CHAPTER 1

INTRODUCTION

Introduction

Security of data to maintain its confidentiality, proper access control, integrity and availability is a major issue in data communication. As soon as a sensitive message was etched on a clay tablet or written on the royal walls, then it must have been foremost in the sender's mind that the information should not get intercepted and read by a rival. Codes, hence, form an important part of our history; starting from the paintings of Da Vinci and Michelangelo to the ancient Roman steganographic practices the necessity of data hiding was obvious.

Today in the e-age, the need to protect communications from prying eyes is greater than ever before. Cryptography, the science of encryption plays a central role in mobile phone communication, e-commerce, Pay-TV, sending private e-mails, transmitting financial information and touches on many aspects of daily lives.

Today's technology can be traced back to earliest ciphers, and have grown as a result of evolution. The initial ciphers were cracked, so new, stronger ciphers emerged. Code breakers set to work on these and eventually found flaws, forcing cryptographers to invent better ciphers and so on. The significance of key is an enduring principle of cryptography.

With the advent of the computer age, the mechanical encryption techniques were replaced with computer ciphers .They operated according to the same principles of substitution and transposition (where the order of letters or bits is altered). Again each cipher depended on choosing a key, known only by the sender and the receiver which defined how a particular message would be. This meant that there still was a problem of getting the key to the receiver so that the message could be deciphered. This had to be done in advance, which was an expensive slow and risky process.

For years, this key distribution problem haunted code makers i.e. if you want to decipher a scrambled text you have to know the key in advance. But there was revolution in cryptography known as public key cryptography, which destroyed the key distribution problem. This was a technology tailor made for the internet. Customers could send credit card details and send them to retailers on the other side of the planet. It formed the basis of all kinds of modern day communications.

1.1The Purpose of Cryptography

Cryptography is the science of writing in secret code and is an ancient art; the first documented use of cryptography in writing dates back to circa 1900 B.C. when an Egyptian scribe used non-standard hieroglyphs in an inscription. Some experts argue that cryptography appeared spontaneously sometime after writing was invented, with applications ranging from diplomatic missives to war-time battle plans. It is no surprise, then, that new forms of cryptography came soon after the widespread development of computer communications. In data and telecommunications, cryptography is necessary when communicating over any untrusted medium, which includes just about *any* network, particularly the Internet.

Within the context of any application-to-application communication, there are some specific security requirements, including:

- Authentication: The process of proving one's identity. (The primary forms of host-to-host authentication on the Internet today are name-based or address-based, both of which are notoriously weak.)
- Privacy/confidentiality: Ensuring that no one can read the message except the intended receiver.
- Integrity: Assuring the receiver that the received message has not been altered in any way from the original.
- Non-repudiation: A mechanism to prove that the sender really sent this message.

Cryptography, then, not only protects data from theft or alteration, but can also be used for user authentication. There are, in general, three types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric) cryptography, public-key (or asymmetric) cryptography, and hash functions, each of which is described below. In all cases, the initial unencrypted data is referred to as plaintext. It is encrypted into cipher text, which will in turn (usually) be decrypted into usable plaintext.

In many of the descriptions below, two communicating parties will be referred to as Alice and Bob; this is the common nomenclature in the crypto field and literature to make it easier to identify the communicating parties. If there is a third or fourth party to the communication, they will be referred to as Carol and Dave. Mallory is a malicious party, Eve is an eavesdropper, and Trent is a trusted third party.

1.1.1 Classical cryptographic techniques

The technique enables us to illustrate the basic approaches to conventional encryption today. The two basic components of classical ciphers are substitution and transposition. Then other systems described that combines both substitution and transposition.

1.1.1.1 Substitution techniques

In this technique letters of plaintext are replaced by or by numbers and symbols. If Plain text is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns.

1.1.1.2 Caesar Cipher

Caesar Cipher replaces each letter of the message by a fixed letter a fixed distance away e.g. uses the third letter on and repeatedly used by Julius Caesar.

For example:

Plaintext: I CAME I SAW I CONQUERED

Cipher text: L FDPH L VDZ L FRQTXHUHG

Mapping is:

ABCDEFGHIJKLMNOPQRSTUVWXYZ

DEFGHIJKLMNOPQRSTUVWXYZABC

Can describe the Cipher as:

Encryption: C=E (P)=(P+3)mod 26-------(2.1)

Decryption: P=D(C) =(C-3) mod26-------- (2.2)

1.1.1.3 Mono-alphabetic Cipher

The Caesar cipher uses only 26 rotations out of the 26 permutations on the alphabet. The mono-alphabetic cipher uses them all. A key k is an arbitrary permutation of the alphabet. Ek(m) replaces each letter of a of m by k(a) to yield c. To decrypt Dk(c) replaces each letter b of c by k-inverse (b). The size of the key space is 26!> 2 pow(74), which is too large for a successful brute force attack. However, mono-alphabetic ciphers can be easily broken down using letter frequency analysis, given a long enough message. Because each occurrence of a letter a in the message is replaced by the same letter k(a), the most frequently occurring letter of m will correspond with the most frequently occurring letter of c. while jack might know what the most frequently occurring letter of m is, if the message is long enough and she knows that it is English, then it is quite likely that the most frequently occurring letter in m is one of the most frequently occurring letters in English i.e. ‘e’ or maybe ‘t’. She can assume then that the most frequent letter ‘b1’ in c is ‘e’, the next most frequent letter b2 in ‘t’., and so forth. Of course, not all these guesses will be correct, but the number of likely candidates for each cipher text letter is greatly reduced. Moreover, many wrong guesses can quickly discarded eve without constructing the entire trial key because they lead to unlikely letter combinations.

1.1.1.4 Playfair Cipher

The Playfair is a substitution cipher bearing the name of the man who popularized but not created it. The method was invented by Sir Charles Wheatstone, in around 1854; however he named it after his friend Baron Playfair. The Playfair Cipher was developed

for telegraph secrecy and it was the first literal digraph substitution cipher. This method is quite easy to understand and learn but not easy to break, because you would need to know the "keyword" to decipher the code. The system functions on how letters are positioned in a 5*5 alphabet matrix. A "KEYWORD" sets the pattern of letters with the other letters the cells of the matrix in alphabetical order (I and j are usually combined in one cell). For instance, suppose we use a keyword Charles then matrix would look like this:

c h a r l

e s b d f

g i/j k m n

o p q t u

v w x y z

Now supposing the message to be enciphered here is "the scheme really works". First of all the plaintext is dividing into two letter groups. If there are double letters occurring, in the message, either an 'x' will be used to separate the double letters or an 'x' will be added to make a two letter group combination. In our example, the phrase becomes:

Enciphered text: th es c hem er ea lx ly wo rk sx

Each of the above two letter combinations will have 3 possible relationships with each other in the matrix: they can be in the same column, same row, or neither. The following rules for replacement should be used:

If two letters are in the same column of the matrix, use letter below it as the cipher text.(columns are cyclical).

If two letters are in the same row of the matrix, use letter to the right as the cipher text.(columns are cyclical).

If neither the same column or row, then each are exchanged with the letter at the intersection of its own row and the other column.

Example:
Plaintext: th es ch em er lx ly wo rk sx
Cipher text: pr sb ha dg bc az rz vp am bw
For deciphering, the rules are exact opposite.

1.2 Three Dimensions Of Cryptographic systems

1. Type of operations used for transforming plaintext to cipher text. All encryption algorithms are based on two general principles. Those are substitution, in which each element in the plain text is mapped into another element and transposition in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost. Most systems referred to as product systems, involved multiple stages of substitution and transposition.

2. The number of keys used: If sender and receiver use the same key, the system is referred to as symmetric, single key or secret key conventional encryption. If the sender and the receiver each uses a different key the system is referred to as asymmetric, two key, or public-key encryption.
3. The way in which the plaintext is processed: A block cipher processes the input on block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

There are several ways of classifying cryptographic algorithms. For purposes of this paper, they will be categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms are:

- Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption

- Public Key Cryptography (PKC): Uses one key for encryption and another for decryption
- Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information

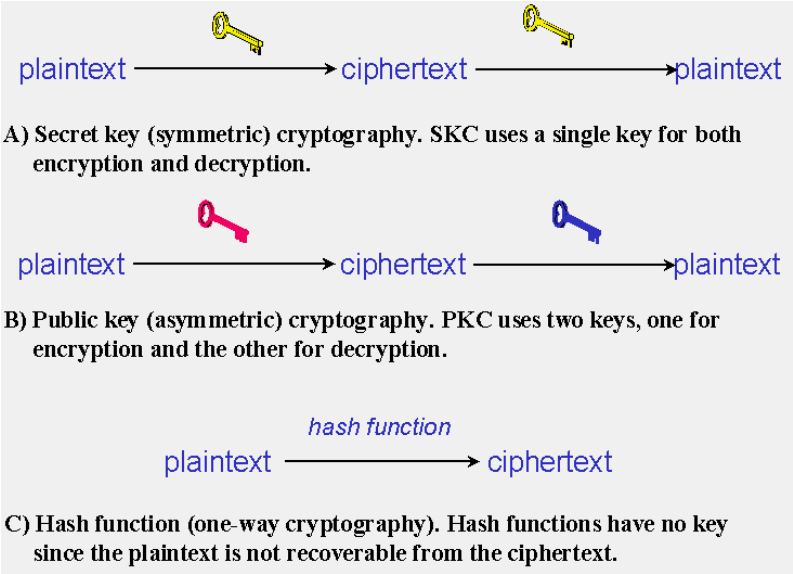
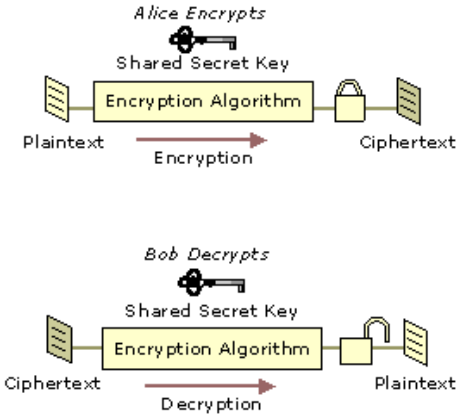


Fig 1: Three ways of Cryptography

1.2.1 Secret Key Cryptography

With secret key cryptography, a single key is used for both encryption and decryption. As shown in Figure 1A, the sender uses the key (or some set of rules) to encrypt the plaintext and sends the cipher text to the receiver. The receiver applies the same key (or rule set) to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called symmetric encryption.

Fig 2 Basic symmetric key encryption and decryption.



With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key.

Secret key cryptography schemes are generally categorized as being either stream ciphers or block ciphers. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same cipher text when using the same key in a block cipher whereas the same plaintext will encrypt to different cipher text in a stream cipher.

Stream ciphers come in several flavors but two are worth mentioning here. Self-synchronizing stream ciphers calculate each bit in the key stream as a function of the previous *n* bits in the key stream. It is termed "self-synchronizing" because the decryption process can stay synchronized with the encryption process merely by knowing how far into the *n*-bit key stream it is. One problem is error propagation; a garbled bit in transmission will result in *n* garbled bits at the receiving side. Synchronous stream ciphers generate the key stream in a fashion independent of the message stream but by using the same key stream generation function at sender and receiver. While stream

ciphers do not propagate transmission errors, they are, by their nature, periodic so that the key stream will eventually repeat.

Block ciphers can operate in one of several modes; the following four are the most important:

- Electronic Codebook (ECB) mode is the simplest, most obvious application: the secret key is used to encrypt the plaintext block to form a cipher text block. Two identical plaintext blocks, then, will always generate the same cipher text block. Although this is the most common mode of block ciphers, it is susceptible to a variety of brute-force attacks.
- Cipher Block Chaining (CBC) mode adds a feedback mechanism to the encryption scheme. In CBC, the plaintext is exclusively-OR ed (XOR ed) with the previous cipher text block prior to encryption. In this mode, two identical blocks of plaintext never encrypt to the same cipher text.
- Cipher Feedback (CFB) mode is a block cipher implementation as a self-synchronizing stream cipher. CFB mode allows data to be encrypted in units smaller than the block size, which might be useful in some applications such as encrypting interactive terminal input. If we were using 1-byte CFB mode, for example, each incoming character is placed into a shift register the same size as the block, encrypted, and the block transmitted. At the receiving side, the cipher text is decrypted and the extra bits in the block (i.e., everything above and beyond the one byte) are discarded.
- Output Feedback (OFB) mode is a block cipher implementation conceptually similar to a synchronous stream cipher. OFB prevents the same plaintext block from generating the same cipher text block by using an internal feedback mechanism that is independent of both the plaintext and cipher text bit streams.

Secret key cryptography algorithms that are in use today include:

- Data Encryption Standard (DES): The most common SKC scheme used today, DES was designed by IBM in the 1970s and adopted by the National Bureau of

Standards (NBS) [now the National Institute for Standards and Technology (NIST)] in 1977 for commercial and unclassified government applications. DES is a block-cipher employing a 56-bit key that operates on 64-bit blocks. DES has a complex set of rules and transformations that were designed specifically to yield fast hardware implementations and slow software implementations, although this latter point is becoming less significant today since the speed of computer processors is several orders of magnitude faster today than twenty years ago. IBM also proposed a 112-bit key for DES, which was rejected at the time by the government; the use of 112-bit keys was considered in the 1990s, however, conversion was never seriously considered..

