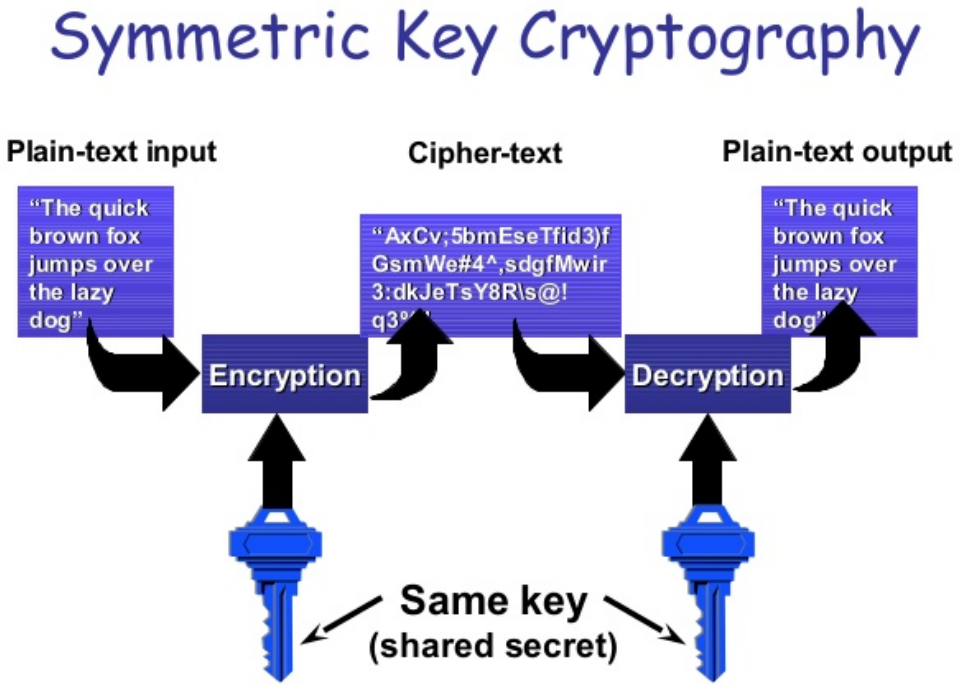
By above diagram, all the basic tasks for designing a particular security service can be divided in following four tasks:

1. Designing algorithms for encryption and decryption

2. Generation of secret key information with the help of above step 1
3. Sharing and distribution for secret information
4. Designing protocols so that both communicating parties can communicate.

A crypto system is simply an algorithm with all possible plain texts, cipher texts and keys. Generally key based algorithms can be categorized in two different types of key based algorithms: symmetric keys and public keys.

In most symmetric algorithms, the same key is used for encryption and decryption both, as shown in Figure:



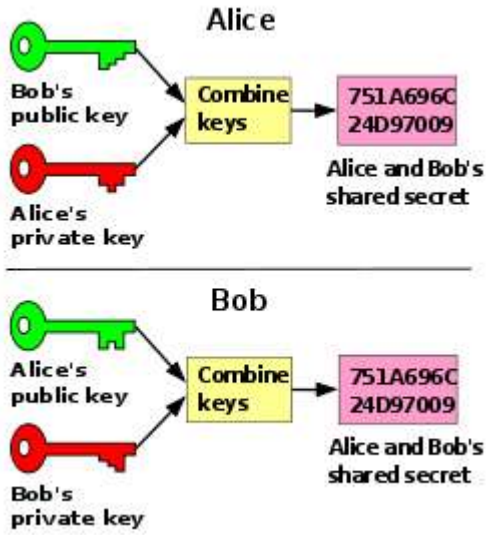
Symmetric-key encryption

The process of symmetric-key encryption can be very fast because the communicating parties don't get any time delay.

Symmetric algorithms can be categorized into two different categories. Some operate on plain text a single bit or a byte at a time, these are called stream cipher algorithms or stream ciphers.

Others operate on a number of bits (typically 64 bits are used in modern ciphers), and encrypt the whole as a single unit. Such algorithms are known as block algorithms.

Asymmetric encryption or public key algorithms uses different keys for encryption and decryption, and the decryption key cannot directly (practically) be calculated or derived from the encryption key. First key which is also known as public key can be declared public and the second one is private that means the key is known only to any specific participating party. Public key cryptography can also be used for digital signing as it supports authentication of users. The information encrypted with one public key will only be decrypted with the other key. Also the decryption key or private which will be used by counter party, cannot be calculated from the public and encryption key. Following figure shows a simplified view of the public-key encryption or asymmetric encryption.



Probabilistic data encryption method was introduced by Goldwasser and Micali. In this method, every message has many different possible encodings schemes and every bit of message is encrypted independently. . They use the predicate "quadratic residue modulo *n".* If security parameter is k then each bit is coded separately by a *k*-bit long string and even, it results in at least a *k*-bit data expansion factor.

Given a composite integer n = p*q and a Z^*_n with (a/n) = 1, decide whether or not a is a quadratic residue modulo n. There is no efficient procedure known for solving the quadratic residuacity

problem if the factorization of n is unknown. This problem is based on the Quadratic residuacity assumption which states that for sufficiently large primes p and q for real-life algorithm it is infeasible to solve Quadratic Residuacity Problem, but if the factorization of n = pq is known, it is easy to solve QRP by computing (a/p), since a is pseudo square if and only if (a/p) = (a/q) = -1. Our encryption scheme is based on the function that maps elements of Z^*_n to quadratic residues modulo n.

Public key methods are important because they can be used for transmitting encryption keys or other data securely even when the parties have no opportunity to agree on a secret key in private. In symmetric-key encryption, security of data is done by common key which is shared by both of participating parties. It provides a good degree of authentication, since the information which is encrypted with one symmetric key cannot be decrypted with any other key. Thus, if the symmetric key is kept secret by the two communicating parties using it to encrypt communications, each party can be confident that it is communicating with the other as long as the decrypted messages specify a meaningful sense.

Sender's Message

Plain text converted to cipher text using private key Cipher text converted back to plain text using the same private key

Receiver's message

Symmetric-key encryption will be successful only if the symmetric key is kept secured by the two parties involved. If anyone else discovers the key, it affects both confidentiality and authentication. The success of a symmetric algorithm rests in the key, divulging the key means that any one could encrypt and decrypt messages. As long as the communication needs to remain secure, the key must be protected between the participating parties.

Encryption and decryption with a symmetric algorithm are denoted by

E K (M) = C

D K (M) = P

Symmetric algorithms can be divided into two categories. Some operate on the plain text a single bit or byte at a time, these are called stream algorithms or stream ciphers. Others operate on group of bits or characters. Such algorithms are called block algorithms. Public Key algorithms use two keys, one key for encryption and the other for decryption. One key can be called as public key which can be declared public and the other one is private that is, the key is known only to the particular participating party. And also public key cryptography can be used for digital signing as it supports authentication of users. The information encrypted with one key will only be decrypted with the other key. Further more the decryption key cannot be calculated from the encryption key. Figure 1.3 shows a simplified view of the way public-key encryption works.

Figure 1.3 Public-key encryption

Sender's Message

Plain text converted to cipher text using public key of receiver Cipher text converted back to plain text using the private key of receiver

Receiver's message

Compared with symmetric-key encryption, public-key encryption requires more computation and is therefore not always appropriate for large amounts of data. However, it's possible to use public-key encryption to send a symmetric key, which can then be used to encrypt additional data. This is the approach used by the SSL protocol. This provides Authentication, Integrity & Confidentiality of Information at low computing power. Since authentication of the users is very important in applications like ecommerce and other similar applications, public key cryptography is of much use.

Encryption and decryption can be represented in a public key scheme is

E Kpu(M) = C

D Kpr(C) = M

Where Kpu is the public key and Kpr is the private key. In public key encryption there is always a possibility of some information being leaked out. A crypto analyst tries to get some information based on ones public key. Complete information can not be gained here but a part of

information may be gained. In probabilistic Encryption, multiple cipher texts are generated for one plain text, a cryptanalyst can not generate any information by chosen plain text and chosen cipher text attacks.

Figure 1.4 Probabilistic encryption

Sender's Message

Plain text converted to multiple cipher texts using key Multiple Cipher texts converted back to plain text using the same key

Receiver's message

Security Analysis of algorithms: Different algorithms offers different degrees of security, it depends on how hard they are to break. If the cost required to break an algorithm is greater than the value of the encrypted data, then the algorithm is supposed to be safe. If the time required breaking an algorithm is longer than the time that the encrypted data must remain secret, and then also it is safe. If the amount of data encrypted with a single key is less than the amount of data necessary to break the algorithm, it is supposed to be safe.

An algorithm is unconditionally secure if, it is difficult to recover the plain text in spite of having substantial amount of cipher text. In such circumstances, only a one time pad is unbreakable in a cipher text only attack, simply by trying every possible key one by one and by checking whether the resulting plain text is meaningful. This is called a brute force attack. Cryptography is more concerned with crypto systems that are computationally infeasible to break. Any algorithm is considered computationally secure if it cannot be broken with available resources.

The complexity of an attack can be measured as Data Complexity, the amount of data needed as input to the attack, processing complexity, the time needed to perform the attack and storage requirements which are the amount of memory needed to do the attack which is space complexity.

As a thumb rule, the complexity of an attack is taken to be minimum of these three factors. Another classification of complexities is by complexity of the algorithm by its construction and complexity of the algorithm by its strength. By its construction, the time complexity of the

algorithm can be calculated by executing through the steps of the algorithm, which will be referred as O(n). Complexities can also be expressed as orders of magnitude. If the length of the key is k, then the processing complexity is given by 2k .

It means that 2 k operations are required to break the algorithm. Then the complexity of the algorithm is said to be exponential in nature. A desirable property of any encryption algorithm is that a small change in plain text or the key should produce significant change in cipher text. Such an effect is known as avalanche effect. The more the avalanche affects of the algorithm, the better the security. Crypto analysis is the study of recovering the plain text with out access to the key. It may also find weakness in a crypto system that identifies patterns which can be useful in knowing the previous results.

An attempted crypto analysis is called an attack. There are five types of attack. Each of them assumes that the crypto analyst has complete knowledge of the encryption algorithm used.

1. Cipher text only attack: Here the intruder is in hold of cipher text only. The crypto analyst has cipher text of several messages, all of which have been encrypted using the same encryption algorithm. The crypto analyst's job is to recover the plain text or the key used to encrypt the messages, in order to decrypt other part of messages encrypted with the same keys.

