International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 2, Issue 9 (May 2013) PP: 06-15

Stegnography Systems for Securing DataFile in Image

M. Charles Arockiaraj
Head & Asst. Professor, Dept. of Computer Science
Arakkonam Arts and Science College Arakkonam, Tamil Nadu-631003, India

Abstract: *Steganography system enhances data protection from hackers, unknown users and others and secure data file in an image. Securing Data file through Image (SDI) is a Steganography systems program. It is a program that allows to "hide" any kind of file inside standard bitmap pictures. The pictures look like normal images, so people will not suspect they contain hidden data. You can use a password to hide your files, and only those who know the password use are able to retrieve them - without it, people cannot even be sure there is something hidden in the image.*

At this information age, a lot of new technologies and methodologies are introduced by computer science experts as to protect the valuable data and information of corporate. But at this juncture, these technologies are not enough to secure data from unknown hackers. Even though there are a lot of technology tools available, data gets corrupted or copied or deleted from powerful enterprise servers without the knowledge of system analysts and administrators. Similarly most of the data files are damaged or copied or deleted when it is transmitted from one terminal to another terminal may be located in another place or even another country.

Key words: *Digital images, Digital watermarks, Vernam cipher, Avalanche effect*

I. Introduction:

Digital images, audio and video files are composed of collections of bits that are translated by their related software into images, sounds or videos. These files contain unused areas or data that is insignificant and can be overwritten. By using this proposed algorithm, we can hide our file of any format in an image. It will not be obvious that the image also contains a secret data in it. We can then send the image via e-mail attachment or post it on the web site and anyone with knowledge that it contains secret information, and who is in possession of the encryption password, will be able to open the file, extract the secret information and decrypt it. So this project would certainly enable security to highest level for data when it is transmitted from one place to another. So others even the hackers can not identify the contents which are kept inside the image. Steganography literally means covered writing. Its goal is to hide the fact that communication is taking place. This is often achieved by using a (rather large) cover file and embedding the (rather short) secret message into this file. The result is an innocuous looking file (the stegofile) that contains the secret message. It has until recently been the poor cousin of cryptography. Now, it is gaining new popularity with the current industry demands for digital watermarking and fingerprinting of audio and video. There are three different aspects in information-hiding systems contend with each other: capacity, security and robustness. Capacity refers to the amount of information that can be hidden in the cover medium, security to an eavesdropper's inability to detect hidden information and robustness to the amount of modification the stego medium can withstand before an adversary can destroy the hidden information. In the field of Steganography, some terminology has been developed, as explained below.

The adjectives cover, embedded and stego were defined at the Information Hiding Workshop held in Cambridge, England. The term "cover" is used to describe the original, innocent message, data, audio, still, video and so on. The information to be hidden in the cover data is known as the "embedded" data. The "stego" data is the data containing both the cover signal and the "embedded" information. Logically, the processing of putting the hidden or embedded data, into the cover data, is sometimes known as embedding. Occasionally, especially when referring to image steganography, the cover image is known as the container.

II. How does steganography work?

Digital images, audio and video files are composed of collections of bits that are translated by their related software into images, sounds or videos. These files contain unused areas or data that is insignificant and can be overwritten. By using this proposed algorithm, we can hide our text in an image. It will not be obvious that the image also contains text. We can then send the image via e-mail attachment or post it on the web site and anyone with knowledge that it contains secret information, and who is in possession of the encryption password, will be able to open the 3file, extract the secret information and decrypt it.

1. 1. Cryptography & Steganography: A Comparison

While cryptography is about protecting the content of messages (their meaning), Steganography is about concealing their very existence. It comes from Greek roots, literally means ‘covered writing’, and is usually interpreted to mean hiding information in other information. Examples include sending a message to a spy by marking certain letters in a newspaper using invisible ink, and adding sub perceptible echo at certain places in an audio recording.

It is often thought that communications may be secured by encrypting the traffic, but this has rarely been adequate in practice. Eneas the Tactician, and other classical writers, concentrated on methods for hiding messages rather than for enciphering them; and although modern cryptographic techniques started to develop during the Renaissance, we find in 1641 that John Wilkins still preferred hiding over ciphering because it arouses less suspicion. This preference persists in many operational contexts to this day.

For example, an encrypted email message between a known drug dealer and somebody not yet under suspicion, or between an employee of a defense contractor and the embassy of a hostile power, has obvious implications. A message in cipher text may arouse suspicion while an invisible message will not.

In contrast to cryptography, where the “enemy” is allowed to detect, intercept and modify messages without being able to violate certain security premises guaranteed by a cryptosystem, the goal of Steganography is to hide messages inside other “harmless” messages in a way that does not allow any “enemy” to even detect that there is a second secret message present. As the purpose of Steganography is having a covert communication between two parties whose existence is unknown to a possible attacker, a successful attack consists in detecting the existence of this communication (e.g., using statistical analysis of images with and without hidden information). Digital watermarks are pieces of information added to digital data/audio, video or still images that can be detected or extracted later to make an assertion about the data. These digital watermarks remain intact under transmission/transformation, allowing us to protect our ownership rights in digital form. Technically, watermarking is not a steganographic form.