Two important variants that strengthen DES are:

- Triple-DES (3DES)*:* A variant of DES that employs up to three 56-bit keys and makes three encryption/decryption passes over the block.

- DESX: A variant devised by Ron Rivest. By combining 64 additional key bits to the plaintext prior to encryption, effectively increases the key length to 120 bits.

- Advanced Encryption Standard (AES): In 1997, NIST initiated a very public, 4-1/2 year process to develop a new secure cryptosystem for U.S. government applications. The result, the Advanced Encryption Standard, became the official successor to DES in December 2001. The algorithm can use a variable block length and key length; the latest specification allowed any combination of keys lengths of 128, 192, or 256 bits and blocks of length 128, 192, or 256 bits.

- CAST-128/256: CAST-128,a DES-like substitution-permutation crypto algorithm, employing a 128-bit key operating on a 64-bit block. CAST-256 is an extension of CAST-128, using a 128-bit block size and a variable length (128,

160, 192, 224, or 256 bit) key. CAST is named for its developers, Carlisle Adams and Stafford Tavares and is available internationally.

- International Data Encryption Algorithm (IDEA): Secret-key cryptosystem written by Xuejia Lai and James Massey, in 1992 and patented by Ascom; a 64-bit SKC block cipher using a 128-bit key.

- Blowfish: A symmetric 64-bit block cipher invented by Bruce Schneier; optimized for 32-bit processors with large data caches, it is significantly faster than DES on a Pentium/PowerPC-class machine. Key lengths can vary from 32 to 448 bits in length. Blowfish, available freely and intended as a substitute for DES or IDEA, is in use in over 80 products.

- Twofish: A 128-bit block cipher using 128-, 192-, or 256-bit keys. Designed to be highly secure and highly flexible, well-suited for large microprocessors, 8-bit smart card microprocessors, and dedicated hardware. Designed by a team led by Bruce Schneier and was one of the Round 2 algorithms in the AES process.

- Camellia: A secret-key, block-cipher crypto algorithm developed jointly by Nippon Telegraph and Telephone Corp. and Mitsubishi Electric Corporation in 2000. Camellia has some characteristics in common with AES: a 128-bit block size, support for 128-, 192-, and 256-bit key lengths, and suitability for both software and hardware implementations on common 32-bit processors as well as 8-bit processors (e.g., smart cards, cryptographic hardware, and embedded systems).

- MISTY1: Developed at Mitsubishi Electric Corp., a block cipher using 128-bit key and 64-bit blocks, and a variable number of rounds. Designed for hardware and software implementations, and is resistant to differential and linear cryptanalysis.

- Secure and Fast Encryption Routine (SAFER): Secret-key crypto scheme designed for implementation in software. Versions have been defined for 40-, 64-, and 128-bit keys.

- KASUMI: A block cipher using a 128-bit key that is part of the Third-Generation Partnership Project (3gpp), formerly known as the Universal Mobile Telecommunications System (UMTS). KASUMI is the intended confidentiality and integrity algorithm for both message content and signaling data for emerging mobile communications systems.

- SEED: A block cipher using 128-bit blocks and 128-bit keys. Developed by the Korea Information Security Agency (KISA) and adopted as a national standard encryption algorithm in South Korea.

- ARIA: A 128-bit block cipher employing 128-, 192-, and 256-bit keys. Developed by cryptographers at the Academy, Research Institute and Agency (ARIA) in Korea in 2003.

- Skipjack: SKC scheme proposed for Capstone. Although the details of the algorithm were never made public, Skipjack was a block cipher using an 80-bit key and 32 iteration cycles per 64-bit block.

1.2.2 Public-Key Cryptography

Public-key cryptography has been said to be the most significant new development in cryptography in the last 300-400 years. Modern PKC was first described publicly by Stanford University professor Martin Hellman and graduate student Whitfield Diffie in 1976. Their paper described a two-key crypto system in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key.

PKC depends upon the existence of so-called *one-way functions*, or mathematical functions that are easy to computer whereas their inverse function is relatively difficult to compute. Let me give you two simple examples:

1. Multiplication vs. factorization: Suppose I tell you that I have two numbers, 9 and 16, and that I want to calculate the product; it should take almost no time to calculate the product, 144. Suppose instead that I tell you that I have a number, 144, and I need you tell me which pair of integers I multiplied together to obtain that number. You will eventually come up with the solution but whereas calculating the product took milliseconds, factoring will take longer because you first need to find the 8 pair of integer factors and then determine which one is the correct pair.

2. Exponentiation vs. logarithms: Suppose I tell you that I want to take the number 3 to the 6th power; again, it is easy to calculate 3^6=729. But if I tell you that I have the number 729 and want you to tell me the two integers that I used, *x* and *y* so that $\log_x 729 = y$, it will take you longer to find all possible solutions and select the pair that I used.

While the examples above are trivial, they do represent two of the functional pairs that are used with PKC; namely, the ease of multiplication and exponentiation versus the relative difficulty of factoring and calculating logarithms, respectively. The mathematical "trick" in PKC is to find a trap door in the one-way function so that the inverse calculation becomes easy given knowledge of some item of information.

Generic PKC employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key. One key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext. The important point here is that it does not matter which key is applied first, but that both keys are required for the process to work. Because a pair of keys is required, this approach is also called asymmetric cryptography.

In PKC, one of the keys is designated the public *key* and may be advertised as widely as the owner wants. The other key is designated the private *key* and is never revealed to another party. It is straight forward to send messages under this scheme. Suppose Alice wants to send Bob a message. Alice encrypts some information using Bob's public key; Bob decrypts the cipher text using his private key. This method could be also used to prove who sent a message; Alice, for example, could encrypt some plaintext with her private key; when Bob decrypts using Alice's public key, he knows that Alice sent the message and Alice cannot deny having sent the message (non-repudiation)

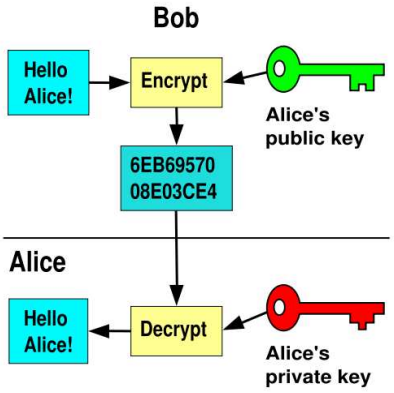


Fig 3: Public Key Cryptography

Public-key cryptography algorithms that are in use today for key exchange or digital signatures include:

- RSA: The first, and still most common, PKC implementation, named for the three MIT mathematicians who developed it — Ronald Rivest, Adi Shamir, and Leonard Adleman. RSA today is used in hundreds of software products and can be used for key exchange, digital signatures, or encryption of small blocks of data. RSA uses a variable size encryption block and a variable size key. The key-pair is derived from a very large number, n, that is the product of two prime numbers chosen according to special rules; these primes may be 100 or more digits in length each, yielding an n with roughly twice as many digits as the prime factors. The

public key information includes n and a derivative of one of the factors of n; an attacker cannot determine the prime factors of n (and, therefore, the private key) from this information alone and that is what makes the RSA algorithm so secure. (Some descriptions of PKC erroneously state that RSA's safety is due to the difficulty in factoring large prime numbers. In fact, large prime numbers, like small prime numbers, only have two factors!) The ability for computers to factor large numbers, and therefore attack schemes such as RSA, is rapidly improving and systems today can find the prime factors of numbers with more than 200 digits. Nevertheless, if a large number is created from two prime factors that are roughly the same size, there is no known factorization algorithm that will solve the problem in a reasonable amount of time; a 2005 test to factor a 200-digit number took 1.5 years and over 50 years of compute time.

- Diffie Hellman: After the RSA algorithm was published, Diffie and Hellman came up with their own algorithm. D-H is used for secret-key key exchange only, and not for authentication or digital signatures.

- Digital Signature Algorithm: The algorithm provides digital signature capability for the authentication of messages.

- Elliptic Curve Cryptography (ECC): A PKC algorithm based upon elliptic curves. ECC can offer levels of security with small keys comparable to RSA and other PKC methods. It was designed for devices with limited compute power and/or memory, such as smartcards and PDAs.

- Cramer shoup: A public-key cryptosystem proposed by R. Cramer and V. Shoup of IBM in 1998.

- Key Exchange Algorithm (KEA): A variation on Diffie-Hellman; proposed as the key exchange method for Capstone.

1.2.3 Hash Functions

Hash functions, also called message digests and one-way encryption, are algorithms that, in some sense, use no key (Figure 1C). Instead, a fixed-length hash value is computed based upon the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered. Hash algorithms are typically used to provide a digital fingerprint of a file's contents often used to ensure that the file has not been altered by an intruder or virus. Hash functions are also commonly employed by many operating systems to encrypt passwords. Hash functions, then, provide a measure of the integrity of a file.

Hash algorithms that are in common use today include:

- MD2: Designed for systems with limited memory, such as smart cards.
- MD4: Developed by Rivest, similar to MD2 but designed specifically for fast processing in software.
- MD5: Also developed by Rivest after potential weaknesses were reported in MD4; this scheme is similar to MD4 but is slower because more manipulation is made to the original data. MD5 has been implemented in a large number of products although several weaknesses in the algorithm were demonstrated by German cryptographer Hans Dobbertin in 1996.

1.3 Why Three Encryption Techniques?

So, why are there so many different types of cryptographic schemes? Why can't we do everything we need with just one?

The answer is that each scheme is optimized for some specific application(s). Hash functions, for example, are well-suited for ensuring data integrity because any change made to the contents of a message will result in the receiver calculating a different hash value than the one placed in the transmission by the sender. Since it is highly unlikely that two different messages will yield the same hash value, data integrity is ensured to a high degree of confidence.

Secret key cryptography, on the other hand, is ideally suited to encrypting messages, thus providing privacy and confidentiality. The sender can generate a *session key* on a per-message basis to encrypt the message; the receiver, of course, needs the same session key to decrypt the message.

Key exchange, of course, is a key application of public-key cryptography (no pun intended). Asymmetric schemes can also be used for non-repudiation and user authentication; if the receiver can obtain the session key encrypted with the sender's private key, then only this sender could have sent the message. Public-key cryptography could, theoretically, also be used to encrypt messages although this is rarely done because secret-key cryptography operates about 1000 times faster than public-key cryptography.

1.4 Visual Cryptography

The rapid growth of computer networks allowed large files, such as digital images, to be easily transmitted over the internet. Data encryption is widely used to ensure security however, most of the available encryption algorithms are used for text data. Due to large data size and real time constrains, algorithms that are good for textual data may not be suitable for multimedia data. The efficiency and success of e-commerce and business was fueled by the underlying growth of available network bandwidth. Over the past few years, internet-enabled business or e-business has drastically improved revenue and efficiency of large scale organizations. It has enabled organizations to lower operating costs and improves customer’s satisfaction. Such applications require networks which accommodate voice, image, video and protected data.

The security of digital images has become more and more important due to the rapid evolution of the Internet in the digital world today. The security of digital images has attracted more attention recently, and many different image encryption methods have been proposed to enhance the security of these images.

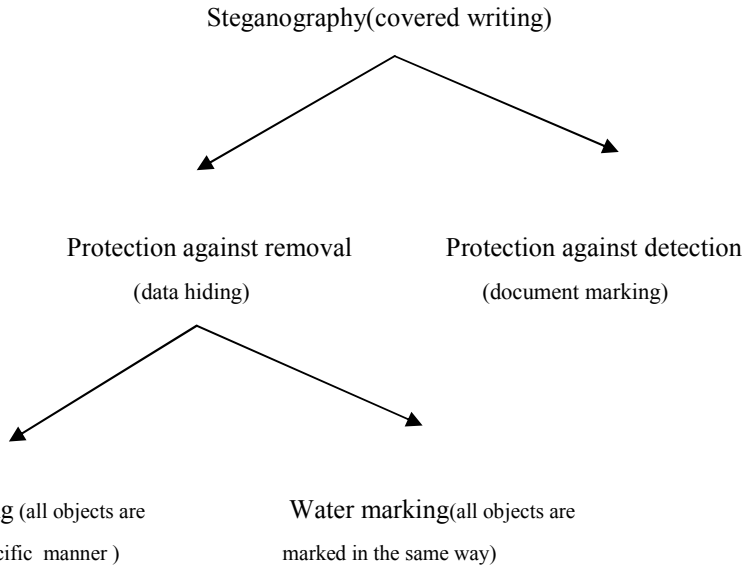
Visual cryptography is a cryptographic technique which allows visual information (pictures, text, etc) to be encrypted in such a way that the decryption can be performed by the human visual system without the aid of computers. As network technology has been greatly advanced, much information is transmitted via the Internet conveniently and rapidly. At the same time, the security issue is a crucial problem in the transmission process. For example, the information may be intercepted from transmission process. This are of cryptography aims to build a cryptosystem that would be able to encrypt any image in any standard format, so that the encrypted image when perceived by the naked eye or intercepted by any person with malicious intentions during the time of transmission of the image is unable to decipher the image. Firstly an image and key is fed into cryptosystem. The encryption algorithm produces a cipher image which is sent into receiver through a communication channel. When the cipher image reaches the destination, the receiver enters the key and the original image is decrypted.