2. Known Plaintext attack: The crypto analyst is in possession of pairs of known plain text and cipher text. His job is to get the key used to encrypt the messages or an algorithm to decrypt any messages encrypted with the same key.

3. Chosen Plaintext Attack (CPA): Here the crypto analyst is in hold of not only cipher text but also parts of chosen plain text. Here the intruder is identified to be placed at encryption site to do the attack. Differential crypto analysis is an example of this mode.

4. Chosen cipher text attack (CCA): Under the CCA model, the crypto analyst is in possession of chosen cipher text and corresponding plain text being decrypted from the private key. After it has chosen the messages, however, it only has access to an encryption machine.

5. Chosen text: In this model, the analyst posses the encipher algorithm, Cipher text to be decrypted, chosen plain text messages and corresponding cipher texts, fabricated cipher text with the corresponding decrypted plain texts developed by the private key.

1.2 Present work:

In this work an attempt has been made to generate two algorithms which provide security to data transmitted. In the work we will discuss various algorithms required for encryption and decryption using probabilistic encryption. In this work both the algorithms are discussed in terms of computational security, computational complexity and computational overhead. Both the algorithms are studied for their strengths and limitations. A crypto analytical study of the algorithms with emphasis on probabilistic encryption is also considered in this study. The encryption algorithms are compared with standard algorithms like RC4 and DES. The algorithms are also discussed in terms of its applications and also about their advantages and limitations in network security environment.

1.2 Objectives of Proposed Study

- To discuss the method of encryption and decryption algorithm of probabilistic encryption.
- To discuss the method and examples of probabilistic encryption
- To develop a software system(Java Based) for encryption and decryption of probabilistic encryption.

1.4 Outline of This Book

In chapter 1, the general overview is given with proposed study and outline of the book.

Chapter 2 introduces detailed literature review. Chapter 3 discussees about the probabilistic encryption with its methods and some of the examples. Chapter 4 describes screenshots and code of java application which is developed for probabilistic encryption In Chapter 5, conclusion and future work is given. At last references are shown.

Chapter- 2

LITERATURE REVIEW

A crypto system **Amjay Kumar et al 2009,** is an algorithm which include all possible plain texts, cipher texts and keys. There are two general types of key based algorithms: symmetric and public key.

2.1 Symmetric Encryption Schemes:

With symmetric-key encryption, the encryption key can be calculated from the decryption key and vice versa. With most symmetric algorithms, the same key is used for both encryption and decryption. Implementations of symmetric key encryption can be highly efficient, so that users do not experience any significant time delay as a result of the encryption and decryption. Symmetric-key encryption also provides a degree of authentication, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key. Thus, as long as the symmetric key is kept secret by the two parties using it to encrypt communications, each party can be sure that it is communicating with the other as long as the decrypted messages continue to make sense.

Encryption functions normally take a fixed-size input to a fixed-size output, so encryption of longer units of data must be done in one of two ways: either a block is encrypted at a time and the blocks are somehow joined together to make the cipher text,

2.1.1 Block ciphers

Block ciphers take as input the key and a block, often the same size as the key. Further, the first block is often augmented by a block called the initialization vector, which can add some randomness to the encryption.

2.1.1.1 DES Algorithm:

The most widely used encryption scheme is based on Data Encryption Standard (DES). There are two inputs to the encryption function, the plain text to be encrypted and the key. The plain text must be 64 bits in length and key is of 56 bits. First, the 64 bits of plain text passes through

an initial permutation that rearranges the bits. This is fallowed by 16 rounds of same function, which involves permutation & substitution functions. After 16 rounds of operation, the pre output is swapped at 32 bits position which is passed through final permutation to get 64 bit cipher text.

Initially the key is passed through a permutation function. Then for each of the 16 rounds, a sub key is generated by a combination of left circular shift and permutation. At each round of operation, the plain text is divided to two 32 bit halves, and the fallowing operations are executed on 32 bit right halve of plain text. First it is expanded to 48 bits using a expansion table, then X-ORed with key, then processed in substitution tables to generate 32 bit output. This output is permuted using predefined table and XORed with left 32 bit plain text to form right 32 bit pre cipher text of first round. The right 32 bit plain text will form left 32 bit pre cipher text of first round. Decryption uses the same algorithm as encryption, expect that the application of sub keys is reversed. A desirable property of any encryption algorithm is that a small change in either plain text or the key should produce a significant change in the cipher text. This effect is known as Avalanche effect which is very strong in DES algorithm. Since DES is a 56 bit key encryption algorithm, if we proceed by brute force attack, the number of keys that are required to break the algorithm is 2^{56} . But by differential crypto analysis, it has been proved that the key can be broken in 2^{47} combinations of known plain texts. By linear crypto analysis it has been proved that, it could be broken by 2^{41} combinations of plain text.

The DES algorithm is a basic building block for providing data security. To apply DES in a variety of applications, four modes of operations have been defined. These four models are intended to cover all possible applications of encryption for which DES could be used. They involve using a initialization vector being used along with key to provided different cipher text blocks.

2.1.1.1.1 Electronic Code Book (ECB) mode: ECB mode divides the plaintext into blocks m1, m2, ..., mn, and computes the cipher text ci = Ei(mi). This mode is vulnerable to many attacks and is not recommended for use in any protocols. Chief among its defects is its vulnerability to

splicing attacks, in which encrypted blocks from one message are replaced with encrypted blocks from another.

2.1.1.1.2 Cipher Block Chaining (CBC) mode: CBC mode remedies some of the problems of ECB mode by using an initialization vector and chaining the input of one encryption into the next. CBC mode starts with an initialization vector iv and XORs a value with the plaintext that is the input to each encryption. Since each block depends on all previous blocks along with the initialization vector. The is a good example of a nonce that needs to satisfy Uniqueness but not unpredictability.

2.1.1.1.3 Cipher Feed-Back (CFB) mode: CFB mode moves the XOR of CBC mode to the output of the encryption. In other words, the cipher text c1 = p1 XOR Sj(E(IV)). This mode then suffers from failures of Non-Malleability, at least locally to every block, but changes to cipher text do not propagate very far, since each block of cipher text is used independently to XOR against a given block to get the plaintext.

These failures can be seen in the following example, in which a message m = m1 m2 ... mn is divided into n blocks, and encrypted with an iv under CFB mode to c1 c2 ... cn. Suppose an adversary substitutes c'2 for c2. Then, in decryption, m1 = Ek(iv) XOR c1, which is correct, but m'2 = Ek(c1) XOR c'2, which means that m'2 = m2 XOR c2 XOR c'2, since m2 = Ek(c1) XOR c2. Thus, in m2, the adversary can flip any bits of its choice. Then m'3 = Ek(c'2) XOR c3, which should lead to random looking message not under the adversary's control, since the encryption of c'2 should look random. But m4 = Ek(c3) XOR c4 and thereafter the decryption is correct.

2.1.1.1.4 Output Feed-Back (OFB) mode OFB mode modifies CFB mode to
feed back the output of the encryption function to the encryption function without XORing the cipher text.

2.1.1.2 Triple DES:
Given the potential vulnerability of DES to brute force attack, a new mechanism is adopted which uses multiple encryptions with DES and multiple keys. The simplest form of multiple

encryptions has two encryption stages and two keys. The limitation with this mechanism is it is susceptible to meet in the middle attack. An obvious counter to meet in the middle attack and reducing the cost of increasing the key length, a triple encryption method is used, which considers only two keys with encryption with the first key, decryption with the second key and fallowed by encryption with the first key. Triple DES is a relatively popular alternative to DES and has been adopted for use in key management standards.

2.1.1.3 Homomorphic DES:

A variant of DES called a homophonic DES **Carlone Fontaine et al,** is considered. The DES algorithm is strengthened by adding some random bits into the plaintext, which are placed in particular positions to maximize diffusion, and to resist differential attack. Differential attack makes use of the exclusive-or homophonic DES. In this new scheme, some random estimated bits are added to the plaintext. This increases the certain plaintext difference with respect to the cipher text.

A homophonic DES is a variant of DES that map search plaintext to one of many cipher texts (for a given key). In homophonic DES a desired difference pattern with the cipher text will be suggested with some key values including the correct one, oppositely wrong pairs of cipher text. For a difference pattern which 56-bit plaintext to a 64-bit cipher text using a 56-bit key. In this scheme, eight random bits are placed in specific positions of the 64-bit input data block to maximize diffusion.

For example, the random bits in HDESS are the bit- positions 25, 27, 29, 31, 57, 59, 61 and 63. In this algorithm, after the initial permutation and expansion permutation in the first round, these eight random bits will spread to bits 2, 6, 8, 12, 14, 18, 20, 24, 26, 30, 32, 36, 38,42,44,48 of the 48-bit input block to the S-boxes and will affect the output of all the S-boxes. The 48 expanded bits must be exclusive-or'd with some key before proceeding to the S-boxes, thus two input bits into the S-boxes derived from the same random bit may have different values. This says that the random bits do not regularize the input to the S-boxes, that is, the property of confusion does not reduce while we try to maximize diffusion.

The decryption of the homophonic DES is similar to the decryption of DES. The only difference is that eight random bits must be removed to get the original plaintext (56 bits). A homophonic DES can easily be transformed into a triple-encryption version by concatenating a DES decryption and a DES encryption after the homophonic DES.

Security analysis: Thus there is a probability of 1/256 between a pair of texts. The differential crypto analysis is also difficult on this mechanism. The diffusion of bits is also more in this mode. Thus this mechanism provides some probabilistic features to DES algorithm which makes it stronger from differential and linear crypto analysis.

2.1.1.4 AES:

The Advanced Encryption Standard (AES) was chosen in 2001. AES is also an iterated block cipher, with 10, 12, or 14 rounds for key sizes 128, 192, and 256 bits, respectively. AES provides high performance symmetric key encryption and decryption.

2.1.1.5 Dynamic substitution:

An apparently new cryptographic mechanism **Terry Ritter 1999,** which can be described as dynamic substitution is discussed in the fallowing topic. Although structurally similar to simple substitution, dynamic substitution has a second data input which acts to re-arrange the contents of the substitution table. The mechanism *combines* two data sources into a complex result; under appropriate conditions, a related inverse mechanism can then *extract* one of the data sources from the result. A dynamic substitution combiner can directly replace the exclusive-OR combiner used in Vernam stream ciphers. The various techniques used in Vernam ciphers can also be applied to dynamic substitution; any cryptographic advantage is thus due to the additional strength of the new combiner.

