P-1604

ADD-ON SECURITY MECHANISM IN STEGANOGRAPHY

A PROJECT REPORT

Submitted by

GANESHRAMANUJAM.A.M.S. **71202104010**

SARAVANAN.M. **71202104062**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641006

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2006

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"ADD-ON SECURITY MECHANISM IN STEGANOGRAPHY"** bonafide work of "Ganeshramanujam.A.M.S.(71202104010)and Saravanan.M.(71202104062)", who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.S.THANGASAMY	**Mr.K.SIVANARULSELVAN**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
	SENIOR LECTURER
Dept. of Computer Science & Engg.,	Dept. of Computer Science & Engg.,
Kumaraguru College of Technology,	Kumaraguru College of Technology,
Chinnavedampatti P.O.,	Chinnavedampatti P.O.,
Coimbatore-641006.	Coimbatore-641006.

Submitted for Viva Voce Examination held on 02/05/2006

Internal Examiner

External Examiner

DECLARATION

We here by declare that the project entitled **"ADD-ON SECURITY MECHANISM IN STEGANOGRAPHY"**, is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any other institution for fulfillment of the requirement of the course study

This report is submitted in partial fulfillment of the requirements for the award of the degree of bachelor of computer science and engineering of Anna University, Chennai

Place: Coimbatore
Date: 26-04-06

(GANESHRAMANUJAM.A.M.S.)

(SARAVANAN.M)

ACKNOWLEDGEMENT

We are extremely grateful to Dr.Padmanaban.K.K, B.Sc. (Engg.), M.Tech., Ph.D., Principal, Kumaraguru College of Technology, Coimbatore for having given us a golden opportunity to embark on this project.

We are deeply obliged to Dr.Thangasamy.S., B.E.(Hons), Ph.D., Dean, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore for his valuable guidance and useful suggestions.

We are grateful to Prof Mrs.Devaki.P B.E.,M.S., Project Coordinator, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore for her encouragement and support at various levels of this project work.

We also express our sincere thanks to our guide Mr.K. SIVANARULSELVAN.M.E.,Senior Lecturer,Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, for his valuable guidance and encouragement at every stage of this project.

Most of all, we thank our parents and friends for their blessings and support, without which we would not be able to do anything.

ABSTRACT

ABSTRACT

Steganography is the art of hiding the fact that communication is taking place by hiding information in other information.Many different carrier file formats can be used.For hiding secret information in images,there exist a a large variety of steganography techniques.Different applications have different requirements of the steganographic technique used.For example some applications may require absolute invisibility of the sectret information,while others require a larger secret message to be hidden.Now a days the threat has been developed in the computerized systems too.Those people who are threatening the network is called intruders.So the security should be provided for all system users using as their communication line.

The main purpose of steganography is to hide the occurrence of communication.While most methods in use today are invisible to the observer's senses,mathematical analysis may reveal statistical discrepancies in the stego medium.These discrepancies expose the fact that hidden communication is happening.For that we can choose an image in which given message can be hidden safely.So it is not easy for intruders to retrieve the data.This project intends to encrypt a message to be transmitted and embedding the encrypted message in an image.We are performing two kinds of security system to make our data secured.The method followed are RSA and Steganography.In RSA we go for encryption where it provides security to the data file that is to be sent to the receiver.The other security system Steganography is followed to send the data through some communication line like internet.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURES

INTRODUTION

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION:

The objective is to hide messages in images.This process is termes as Steganography.Steganography can be defined as the process of hiding the message with carrier file whose primart intend is to conceal the very existence of a message in a carrier of any form.

For example,a text file could be hidden "inside" an image or a sound file.By looking at the image,or listening to the sound,you would not know that there is extra information present.

Here our project is aimed at embedding message which is encrypted by RSA algorithm and the message will be embedded using STILL images as cover medium by Pixel Grabber using Java.

1.2 CURRENT STATUS OF THE PROBLEM TAKEN UP:

Over the past few years, numerous Steganography techniques that embed hidden messages in images and multimedia objects have been proposed. In specific, multimedia objects are excellent covers for Steganography. This is largely due to the fact that multimedia objects often have a highly redundant representation. This redundancy usually permits the addition of significantly large amounts of stego-data by means of simple and subtle modifications that preserve the perceptual

content of the underline cover object. Hence, they have been found to be perfect candidates for use as cover messages

In fact, there are many freely available public tools that embed a message into images, audio and video in different formats. To name a few tools that have made their mark in popularity and usage are STEG, F5, OUTGUESS, EZSTEGO, GIF_SHUFFLE, STEGNOS and DIGIMARK (Appendix 1).Appendix 2 gives details of few commercially available tools. These tools work either in the spatial or frequency domain of the cover. The methods followed are usually centered on techniques such as LSB embedding, algorithms and transformations or masking and filtering techniques.