1.5 Steganography

Steganography is derived from the Greek for covered writing and essentially means “to hide in plain sight”. As defined by Cachin steganography is the art and science of communicating in such a way that the presence of a message cannot be detected. Simple steganographic techniques have been in use for hundreds of years, but with the increasing use of files in an electronic format new techniques for information hiding have become possible.

Information hiding can be broken down into different areas. Steganography can be used to hide a message intended for later retrieval by a specific individual or group. In this case the aim is to prevent the message being detected by any other party. The other major area of steganography is copyright marking, where the message to be inserted is used to assert copyright over a document. This can be further divided into watermarking and fingerprinting r.

Fig 4: Types of steganography



Steganography and encryption are both used to ensure data confidentiality. However the main difference between them is that with encryption anybody can see that both parties are communicating in secret Steganography hides the existence of a secret message and in the best case nobody can see that both parties are communicating in secret. This makes steganography suitable for some tasks for which encryption aren't, such as copyright marking. Adding encrypted copyright information to a file could be easy to remove but embedding it within the contents of the file itself can prevent it being easily identified and removed

Encryption allows secure communication requiring a key to read the information. An attacker cannot remove the encryption but it is relatively easy to modify the file, making it unreadable for the intended recipient.

Digital signatures allow authorship of a document to be asserted. The signature can be removed easily but any changes made will invalidate the signature, therefore integrity is maintained.

Steganography provides a means of secret communication which cannot be removed without significantly altering the data in which it is embedded. The embedded data will be confidential unless an attacker can find a way to detect it.

1.5.1 Requirements of Hiding Information Digitally

There are many different protocols and embedding techniques that enable us to hide data in a given object. However, all of the protocols and techniques must satisfy a number of requirements so that steganography can be applied correctly. The following is a list of main requirements that steganography techniques must satisfy:

- The integrity of the hidden information after it has been embedded inside the stego object must be correct. The secret message must not change in any way, such as additional information being added, loss of information or changes to the secret information after it has been hidden. If secret information is changed during steganography, it would defeat the whole point of the process.

- The stego object must remain unchanged or almost unchanged to the naked eye. If the stego object changes significantly and can be noticed, a third party may see that information is being hidden and therefore could attempt to extract or to destroy it.

- In watermarking, changes in the stego object must have no effect on the watermark.

Imagine if you had an illegal copy of an image that you would like to manipulate in various ways. These manipulations can be simple processes such as resizing, trimming or rotating the image. The watermark inside the image must survive these manipulations, otherwise the attackers can very easily remove the watermark and the point of steganography will be broken.

- Finally, we always assume that the attacker knows that there is hidden information inside the stego object.

1.5.2 Types of Steganography

Steganography can be split into two types, these are Fragile and Robust. The following section describes the definition of these two different types of steganography.

• Fragile

Fragile steganography involves embedding information into a file which is destroyed if the file is modified. This method is unsuitable for recording the copyright holder of the file since it can be so easily removed, but is useful in situations where it is important to prove that the file has not been tampered with, such as using a file as evidence in a court of law, since any tampering would have removed the watermark. Fragile steganography techniques tend to be easier to implement than robust methods.

• Robust

Robust marking aims to embed information into a file which cannot easily be destroyed. Although no mark is truly indestructible, a system can be considered robust if the amount of changes required to remove the mark would render the file useless. Therefore the mark should be hidden in a part of the file where its removal would be easily perceived. There are two main types of robust marking.

Fingerprinting involves hiding a unique identifier for the customer who originally acquired the file and therefore is allowed to use it. Should the file be found in the possession of somebody else, the copyright owner can use the fingerprint to identify which customer violated the license agreement by distributing a copy of the file.

Watermarks identify the copyright owner of the file, not the customer. Whereas fingerprints are used to identify people who violate the license agreement watermarks help with prosecuting those who have an illegal copy.

One application of watermarking is in copyright protection systems, which are intended to prevent or deter unauthorized copying of digital media. In this use a copy device retrieves the watermark from the signal before making a copy; the device makes a decision to copy or not depending on the contents of the watermark. Another application is in source tracing. A watermark is embedded into a digital signal at each point of distribution. If a copy of the work is found later, then the watermark can be retrieved from the copy and the source of the distribution is known. This technique has been reportedly used to detect the source of illegally copied movies.

1.6 Cryptanalysis

Code making involves the creation of encryption products that provide protection of confidentiality. Code breaking involves defeating this protection by some mens other than the standard decryption process used by a intended recipient. Five scenarios for which code breaking is used. They are ensure accessibility, spying on opponents, selling cracking products and services, pursuing the intellectual aspects of code breaking and testing whether one's codes are strong enough.

Cryptanalysis is the process of attempting to discover either the plaintext Xp or the key K. Discovery of the encryption is the most desired one as with its discovery all the subsequent messages can be deciphered. Therefore, the length of encryption key, and

the volume of the computational work necessary provides for its strength i.e. resistance to breakage. The longer the key, the stronger the protection, the more brute force is needed.

Neither conventional encryption nor public key encryption is more resistant to cryptanalysis than the other. All that the user of an encryption algorithm can strive for is an algorithm that meets one or both of the following criteria: the cost of breaking the cipher exceeds the value of the encrypted information, the time required to break to exceed the normal lifetime of the information.

1.7 Compression

In computer science and information theory, data compression or source coding is the process of encoding information using fewer bits (or other information-bearing units) than a uuencoded representation would use, through use of specific encoding schemes.

Another concept related to compression is that of Data deduplication. In computing, data deduplication is a specialized data compression technique for eliminating coarse-grained redundant data, typically to improve storage utilization.

As with any communication, compressed data communication only works when both the sender and receiver of the information understand the encoding scheme. For example, this text makes sense only if the receiver understands that it is intended to be interpreted as characters representing the English language. Similarly, compressed data can only be understood if the decoding method is known by the receiver.

Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth. The design of data compression schemes therefore involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (if using a lossy compression scheme), and the computational resources required to compress and uncompress the data.

1.7.1 Lossless and Lossy Compression Techniques

Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely without error. Lossless compression is possible because most real-world data has statistical redundancy. For example, in English text, the letter 'e' is much more common than the letter 'z', and the probability that the letter 'q' will be followed by the letter 'z' is very small. Another kind of compression, called lossy data compression or perceptual coding, is possible if some loss of fidelity is acceptable. Generally, a lossy data compression will be guided by research on how people perceive the data in question. For example, the human eye is more sensitive to subtle variations in luminance than it is to variations in color. JPEG image compression works in part by "rounding off" some of this less-important information. Lossy data compression provides a way to obtain the best fidelity for a given amount of compression. In some cases, *transparent* (unnoticeable) compression is desired; in other cases, fidelity is sacrificed to reduce the amount of data as much as possible.

Lossless compression schemes are reversible so that the original data can be reconstructed, while lossy schemes accept some loss of data in order to achieve higher compression.

However, lossless data compression algorithms will always fail to compress some files; indeed, any compression algorithm will necessarily fail to compress any data containing no discernible patterns. Attempts to compress data that has been compressed already will therefore usually result in an expansion, as will attempts to compress all but the most trivially encrypted data.

1.7.2 Example algorithms and applications

The above is a very simple example of run-length encoding, wherein large runs of consecutive identical data values are replaced by a simple code with the data value and length of the run. This is an example of lossless data compression. It is often used to optimize disk space on office computers, or better use the connection bandwidth in a

computer network. For symbolic data such as spreadsheets, text, executable programs, etc., losslessness is essential because changing even a single bit cannot be tolerated (except in some limited cases).

For visual and audio data, some loss of quality can be tolerated without losing the essential nature of the data. By taking advantage of the limitations of the human sensory system, a great deal of space can be saved while producing an output which is nearly indistinguishable from the original. These lossy data compression methods typically offer a three-way tradeoff between compression speed, compressed data size and quality loss.

1.7.2.1 Lossy

Lossy image compression is used in digital cameras, to increase storage capacities with minimal degradation of picture quality. Similarly, DVDs use the lossy MPEG-2 Video codec for video compression.

In lossy audio compression, methods of psychoacoustics are used to remove non-audible (or less audible) components of the signal. Compression of human speech is often performed with even more specialized techniques, so that "speech compression" or "voice coding" is sometimes distinguished as a separate discipline from "audio compression".

1.7.2.2 Lossless

The Lempel-Ziv (LZ) compression methods are among the most popular algorithms for lossless storage. DEFLATE is a variation on LZ which is optimized for decompression speed and compression ratio, therefore compression can be slow. DEFLATE is used in gzip and PNG. LZW (Lempel-Ziv-Welch) is used in GIF images. . LZ methods utilize a table-based compression model where table entries are substituted for repeated strings of data. For most LZ methods, this table is generated dynamically from earlier data in the input.

1.8 Image Compression

Image compression is minimizing the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in file size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

There are several different ways in which image files can be compressed. For Internet use, the two most common compressed graphic image formats are the JPEG format and the GIF format. The JPEG method is more often used for photographs, while the GIF method is commonly used for line art and other images in which geometric shapes are relatively simple.

Other techniques for image compression include the use of fractals and wavelets. These methods have not gained widespread acceptance for use on the Internet as of this writing. However, both methods offer promise because they offer higher compression ratios than the JPEG or GIF methods for some types of images. Another new method that may in time replace the GIF format is the PNG format.

A text file or program can be compressed without the introduction of errors, but only up to a certain extent. This is called lossless compression. Beyond this point, errors are introduced. In text and program files, it is crucial that compression be lossless because a single error can seriously damage the meaning of a text file, or cause a program not to run. In image compression, a small loss in quality is usually not noticeable. There is no "critical point" up to which compression works perfectly, but beyond which it becomes impossible. When there is some tolerance for loss, the compression factor can be greater than it can when there is no loss tolerance. For this reason, graphic images can be compressed more than text files or programs.

1.9 DCT (Discrete Cosine Transformation)

In today's technological world as our use of and reliance on computers continues to grow, so too does our need for efficient ways of storing large amounts of data and due to the bandwidth and storage limitations, images must be compressed before transmission and storage. For example, someone with a web page or online catalog that uses dozens or perhaps hundreds of images will more than likely need to use some form of image compression to store those images. This is because the amount of space required holding images can be prohibitively large in terms of cost. Fortunately, there are several methods of image compression available today. This fall into two general categories: lossless and lossy image compression.

However, the compression will reduce the image fidelity, especially when the images are compressed at lower bit rates. The reconstructed images suffer from blocking artifacts and the image quality will be severely degraded under the circumstance of high compression ratios. In order to have a good compression ratio without losing too much of information when the image is decompressed we use DCT.

A discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. The JPEG process is a widely used form of lossy image compression that centers on the Discrete Cosine Transform. DCT and Fourier transforms convert images from time-domain to frequency- domain to decorrelate pixels. The DCT transformation is reversible .The DCT works by separating images into parts of differing frequencies. During a step called quantization, where part of compression actually occurs, the less important frequencies are discarded, hence the use of the term "lossy". Then, only the most important frequencies that remain are used retrieve the image in the decompression process. As a result, reconstructed images contain some distortion; but as we shall soon see, these levels of distortion can be adjusted during the compression stage. The JPEG method is used for both color and black- and-white images.

1.9.1The JPEG process

The following is a general overview of the JPEG process. JPEG stands for Joint Photographic Experts Group which is a commonly used method of compression for photographic images. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. JPEG typically achieves 10:1 compression with little perceptible loss in image quality. More comprehensive understanding of the process may be acquired as such given under:

1. The image is broken into 8x8 blocks of pixels.
2. Working from left to right, top to bottom, the DCT is applied to each block.
3. Each block is compressed through quantization.
4. The array of compressed blocks that constitute the image is stored in a drastically reduced amount of space.
5. When desired, the image is reconstructed through decompression, a process that uses the Inverse Discrete Cosine Transform (IDCT).

1.9.2 One Dimensional DCT

Like other transforms, the Discrete Cosine Transform (DCT) attempts to decorrelate the image data. After decorrelation each transform coefficient can be encoded independently without losing compression efficiency. This section describes the DCT and some of its important properties.

The One-Dimensional DCT Equation:

$$xc(k) = (1/N) \sum_{n=0}^{N-1} xn \cos(k2\pi n/N)$$

where k = 0, 1, 2, …, N-1

One-Dimensional IDCT Equation:

$$xc(k) = \sum_{n=0}^{N-1} c[u]\, xn \cos(k2\pi n/N),$$

where k = 0, 1, 2, …, N-1

Xn is the DCT result

c[u] = 1 for u=0

c[u] = 2 for u=1,2,3,…N-1

1.9.3 Two-Dimensional DCT

The Discrete Cosine Transform (DCT) is one of many transforms that takes its input and transforms it into a linear combination of weighted basis functions. These basis functions are commonly the frequency. The 2-D Discrete Cosine Transform is just a one dimensional DCT applied twice, once in the x direction, and again in the y direction. One can imagine the computational complexity of doing so for a large image. Thus, many algorithms, such as the Fast Fourier Transform (FFT), have been created to speed the computation.