2.1.1.5.1 The Vernam Cipher: A Vernam cipher maps plaintext data with a pseudo-random sequence to generate cipher text. Since each ciphertext element from a Vernam combiner is the (mod 2) sum of two unknown values, the plaintext data is supposed to be safe. But this mode is susceptive to several cryptanalytic attacks, including known plain text and cipher text attacks. And if the confusion sequence can be penetrated and reproduced, the cipher is broken. Similarly,

if the same confusion sequence is ever re-used, and the overlap identified, it becomes simple to break that section of the cipher.

2.1.1.5.2 Cryptographic Combiners: An alternate approach to the design of a secure stream cipher is to seek combining functions which can resist attack; such functions would act to hide the pseudo-random sequence from analysis.

The mechanism of this work is a new combining function which extends the weak classical concept of simple substitution into a stronger form suitable for computer cryptography.

2.1.1.5.3 Substitution Ciphers: In simple substitution ciphers each plain text character is replaced with fixed cipher text character. But this mechanism is weak from statistical analysis methods where by considering the rules of the language, the cipher can be broken. This work is concerned with the cryptographic strengthening of the fundamental substitution operation through *dynamic* changes to a substitution table. The substitution table can be represented as a function of not only input data but also a random sequence. This combination gives a cryptographic combining function; such a function may be used to combine plaintext data with a pseudo-random sequence to generate enciphered data.

2.1.1.5.4 Dynamic Substitution: A simple substitution table supported with combining function gives the idea of dynamic substitution. A substitution table is used to translate each data value into an enciphered value. But after each substitution, the table is re-ordered. At a minimum, it makes sense to exchange the just-used substitution value with some entry in the table selected at random. This generally changes the just-used substitution value to help prevent analysis, and yet retains the existence of an inverse, so that the cipher can be deciphered.

2.1.1.5.5 Black Box Analysis: Dynamic substitution may be considered to be a *black box*, with two input ports Data In and Random In, and one output port Combiner Out. In the simple version, each data path has similar width; evidently the mechanism inside the box in some way *combines* the two input streams to produce the output stream. It seems reasonable to analyze the output statistically, for various input streams.

2.1.1.5.6 Polyalphabetic Dynamic Substitution: A means to defend to known plain text and chosen-plaintext attacks would be to use multiple different dynamic substitution maps and to select between them using a hidden pseudo-random sequence. Thus the dynamic substitution if free from statistical attacks where each character of plain text is replaced with multiple characters of cipher text which makes the mechanism robust.

2.1.1.5.7 Internal State: Dynamic substitution contains internal data which after initialization is continuously re-ordered as a consequence of both incoming data streams; thus, the internal state is a function of initialization and all subsequent data and confusion values. The changing internal state of dynamic substitution provides necessary security to the data streams.

Thus dynamic substitution provides a probabilistic nature to the enciphering mechanism. The limitation with this scheme is, not only different dynamic substitution tables has to be maintained but also the pseudo random sequence which selects between these dynamic substitution tables has to be shared between sender and receiver**.**

2.1.1.6 Nonces

Phillip Rogaway, A nonce is a bit string that satisfies Uniqueness, which means that it has not occurred before in a given run of a protocol. Nonces might also satisfy Unpredictability, which effectively requires pseudo-randomness: no adversary can predict the next nonce that will be chosen by any principal. There are several common sources of nonces like counters, time slots and so on.

2.1.1.6.1 Nonce Based Encryption: In this work a different formalization for symmetric encryption is envisaged. The encryption algorithm is made to be a deterministic function, but it is supported with initialization vector (IV). Efficiency of the user is made success of this mode. The IV is a nonce like value, used at most once within a session. Since it is used at most once having any sort of crypto analysis is practically not possible which provides sufficient security.

2.1.1.7 One-Time Pad Encryption

One more encryption mechanism for providing security to data is one time pad **Henry Baker et al 1982,** encryption. The functions are computed as follows: A and B agree on a random number k that is as long as the message they later want to send.

Ek(x) = x XOR k

Dk(x) = x XOR k

Note that since k is chosen at random and not known to an adversary, the output of this scheme is indistinguishable to an adversary from a random number. But it suffers from several limitations. It is susceptible to chosen plain text and chosen cipher text attacks. Again the limitation is here is sharing of one time keys by the participating parties of the encryption scheme. As a new key is always used for encryption, a continuous sharing of key mechanism has to be employed by the participating parties.

2.1.2 Stream ciphers

Unlike block ciphers, stream ciphers **J.William stalling 1998,** (such as RC4) produce a pseudorandom sequence of bits that are then combined with the message to give an encryption. Since the combining operation is often XOR, naive implementations of these schemes can be vulnerable to the sort of bit-flipping attacks on Non-Malleability. Two types of stream ciphers exist: synchronous, in which state is kept by the encryption algorithm but is not correlated with the plaintext or cipher text, and self synchronizing, in which some information from the plaintext or cipher text is used to inform the operation of the cipher.

2.1.2.1 RC4 Encryption Algorithm:

Ronald Rivest of RSA developed the RC4 algorithm, which is a shared key stream cipher algorithm requiring a secure exchange of a shared key. The algorithm is used identically for encryption and decryption as the data stream is simply XORed with the generated key sequence. The algorithm is serial as it requires successive exchanges of state entries based on the key sequence. Hence implementations can be very computationally intensive. In the algorithm the key stream is completely independent of the plaintext used. An 8 * 8 S-Box (S0 S255), where

each of the entries is a permutation of the numbers 0 to 255, and the permutation is a function of the variable length key.
There are two counters i, and j, both initialized to 0 used in the algorithm.

2.1.2.1.1 Algorithm Features:

1.It uses a variable length key from 1 to 256 bytes to initialize a 256-byte state table. The state table is used for subsequent generation of pseudo-random bytes and then to generate a pseudo-random stream which is XORed with the plaintext to give the cipher text. Each element in the state table is swapped at least once.

2. The key is often limited to 40 bits, because of export restrictions but it is sometimes used as a 128 bit key. It has the capability of using keys between 1 and 2048 bits. RC4 is used in many commercial software packages such as Lotus Notes and Oracle Secure.

3. The algorithm works in two phases, key setup and ciphering. During a N-bit key setup (N being your key length), the encryption key is used to generate an encrypting variable using two arrays, state and key, and N-number of mixing operations. These mixing operations consist of swapping bytes, modulo operations, and other formulas.

2.1.2.1.2 Algorithm Strengths:

The difficulty of knowing which location in the table is used to select each value in the sequence. A particular RC4 Algorithm key can be used only once and Encryption is about 10 times faster than DES.

Algorithm Weakness: One in every 256 keys can be a weak key. These keys are identified by cryptanalysis that is able to find circumstances under which one of more generated bytes are strongly correlated with a few bytes of the key. Thus some symmetric encryption algorithms have been discussed in this chapter.

They varies from block ciphers like DES, Triple DES, Homomorphic DES to stream ciphers like RC4. To the symmetric encryption mechanisms concepts like application of Nounce and

dynamic substitution are discussed which provides randomness to the encryption mechanism. This probabilistic nature to the encryption mechanism provides sufficient strength to the algorithms against Chosen Cipher text attacks(CCA). The security with all these mechanisms lies with proper sharing of keys among the different participating parties.

2.1.3 Adoptability of some mathematical functions in Cryptography:

Sign Function: Pandit S.N.N 1963, This function when applied on when applied on a matrix of values, converts all the positive values to 1, negative values to -1 & zero with 0. The advantage of using this function in cryptography is it cannot be a reversible process ie we cannot get back to the original matrix by applying a reverse process.

Modular Arithmetic: One more function that is widely used in cryptography is modular arithmetic of a number with a base value. It will generate the remainder of a number with respect to the base value. This function is widely used in public key cryptography.

2.2 Public-Key Encryption

The most commonly used implementations of public-key **Henry Baker et al 1982,** encryption are based on algorithms patented by RSA Data Security. Therefore, this section describes the RSA approach to public-key encryption.

Public-key encryption (also called *asymmetric encryption*) involves a pair of keys a *public key* and a *private key*, used for security & authentication of data. Each public key is published, and the corresponding private key is kept secret. Data encrypted with one key can be decrypted only with other key.

The scheme shown in Figure 1.2 says public key is distributed and encryption being done using this key. In general, to send encrypted data, one encrypt's the data with the receiver's public key, and the person receiving the encrypted data decrypts it with his private key.

Compared with symmetric-key encryption, public-key encryption requires more computation and is therefore not always appropriate for large amounts of data. However, a combination of

symmetric & Asymmetric schemes can be used in real time environment. This is the approach used by the SSL protocol.

As it happens, the reverse of the scheme shown in Figure 1.2 also works: data encrypted with one's private key can be decrypted only with his public key. This may not be an interesting way to encrypt important data, however, because it means that anyone with receiver's public key, which is by definition published, could decipher the data. And also the important requirement with data transfer is authentication of data which is supported with Asymmetric encryption schemes, which is an important requirement for electronic commerce and other commercial applications of cryptography.

2.2.1 Key Length and Encryption Strength:

In general, the strength of encryption algorithm depends on difficulty in getting the key, which in turn depends on both the cipher used and the length of the key. For the RSA cipher, the strength depends on the difficulty of factoring large numbers, which is a well-known mathematical problem. Encryption strength is often described in terms of the length of the keys used to perform the encryption, means the more the length of the key, the more the strength. Key length is measured in bits. For example, a RC4 symmetric-key cipher with key length of 128 bits supported by SSL provide significantly better cryptographic protection than 40-bit keys for use with the same cipher. It means 128-bit RC4 encryption is 3 x 1026 times stronger than 40-bit RC4 encryption. Different encryption algorithms require variable key lengths to achieve the same level of encryption strength.

Other ciphers, such as those used for symmetric key encryption, can use all possible values for a key of a given length, rather than a subset of those values. Thus a 128-bit key for use with a symmetric-key encryption cipher would provide stronger encryption than a 128-bit key for use with the RSA public-key encryption cipher. This says that a symmetric encryption algorithm with a key length of 56 bits achieve a equal security to Asymmetric encryption algorithm with a key length of 512 bits,

2.2.2 RSA Key Generation Algorithm

1. Two large prime numbers are considered. Let them be p,q.
2. Calculate n = pq and (_) phi = (p-1)(q-1).
3. Select e, such that 1 < e < phi and gcd(e, phi) = 1.
4. Calculate d, the private key, such that de =1 mod phi.