Steganalysis against the above said tools have also emerged. But most of these tools work with the knowledge of the embedded algorithm. Attempts have been successful in recovering the embedded messages too. But it is indeed a difficult task to detect with ignorance of the tool that had been used in the Steganography process. In such cases analysis of the statistics of the image or in general the cover helps. Both first order as well as second order statistics has been much used. Hence the current trend in this field poses a lot of questions such as the following

- Can the current and future Steganography algorithms be categorized into distinct classes of mathematical techniques?
- What is a good mathematical definition of Steganalysis?
- What prior knowledge can we assume the steganalyst possesses?
- What mathematical properties a class of Steganography algorithms must satisfy for which good Steganalysis techniques can be developed?

Hence tests are underway to answer all the above said questions.

The pioneers of this who seem to have much contributed are Niels Provos and Peter Honeyman who have attached the JSTEG, F5 and OUTGUESS tools. Andreason Westfield and Andreas Pfltzman have proposed attacks against the F5.Ross.J.Anderson, Fabien Petitcolas and Mark Kuhn have much contributed to survey of information hiding. His contributions include producing a mathematical approach to Steganalysis, a distributed detection framework for detection and a mathematical model to the question of steganographic capacity. Jessica has made her mark through extensive contribution. Her steganalystic approaches towards tools F5, OUTGUESS etc and an approach named RS Steganalysis are worth noting.

1.3 TERMINOLOGIES, STEGANOGRAPHIC TOOLS

1.3.1 Terminologies:

A COVER or CARRIER is the file used to carry the covert message. The possible cover carriers are innocent looking carriers (images, audio, video, text, or some other digitally representative code) which will hold the hidden information.

A MESSAGE is the information hidden and may be plaintext, cipher text, images, or anything that can be embedded into a bit stream.

A STEGO-CARRIER is the cover carrier and the embedded message together. Hiding information may require a stego key which is additional secret information, such as a password, required for embedding the information. For example, when a secret message is

DETAILS OF LITERATURE SURVEY

CHAPTER 2
DETAILS OF LITERATURE SURVEY

2.1 SURVEY OF A PAPER:

"Exploring Steganography: Seeing the unseen"-(Neil F. Johnson & Sushil Jajodia,1998).

This textual matter gives an insight towards various steganographic methods available from the past to the present. A brief history on the evolution of these methods has been surveyed.

Historical forms of steganography cited were forms of text being written on wax-covered tablets, Invisible inks, Microdots and so on.Modern forms of steganography relate to information being sent covertly using computers.Some of the examples he cites are hiding messages in covers such as HTML documents,images.multimedia files audio and video,TCP packets etc.The methods of hiding,the advantages and disadvantages have also been discussed.

A technique to hide information is to include extra spaces in documents.These aspects may contain hidden characters.This is a simple technique for hiding information is easy to detect and defeat. By opening such a document in a word processor the unusual spacing becomes readily apparent.Reformatting the document can remove the hidden message.

The most prevalent cover objects in use today are digital images because of their potencial payload(hidden information).The use of audio files can provide a good carrier for hidden messages.By their very nature sound files tend to be large in size and thus do not attract attention.

The standard protocol suite used on the internet is the Transmission Control Protocol /Internet Protocol(TCP/IP).The headers of these packets allow the use of flags and certain reserved fields.Information can be inserted into these fields using appropriate technique.The advantage of this technique is that headers are rarely read by humans and thus make an ideal place to hide data.The disadvantage of this method is that firewalls can be configured to filter out packets that contain inappropriate data in the reserved fields,thus defeating the steganographic transmission.

SYSTEM ENVIRONMENT

CHAPTER 3

3. SYSTEM ENVIRONMENT

3.1 HARDWARE SPECIFICATION

PROCESSOR	:	PENTIUM III
CLOCK SPEED	:	800 MHZ
MEMORYSIZE	:	128 MB SD RAM
HARD DISC	:	20 GB
FLOPPY DISK DRIVE	:	1.44 MB
DISPLAY	:	15" SVGA MONITOR
KEYBOARD	:	SAMSUNG 104 KEYS
PRINTER	:	HP 1000 LASER

3.2 SOFTWARE SPECIFICATION

OPERATING SYSTEM	:	WINDOWS '9x, 2000, XP, 2003
LANGUAGE	:	JAVA(jdk 1.3)

DETAILS OF METHODOLOGY EMPLOYED

CHAPTER 4

DETAILS OF METHODOLOGY EMPLOYED

The process of encrypting data and embedding within the cover image were done by using the RSA algorithm.Two basic kinds of steganography were attempted.The former encrypted text message and the latter concealed it within the cover image.