Two-Dimensional IDCT Equation:

$$f[m, n] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} c[u]\, c[v]\, F[u, v] \cos[(2m + 1)u\pi/2N] \cos[(2n + 1)v\pi/2N]$$

where:

m, n = image result pixel indices(0, 1, 2, …, N – 1),

F[u, v] = N by N DCT result,

c[λ] = 1 for λ=0 and c[λ]=2 for λ=1,2,3,…N-1

f[m, n] = N by N IDCT result

Two-Dimensional IDCT Equation:

$$f[m, n] = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} c[u]\, c[v]\, F[u, v] \cos[(2m + 1)u\pi / 2N] \cos[(2n + 1)v\pi / 2N]$$

where: m, n = image result pixel indices(0, 1, 2, …, N – 1),

F[u, v] = N by N DCT result,

c[λ] = 1 for λ=0 and c[λ]=2 for λ=1,2,3,…N-1

f[m, n] = N by N IDCT result

CHAPTER 2

LITERATURE REVIEW

LITERATURE REVIEW

2.1 Image Encryption Techniques

In this section, a few newly proposed techniques for image encryption, has been introduced.

1) **A New Block Image Encryption Algorithm by Fridrich, 1997**

Jiri Fridrich presented an encryption algorithm that adapted certain invertible chaotic two-dimensional maps to create new symmetric block encryption schemes. This scheme is especially useful for encryption of large amount of data, such as digital images

2) **A Technique for Image Encryption using Digital Signatures, 2003**

Aloka Sinha and Kehar Singh have proposed a new technique to encrypt an image for secure image transmission. The digital signature of the original image is added to the encoded version of the original image. Image encoding is done by using an appropriate error control code, such as a Bose-Chaudhuri Hochquenghem (BCH) code. At the receiver end, after the decryption of the image, the digital signature has been used to verify the authenticity of the image.

3) **A Technique for Image Encryption using multi level and image dividing technique, 2003**

Chang-Mok Shin**,** Dong-Hoan Seo**,** Kyu-Bo Chol, Ha-Wmn Lee, and SmJmng Kim[16] proposed image encryption by using binary exclusive OR operation and image dividing technique. They converted binary images to binary

phase encoding and then encrypt these images with binary random phase images by binary phase XOR operation. Encrypted gray image was then obtained by combining each binary encrypted images.

4) Image Encryption Using Advanced Hill Cipher Algorithm

In this paper, we have proposed an advanced Hill (AdvHill) cipher algorithm which uses an Involutory key matrix for encryption The objective of this paper is to overcome the drawback of using a random key matrix in Hill cipher algorithm for encryption, where we may not be able to decrypt the encrypted message, if the key matrix is not invertible. Divide the image into blocks apply the involutory key matrix to each block and create a temporary block using the ith pixel value of each block again multiply it with involuntary key matrix and find transpose of it transfer it to destination

5) Image Encryption Using Block-Based Transformation Algorithm, 2006

In this paper the original image is divided into random number of blocks the original image is divided into a random number of blocks that are then shuffled within the image. The generated (or transformed) image is then fed to the Blowfish encryption algorithm.

6) H-S-X Cryptosystem and Its Application to Image Encryption, 2009

In this paper, we have proposed a novel technique which is a modified version of Hill cipher algorithm for image encryption named H-S-X (Hill-Shift-XOR) which can be applied to any type of images whether they are colour or gray. First color image is decomposed into (R-G-B) components. Second, encrypt each component (R-G-B) separately by the algorithm. Finally, concatenate the encrypted components together to get the encrypted color image

7) Image Encryption Using DCT and Stream Cipher, 2009

The proposed method based on the idea of decomposing the image into 8x8 blocks, these blocks are transformed from the spatial domain to frequency domain by the DCT. Then, the DCT coefficients correlated to the higher frequencies of the image block are encrypted using Non-Linear Shift Back. The concept behind encrypting only some selective DCT coefficients based on the fact that the image details are situated in the higher frequencies, while the human eye is most sensitive to lower frequencies than to higher frequencies. Encrypt the selected coefficients by XORing the generated bit stream from the NLFSR +Key with the coefficient bits, the sign bit of the selected coefficients will not be encrypted.

8) Choase Based Image Encryption Using Block-Based Transformation Algorithm

The proposed algorithm is for image compression and encryption. The proposed algorithm with block size of 8-bit applies wavelet transform for each block for image compression and 256-bit secret key used for image encryption. The key is used to generate a pad that is then merged with the plaintext a byte at a time.

9) New Image Encryption and Compression Method Based on Independent Component Analysis

In this paper a new method is proposed combining encryption and compression based on Embedding and Discrete Cosine Transform (DCT). For encryption, target images are covered with an insignificant image to hide them and their mixtures to be transmitted are obtained. The receiver reconstructs the

original images by removing the embedded random image and for compression I am using DCT and for decompression IDCT.

2.2 Authors Contribution

Telecommunication becomes one of our modern society's characteristics, which requires more and more new techniques to meet the increasing needs of a modern society. However, for any communication system, it is necessary to take into account two major requirements: a fast transmission to send the information from a transmitter to a receiver (that can be done using an efficient compression technique) and a secure transmission of information (which can be achieved using a powerful encoding algorithm).

To satisfy these constraints, new compression and encryption methods allowing a fast and secure information transmission are proposed in the literature. However, these methods (compression and encoding) are often developed in an independent manner although they are strongly connected and one influences the other.

After doing the literature review in my paper I am trying to combine the encryption and compression technique and to modify an all ready existing algorithm titled "**New Image Encryption and Compression Method Based on Independent Component Analysis and Discrete Cosine Transformation** "and trying to implement it in MatLab. In the reviewed paper they are trying to fuse to an image that is the original image and random image by using a model of Blind Source Separation (BSS) problem in the encoding phase and an Independent Component Analysis (ICA) algorithm in decoding phase. I have modified the methodology by using which they have encrypted the image. Instead of Independent Component Analysis I am covering the original image with random image (Embedding) and at the destination I am extracting the random image out (Extracting).

In addition we would like to compress the transmitting data, to achieve a high speed communication. For this purpose we utilized Discrete Cosine Transform (DCT)

and cut out the higher-frequency components because most of the power is concentrated in the lower frequency bands by DCT. Then the compressed DCT components are rotated, all the DCT components have energy in the lower frequencies and they are highly correlated to each other. The compressed original image is covered with random image and send to destination.

It is supposed that we transmit important images to a receiver, preventing non-authorized people from intercepting the images. In order to encrypt the images we cover the images with an insignificant image or random image .This process is called as Embedding and once after reaching the destination the random image is Extracted out and the original image is retrieved .This process is called Extraction

CHAPTER 3
METHODOLOGY

METHODOLOGY

3.1 Design Used

Nowadays, due to the prevalence of the network communication, there has been more and more multimedia data transmission on the Internet. In the multimedia data transmission, images are transmitted in high rate. Preventing an important image data from being stolen by the eavesdroppers is becoming an important one.

For any communication system, it is necessary to take into account two major requirements: a fast transmission to send the information from a transmitter to a receiver (that can be done using an efficient compression technique) and a secure transmission of information which can be achieved using a powerful encoding algorithm. To satisfy these constraints, new compression and Encryption methods allowing a fast and secure information transmission are proposed in the literature. Accordingly, we propose to combine compression with encryption and propose a new method of compression and encryption at same time.

In order to encrypt the images we cover the images with an insignificant image. Our new method is based on the hiding of information (Embedding) in the transmitter side and taking out (Extracting) algorithm in receiver side the decoding phase.. The simplicity and the good performances of these techniques have been very much motivating for me. In addition we would like to compress the transmitting data, to achieve a high speed communication.

For this purpose we utilized Discrete Cosine Transform (DCT) and cut out the higher-frequency components because most of the power is concentrated in the lower frequency bands by DCT. Then the compressed DCT components are rotated, the rotations have another aspect. The directions and degrees of the rotations are saved as

"key" to restore the original images. If the receiver does not know "key," it is hard to restore the original images.

Data hiding is defined as the process by which a message or image is imperceptibly embedded into a host or cover to get a composite signal. Generally, in encryption, the actual information is not maintained in its original format and thereby it is converted into an alternative equivalent multimedia file like image, video or audio, which in turn is being hidden within another object. This apparent message is sent through the network to the recipient, where the actual message is separated from it.

In this paper, we are using the method of digital embedding where the original image is hidden by a random image, which could survive attacks on the network. The digital embedding technique proposed for hiding an image into another image helps to maintain the quality of the recovered image. The image file, which is to be hidden, is here referred as Target Image and the image behind which it is to be hidden is termed as Cover Image

In this paper the whole process is divided into two sections.

- Transmitter side process
- Receiver side process

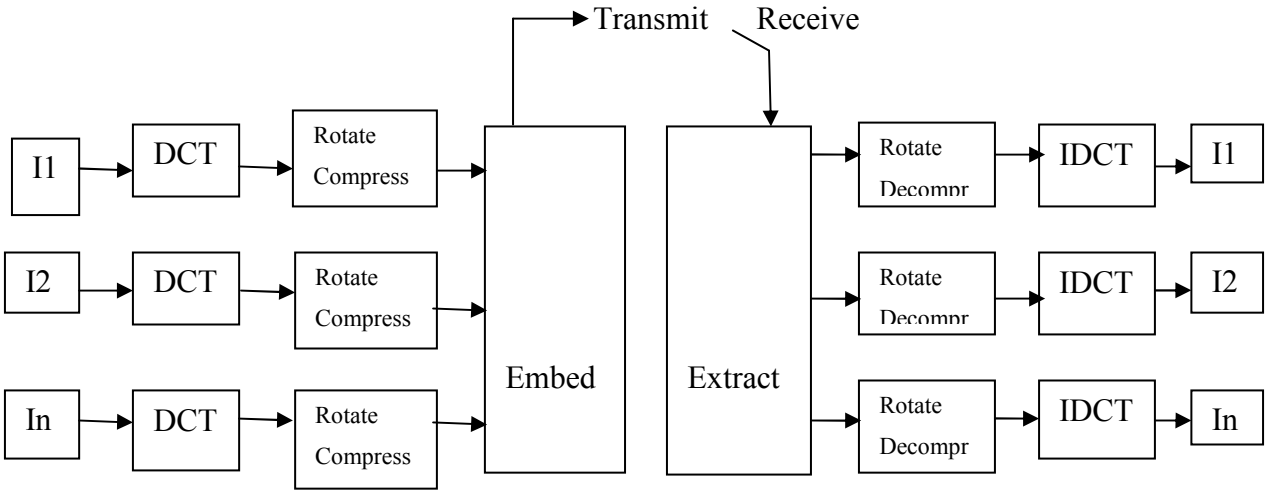


Fig 5: Our approach for encryption and compression

3.2 Approach followed in the paper

3.2.1 Transmitter side process

- **Discrete Cosine Transform:**

At first we divide original images to be transmitted into small square blocks and apply two-dimensional discrete cosine transform to each block and we obtain DCT components of each block.

- **Compression:**

For a fast communication, we would like to reduce the amount of transmitting data. Consequently, compression of the DCT components is required. In each block, most of DCT components have high energies in low frequency bands, we only use low frequency components through a simple low pass filter that is, left-up corners of each block with size of NC × NC are selected and higher frequency components are dropped. As a result of this process we can compress the transmitting images.

DCT	DCT	DCT
DCT	DCT	DCT
DCT	DCT	DCT

Fig 6: Dividing original image into small blocks and apply DCT

- **Rotation:**

After the compression we have to rotate the small blocks randomly. Therefore, in order to overcome this problem we rotate each block randomly, so as to make rotated DCT components be independent of each other. DCT blocks are rotated

1	2	3
4	5	6
7	8	9

No rotation

1	4	7
2	5	8
3	6	9

Rotation 1

7	4	1
8	5	2
9	6	3

Rotation 2

3	2	1
6	5	4
9	8	7

Rotation 3

9	8	7
6	5	4
3	2	1

Rotation 4

9	6	3
8	5	2
7	4	1

Rotation 5

3	6	9
2	5	8
1	4	7

Rotation 6

7	8	9
4	5	6
1	2	3

Rotation7

Fig 7: Rotation patterns

- **Embedding**

The compressed rotated image is covered by a random image and sends to destination

3.2.2 Receiver side process

- **Extracting**

Authorized people receive the mixtures and extract it. Turning back the rotated DCT components and using inverse discrete cosine transform (IDCT), original images are decrypted.

- **Inverse rotation and inverse discrete cosine transform:**

After separation of the original image and random image original images are reconstructed. The rotated DCT components have to be restored. The receiver is beforehand given the rotation "key" by the transmitter. Based on the rotation "key", the receiver can reconstruct the original images, rotating the DCT components contrary to the encryption stage. Without rotation key, it is difficult to reconstruct the original DCT components. Finally he can apply inverse discrete cosine transform (IDCT) to them. In this way the receiver can obtain the estimated original images from the observed mixtures.

3.3 Algorithm

3.3.1 Transmitter Algorithm

- First divide the original or target image into blocks and apply DCT(Discrete Cosine Transformation) on each blocks
- Then rotate the DCT blocks keep the direction of rotation as key for reconstructing the image
- Then cover the original image with another random image
- The random image is also divided into blocks and applies DCT on each block and the original image is covered by random image and it is send to the destination. This process is called embedding

3.2.2 Receiver Algorithm

- The random image is taken out .This process is called extraction
- Using the rotation key the DCT blocks are reconstructed for the source image
- Then to each block we apply inverse DCT
- The image is reconstructed

3.4 Tool Survey

MATLAB (matrix laboratory) is MATLAB is a high-level technical computing language and interactive environment for algorithm development; data visualization, data analysis, and numeric computation We can use MATLAB in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. Add-on toolboxes (collections of special-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems in these application areas.

MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications. Using the MATLAB product, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and FORTRAN. MATLAB also provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, object-oriented programming (OOP), and debugging features.

3.4.1 Key Features

- High-level language for technical computing
- Development environment for managing code, files, and data
- Interactive tools for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration

- 2-D and 3-D graphics functions for visualizing data
- Tools for building custom graphical user interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java, COM, and Microsoft Excel

3.5 Function Used in the Code

3.5.1 Functions used in transmitter side

- **mainTx Function**

  This is the main function in our algorithm. This function will be used in the sender side and will call other modules of our algorithm.

  Input: This function will take Target Image and Cover Image as input.

  Output: It will output the encoded stego-image.

- **Dctb Function**

  This function is called inside mainTx function. Dctb is used to do Discrete Cosine Transformation. It helps to compress the image .It is applied on target image as well as the cover image or random image

  Input: It takes the target image as input t for the first time and it takes cover image as input for the second time

  Output: The compressed image.

- **Hide_image Function**

  This function is also called inside the mainTx function to embed the original image with random image.

  Input: The compressed random image and compressed original image.

  Output: The image to be transmitted to the receiver side ie Txoutput (the output of mainTx)

3.5.2 Functions used in receiver side

- **mainRx.m**

  This is the main function in our algorithm. This function will be used in the receiver side and will call other modules of our algorithm.

  Input: This function will take output of mainTx.m as input.

  Output: It will output the original image.

- **Hide_image Function**

  This function is also called inside the mainRx function to extract the random image

  Input: The output of mainTx.m ie the compressed embedded image

  Output: The compressed original image to be transmitted to the receiver side ie Txoutput (the output of mainTx)

- **IDctb Function**

  This function is called inside mainRx function. IDctb is used to do Inverse Discrete Cosine Transformation. It helps to decompress the image .It is applied on compressed original image

  Input: It takes the compressed original image

  Output: The decompressed original image

CHAPTER 4

IMPLEMENTATION

IMPLEMENTATION

4.1 Coding in MATLAB

mainTx.m

```
close all;clear all;clc;
Im1=imread('lena256gray.bmp');    %%%%%%%%  first image in which an another
image will embedded
figure('name','Lena Image');
imshow(Im1);title('Input Image');

%%%%%%%%%%%%%%%%%% Dividing the input image into 16X16 block size
%%%%%%%%%

[imm imn]=size(Im1);       %%%%%%%%%%%%%%% Size of the Im1 image
Sr=16;                  %%%%%%%%%%%%%%% size of image which we have to divide
the image
Lnum=(imm/Sr)*(imn/Sr);
ln=zeros(Lnum,Sr,Sr);
for i=1:imm/Sr
   for j=1:imn/Sr
      k=(i-1)*imn/Sr+j;
      ln(k,:,:)=Im1((i-1)*Sr+1:i*Sr,(j-1)*Sr+1:j*Sr);

   end
end

%%%%%%%%%%%%%%%%%%%% applying dct to blocks divided
%%%%%%%%%%%%%%%%%55
```

```
ldn=dctb(ln);          %%%%%%%%% function calling
%%%%%%%%%%%%%%%%%%% compression of lena image
%%%%%%%%%%%%%5
[nb sb]=size(ldn);
for n=1:nb
   ldc(:,:,n)=ldn(n,1:4,1:4);
end

Ikey=randperm(4);
for n=1:nb
   for r=1:4
      ldr(:,r,n)=ldc(:,Ikey(r),n);
   end
end



k=1;
A=[];
%for n=1:nb
%lde(:,:,n)=ln(n,1:4,1:4);
%end
for i=1:imm/Sr
   ci=[];
   for j=1:imn/Sr
      ci=[ ci ldr(:,:,k)];
      k=k+1;
   end
   A=[A;ci];
end
Aa=A;
```

```
figure('name','Compressed image');imshow(uint8(Aa));title('The Compressed output of lena image');
Aa=imresize(Aa,[12 12]);
imwrite(Aa,'tiny.bmp');    % This is a image of compressed lena images

%%%%%%%%%%%%%%%%%%%%%%%%%%

Im2=imread('Megastar.bmp');figure('name','Cover image');
imshow(Im2); title('The Random image');

[imm imn]=size(Im2);
Lnum=(imm/Sr)*(imn/Sr);
ln1=zeros(Lnum,Sr,Sr);
for i=1:imm/Sr
   for j=1:imn/Sr
      k=(i-1)*imn/Sr+j;
      ln1(k,:,:)=Im2((i-1)*Sr+1:i*Sr,(j-1)*Sr+1:j*Sr);

   end
end
%%%%%%%%% applying dct to blocks divided %%%%%%%%%

%ldn1=dctb(ln1);
[nb1 sb1]=size(ln1);
for n=1:nb1%compression
   ldc1(:,:,n)=ln1(n,1:4,1:4);
end
k1=1;
B=[];
for i=1:imm/Sr
   ci1=[];
```

```
    for j=1:imn/Sr
        ci1=[ ci1 ldc1(:,:,k1)];
        k1=k1+1;
    end
    B=[B;ci1];
end

figure('name','Compressed Image');imshow(uint8(B));title('Compressed image of
Random Image');
imwrite(uint8(B),'compressedcover.bmp');
[row col]=size(Aa);% size(B)
C=zeros(row,col);
for p=1:row
    for q=1:col
        C(p,q)=Aa(p,q)+B(p,q);
    end
end

%figure('name',' output');imshow(uint8(C));title('The final output figure');

save dummy.mat I*;
hide_image('compressedcover.bmp','tiny.bmp');
TOI=imread('compressedc

over.bmp');

%%%%%%%% to show the final transmitter output in figure window write the
%%%%%%%% below in Command window;
%%%%%%%% figure;imshow(TOI); title('The TxOutput');
```

Dctb.m

```
function c=dctb(a)

[l l1 l2]=size(a);
for m=1:l2
for k=1:l1
for i= 1:l
   sum=0;
   for j=1:l
      sum=sum+a(j,k,m)*cos(3.14*(2*j+1)*i/2*l*l1);
   end
   c(i,k,m)=sum*sqrt(2/l);
end
end

end
```

hide_image.m

```
function hide_image(src_img,varargin)

NOI=nargin-1;
AI=[];
image1props=[];
szim=zeros(NOI,3);
check_limit=0;
for i=1:NOI
   b=double(imread(char(varargin(i))));
   szim(i,1)=size(b,1);szim(i,2)=size(b,2);szim(i,3)=size(b,3);
   check_limit=check_limit+size(b,1)*size(b,2)*size(b,3);
   AI=[AI reshape(b,1,size(b,1)*size(b,2)*size(b,3))];
```

```
end
for i=1:NOI
   image1props=[image1props reshape([floor(szim(i,3)/3) binary_vector(szim(i,1))'
zeros(1,15-length(binary_vector(szim(i,1)))) binary_vector(szim(i,2))' zeros(1,16-
length(binary_vector(szim(i,2))))],8,4)];
end
next=4*NOI+1;
a=double(imread(src_img));
%POI=(100*8*(check_limit+4*NOI+1))/(size(a,1)*size(a,2)*size(a,3));
a2=[reshape(a,1,size(a,1)*size(a,2)*size(a,3)) zeros(1,8-
rem(size(a,1)*size(a,2)*size(a,3),8))];
a11=2.*floor(reshape(a2,8,length(a2)/8)/2);
a11(:,1)=a11(:,1) + binary_vector(NOI);
a11(:,2:next)=a11(:,2:next) + image1props;
for j=1:NOI
   b1=AI(next-4*NOI:next-4*NOI-1+szim(j,1)*szim(j,2)*szim(j,3));
   for i=1:length(b1)
      a11(:,i+next)=a11(:,i+next)+ binary_vector(b1(i));
   end
   next=next+szim(j,1)*szim(j,2)*szim(j,3);
end
a111=reshape(a11,1,size(a11,1)*size(a11,2)*size(a11,3));
a111=a111(1:end-(8-rem(size(a,1)*size(a,2)*size(a,3),8)));
a22=reshape(a111,size(a,1),size(a,2),size(a,3));
imwrite(uint8(a22),'The TxOutput.png');
function out = binary_vector(in)
out=zeros(8,1);
if ~in
   return;
else
   while(1)
```

```
        out(floor(log2(in))+1)=1;
        in= in - power(2,floor(log2(in)));
        if ~in
            break;
        end
    end
end
```

mainRx.m

```
clear all;
load 'dummy.mat' I*
extract_image('The TxOutput.png');    %%%%%%%%%%%%%%%% function calling two
extract those images
IE=imread('The TxOutput.png') ;  %%%%%%%%%%%%%%%%%%%% reading that
extracted image

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%

for i=1:16
    for j=1:16
        k=(i-1)*16+j;
        ildn(k,:,:)=IE((i-1)*4+1:i*4,(j-1)*4+1:j*4);

    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Applying inverse DCT %%%%%%%%%%%%%%%%%%%%%%%%%

iln=idctb(ildn);            %%%%%%%%%%%% function calling
[nb nb1]=size(iln);
```

```
      %%%%%%%%%%%inverse rotation %%%%%%%%%%%

for n=1:nb
   for r=1:4
      ildr(n,Ikey(r),:)=iln(n,r,:);
   end
end
%RI=reshape(ildr,[64 64]);
for n=1:nb
   ildc(:,:,n)=ildr(n,1:4,1:4);
end

 %%%%%%%%%%%%%%%%%% decompression of the image
%%%%%%%%%%%%
k=1;im1=[];
for n=1:2:nb
%ldm(:,:,n)=ldc(:,:,n)*ldc1(:,:,n);
ildm(:,:,n)=ildc(:,:,n);
ildm1(:,:,n)=ildc(:,:,n+1);
end
ildm(:,:,nb)=zeros(4,4);
ildm1(:,:,nb)=zeros(4,4);

%%%%%%%%%%%%%%%%%% reconstructed image %%%%%%%%%%%%
[imm imn]=size(Im1);
Sr=16;
for i=1:imm/Sr
   for j=1:imn/Sr
      k=(i-1)*imn/Sr+j;
      ln(k,:,:)=Im1((i-1)*Sr+1:i*Sr,(j-1)*Sr+1:j*Sr);
```

```
    end
end
for n=1:k
    ldc(:,:,n)=ln(n,:,:);
end
t=1;
s1=1;

Out=[];
for i=1:imm/Sr
    var=[];
    for j=1:imn/Sr
        var=[ var ldc(:,:,t)];
       t=t+1;
    end
   Out=[Out;var];
end
imwrite(Out,'TheOutputRx.jpg');
%%%%%%%%%%%    figure();imshow(Out);title('The Rx Ouput')    %%%%
%%%%%%%% The above line should be write in Command Window then only you can
%%%%%%%% see the output of the Rx %%%%%%%%%
```

Extract_image.m

```
function extract_image(source_image)
a=double(imread(source_image));
a1=reshape(a,1,size(a,1)*size(a,2)*size(a,3));
a2=rem([a1 zeros(1,8-rem(length(a1),8))],2);
a11=reshape(a2,8,length(a2)/8);
num_of_images=sum(a11(:,1).*power(2,0:7)');
```

```
for i=1:num_of_images
   x=4*i-2;
szim(i,1)=sum([a11(2:end,x) ; a11(:,x+1)].*power(2,0:14)');
szim(i,2)=sum([a11(:,x+2) ; a11(:,x+3)].*power(2,0:15)');
szim(i,3)=a11(1,x)*2+1;
end
next=4*num_of_images+1;
for j=1:num_of_images
a22=zeros(1,szim(j,1)*szim(j,2)*szim(j,3));
for i=1:szim(j,1)*szim(j,2)*szim(j,3)
a22(i)=sum(a11(:,i+next).*power(2,0:7)');
end
a33=reshape(a22,szim(j,1),szim(j,2),szim(j,3));
outout_img_filename=['extracted' num2str(j) '.png'];
imwrite(uint8(a33),outout_img_filename);
next=next+szim(j,1)*szim(j,2)*szim(j,3);
end
```