One key is (n, e) and the other key is (n, d). The values of p, q, and phi should also be kept secret.

• n is known as the *modulus*.

• e is known as the *public key*.

• d is known as the *secret key*.

Encryption

Sender A does the following:-

1. Get the recipient B's public key (n, e).
2. Identify the plaintext message as a positive integer m.
3. Calculate the ciphertext c = m^e mod n.
4. Transmits the ciphertext c to receiver B.

Decryption

Recipient B does the following:-

1. Consider his own private key (n, d) to compute the plain text m = c^d mod n.
2. Convert the integer to plain text form.

2.2.3 Digital signing

Sender A does the following:-

This concept can also be used in digital signing as well. The message to be transmitted is converted to some message digest form. This message digest is converted to encryption form using his private key. This encrypted message digest is transmitted to receiver.

Signature verification

Recipient B does the following:-

1. Using the sender's public key, the received message digest is decrypted. From the received message, the receiver independently computes the message digest of the information that has been signed.
2. If both message digests are identical, the signature is valid. Compared with symmetric-key encryption, public-key encryption provides authentication & security to the data transmitted but requires more computation and is therefore not always appropriate for large amounts of data.

2.3. Probabilistic encryption schemes

In public key encryption there is always a possibility of some information being leaked out. Because a crypto analyst can always encrypt random messages with a public key, he can get some information. Not a whole of information is to be gained here, but there are potential problems with allowing a crypto analyst to encrypt random messages with public key. Some information is leaked out every time to the crypto analyst, he encrypts a message.

With probabilistic encryption algorithms **Georg J.Fuchsbauer 2006,** a crypto analyst can no longer encrypt random plain texts looking for correct cipher text. Since multiple cipher texts will be developed for one plain text, even if he decrypts the message to plain text, he does not know how far he had guessed the message correctly. To illustrate, assume a crypto analyst has a certain cipher text ci. Even if he guesses message correctly, when he encrypts message the result will be completely different cj. He cannot compare ci and cj and so cannot know that he has guessed the message correctly. Under this scheme, different cipher texts will be formed for one plain text. Also the cipher text will always be larger than plain text. This develops the concept of multiple cipher texts for one plain text. This concept makes crypto analysis difficult to apply on plain text and cipher text pairs.

An encryption scheme consists of three algorithms: The encryption algorithm transforms plaintexts into cipher texts while the decryption algorithm converts cipher texts back into plaintexts. A third algorithm, called the key generator, creates pairs of keys: an encryption key, input to the encryption algorithm, and a related decryption key needed to decrypt. The encryption key relates encryptions to the decryption key. The key generator is considered to be a probabilistic algorithm, which prevents an adversary from simply running the key generator to

get the decryption key for an intercepted message. The following concept is crucial to probabilistic cryptography:

2.3.1 Definition [Probabilistic Algorithm]:

Georg J.Fuchsbauer 2006, A probabilistic algorithm is an algorithm with an additional command RANDOM that returns “0” or “1”, each with probability 1/2. In the literature, these random choices are often referred to as coin flips.

2.3.1.1 Chosen Cipher Text Attack:

In the simplest attack model, known as Chosen Plaintext Attack (CPA) **Brassard G**, the adversary has access to a machine that will perform arbitrary encryptions but will not reveal the shared key. This machine corresponds intuitively to being able to see many encryptions of many messages before trying to decrypt a new message. In this case, Semantic Security requires that it be computationally hard for any adversary to distinguish an encryption Ek(m) from Ek(m') for two arbitrarily chosen messages m and m'. Distinguishing these encryptions should be hard even if the adversary can request encryptions of arbitrary messages. Note that this property cannot be satisfied if the encryption function is deterministic! In this case, the adversary can simply request an encryption of m and an encryption of m' and compare them. This is a point that one should all remember when implementing systems: encrypting under a deterministic function with no randomness in the input does not provide Semantic Security. One more crypto analytical model is Chosen Cipher text Attack (CCA) Model. Under the CCA model, an adversary has access to an encryption and a decryption machine and must perform the same task of distinguishing encryptions of two messages of its choice. First, the adversary is allowed to interact with the encryption and decryption services and choose the pair of messages. After it has chosen the messages, however, it only has access to an encryption machine. An advancement to CCA Model is Chosen Cipher text Attack 2 (CCA2). CCA2 security has the same model as CCA security, except that the adversary retains access to the decryption machine after choosing the two messages. To keep this property from being trivially violated, we require that the adversary not be able to decrypt the cipher text it is given to analyze.

To make these concepts of CCA & CCA2 adoptable in real time environment, recently Canetti, Krawczyk and Nielsen defined the notion of replayable adaptive chosen ciphertext attack **Brics 2004,** secure encryption. Essentially a cryptosystem that is RCCA secure has full CCA2 security except for the little detail that it may be possible to modify a ciphertext into another ciphertext containing the *same* plaintext. This provides the possibility of *perfectly* replayable RCCA secure encryption. By this, we mean that anybody can convert a ciphertext *y* with plaintext *m* into a different ciphertext *y* that is distributed identically to a fresh encryption of *m*. It propose such a rerandomizable cryptosystem, which is secure against semi-generic adversaries. To improve the efficiency of the algorithm, a probabilistic trapdoor one way function is presented. This adds randomness to the proposed work which makes crypto analysis difficult.

2.3.1.2 Neural networks in cryptography:

One more technique that is used in probabilistic encryption is to adopt Neural Networks **Guo D et al 1999,** on encryption mechanisms. Neural network techniques are added to probabilistic encryption to make cipher text stronger. In addition to security it can also be seen that data over head could be avoided in the conversion process A new probabilistic symmetric probabilistic encryption scheme based on chaotic attractors of neural networks can be considered. The scheme is based on chaotic properties of the Over storaged Hopfield Neural Network (OHNN). The approach bridges the relationship between neural network and cryptography. However, there are some problems in the scheme:

(1) exhaustive search is needed to find all the attractors;

(2) problem exists on creating the synaptic weight matrix.

2.3.1.3 Knapsack-based crypto systems:

Knapsack-based cryptosystems **Baocang Wang et al 2007,** had been viewed as the most attractive and the most promising asymmetric cryptographic algorithms for a long time due to their NPcompleteness nature and high speed in encryption/decryption. Unfortunately, most of them are broken for the low-density feature of the underlying knapsack problems. To improve the performance of the model a new easy compact knapsack problem and propose a novel knapsack-based probabilistic public-key cryptosystem in which the cipher-text is non-linear with the plaintext.

2.3.1.4 On Probabilistic Scheme for Encryption Using Nonlinear Codes Mapped from Z_4 Linear Codes:

Probabilistic encryption becomes more and more important since its ability to against chosen-cipher text attack. To convert any deterministic encryption scheme into a probabilistic encryption scheme, a randomized media is needed to apply on the message and carry the message over as an randomized input **Lester S. Hill 1929**. Thus nonlinear codes obtained by certain mapping from linear error-correcting codes are considered to serve as such carrying media. Thus some algorithms are discussed in literature which is symmetric and probabilistic in nature.

2.4 Numerical Model for data development

2.4.1 Partial differential equations: Partial differential equations to model multiscale phenomena are ubiquitous in industrial applications and their numerical solution is an outstanding challenge within the field of scientific computing **Suhas V. Patenkar 1991**. The approach is to process the mathematical model at the level of the equations, before discretization, either removing non-essential small scales when possible, or exploiting special features of the small scales such as self-similarity or scale separation to formulate more tractable computational problems. Types of data ,

1.Static: Each data item is considered free from any time based and the inferences that can be derived from this data are also free of any time based aspects

2.Sequence. In this category of data, though there may not be any explicit reference to time, there exists a sort of qualitative time based relationship among data values.

3.Time stamped. Here we can not only say that a transaction occurred before another but also the exact temporal distance between the data elements. Also with the activities being uniformly spaced on the time parameter.

4.Fully Temporal: In this category, the validity of the data elements is time dependent. The inferences are necessarily time dependent in such cases.

2.4.2 Numerical Data Analysis

The following are the steps to generate a numerical method for data analysis **Raja Ramanna 1990**.

2.4.2.1 Discretisation Methods.

The numerical solution of data flow and other related process can begin when the laws governing these processes are represented in differential equations. The individual differential equations follow a certain conservation principle. Each equation employs a certain quantity as its dependent variable and implies that there must be a balance among various factors that influence the variable. The numerical solution of a differential equation consists of a set of numbers from which the distribution of the dependent variable can be constructed. It means a numerical method is equal to a experiment in which a set of experimental values gives a means of the measured quantity in the domain under study.

Let us suppose that we decide to represent the variation of _ by a polynomial in x_ = a 0 + a 1 x + a 2 x2 + ……..a n x n and employ a numerical method to find the finite number of coefficients a1 , a2…….an. This will enable us to evaluate _, at any location x by substituting the value of x and the values of a's in the above equation.

Thus a numerical method treats as its basic unknowns the values of the dependent variable at a finite number of location called the grid points in the calculation domain. This method includes the task of providing a set of algebraic equations for these unknowns and of prescribing an algorithm for solving the equations. A discretisation equation is an algebraic equation connecting the values of _ for a set of grid points. Such an equation is derived from the differential equation governing _ and thus expresses the same physical information as the differential information. That is only a few grid points are represented in the given differential equation. The value of _ at a grid point is represented by values at its neighborhood values. As more and more grid points are considered, the solutions of discritization equations reach the exact solution of the corresponding differential equations.

2.4.2.2 Control Volume Formulation.

The considered area is divided into a number of grid points each with control volumes surrounding each grid point. The differential equation is integrated over each control volume piecewise to identify the data values. The feature of the control volume formulation is that the

output data to the control volume is equal to input data values of the control volume. It means that conservation principle is identified over the control volume. This characteristic exists for any number of grid points. Thus even the course grid solution exhibits exact integral balances.