4.1 RSA Algorithm

RSA algorithm is mainly a public key encryption technique used widely in network communication like in Virtual Private Networks (VPNs). Public key is advertised to the world and private key is kept secret. It is not possible to generate private key using the public key. So, someone who knows the public key cannot decrypt a message after it has been encrypted using the public key.

RSA algorithm is a block cipher technique in which plain text and cipher text are integers between '0' and 'n-1' from some 'n'. In RSA algorithm encryption and decryption are of following form, for some plain text M and cipher text C:

$C = M^e \bmod n$

$M = C^d \bmod n$

Both sender and receiver must know the value of 'n'. The sender knows the value of 'e' and only receiver knows the value of 'd'. Thus, this is a public-key encryption algorithm with a public key of KU={e, n} and private key of KR={d, n}. For the algorithm to be satisfactory for public-key encryption, the following requirement must be met

1. It is possible to find values of e, d, n such that M^ed = M mod n for all M<n.
2. It is relatively easy to calculate M^e and C^d for all values of M<n.
3. It is infeasible to determine d given e and n.

4.2 The RSA Algorithm and the RSA Patent

The RSA Algorithm was named after Ronald Rivest, Adi Shamir and Leonard Adelman, who first published the algorithm in April, 1977.Since that time, the algorithm has been employed in the most widely-used Internet electronic communications encryption program, Pretty Good Privacy (PGP). It is also employed in both the Netscape Navigator and Microsoft Explorer web browsing programs in their implementations of the Secure Sockets Layer (SSL), and by Mastercard and VISA in the Secure Electronic Transactions (SET) protocol for credit card transactions.

The RSA Algorithm is claimed in the RSA Patent, which was issued to Drs. Rivest, Shamir and Adelman, who exclusively licensed the patent nine days later to RSA Data Security, Inc., a company which was originally controlled by the inventors but is now a wholly-owned subsidiary of a Boston based company called Security Dynamics Technology, Inc. RSA Data has to date filed three lawsuits alleging

P-1604

infringement of the RSA Patent. Two were settled prior to trial, and the third is still pending. Other litigation threats have been made regarding alleged infringements of the patent, including threats against non-commercial implementations for use by the Internet community. The patent expires on September 20, 2000, but that will be enough time for the patent to have a profound impact on the development of electronic commerce.

The existence of the patent, and RSA Data's aggressive litigation posture, have chilled the interest in both commercial and non-commercial implementations of public key encryption and digital signature technologies. Many have taken for granted the bald assertion that the "RSA Algorithm is patented," without examining the patent itself, or more particularly, the claims of the patent. As we set forth in this article, however, a careful review of the patent reveals that the patent is not necessarily as broad as publicly asserted. More particularly, the decryption operation, standing alone, is not independently claimed at all in the patent. These weaknesses in the patent may be particularly relevant for digital signature operations because they may allow a developer to implement the protocol for verifying an RSA-generated digital signature without infringing the patent. In addition, if one separates the generation of the key pairs from the encryption operation, the claims of the patent do not cover the encryption (or signing) function by itself.

4.3 Basic Uses of Public Key Encryption

The RSA Algorithm is only one implementation of the more general concept of public key cryptography, which permits two parties who have never met and who can only communicate on an insecure channel to nonetheless send secure and verifiable messages to each other. The Internet as currently structured is an insecure communications

channel with an obvious use for such technologies. Indeed, the greatest expected growth for public key techniques is in Internet-related communications.

With public key techniques, each user has two different keys, one made available to the public and the other kept secret. One of the keys is used to encrypt a message, and the other is used to decrypt the message. If Alice wants to send a secret message to Bob, for example, she looks up Bob's public key and uses it to encrypt the message. Because Bob's public key cannot undo the encryption process, no one who intercepts the message can read it. Only Bob, who possesses the secret key corresponding to his public key, can read the message. Alice never has to meet Bob out of the hearing of others to exchange keys or passwords; this is a substantial improvement over older encryption methods in which an exchange of private keys was necessary.

This system can also be used as a means for Bob to be sure a message comes from Alice. If Alice wants to sign a message, she can encrypt it with her private key. When Bob receives an encrypted message which purports to be from Alice, he can obtain Alice's public key and

decrypt the message. If a readable message emerges, Bob can have confidence that the message came from Alice, because Alice's public key would only properly unlock a message which was locked with her private key (known only to Alice).

Of course, digitally signing the message does not make the content of the message private, because anyone with Alice's public key can read a message she encrypted with her private key. Alice can send a private, signed message to Bob, however, by first encrypting the message with Bob's public key (so only Bob can read it with his private key) and then encrypting the message a second time with her private key, forming her signature. Anyone who receives the message can use Alice's public key to undo the second encryption, but only Bob (or someone with Bob's private key) can undo the first encryption step and actually read the message. All of these complex-sounding manipulations can be made quite manageable with well-written software.