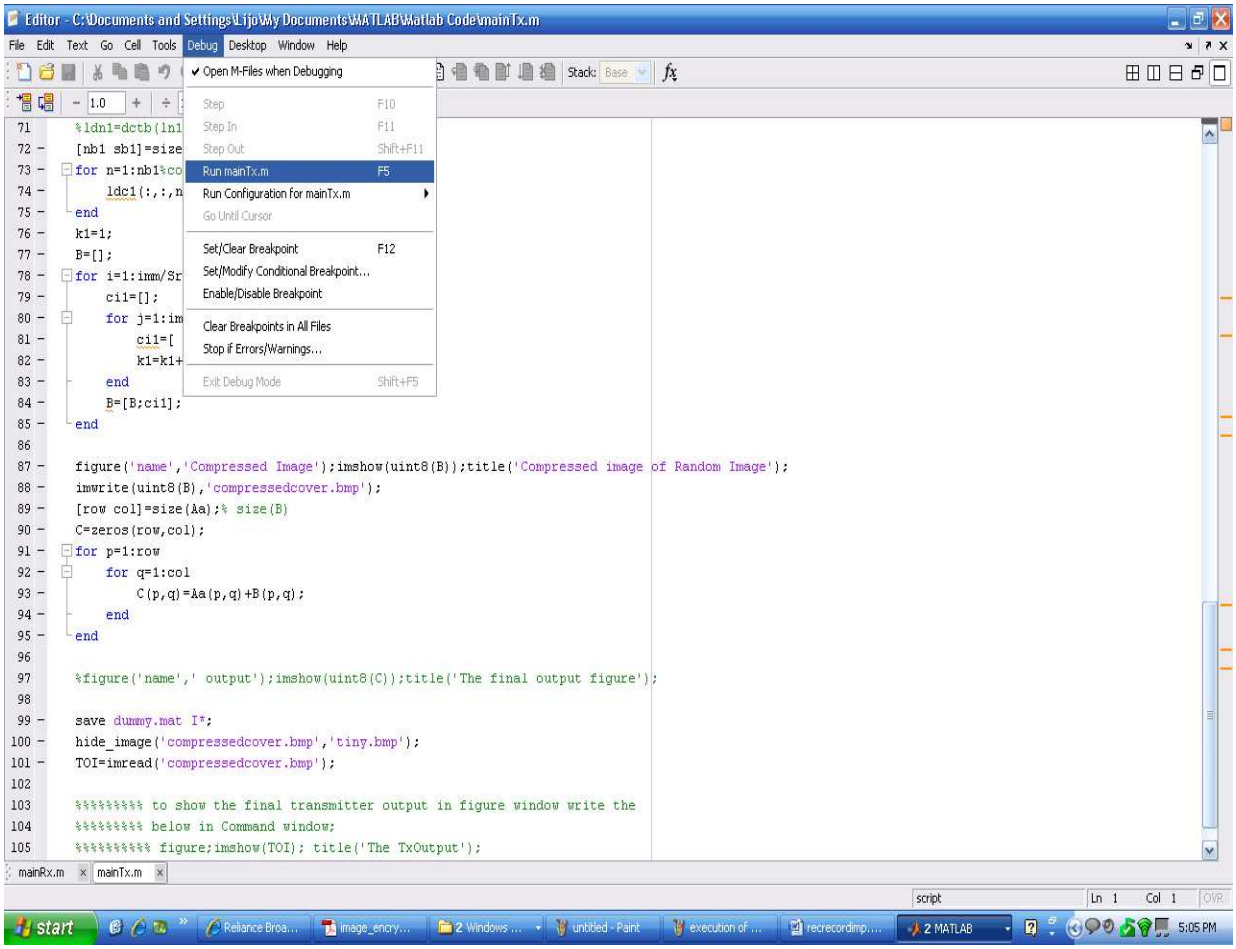
idctb.m

```
function c=idctb(a)

[l l1 l2]=size(a);
for m=1:l2
for k=1:l1
for i= 1:l
   sum=0;
   for j=1:l
      sum=sum+sqrt(2/l)*a(j,k,m)*cos(3.14*(2*j+1)*i/2*l*l1);
   end
```