2.4.2.3 Steady One Dimensional data flow.

Steady state one-dimensional equation is given by _./_ x(k. _T/_x) +s =0. 0 (eq. 1) where k & s are constants. To derive the discretisation equation we shall employ the grid point cluster. We focus attention on grid point P, which has grid points E, W as neighbors. For one dimensional problem under consideration we shall assume a unit thickness in y and z directions. Thus the volume of control volume is delx*1*1. Thus if we integrate the above equation over the control volume, we get (K _.T/_X)e – (K _T/_X)w + _S _X = 0.0 (eq. 2)

If we evaluate the derivatives . _T/ _X in the above equation from piece wise linear profile , the resulting equation will be Ke(Te – Tp)/(_X)e – Kw(Tp – Tw)/(_X)w + S *del x=0.0 where S is average value of s over control volume. (eq. 3)

This leads to discretisation equation

apTp = aeTe + awTw +b Where ae= Ke/_Xe (eq. 4)

aw = Kw/dXw

ap= ae+aw-sp.delX

b=se.delX .

2.4.2.4 Grid Spacing

For the grid points the distances (dX)e and (dX)w may be or may not be equal. For simplicity we assume the grid spacing as equal on the left side and right side of grid points. Indeed, the use of non uniform grid spacing is often desirable, for it enables us to deploy more efficiently. Infact we shall obtain an accurate solution only when the grid is sufficiently fine. But there is no need to employ a fine grid in regions where the dependent variable T changes slowly with X. On the other hand, a fine grid is required where the T_X variation is steep. The number of grid points and the way they are distributed gives the nature of problem to be solved. Theoretical calculations using only a few grid points specify a convenient way of learning.

2.4.2.5 Boundary Conditions

There is one grid point on each of the two boundaries. The other grid points are called internal points, around each of which a control volume is considered. Based on the grid points at boundary, internal grid points are evaluated by Tri diagonal matrix algorithm.

2.4.2.6 Solution Of Linear Algebraic Equations

The solution of the discretisation equations for the one-dimensional situation can be obtained by the standard Gaussian elimination method. Because of the particularly simple form of equations, the elimination process leads to a delightfully convenient algorithm. For convenience in presenting the algorithm, it is necessary to use somewhat different nomenclature. Suppose the grid points are numbered 1,2,3…ni where 1 and ni denoting boundary points.

The discretisation equation based on equations (1-4) can be written as

Ai Ti + BiTi+1 +CiT i-1 = Di (eq. 5)

For I = 1,2,3…….ni. Thus the data value T is related to neighboring data values T i+1 and T i-1.

For the given problem

C1=0 and Bn=0;

These conditions imply that T1 is known in terms of T2. The equation for I=2, is a relation between T1, T2 & T3. But since T1 can be expressed in terms of T2 , this relation reduces to a relation between T2 and T3. This process of substitution can be continued until Tn-1 can be formally expressed as Tn. But since Tn is known we can obtain Tn-1.This enables us to begin back substitution process in which Tn-2,Tn-3…….T3,T2 can be obtained.

For this tridiogonal system , it is easy to modify the Gaussian elimination procedures to take advantage of zeros in the matrix of coefficients. Referring to the tridiogonal matrix of coefficients above, the system is put into a upper triangular form by computing new Ai.

Ai = Ai – (C i-1 /Ai)* Bi where i = 2,3……ni. (eq. 6)

Di= Di – (C i-1 /Ai) * Di (eq. 7)

Then computing the unknowns from back substitution

Tn = Dn / An. (eq. 8)

Then Tn = Dk – Ak * T k+1 / Ak, k= ni-1, ni-2…3,2,1. (eq. 9)

Thus a Sequence of values are generated using tridiogonal matrix algorithm which can be used as sub key in cryptographic techniques.

2.5 Key Distribution Mechanism

In most of the schemes, a key distribution centre (KDC) is employed which handles the task of key distribution for the participating parties. Generally two mechanisms are employed **Donavan G.Govan et al 2008**. In the first mechanism user A, requests KDC for a session with another user say, B. Initially the KDC sends session key encrypted with private key of A, to the user A. This encrypted session key is appended with encrypted session key by private key of B. On receiving this User A, gets session key and encrypted message with private key of B. This encrypted message is sent to B, where B decrypts it and gets the session key. Now both A & B are in hold of session key which they can use for secured transmission of data. Other wise it is the KDC which sends encrypted session key to the participating parties based on the request of user.

In the second mechanism, the scenario assumes that each user shares a unique master key with the key distribution centre. In such a case, the session key is encrypted with the master key and sent to participating parties. A more flexible scheme, referred to as the control vector **Donavan G.Govan et al 2008**. In this scheme, each session key has an associated control vector consisting of a number of fields that specify the uses and restrictions for that session key. The length of the control vector may vary.

As a first step, the control vector is passed through a hash function that produces a value which is equal to encryption key length. The hash value is XOR ed with the master key to produce an output that is used as key to encrypt the session key. When the session key is delivered to the user the control vector is delivered in its plain form. The session key can be recovered only by

using both master key that the user shares with the KDC and the control vector. Thus the linkage between session key & control vector is maintained. Some times keys get garbled in transmission. Since a garbled key can mean mega bytes of unacceptable cipher text, this sis a problem. All keys should be transmitted with some kind of error detection and correction bits. This is one way errors of key can be easily detected and if required the key can be reset. One of the most widely used methods is to encrypt a constant value with the key and to send the first 2 to 4 bytes of that cipher text along with the key. At the receiving end, the same thing is being done. If the encrypted constants match then the key has been transmitted with out error. The chance of undetected error ranges from one in 2 16 to one in 2 32. The limitation with this approach is in addition to the key, even the constant has to be transmitted to participating parties.

Some times the receiver wants to check if a particular key he has, is the correct decryption key. The naïve approach is to attach a verification block, a known header to the plain text message before encryption. At the receiver's side, the receiver decrypts the header and verifies that it is correct. This works, but it gives intruder a known plain text to help crypto analyze the system.

Chapter -3

ALGORITHM AND EXAMPLES OF GOLDWASSER-MICALI SCHEME

3.1 History

In 1984 Shafi Goldwasser and Silvio Micali theorized the idea of probabilistic cryptography [6]. Their scheme gave the ability to encrypt the same text in many different ways without changing the modulus. This scheme was one of the first to be semantically secure, if not the first. Semantic security is defined as the ciphertext not giving any useful information about the plaintext in polynomial time, except possibly the length [20]. This scheme introduced some new ideas, that helped to inspire a new line of probabilistic schemes. In fact, the Paillier scheme seems to have emerged from this line. Although it is significant for this purpose, Goldwasser-Micali doesn't seem to have been studied much because of its slow speed and huge expansion. For these reason it is doubtful that it could be used for encrypting normal amounts of text. However, the scheme does well at bit encryption because of its ability to encrypt bits to different values. For this reason the scheme may find some practical uses.

3.2 Mathematical Background

Goldwasser-Micali encryption uses the quadratic residuosity problem as the basis for the encryption. Assuming this problem is intractable then this scheme is secure. Of course, like most encryption schemes, this has not been proven secure but seems to be a good assumption.

A quadratic residue is simply an integer in Z_n^* that for some x ε N is equivalent to x^2 mod n [15]. Since we are in mod n, the integers can be quadratic residues in several ways. Most are values that would never be squares without evaluating with the modulus.

To understand the quadratic residues, we first need to understand the concepts of the Legendre and Jacobi symbols [15]. The Legendre symbol can easily be found with the formula:

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \bmod p, \text{ where } p \text{ is any odd prime.}$$

If the value n is not prime then one has the Jacobi symbol (a/n) (which looks just like the Legendre symbol). If the factorization of $n = p^{e1}_1 \ p^{e2}_2 \ldots p^{ek}_k$ the Jacobi symbol can be defined with Legendre symbols as:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1}\left(\frac{a}{p_2}\right)^{e_2}\ldots\left(\frac{a}{p_k}\right)^{e_k}.$$

The quadratic residuosity problem is defined as the following.

Definition 1 Given integers n and a ε J_n, where J_n is defined as all a ε Z_n^* whose Jacobi symbol is 1, find whether or not a is a quadratic residue modulo n.

To solve the Legendre symbol the value of n needs to be factored and then the equation can be solved with the resulting Jacobi symbols. The Jacobi symbols can be solved by the above method and the solutions can be multiplied together for the final solution.

With the mathematical background of the quadratic residuosity problem, we are now able to create the keys needed for encryption, and then encrypt and decrypt a message. The following is the method used for generating the public and private keys.

Steps for Key Generation

1. Select two primes p and q that are about the same size.
2. Calculate modulus n = pq, the product of two primes.
3. Select y ε Z_n that is also a pseudosquare mod n. A pseudosquare is a quadratic non-residue modulo n with (a/n) = 1.
4. We now have the public key (n, y) and the private key (p, q)

The key generation is mostly straightforward for this scheme and resembles other public keys schemes like RSA. The tricky part is creating the pseudosquare y. From the name, one can infer that this value will look like a square. In this case, the Jacobi symbol (a/n) = 1 is a test to see if a is quadratic residue modulo n. But just because the Jacobi symbol is 1 doesn't mean it is a

quadratic residue. There is also a chance that that this value is not a quadratic residue and it is precisely these values that represent pseudosquares. There is a good trick for creating these values when n is a composite of two primes p and q [10].

Steps for Finding Pseudosquare

1. Find two quadratic non-residues, a mod p and b mod q.
2. You want to find the number y mod n (remember n = pq) where $y \equiv a \bmod p$ and also $y \equiv b \bmod q$. This can be done using Chinese Remaindering.
3. We know y is a quadratic non-residue mod n. This is because a mod p is in Q_p (Q is the set of quadratic non-residues) and b mod q is in Q_q , which implies they are both quadratic non-residues mod n (see fact 2.137 in [10]). Remember that $y \equiv a \bmod p$ and $y \equiv b \bmod q$.
4. We also know by the properties of Jacobi and Legendre symbols that the Jacobi symbol(a/n) = (a/p)(b/q) = (-1)(-1) = 1, since the Legendre symbol of quadratic non-residues must be -1.
5. From all the above we can conclude we have a pseudosquare, since the Legendre symbol (a/p) = 1 and the Jacobi symbol (a/n) = 1.

Now that we have the public and private keys we can see how to do encryption with Goldwasser-Micali. In our case Bob is encrypting a message m for Alice into a ciphertext c. Bob has received the public key (n, y) that was generated by Alice.