4.4 The RSA Implementation of Public Key Encryption

4.4.1 The Arithmetic in the RSA System

Typical encryption techniques use mathematical operations to transform a message (represented as a number or a series of numbers) into a ciphertext. Mathematical operations called one way functions are particularly suited to this task. A one way function is one which is comparatively easy to do in one direction but much harder to do in reverse. As a trivial example, it is comparatively easy to square a two digit number; with a little concentration, many people can probably multiply 24 by 24 without using a pencil and paper. One the other hand,

calculating the square root of the number 576 is much harder, even with a pencil and paper.

The RSA system uses one way functions of a more complex nature. Specifically, the system uses modular arithmetic to transform a message (or pieces of the message, one piece at a time) into unreadable ciphertext. Modular arithmetic is often called "clock" arithmetic, because addition, subtraction, and the like, work like telling time. In a 12-hour system, four hours after 10:00 is not 14:00 (10 + 4 is not equal to 14); it is 2:00. This is because we subtract out 12 (or any multiples of 12) after doing the addition. In modular arithmetic notation, the operation might look like this:

$2 = (10+4) \bmod 12$

$2 = 14 \bmod 12$

One can do multiplication in modular arithmetic much the same way addition is done in the above example:

$2 = (7*2) \bmod 12$

$2 = 14 \bmod 12$

This process is sometimes called modular reduction. By subtracting out the modulus (and all multiples of the modulus) a number is "reduced" to a much smaller number. When the number 14 is "reduced" to the number 2 in the above example, one can say that "14 is reduced modulo 12."

The RSA system uses multiplication in modular arithmetic. Instead of multiplying one number by a different number (as (7) is multiplied by (2) in the above example), The RSA system multiplies one number (called

the base) by itself a number of times. The number of times a base is multiplied by itself is called the exponent:

$16 = 2*2*2*2$

$16 = 2^4$

In this example, the number (2) is the base, and is multiplied by itself four times, making the exponent the number (4).

In the RSA encryption formula, the message (represented by a number M) is multiplied by itself (e) times (called "raising (M) to the power (e)"), and the product is then divided by a modulus (n), leaving the remainder as a ciphertext (C):

$C = M^e \bmod n$

This is a hard operation to undo -- when (n) is very large (200 digits or so) -- even the fastest computers using the fastest known methods could not feasibly recover the message (M) simply from knowing the ciphertext (C) and the key used to create the message ((e) and (n)).

In the decryption operation, a different exponent, (d) is used to convert the ciphertext back into the plain text:

$C = M^d \bmod n$

The modulus (n) is a composite number, constructed by multiplying two prime numbers, (p) and (q), together:

$n = p * q$

The encryption and decryption exponents, (d) and (e), are related to each

other and the modulus (n) in the following way:

$d = e^{-1} \bmod ((p-1)(q-1))$

To calculate the decryption key, one must know the numbers (p) and (q) (called the factors) used to calculate the modulus (n). When (n) is a sufficiently large number, it is infeasible, using known algorithms and the fastest computing techniques, to calculate the prime number factors of (n).

The RSA Algorithm may be divided, then, into three steps:

(1) key generation: in which the factors of the modulus (n) (the prime numbers (p) and (q)) are chosen and multiplied together to form (n), an encryption exponent (e) is chosen, and the decryption exponent (d) is calculated using (e), (p), and (q).

(2) encryption: in which the message (M) is raised to the power (e), and then reduced modulo (n).

(3) decryption: in which the ciphertext (C) is raised to the power (d), and then reduced modulo (n).

4.5 Using the RSA Algorithm for Privacy

When the RSA Algorithm is used in a public key system, the modulus (n) and one of the exponents (arbitrarily, we can assume (e)) are published.

The other exponent (d) is kept secret, as are (p) and (q), the factors of (n). Each user holds his or her own keys, and knows the public key of the

other user or users. Alice, for example, knows her own public key (e_{alice} and n_{alice}), her own private key (d_{alice}), and Bob's public key (e_{bob} and n_{bob}). Bob knows the converse: his public key (e_{bob} and n_{bob}), his private key (d_{bob}) and Alice's public key (e_{alice} and n_{alice}).For Alice to send Bob a private message only Bob can read, she performs the following operation on the message (M):

$C = M^{ebob} \bmod n_{bob}$

Bob, who is the only one to possess his private key (d_{bob}), performs the following to recover the message (M):

$M = C^{dbob} \bmod n_{bob}$

To sign the message, Alice encrypts with her own private key:

$C = M^{dalice} \bmod n_{alice}$

Because only Alice possesses d_{alice}, only she can create this ciphertext C. Anyone in possession of her public key (e_{alice} and n_{alice}) can verify the signature, however:

$M = C^{ealice} \bmod n_{alice}$

It bears note that (p) and (q), the factors of (n), are not needed for encryption or decryption; they are only used in the key generation step (creating the modulus (n) and the second exponent). In addition, while it is important for key generation purposes that the modulus (n) be the product of two prime numbers, the exponentiation and modular arithmetic operation would work just as well with prime numbers (which are by definition evenly divisible only by themselves and the number 1).