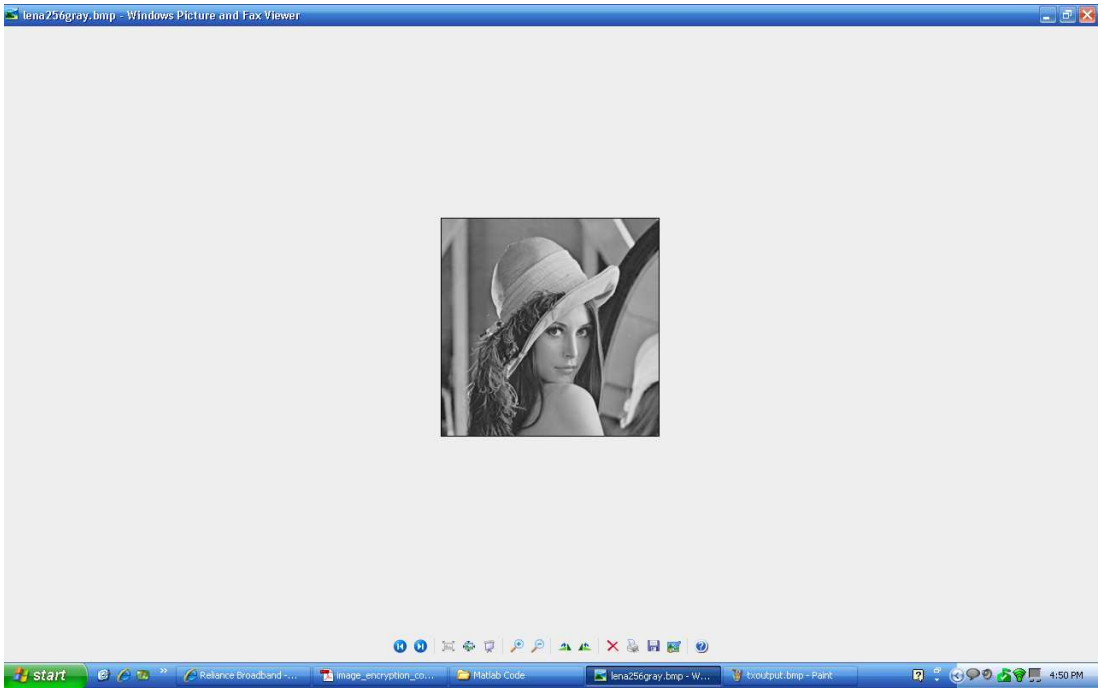
```
    c(i,k,m)=sum;
end
end

end
```

4.2 Screen Shots

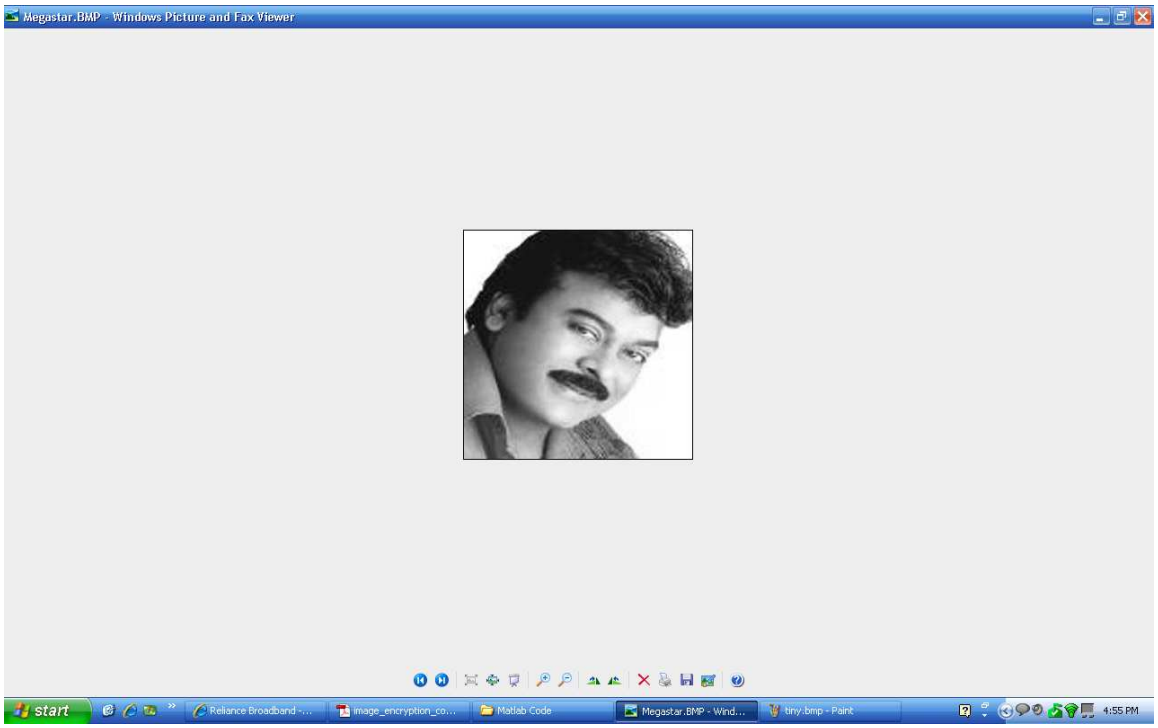
Execution of mainTx.m



Input of mainTx.m

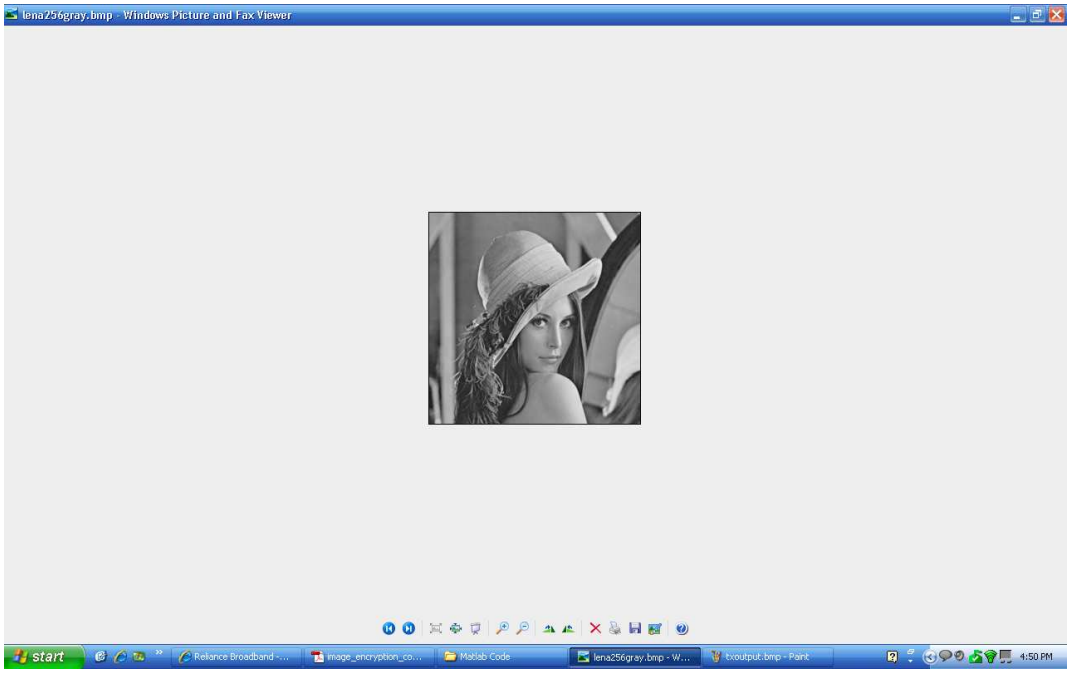


Input of mainTx.m

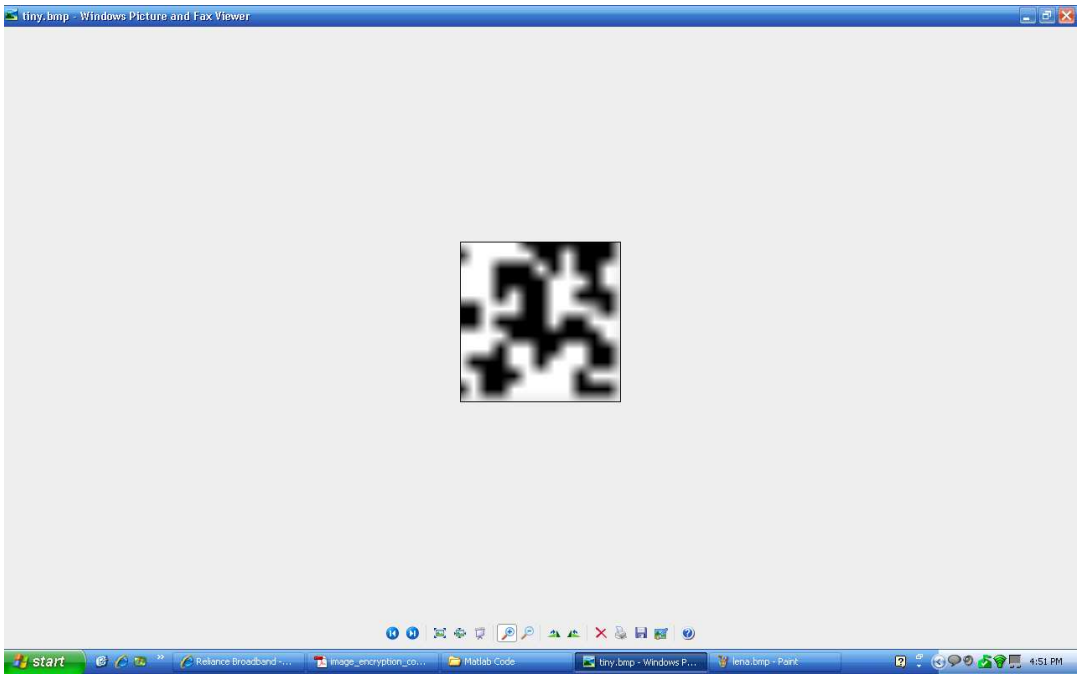


Output of mainTx.m

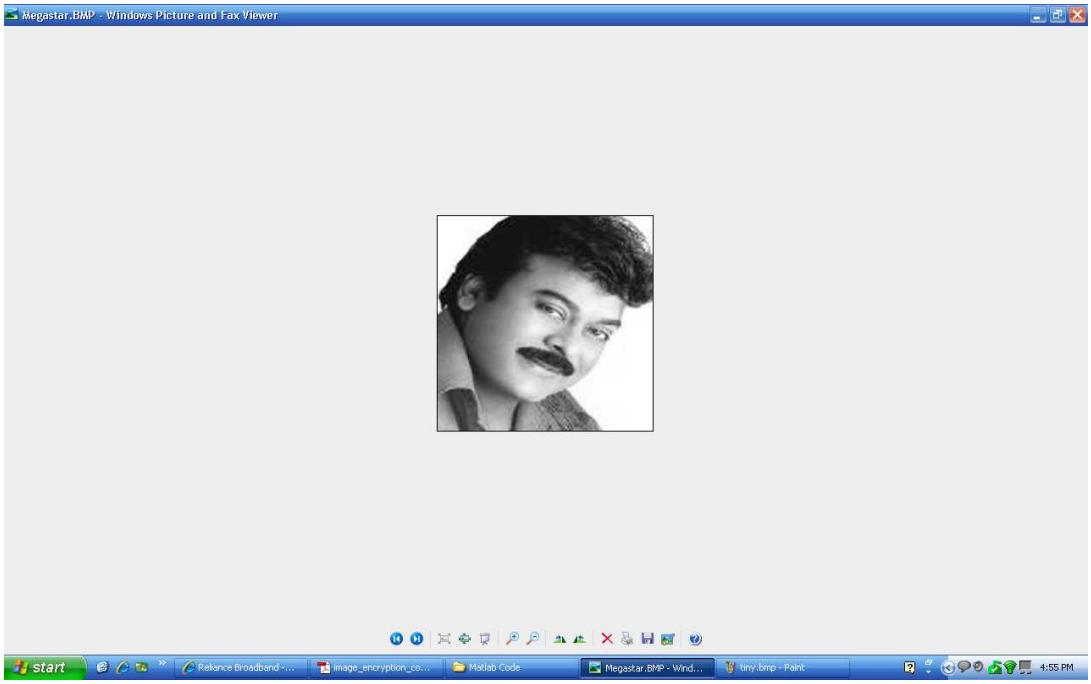
Original image



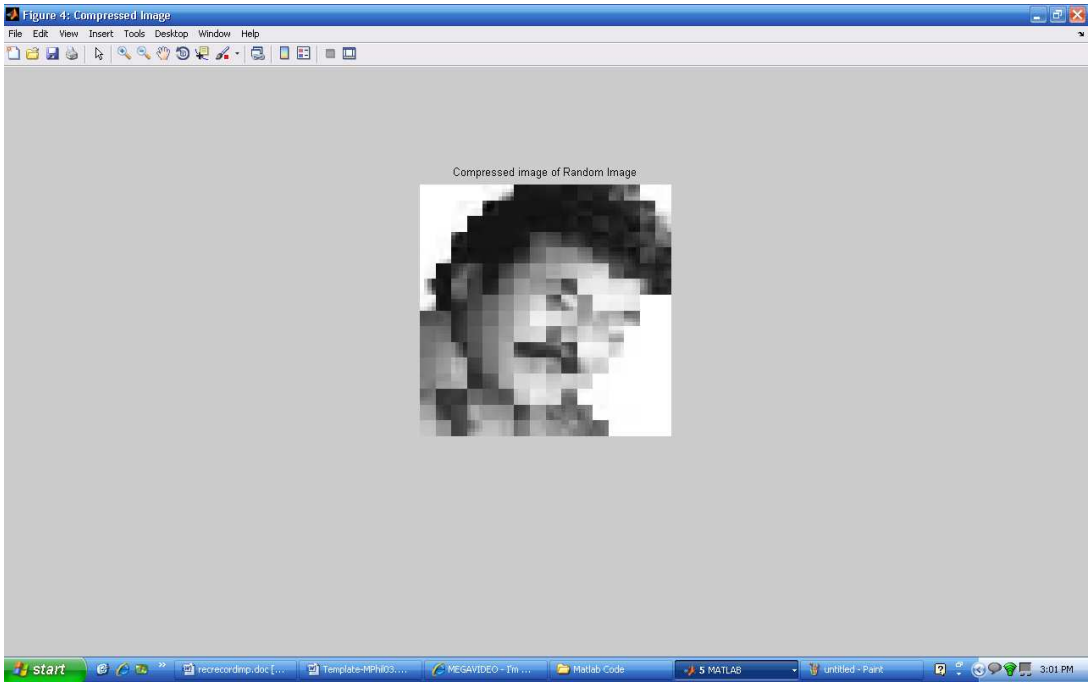
Compressed original image



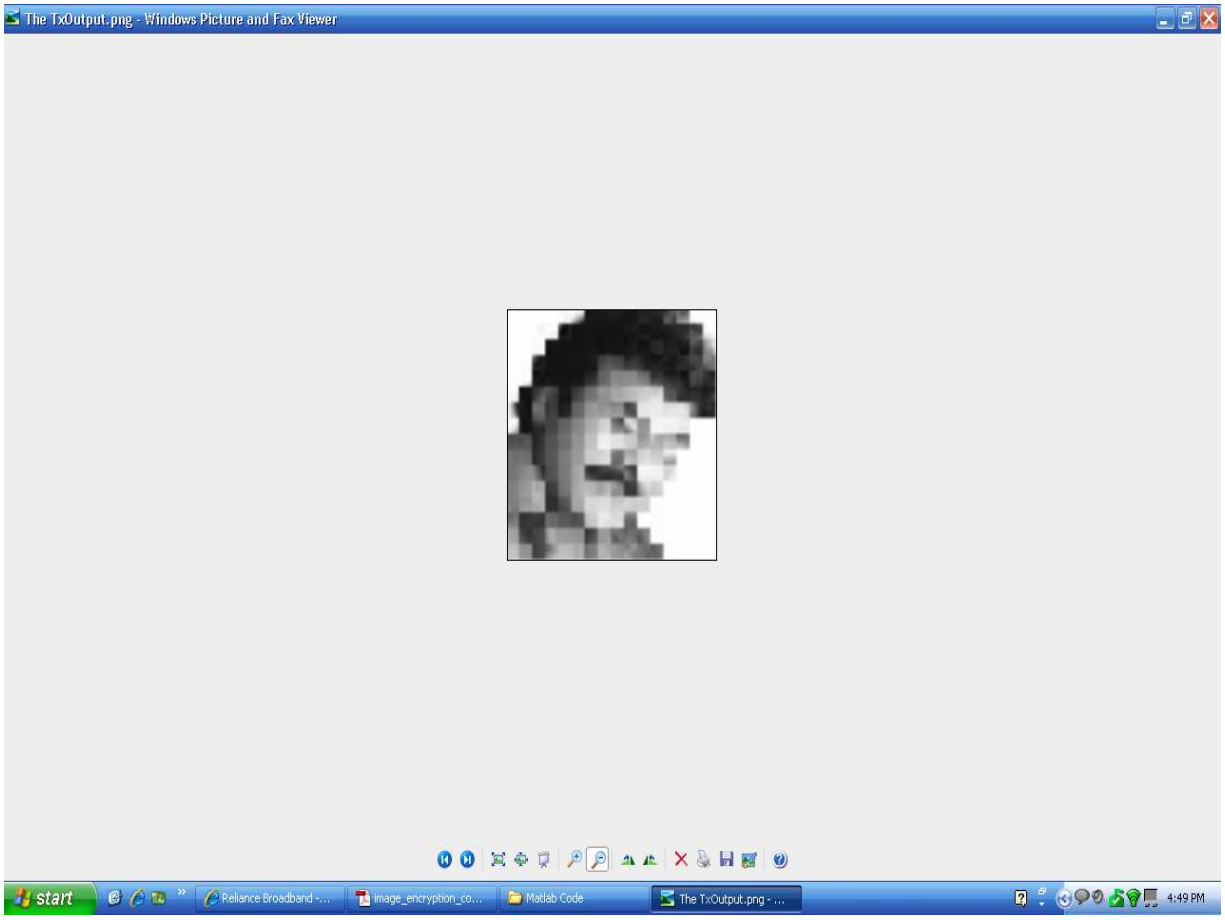
Cover(Random) image



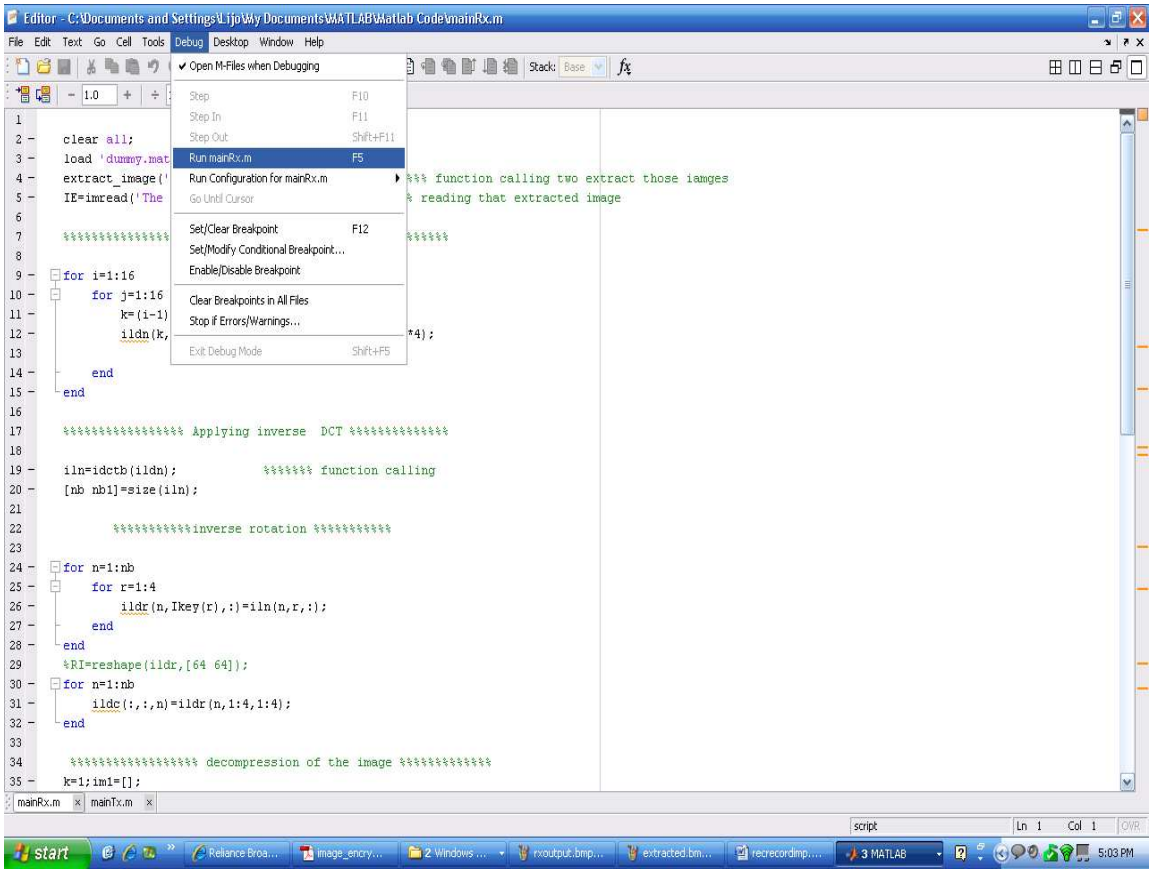
Compressed Random Image



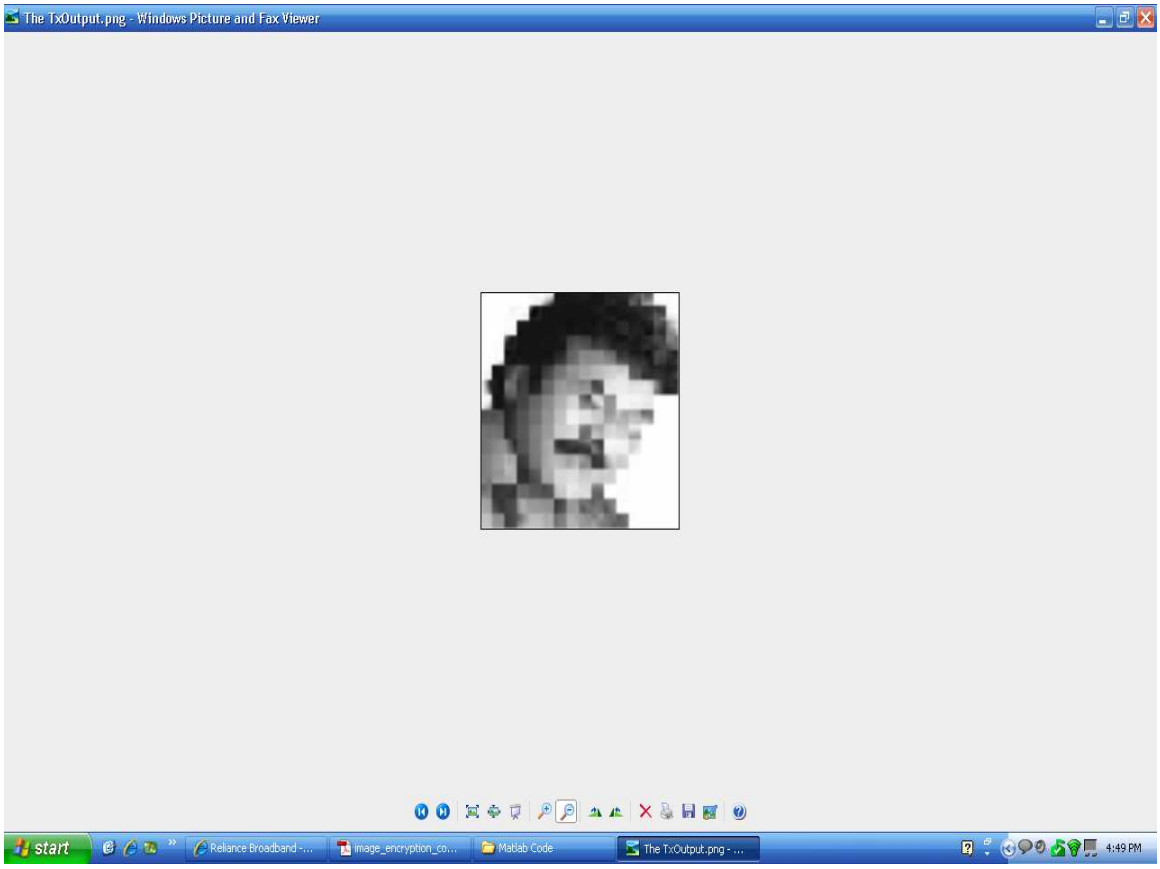
Compressed Encrypted output of mainTx.m



Execution of mainRx.m

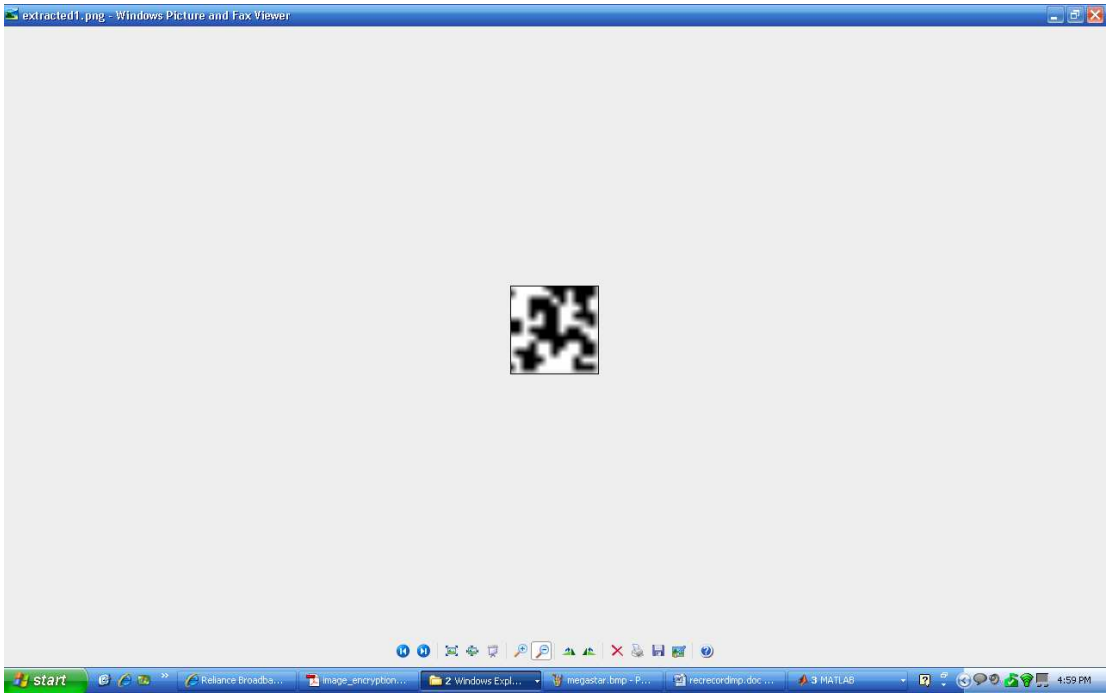


Input of mainRx.m

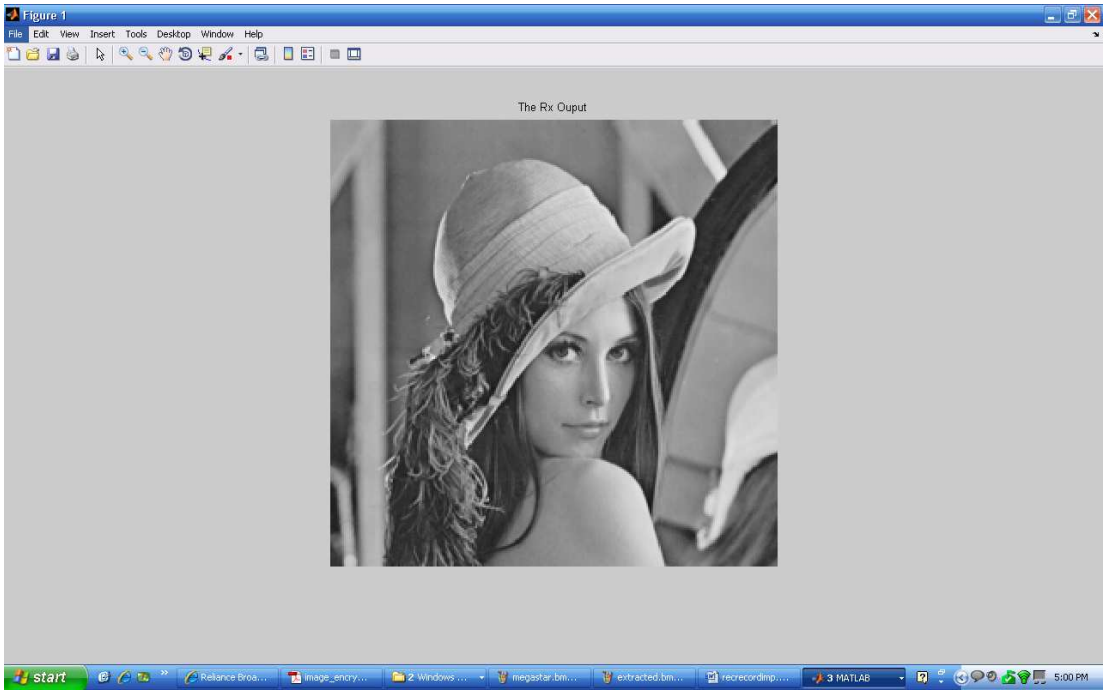


Output of mainRx.m

Extracted image(compressed lena image)



The original image



4.3 Result

In order to validate our approach several simulations are conducted. “Lena” and “Mandrill” images are encrypted and decrypted by the proposed method. 256×256 grayscale bitmap files are used as the original images. An example of the simulations is given above. After execution of extract function and applying IDCT to the separated DCT components with the rotation key, we can get the source images. The quality of the reconstructed images, however, is not as same as the original ones, because compression cut off higher frequency components.

CHAPTER 5

CONCLUSION

CONCLUSION

In this paper we proposed a new image encryption and compression method based on Embedding and Discrete Cosine Transform (DCT). Using DCT images can be compressed. For encryption, DCT blocks of transmitted images are rotated and mixed with a random image to hide them.

In the decryption stage, the covered images can be extracted from the mixtures by applying extraction algorithm. Finally using rotation keys and inverse discrete cosine transform, the original images can be reconstructed.

Therefore we can achieve a fast and secure image transmission. As a result of several computer simulations, the behavior of the proposed approach is confirmed. When compression ratio is between 0.1 and 0.5, the best performance can be achieved. In this paper color images e used as original images, but grey color images can be applied in the same way.

Our future works include a more secure encryption method with an alternative rotation method and a reconstruction key. More complex rotation manner makes it harder for unauthorized people to reconstruct images without keys.

CHAPTER 6

REFERENCES

Reference Text Books

1) Abrams, M., and Podell, H. "Cryptography" Potentials, IEEE Page No 36-38.Issue:1,Volume: 20, Feb-Mar,2001
2) Bruce Schneir: Applied Cryptography, 2nd edition, John Wiley & Sons, 1996
3) Christof Paar and Jan Pelzl – Understanding Cryptography A text book for Students and Practitioners, Springer, 2009. Very accessible introduction to applied cryptography which covers most schemes of practical relevance. The focus is on being a textbook, i.e., it has pedagogical approach, many problems and further reading sections. The main target audience are readers without a background in pure mathematics
4) Garfinkel, S.L; "Public Key Cryptography" , Computer, IEEE, Volume: 29, Issue:6, June 1996.
5) Mel, H.X., and Baker, Doris -- *Cryptography Decrypted*, Addison Wesley 2001, This technical overview of basic cryptographic components (including extensive diagrams and graphics) explains the evolution of cryptography from the simplest concepts to some modern concepts. It details the basics of symmetric key, and asymmetric key cipher.
6) Schneier,Bruce – *Applied Cryptography*, 2 ed, Wiley, 1996, The most accessible single volume available covering modern cryptographic practice, and approachable by the non mathematically oriented. Extensive bibliography which can serve as an entry into the modern literature. It is a great book for beginners but note that it is getting a bit dated—many important schemes such as AES or the eSTREAM candidates are missing entirely, others like elliptic curves are only very briefly treated.