Steps for Encryption

1. Convert message m to a binary string where $m = m_1m_2m_3...m_t$ and t is the length of string and each m_i is either 0 or 1.
2. For i = 0 to t
   (a) Pick a random value $x \varepsilon Z_n^*$.
   (b) If $m_i = 1$ then $c_i = yx^2 \bmod n$ and if $m_i = 0$ then $c_i = x^2 \bmod n$.

3. The encrypted values c_i are then placed in c where $c = c_1c_2c_3...c_t$ where again t is the size c.

It is clear from the encryption algorithm that this is an inefficient scheme and some-what simplistic. Only one bit can be encrypted at a time and the expansion is very great. Even with these characteristics I am fascinated that with the randomness one can encrypt 0's and 1's in so many ways and yet this is considered secure. It makes you wonder if there are other easier methods to encrypt each bit, since all you need is two different outcomes. This type of scheme could be usable when doing bit encryption but would take too long for normal size encryption.

The decryption of ciphertext c of length t is as follows using the private key (p, q).

Steps for Decryption

1. For i = 0 to t
    (a) Find the Legendre symbol $l_i = (c_i/p)$.
    (b) If $l_i = 1$ then $m_i = 0$, else $m_i = 1$.
2. The decrypted message is the string $m = m_1m_2m_3...m_t$.

The decryption process, as you can see, is as simple as figuring out if c_i was encrypted as quadratic residue modulo n or if it was encrypted as a pseudosquare. If c_i is encrypted as x^2 mod n then we obviously get a quadratic residue mod n, since that is the definition of a quadratic residue. If $c_i = yx^2$ then we must have a pseudosquare. This is because the Jacobi symbol of y is $(y/n) = -1$ and the Jacobi symbol of x^2 is $(x^2/n) = 1$. Therefore we get $(-1)(1) = -1$ which implies that yx^2 is a pseudosquare modn. To quickly show whether or not c_i is a quadratic residue modulo n, we use a fact stated in [10] in section 2.137, which says that if c_i is a quadratic residue modulo q or modulo p then we know it is a quadratic residue modulo n. This can be done easily, since p and q are primes, with either Legendre symbol (c_i/p) or (c_i/q).

The ability to find if a number is a quadratic residue modulo n using its factors p and q is especially useful because no one knows a method of finding quadratic residue without these factors. Since an effective way to factor large numbers is unknown, this scheme is considered secure. The security of this scheme relies on the assumption that the quadratic residuosity problem is difficult, but for now it seems to be secure. If a person were able to intercept a message, all they would see would be pseudosquares and quadratic residues modulo n. Assuming

the value of n is large enough, they would not be able to get the factors p and q and would not be able to calculate the Legendre symbol.

Below is a simple example of the Goldwasser-Micali scheme. You can tell from the size of the numbers in the public and private keys that this is not realistic for a truely secure example. But the mathematics are the same and help to cement in your mind how the scheme works. The steps shown in the example below correspond to the algorithms above, so you can compare.

Key Generation

1. p = 71, q = 61.
2. n = 4331.

The equivalent of both these numbers modn. We know the gcd(p, q) = 1 since they are both prime, so we can first use the formula $k \equiv (a - b)p^{-1} \pmod{q}$. Plugging in the numbers we get $k \equiv (23 - 17)71^{-1} \pmod{61} \equiv 25 \pmod{61}$. Then we can substitute k into b + pk (mod n) giving us $17 + 71 * 25 \equiv 1792 \pmod{4331}$. This means we can use y = 1792 as our pseudosquare mod n.

This gives us the public key (n, y) = (4331, 1792) and the private key (p, q) = (71, 61).

Encryption

We will now encrypt the message m = 9. This means m = 1001 in binary, which is a good number to encrypt since there are two 0's and two 1's to encrypt. Obviously we can encrypt much bigger numbers but we would just be encrypting the same two things over and over in a certain order. Having two examples of each should be sufficient to understand the process.

- $m_1 = 1$

  We choose $x \varepsilon Z_n^*$ so x = 12.

  $m_1 = 1$ so $c_1 \leftarrow yx^2 \bmod n$ which gives us $c_1 =$

  $1792 * 12^2 = 2512 \bmod 4331$.
- $m_2 = 0$

  We choose random x = 22.

  $m_2 = 0$ so $c_2 \leftarrow x^2 \bmod n$ which gives us $c_2 =$

$22^2 = 484 \bmod 4331$.

- $m_3 = 0$

  We choose random x = 81.

  $m_3 = 0$ so $c_3 \leftarrow x^2 \bmod n$ which gives us $c_3 =$

  $81^2 = 1378 \bmod 4331$

- $m_4 = 1$

  We choose random x = 3001.

  $m_4 = 1$ so $c_4 \leftarrow yx^2 \bmod n$ which gives us $c_4 =$

  $1792 * 3001^2 = 2421 \bmod 4331$

This gives us the ciphertext c = (2519, 484, 1378, 2421) that is sent to Alice to be decrypted.

Decryption

- $c_1 = 2519$

  We calculate the Legendre symbol $(c_1/p) \equiv c_1^{(p-1)} \bmod p$

  This gives us $(2519/71) = 2519^{((71-1)/2)} \bmod 71 = -1$.

  Because we get −1, $m_1 = 1$.

  $c_2 = 484$

  The Legendre symbol $(484/71) \equiv 484^{((71-1/2)} \bmod 71 = 1$.

  Since we get 1, $m_2 = 0$.

- $c_3 = 1378$

  The Legendre symbol $(1378/71) \equiv 1378^{((71-1/2)} \bmod 71 = 1$.

  Since we get 1, $m_3 = 0$.

  $c_4 = 1238$

  The Legendre symbol $(1238/71) \equiv 1238^{((71-1/2)} \bmod 71 = -1$.

  Since we get −1, $m_4 = -1$.

As you can see when we concatenate m together again we get m = 1001 in binary or m = 9, which is what we started with.

3.3 Summary of Scheme

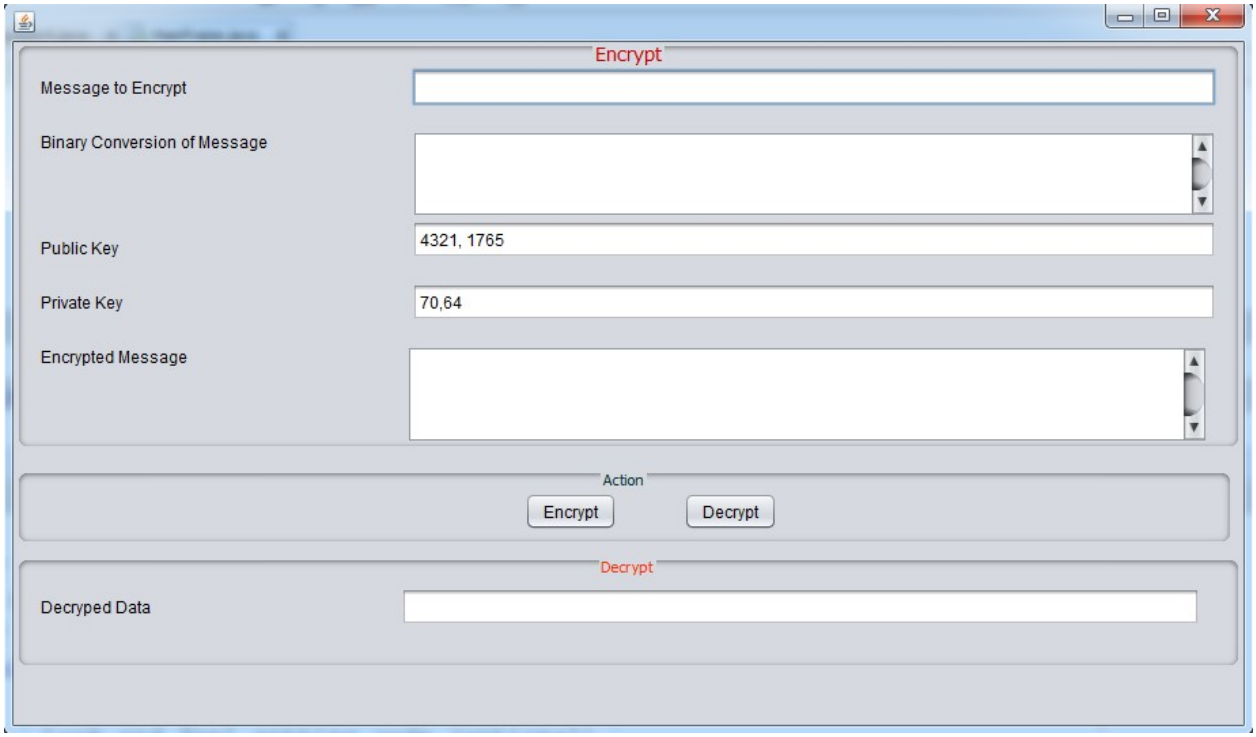
As you can see from the information provided, this scheme works and appears to be semantically secure. The problem is that the scheme has a message expansion around $\lg_2 n$ [10]. The value of n would need to be hundreds of bits long to prevent factorization and finding the private key. This means that each bit of the ciphertext would need to be expanded to just smaller than n. In an example that was reasonably secure this would make the ciphertext hundreds of times larger than the original message. Because of this fact, this scheme is not used practically but it may become useful for encrypting small amounts, such as invidual bits. The other probabilistic schemes covered later have much potential for normal encryption than this one.