4.6 The RSA Patent

Anyone who "makes, uses, offers to sell or sells" a patented invention without the permission of the patent owner can be liable for patent infringement. The boundaries of the patent are defined in the claims portion of the patent.Accordingly, in order to determine whether a particular product, method or process infringes a patent, one must start with the text of the claims themselves.

There are 40 claims in the RSA Patent, but only ten of them are independent claims.Independent claims are claims which do not incorporate other claims by reference. To infringe a dependent claim, one must first infringe the independent claim (or claims) incorporated by reference in the dependent claim. Conversely, if one does not infringe the independent claim(s) incorporated by the dependent claim, by definition one does not infringe the dependent claim. Accordingly, a review of the independent claims in the RSA Patent is sufficient for purposes of this discussion. As we will also see, a detailed review of the broadest independent claim in the RSA Patent (Claim 23) will lead without much further ado to a logical conclusion about the other nine independent claims, and in turn to a conclusion about all the claims in the RSA Patent: that none of these claims are infringed by performing a typical digital signature verification.

The claim with the least number of elements (and thus the broadest claim in the patent) is Claim 23, which provides:

23. A method for establishing cryptographic communications comprising the step of:

encoding a digital message word signal M to a ciphertext word signal C, where M corresponds to a number representative of a message and

$0 <= M <= n-1$

where n is a composite number of the form

$n=p*q$

where p and q are prime numbers, and

where C is a number representative of an encoded form of message word M,

wherein said encoding step comprises the step of:

transforming said message word signal M to said ciphertext word signal C whereby

C [is congruent to] $M^e \pmod n$

where e is a number relatively prime to (p-1)*(q-1).

Accordingly, the elements of this claim require an accused infringer to perform the following steps:

- Establish cryptographic communications;
- Ensure that the message (M) is greater than or equal to zero and less than or equal to (n-1);

- Define the modulus (n) by selecting prime numbers (p) and (q) and multiplying them together;
- Define the encryption exponent (e) such that it is relatively prime to (p-1)*(q-1); and
- Encode message (M) into ciphertext (C) by raising (M) to the power of (e) and
- then reducing modulo (n).

Does Claim 23 Cover Signature Verification?

As noted above, to verify an RSA-generated digital signature, the recipient takes the ciphertext transmitted to him and decrypts the ciphertext with the sender's public key. If the message decrypts properly, the signature is genuine. Importantly, two of the three fundamental steps in the RSA system are not performed in the signature verification step: key generation, where the parameters of the modulus (n) and the exponents (e) and (d) are set; and encryption, where the message (M) is raised to the power of (e) and then reduced by the modular operation. Only the third, decryption step is performed. The keys are generated by the sender (signer) and the encryption step has already been completed in the signing step.

Accordingly, a person who merely verifies an RSA signature arguably does none of the steps contained in Claim 23, and certainly does not do them all. Most significantly, the decryption operation plainly does not constitute "encoding a digital message word signal" into "ciphertext." The verification operation transforms "ciphertext" into a "message word signal," not the reverse. The patent makes clear; moreover, what constitutes a "message" and what constitutes "ciphertext," and how

"decoding" and "encoding" are different. These terms are not interchangeable.

The RSA signature verification steps of transforming ciphertext C into a message M using the decryption exponent (d) are separately claimed in dependent claim 24:

24. The method according to claim 23 comprising the further step of:

Decoding said ciphertext word signal C to said message word signal M,

wherein said decoding step comprises the step of:

transforming said ciphertext word signal C, whereby:

M [is congruent to] C^d (mod n)

where d is a multiplicative inverse of e (mod (lcm ((p-1), (q-1)))).

Because Claim 24 incorporates all of the elements of Claim 23 by reference, one cannot infringe Claim 24 simply by performing the decryption steps alone.

Do the Other RSA Patent Independent Claims Cover Signature Verification?

The analysis of Claim 23 above should apply with equal force to the other independent claims. Claims 1, 13, and 18 require key generation, encoding, and de-coding. Claims 3, 8, 25, 29 do not contain the decoding step (subsequent dependent claims add that step), but have at least the elements of Claim 23, plus others. Claims 33 and 37 claim a special case of the use of the RSA method, and thus are also more limited

than Claim 23. Thus, under this analysis none of the independent claims of the RSA Patent (and therefore none of the dependent claims) are infringed by performing typical digital signature verification.

Does Claim 23 Cover Generation of a Digital Signature?