7) W.Stallings; "Cryptography and Network Security" 2nd Edition, PrenticeHall,1999

Research Papers

1) Ahmed, N., T. Natarajan, and K. R. Rao. 1974. On image processing and a discrete cosine transform. IEEE Transactions on Computers C-23(1): 90-93.
2) An intelligent Data Encryption and Compression for high speed and secure data transmission over internet. Dr. V.K. Govindan, B.S. Shajee mohan.
3) Bibhudendra Acharya, Girija Sankar Rath, Sarat Kumar Patra, Saroj Kumar Panigrahy. 2007. Novel Methods of Generating Self-Invertible Matrix for Hill Cipher Algorithm,International Journal of Security, Vol 1, Issue 1, 2007
4) Image Compression Using the Discrete Cosine Transform *Andrew B. Watson* NASA Ames Research Center, Mathematica Journal, 4(1), 1994, p. 81-88
5) Image Watermarking by using DCT byV. Manimarani, S.P.Raja, N. Narayanan Prasanth, J. Francis, S.P. Princess , Published in Proceedings of the Int. Conf. on Information Science and Applications ICISA 2010 ,6 February 2010, Chennai, India
6) Image Encryption Using Advanced Hill Cipher Algorithm Bibhudendra Acharya1, Saroj Kumar Panigrahy2, Sarat Kumar Patra3, and Ganapati Panda3, Published in International Journal of Recent Trends in Engineering, Vol. 1, No. 1, May 2009
7) Invisible Digital Watermarking Through Encryption by Samir Kumar Bandyopadhyay, Tuhin Utsab Paul, Avishek Raychoudhury,International Journal of Computer Applications (0975 – 8887) Volume 4– No.8, August 2010
8) New Image Encryption and Compression Method Based on Independent Component Analysis by Masanori Ito and Noboru Ohnishi, Ayman Alfalou,Ali Mansour

Websites

- en.wikipedia.org/wiki/**Cryptography**
- www.**cryptography**world.com
- en.wikipedia.org/wiki/Discrete_cosine_transform
- citeseerx.ist.psu.edu/viewdoc
- www.dip.ee.uct.ac.za/~nicolls/lectures/eee401f/projects/**dct**.pdf
- users.ece.gatech.edu/~skhire3/files/**DCT**_image_enc.pdf
- en.wikipedia.org/wiki/Digital_**watermarking**
- en.wikipedia.org/wiki/**Steganography**
- www.cs.bham.ac.uk/~mdr/teaching/modules03/.../Steganography.pdf
- en.wikipedia.org/wiki/**Compression**
- www.ams.org/samplings/feature.../fcarc-**image-compression**
- en.wikipedia.org/wiki/MATLAB
- http://www.mathworks.com/products/matlab/

Acknowledgement

The satisfaction that accompanies the successful completion of any project would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crowned my efforts with success

It is my great privilege to thank our Vice Chancellor Dr.Fr Thomas C Mathew, Christ University, Bangalore for giving me the permission to do the thesis work. I also express my gratitude to Pro Vice Chancellor, Dr. Fr Abraham V M, Christ University, Bangalore for guidance and support throughout the course. I express my deep gratitude to the Director, Additional Director, staff, Centre for Research of Christ University for all guidance and support throughout the course of my Mphil degree. It is a pleasure for me to acknowledge the assistance and contributions of a large number of individuals to this effort .I am deeply grateful to each and every one of them for their constructive criticism and support

This thesis would never have become a reality without the assistance of my Research Guide Mrs. C.N Rajeswari, Professor, Dept of Computer Science, Christ University, Bangalore. I would like to thank her for the substantial contribution of guidance, encouragement and valuable suggestions during the course of the thesis work. I would also like to thank Mr. Balachandran, Professor, Coordinator, Dept of Computer Science for his constant motivation throughout my research. I would also like to thank Mr. Joy Paulose, HOD, Dept of Computer Science for his timely guidance throughout my Mphil course.

I would like to thank all the colleagues of my department who rendered their valuable suggestions and criticism during the research period.

Above all I am grateful to the Almighty God for helping me to complete my project in time and as desired. Last but not the least I would like to extend my gratitude to my great motivators, my family members who supported me through the entire course of my research.