Chapter - 4

SOFTWARE TOOL (CODE AND SCREESHOTS)

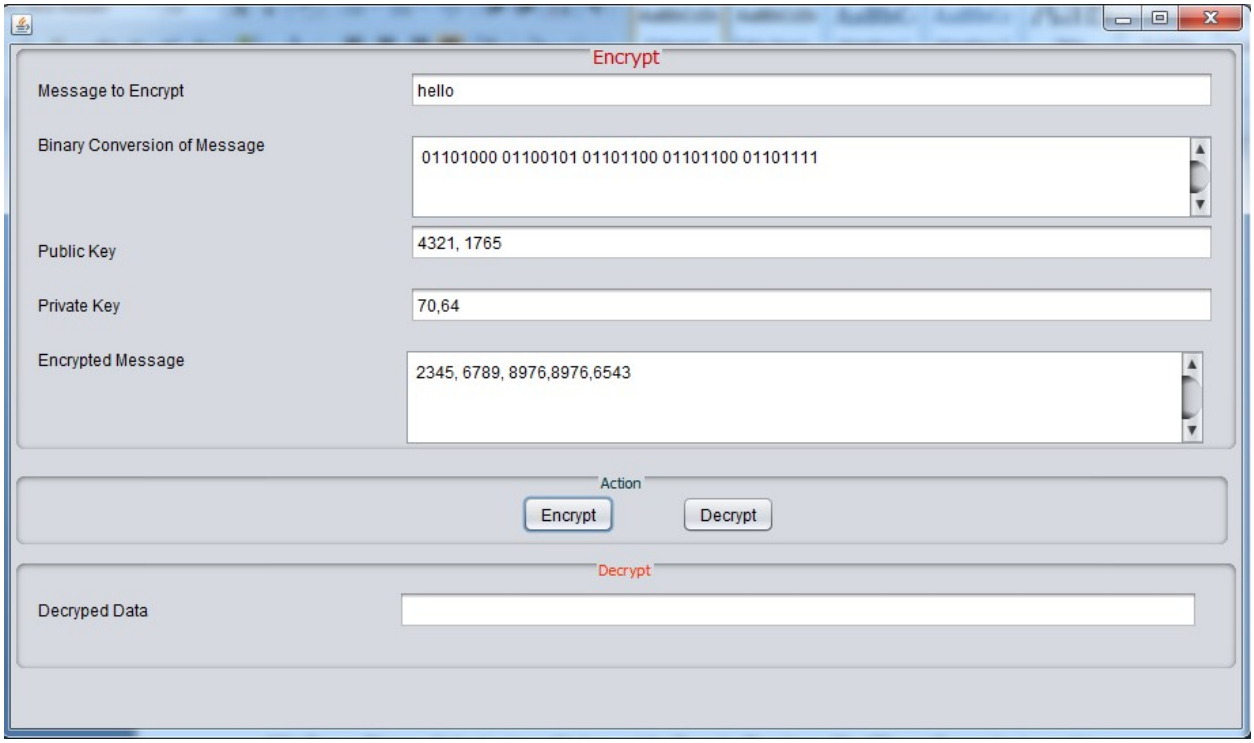
4.1 Screenshots

4.1.1 This is the main screen which will be visible as the first page. We just have to enter message which is to be encrypted in first test field (Message to encrypt). The public key and private key which are already defined are given and shown.



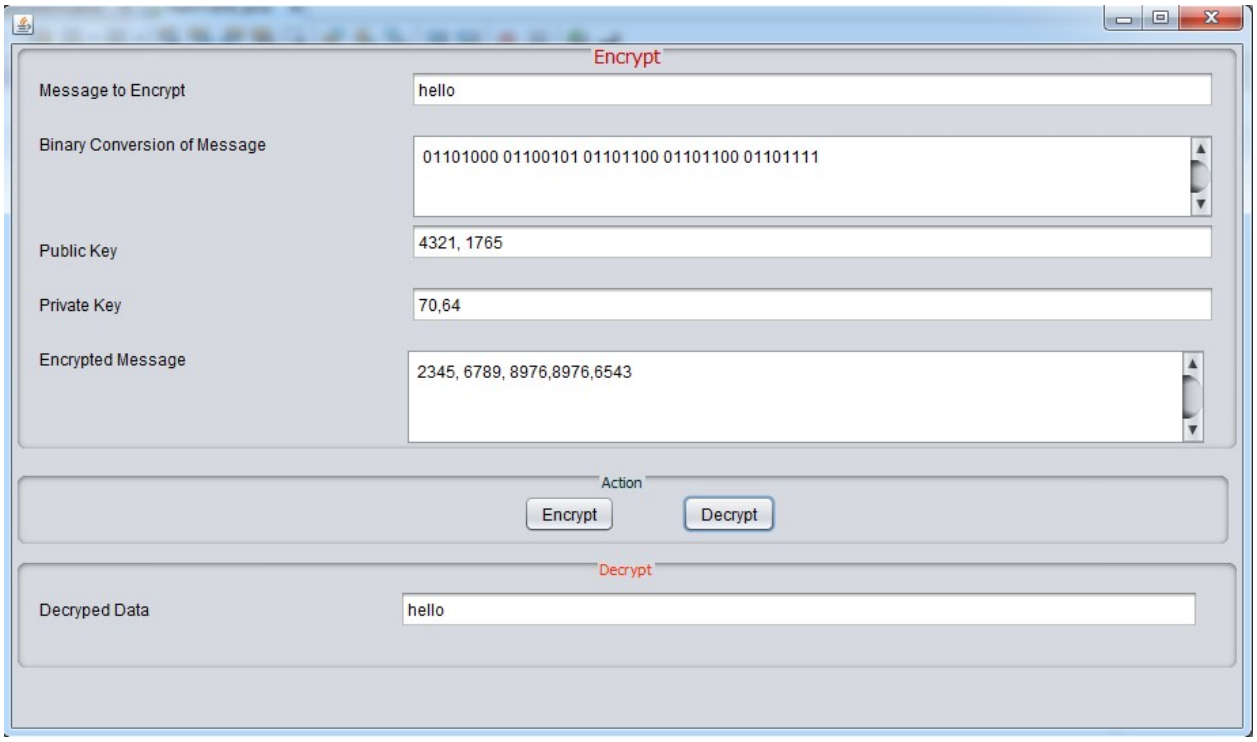
4.1.2

The given screen is showing the result of encrypted message which is coming after calculation of message using public key and private key. As the algorithm works on binary conversion of data so binary conversion of message and encrypted message is also shown for simplicity and explanation. This will be shown on clicking on encrypt button.



4.1.3

On clicking on decrypt button following screen will be shown after decrypting of the message.



Code of the application

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package nishu;

/**
 *
 * @author Nishu
 */
public class MainFrame extends javax.swing.JFrame {
    String message;
    /**
     * Creates new form MainFrame
     */
    public MainFrame() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel5 = new javax.swing.JLabel();
```

```
        jPanel1 = new javax.swing.JPanel();
        jTextField1 = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jLabel9 = new javax.swing.JLabel();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextArea2 = new javax.swing.JTextArea();
        jPanel2 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jPanel3 = new javax.swing.JPanel();
        jTextField4 = new javax.swing.JTextField();
        jLabel8 = new javax.swing.JLabel();

        jLabel5.setText("jLabel5");

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Encrypt",
javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP, new
java.awt.Font("Tahoma", 0, 14), new java.awt.Color(204, 0, 0))); // NOI18N
        jPanel1.setToolTipText("Encrypt ");
        jPanel1.setName("encryptPanel"); // NOI18N
```

```
jTextField1.setName("txtMessage"); // NOI18N

jLabel1.setText("Message to Encrypt");

jTextArea1.setEditable(false);
jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jTextArea1.setName("txtBinary"); // NOI18N
jScrollPane1.setViewportView(jTextArea1);

jLabel2.setText("Binary Conversion of Message");

jLabel3.setText("Public Key");

jLabel6.setText("Private Key");

jTextField2.setEditable(false);
jTextField2.setText("4321, 1765");
jTextField2.setName("txtPublic"); // NOI18N
jTextField2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField2ActionPerformed(evt);
    }
});

jTextField3.setEditable(false);
jTextField3.setText("70,64");
jTextField3.setName("txtPrivate"); // NOI18N

jLabel9.setText("Encrypted Message");
```

```
        jTextArea2.setColumns(20);
        jTextArea2.setRows(5);
        jScrollPane2.setViewportView(jTextArea2);

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE,
105, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addContainerGap()
                            .addComponent(jLabel9, javax.swing.GroupLayout.PREFERRED_SIZE,
116, javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(0, 0, Short.MAX_VALUE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE,
590, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                .addComponent(jLabel4))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addGap(18, 18, 18)
                    .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 592,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
62, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE,
62, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                    .addComponent(jTextField3)
                    .addComponent(jTextField2, javax.swing.GroupLayout.DEFAULT_SIZE,
592, Short.MAX_VALUE)))
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
```

```
                        .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addGap(166, 166, 166)
                        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 592,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel1))
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 64,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel2))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                    .addComponent(jLabel4)
                    .addGap(84, 84, 84))
```

```
                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING
)
                    .addComponent(jLabel3)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel1Layout.createSequentialGroup()
                        .addGap(2, 2, 2)
                        .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(jLabel6)
                    .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(42, 42, 42)
                        .addComponent(jLabel7))
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel9)
```

```
                        .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 72,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addGap(72, 72, 72))))
        );

        jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Action",
javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP, new
java.awt.Font("Tahoma", 0, 11), new java.awt.Color(0, 51, 51))); // NOI18N

        jButton1.setText("Encrypt");
        jButton1.setName("btnEncrypt"); // NOI18N
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setText("Decrypt");
        jButton2.setName("btnDecrypt"); // NOI18N
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
```

```
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jButton1)
            .addGap(49, 49, 49)
            .addComponent(jButton2)
            .addGap(323, 323, 323))
        );
        jPanel2Layout.setVerticalGroup(
           jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
           .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
            .addGap(0, 0, Short.MAX_VALUE)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                .addComponent(jButton1)
                .addComponent(jButton2)))
        );

        jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Decrypt",
javax.swing.border.TitledBorder.CENTER, javax.swing.border.TitledBorder.TOP, new
java.awt.Font("Tahoma", 0, 11), new java.awt.Color(255, 51, 0))); // NOI18N

        jTextField4.setName("txtDecrypt"); // NOI18N

        jLabel8.setText("Decryped Data");

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
           jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel8, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE, 588,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18))
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELIN
E)
                    .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel8))
                .addContainerGap(21, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
```

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 303,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(38, 38, 38))
        );

        pack();
    }// </editor-fold>

    private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling ecode here:
        message=jTextField1.getText();
```

```
    String binaryMessage="";
    byte[] bytes = message.getBytes();
    StringBuilder binary = new StringBuilder();
    for (byte b : bytes)
    {
       int val = b;
       for (int i = 0; i < 8; i++)
       {
           binary.append((val & 128) == 0 ? 0 : 1);
          val <<= 1;
       }
       binary.append(' ');
    }
    binaryMessage=binary.toString();
    jTextArea1.setText(binaryMessage);
    jTextArea2.setText(encryptedMessage());

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTextField4.setText(decryptedMessage());
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
```

```
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
```

```
            public void run() {
                new MainFrame().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextArea jTextArea2;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    // End of variables declaration
}
```

Chapter 5

Conclusion and Future work

5.1 Summary

This study represents the importance of Encryption of data for storage and transmission. The significance of encrypted data can be identified in light of the various type of applications and globalization of communication. The advantages of encrypting data manifest themselves in the form of security & confidentiality in real time applications. Encryption of data is of particular significance in applications like email, ecommerce, e-cash where highly vulnerable communication lines is accessed for transmission of highly volatile data.