It appears that the process of generating an RSA signature also may be done without infringing Claim 23. To create an RSA digital signature, the sender encrypts the message M with her private key, and transmits it to the receiver. The encoding step is clearly claimed in Claim 23 (and other independent claims), and thus the argument stated above regarding verification (decryption) is not available. However, Claim 23 also requires key generation, and that step need not be performed by software creating a digital signature if the keys are already supplied. (All of the other independent claims require the generation of the keys meeting the RSA Algorithm parameters.)

The step of generating the numbers comprising RSA keys (p, q, n, e and d) can easily be separated from the encryption step. The same keys can be used over and over again for multiple signatures; indeed, they can be used for as long as one has confidence that the keys have not been compromised. Moreover, many RSA-licensed products generate keys which can be separated from the software which generated them. Thus, as a practical matter, it should be possible to use keys generated by licensed RSA software in order to create digital signatures with other, unlicensed software.

The owner of the RSA Patent would have difficulty arguing successfully that the encryption operation itself is covered by the patent,

separate from the generation of the keys. This is because the concept of using exponents in modular arithmetic for encryption was invented and disclosed before the RSA system was invented. In 1975, two years before the RSA method was invented, Martin Hellman and Stephen Pohlig at Stanford University invented the Pohlig-Hellman encryption system, which is identical to the RSA method, except that the modulus is a prime number, as opposed to the product of two primes:

Comparison of RSA and Pohlig-Hellman		
	RSA System	Pohlig-Hellman
Encryption Operation	$C = M^e \bmod n$	$C = M^e \bmod p$
Decryption Operation	$M = C^e \bmod n$	$M = C^e \bmod p$
modulus	p*q(prime numbers)	p (prime number)
Encryption exponent (e)	e relatively prime to (p-1)*(q-1)	e relatively prime to (p-1)
Decryption exponent (d)	$d=e^{-1} \bmod ((p-1)*(q-1))$	$d = e^{-1} \bmod (p-1)$

TABLE 4.6 Comparison of RSA and Pohlig-Hellman

The exponentiation and modular reduction steps in RSA and Pohlig-Hellman will work exactly the same regardless of whether the modulus is a prime number or the product of two primes. Once the modulus and the exponents are defined, the mathematical operations for encryption and decryption are identical in both the Pohlig-Hellman and RSA systems. A software module written to perform Pohlig-Hellman encryption or decryption would work just as well using a modulus and exponents generated with an RSA system. Accordingly, even if the inventors had

attempted to claim the encryption step alone, without regard to key generation, the prior Pohlig-Hellman invention would have prevented such a claim from being valid.

CONCLUSION AND FUTURE ENHANCEMENT

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

To detect the presence of hidden messages an analysis based on statistics has been employed. The basic statistical measures such as the mean and standard deviation, and correlation were analyzed to distinguish stego images from non-stego ones. Also an instance of visual attack was demonstrated making use of the ability of humans to clearly between noise and visual patterns.

In such an effort, the input to the steganalyzer the stego-medium was required. Hence instances of Steganography such as textual embedding and image hiding using LSB hiding were demonstrated. In the latter process nearly 40 to 50 percentage of the cover image had been used in the embedding process. Here 8 bit grayscale, uncompressed still images were taken the case study.

Results showed that the statistical tests could distinguish the stego from non-stego images for the steganographic case taken

There is work yet to be done in this field. Steganalysis has to be established for cover images with lesser steganographic capacity, not to mention a larger bit format and color images.

There is a lot of scope for future work in this field since Steganography goes well beyond simply embedding images in an image. It also pertains

to other media, including multimedia, and communication channel. For example, the plans of a top-secret project, device, aircraft, covert operations, or trade secrets can be embedded, using same steganographic method, even on an ordinary audio cassette tape. The alterations of the expected contents of the tape cannot be detected by human ear and probably not easily by digital means. Part of secrecy is of course in selecting the proper mechanisms.

So analyzing the various kinds of media for potentially vulnerable steganographic content requires a lot of research and exploration. Commercially and freely available software tools, are easy to use tools in performing steganalysis, but are known to generate many false positives. Hence much work is yet to be done.

APPENDICES

CHAPTER 6

APPENDIX-A

SAMPLE CODING

```
import java.math.*;
import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class RSATestFastEnc extends JFrame
{
        private JLabel lfilepath;
        private JTextField filepath;
        private JTextArea filecontent;
        private JButton br;
        private ObjectInputStream input;
        private JButton encrypt;
        private JButton encode;
        private JButton decode;
        private JButton decrypt;
        public String fnten;
        JFileChooser fileChooser;
        JFileChooser filech;//=new JFileChooser();
        JFileChooser filechen;
        JDialog jd;
```



```
        String location;
        private Random rnd;
        private BigInteger m, m1, m2, m3, c, s, s1;
        private String msg="";
        private String tmpmsg="";
        private RSAPrivateKeyFast alice;
        private RSAPrivateKeyFast bob;
        private RSAUtil rsautil;

public RSATestFastEnc()
{
        filepath=new JTextField(15);
        Panel tp=new Panel();
        Panel bp=new Panel();
        lfilepath=new JLabel("File for Encryption");
        br=new JButton("Browse");
        encrypt=new JButton("Encrypt");
        encode=new JButton("Encode");
        decode=new JButton("Decode");
        decrypt=new JButton("Decrypt");
        tp.add(lfilepath);
        tp.add(filepath);
        tp.add(br);
        filecontent=new JTextArea();
        Container c=getContentPane();
        ScrollPane p=new ScrollPane();
        p.add(filecontent);
```



```
c.add(p);
c.add(tp,BorderLayout.NORTH);
bp.add(encode);
bp.add(encrypt);
bp.add(decrypt);
bp.add(decode);
c.add(bp,BorderLayout.SOUTH);
encrypt.setEnabled(false);
encode.setEnabled(false);
decrypt.setEnabled(false);
decode.setEnabled(false);
setSize(400,400);
show();
br.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e)
        {
                openFile();
        }
});


encrypt.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e)
    {
            Encrypt();
     }}
    );
```



```
encode.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e)
{
        encodemsg();
 }}
 );

 decrypt.addActionListener(new ActionListener(){
 public void actionPerformed(ActionEvent e)
 {
        decrypt();
  }}
  );
                decode.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e)
  {
          decodemsg();
   }}
   );
   setSize(400,500);
   show();
   }
   public void ReadFile(File f) {
     filepath.setText("");
filecontent.setText("");
if(f.isFile()) {
     try {
```

```
            RandomAccessFile r=new RandomAccessFile(f,"r");
            StringBuffer buf=new StringBuffer();
            String text;
            filecontent.append("\n\n");
            while((text=r.readLine())!=null)
            buf.append(text +"\n");
            filecontent.append(buf.toString());
            }
        catch(IOException e2) {
        JOptionPane.showMessageDialog(this,"FILE      ERROR","FILE
ERROR",
        JOptionPane.ERROR_MESSAGE);
      }
     }
    else {
         JOptionPane.showMessageDialog(this,
         "File  Does Not Exists ","FILE
 ERROR",JOptionPane.ERROR_MESSAGE);
                }
  }
 public void openFile()
  {
          filech=new JFileChooser();
          filech.setFileSelectionMode(JFileChooser.FILES_ONLY);
          int result=filech.showOpenDialog(this);
          if(result==JFileChooser.CANCEL_OPTION)
          return;
          File filename=filech.getSelectedFile();
```



```
        if(filename==null )
        JOptionPane.showMessageDialog(this,"Invalid File
Name","Invalid",
        JOptionPane.ERROR_MESSAGE);
        else {
                String s=filename.getPath();
                fnten=s;
                    filepath.setText(s);
                    br.setEnabled(true);
                    encrypt.setEnabled(false);
                    encode.setEnabled(true);
                    ReadFile(filename);
              }
  }
  public void Encrypt()
  {
   System.out.println("ALICE ENCRYPTS m FOR BOB; BOB
  DECRYPTS IT:");
      c = bob.RSAEncrypt(m);// Using Bob's public key
      filecontent.setText("");
      filecontent.append(c.toString());
      System.out.println("Message encrypted with Bob's public key:\n" +
        c + "\n");
      String outenmsg=c.toString();
        WriteFile(outenmsg,"D:\\Cryptography\\rsa\\encrypt.txt");
        decrypt.setEnabled(true);
   }
   public void encodemsg()
```



```
  System.out.println("The encoded msg :"+enmsg);
  m=new BigInteger(enmsg);
  filecontent.setText("");
  filecontent.append(enmsg);
  encrypt.setEnabled(true);
  String encodefile="D:\\Cryptography\\rsa\\encoded.txt";
  WriteFile(enmsg,encodefile);
  }
public void decrypt()
{
    m1 = bob.RSADecrypt(c);
    System.out.println("Original message back, decrypted:\n" + m1 +
"\n");
    tmpmsg=m1.toString();
    String dec=m1.toString();
    WriteFile(dec,"D:\\Cryptography\\rsa\\decrypted.txt");
    filecontent.setText("");
    filecontent.append(dec);
    decode.setEnabled(true);
  }
  public void decodemsg()
  {
          String orgmsg=rsautil.decode(tmpmsg);
      System.out.println("the original msg back: "+orgmsg);
      WriteFile(orgmsg,"D:\\Cryptography\\rsa\\decoded.txt");
          filecontent.setText("");
      filecontent.append(orgmsg);
   }
```



```
public void WriteFile(String inmsg,String fname)
{
        try
        {
                BufferedWriter w=new BufferedWriter(new
FileWriter(fname));
                w.write(inmsg);
                w.close();
         }
         catch(IOException e2)
         {
         System.out.println(e2.getMessage());
         }
   }
   public static void elapsedTime(long startTime) {
     long stopTime = System.currentTimeMillis();
     double elapsedTime = ((double)(stopTime - startTime))/1000.0;
     System.out.println("Elapsed time: " + elapsedTime + " seconds");
   }
   public static void main(String[] args) {
                RSATestFastEnc objfrm= new RSATestFastEnc();
                objfrm.setTitle("ALICE ENCRYPTS m FOR BOB; BOB
  DECRYPTS IT:");  }}
```

```
{
    rnd=new Random();
    alice=new RSAPrivateKeyFast(1024, rnd, "Alice");
     bob=new RSAPrivateKeyFast(1024, rnd, "Bob  ");
     File inputf=new File(fnten);
    if(inputf.isFile())
    {
        try
        {
                RandomAccessFile f=new RandomAccessFile(inputf,"r");
                StringBuffer buf=new StringBuffer();
                String text;
                while((text=f.readLine())!=null)
                buf.append(text);
                msg=buf.toString();
                System.out.println("The input file message :"+"\n"+msg);
          }
          catch(IOException e2)
          {
          System.out.println(e2.getMessage());
          }
      }
      else
      {
      System.out.println("File not Found");
      }
      rsautil=new RSAUtil();
      String enmsg=rsautil.encode(msg);
```