The study traces the development of various probabilistic encryption algorithms models in a real time environment in all their breath taking diversity and breakthroughs in Chapters 2. Discusses various aspects of encryption with review of literature. A lots of literature were reviewed for discussed in the chapter.

The chapter 3 pays special emphasis on probabilistic method of encryption, its algorithm, methods and examples.

In chapter 4 we have implemented various encryption algoritms in java. Various screenshots with code is given in this chapter. The code is developed using IDE NetBeans8.

5.2 Future Work

The present work is using the method given by Goldwasser-Micali Scheme. A lots of different researchers have been working on the probabilistic approach of encryption and decryption. A lots of research can be done on all these methods of probabilistic approaches.

The proposed system is using Java1.8 for coding. The user interface is designed in Swing. A lots of researchers have worked on concept of probabilistic approach of encryption and decryption. Some of them have designed good algorithm. But a very least number of researchers have taken it to implementation level. Our focus of the work was to implement one of the best models, in

which the data can be moved in secure and safe way. We tried to study various language specifications. Java has implemented one of the best security model so we used java security for designing the tool.

For security of encrypted data, a lots of complex algorithms were suggested by researchers. Most of these algorithms were very effective but these algorithms are a bit complex in understanding and implementation. Overall the proposed tool is very nice, compact and effective tool for implementation and understanding of probabilistic approach of encryption and decryption. Also the tool is developed by using complete object oriented methodology which can later be extended as per need.

The work of probabilistic approach of encryption and decryption can never ends. This is the race condition in between hackers and these methods. Although java is using Base64 method which is one of the best methods as on date but still it is not destination. As and when new methodologies are evaluated, it is essential to update the tool as per requirement

Our ability to discover hidden information during our investigations is vital, especially as new and innovative methods continue to evolve. During the past decade, data hiding technologies have advanced from limited use to ubiquitous deployment. With the rapid advancement of smart mobile devices, the need to protect valuable proprietary information has generated a plethora of new methods and technologies for both good and evil. Most dangerous among these are those that employ hiding methods along with cryptography, thus providing a way to both conceal the existence of hidden information while strongly protecting the information even if the channel is discovered.

Many vendors provide excellent technologies for protecting the privacy of information for the desktop. In addition, many of the latest smart mobile platforms (Android and iPhone) include built-in cryptographic capabilities. What is more dangerous and difficult to discover/decipher are data hiding methods that exploit multimedia and protocol weaknesses to both hide and communicate covertly. These new techniques provide hybrid solutions that combine the best of

cryptography with the best of probabilistic approach of encryption and decryption. The interest, innovation, and advancement of these threats continue to go unchecked for the most part.

The present study can be extended for the use of different mobile technologies like window mobile, android, iphone etc. Also the study can be extended for audio and vedio type of media. In the present system, we are using encryption methods which are given by java only. Later on, the tool and such applications can be developed in almost all the available technologies. As java is open source and source code of encryption/decryption methods are available, these methods/classes can be re-written to extend their algorithm and our new ideas can be included in these methods.

REFERENCES

1. Shashi Bala et al 2015, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 6, June 2015, ISSN: 2277 128X.
2. Pinki Singh et al 2015, Encryption Algorithms With Emphasis On Probabilistic
3. Encryption & Time Stamp In Network Security, International Journal of Research in Engineering & Technology (IMPACT: IJRET) ISSN(E): 2321-8843; ISSN(P): 2347-4599 Vol. 3, Issue 5, May 2015, 39-46.
4. Dheeraj et al 2015, Goldwasser-Micali Cryptosystem.
5. Arun Singh Chouhan et al 2014, Design and Performance Analysis of new Cryptographic Algorithm for Wireless Sensor Networks & Broadcasting Applications Security, International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 3, Issue 11, November 2014, ISSN 2319 - 4847 .
6. Sage.math.Washington.edu/home/jetchev/Public.html/docs/jetchev-talk.ppt- Broadcast encryption schemes.
7. Brassard G.: Modern Cryptology , a tutorial lecture Notes on computer science , (325) ,(spring-verlas) .
8. Bruce Scheneier: Applied cryptography (John Wiley & sons (ASIA) Pvt. Ltd.
9. Carlone Fontaine & Fabien Galand: A Survey of Homomorphic Encryption for non specialists, EURASIP Journal,Vol 07, Article 10.
10. Aldrin W. Wanambisi et al 2013, Journal of Natural Sciences Research, ISSN 2224-3186 (Paper) ISSN 2225-0921 (Online) Vol.3, No.1, 2013.
11. Vinod Vaikuntanathan 2013, Computing on Encrypted Data.
12. Nishu Sharma et al, Performance rating of new encryption algorithms with prominence on probabilistic encryption, International Journal of Engineering Technology and Management (IJETM), Volume 2: Issue 1: Page No. 16-19.
13. Sanjib Kumar Baral et al, Development of time-stamped signcryption scheme and its application in e-cash system.
14. A.V.N.Krishna et al, A New Non Linear, Time Stamped & Feed Back Model Based Encryption Mechanism with Acknowledgement Support, International Journal of Advancements in Technology, ISSN 0976-4860.

15. Stuart Haber et al, HowtoTime-StampaDigitalDocument.
16. A.V.N.Krishna, Probabilistic Encryption Based ECC Mechanism, International Journal of Advancements in Technology, ISSN 0976-4860.
17. Dogan Kesdogan et al, Stop-And-Go-MIXes Providing Probabilistic Anonymity in an Open System.
18. Amjay Kumar, Ajay Kumar: Development of New Cryptographic Construct using Palmprint Based Fuzzyvoult, EURASIP Journal on Adv. In Signal Processing, Vol 21, pp 234-238, 2009
19. Bluekrypt 2009: Cryptographic Key length Recommendations, http://www.keylength.com
20. Hamid Mirvazri, Kashmiran Jumari Mahamod Ismail, Zurina Mohd. Hanapi: Message based Random Variable Length Key Encryption Algorithm, Journal of Computer Science, pp 573-578, 2009.
21. Hianyi Hu, Gufen Znu, Guanning Xu: Secret Scheme for Online Banking based on Secret key Encryption, Second International Workshop on Knowledge Discovery & Data Mining, Jan 23-25 2009.
22. Kaiping Xue: Study of improved key Distribution Mechanisms based on two length structure for wireless sensor networks, International conference on adv. Information Technology, 2008.
23. Donavan G.Govan, Nathen Lewis: Using Trust for Key Distribution & Route Selection in Wireless Sensor Networks, International Conference on Network Operations & Management, IEEE Symposium 2008, PP 787-790.
24. Donavan G.Govan, Nathen Lewis: Using Trust for Key Distribution & Route Selection in Wireless Sensor Networks, International Conference on Network Operations & Management, IEEE Symposium 2008, PP 787-790.
25. E.C.Park, I.F.Blake: Reducing communication overhead of Key Distribution Schemes for Wireless Sensor Networks: Computer Communications & Networks, ICCCN 2007, pp 1345-1350.
26. Baocang Wang, Qianhong Wu, Yupu Hu: A Knapsack Based Probabilistic Encryption Scheme, On Line March 2007, www.citeseer.ist.psu.edu.

27. Krishna A.V.N, S.N.N.Pandit, A.Vinaya Babu: A generalized scheme for data encryption technique using a randomized matrix key, Journal of Discrete Mathematical Sciences & Cryptography, Vol 10, No. 1, Feb 2007, pp73-81
28. Krishna A.V.N, A.Vinaya Babu: Pipeline Data Compression & Encryption Techniques in e-learning environment, Journal of Theoretical and Applied Information Technology, Vol 3, No.1, Jan 2007, pp37-43.
29. Krishna A.V.N, A.Vinaya Babu: A Modified Hill Cipher Algorithm for Encryption of Data in Data Transmission, Georgian Electronic Scientific Journal: Computer Science and Telecommunications 2007 N0. 3(14) pp 78-83.
30. Krishna A.V.N., A.Vinaya Babu: Web and Network Communication security Algorithms, Journal on Software Engineering, Vol 1,No.1, July 06, pp12-14
31. Georg J. Fuchsbauer 2006, An Introduction to Probabilistic Encryption, Osjeˇcki matematiˇcki list 6(2006), 37–44.
32. Georg J.Fuchsbauer: An Introduction to Probabilistic Encryption, 'Osjecki Matematicki List 6(2006), pp37-44.
33. Krishna A.V.N., Vishnu Vardhan.B.: Utility and Analysis of some Encryption algorithms in E learning environment, International Convention Proc. Of CALIBER 2006, 02-04 Feb. 2006, Gulbarga, India.
34. Luminiţa SCRIPCARIU et al 2006, Java Implemented Encryption Algorithm.
35. Krishna A.V.N.: A new algorithm in network security, International Conference Proc. Of CISTM-05, 24-26 July 2005, Gurgoan, India.
36. Pci Yihting: A Temporal order Resolution algorithm in the multi server time stamp service frame work, International Conference on Advanced Information Networking & Applications, AINA 2005, Vol 2m 28-30 March, pp 445-448.
37. Brics: Universally comparable notions of key exchange and secure channels, Lecture Notes in Computer Science, Springer, Berlin, March 2004.
38. Krishna A.V.N., S.N.N.Pandit: A new Algorithm in Network Security for data transmission, Acharya Nagarjuna International Journal of Mathematics and Information Technology, Vol: 1, No. 2, 2004 pp97-108

Contact Us:

University Campus Address:

Jayoti Vidyapeeth Women's University

Vadaant Gyan Valley, Village-Jharna, Mahala Jobner Link Road,

Jaipur Ajmer Express Way, NH-8, Jaipur- 303122, Rajasthan (INDIA)

(Only Speed Post is Received at University Campus Address, No. any Courier Facility is available at Campus Address)

Pages	: 60
Book Price	: ₹ 150/-

JPD Women University Press

Jayoti Publication Desk