APPENDIX-B

SCREEN SHOTS

MESSAGE TO BE TRANSMITTED:

Fig.6.1:snap shot1

ENCRYPTION USING RSA ALGORITHM:

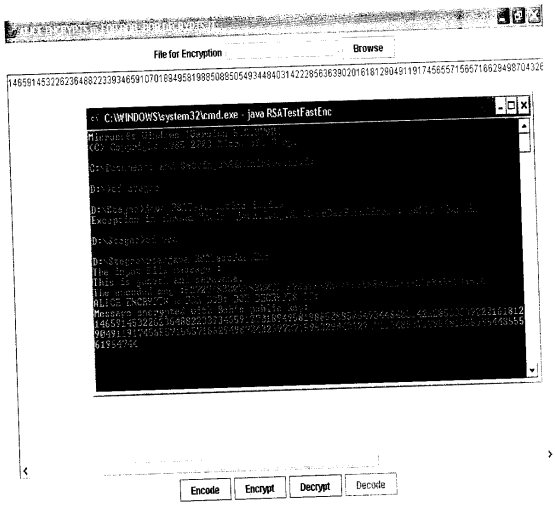

Fig.6.2:snap shot2

EMBEDDING ENCRYPTED MESSAGE:

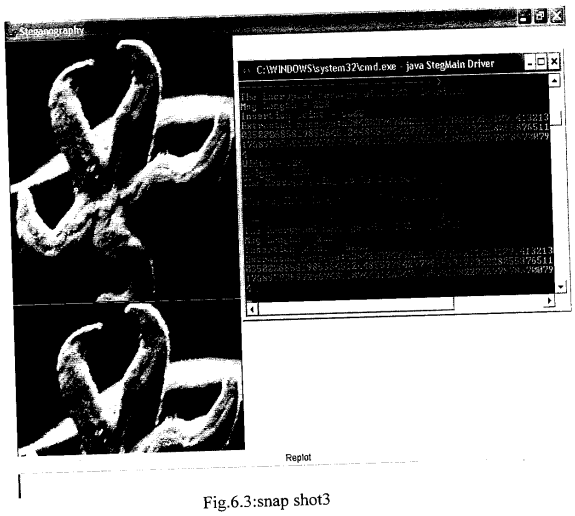

Fig.6.3:snap shot3

REFERENCES

CHAPTER 7

REFERENCES

BOOKS:

- F. Alturki and R.Mersereau (2001), " A Novel Approach for Increasing security and data embedding capacity in images for data hiding applications".

- R.J. Anderson and F.A.P. Petitcolas (1998), " On the limits of Steganography", IEEE journal of selected Areas in communications, Special Issue on copyright and privacy protection, pp., 1998.

- C.Cachin (1998), "An information-Theoretic Model for Steganography ", Lecture Notes on computer science, vol. 1525, springer-Verlag, New york, 1998, pp. 306-318

- H. Farid (2001), " Detecting steganographic Message in digital Images", Report TR2001-412, Darmouth college, hanover, NH, 2001.

- J. Fridrich, M. Goljan,and R.Du (2001), "Steganalysis based on JPEG compatibility", SPIE Multimedia systems and Applications IV, Denver, CO, August 20 24, 2001