Strictly, Steganography conceals data in the image, watermarking extends the image information and becomes an attribute of the cover image, providing license, ownership or copyright details. Watermarking, as opposed to Steganography, has the (additional) requirement of robustness against possible attacks. In this context, the term ‘robustness’ is still not very clear; it mainly depends on the application. Copyright marks do not always need to be hidden, as some systems use visible digital watermarks, but most of the literature has focused on imperceptible (e.g., invisible, inaudible) digital watermarks, which have wider applications.

Visible digital watermarks are strongly linked to the original paper watermarks, which appeared at the end of the 13th century to differentiate paper makers of that time. Modern visible watermarks may be visual patterns (e.g., a company logo or copyright sign) overlaid on digital images. The intent of use is also different: the payload of a watermark can be perceived as an attribute of the cover-signal (e.g., copyright information, license, ownership, etc.). In most cases the information hidden using steganographic techniques is not related at all to the cover. These differences in goal lead to very different hiding techniques.

A watermarking system’s primary goal is to achieve a high level of robustness- that is, it should be impossible to remove a watermark without degrading the data object’s quality. Steganography, on the other hand, strives for high security and capacity, which often entails that the hidden information is fragile. Even trivial modifications to the stego medium can destroy it.

Embed data in an image: Image Steganography:

Before proceeding into the algorithm, a small note on how images are stored into files. To a computer, an image is an array of numbers that represent light intensities at various points, or pixels. These pixels make up the image’s raster data. An image size of 640 x 480 pixels, utilizing 256 colors (8 bits per pixel) is fairly common. Digital images are typically stored in either 24-bit or 8-bit per pixel files. 24-bit images are known as true color images. Obviously, a 24-bit image provides more space for hiding information; however, a 24-bit images are generally large and not that common. A 24- bit image 1024 pixels wide by 768 pixels high would have a size in excess of 2MB. Alternatively, 8-bit color images can be used to hide information. In 8-bit color images, each pixel is represented as a single byte. Each pixel merely points to a color index table, or palette, with 256 possible colors. The pixel’s value, then, is between 0 and 255. The image software merely needs to paint the indicated color on the screen at the selected pixel position.

Image compression:

Image compression offers a solution to large image files. Two kinds of image compression are loss less and lossy compression. Both methods save storage space but have differing effects on any uncompressed hidden data in the image. Lossy compression, as typified by jpeg format files, offers high compression, but may not maintain the original image’s integrity. This can impact negatively on any hidden data in the image. This is due to the lossy compression algorithm, which may lose unnecessary image data, providing a close approximation to high quality digital images, but not an exact duplicate. Hence, the term lossy compression. Lossy compression is frequently used on true-color images, as it offers high compression rates. Loss less compression maintains the original image data exactly: hence it is preferred when the original information must remain intact. It is thus more favored by steganographic techniques. Unfortunately, loss less compression does not offer such high compression rates as lossy compression. Typical examples of loss less compression formats are GIF, BMP and PCX. This algorithm hides the encrypted file into a carrier medium.(The image file which carries the encrypted data),using the least significant bit insertion technique. The stego medium may be any image file compressed with loss less compression. The message is hidden into the LSBs of the image file. This algorithm handles the carrier file in a much careful way, since a very small change in the stego file, which is noticeable, will reveal the fact that it contains some data.

2.1 Least significant bit insertion

The least significant bit insertion method is probably the most well known image Steganography technique. It is a common, simple approach to embed information in a graphical image file. Unfortunately, it is extremely vulnerable to attacks, such as image manipulation. A simple conversion from a GIF or BMP format to a lossy compression format such as JPEG can destroy the hidden information in the image. When applying LSB techniques to each bytes of a 8-bit image, one bit can be encoded to each pixel. Any changes in the pixel bits will be indiscernible to the human eye. The main advantage of LSB insertion is that data can be hidden in the least and second to least bits and still the human eye would be unable to notice it. Care needs to be taken in the selection of the cover image, so that changes to the data will not be visible in the stego-image.

2.2 Multi-level securities proposed in the algorithm

1. To hide a text into the image
   a) Apply encryption algorithm. This algorithm encrypts the text with a strong block cipher mechanism by applying the cipher block- chaining mode. It also provides the password protection.
   b) The cipher text file is embedded into the stego medium
2. To extract a text from the image
   a. By extracting the LSBs from the stego image, a file containing cipher text is obtained.
   b. This file is decrypted using the encryption algorithm to get the Original file.

2.3 How security is enforced in the proposed algorithm?

The stego image file along with the message embedded into it is available to the hacker. The stego file does not reveal any difference in attributes like size, content etc., from that of the original file. Hence it is difficult for a hacker to find out that this image contains a message. Even if he doubts so, he has to apply the same algorithm to retrieve the embedded file. Unfortunately if he finds out the algorithm by which the text has been embedded into the medium, he could get back only the cipher text file. In the retrieved file, he could see nothing but the junk characters. This provides an extra layer of protection. To get the original message, this junk file has to be decrypted with the encryption algorithm with the correct password. This also adds an extra layer of protection. If the algorithm is found out, he must supply the correct password to get the correct message. Even if one character is altered in the password the original message can’t be obtained. Hence a hacker must know the following in order to extract the embedded message from the image file.

1. Algorithm to extract the message from the image. (Stego algorithm)
2. Encryption algorithm.
3. Correct password for algorithm.

With these increased levels of protection using encryption algorithm, the proposed system for Steganography is more strong from attacks than any other existing system.

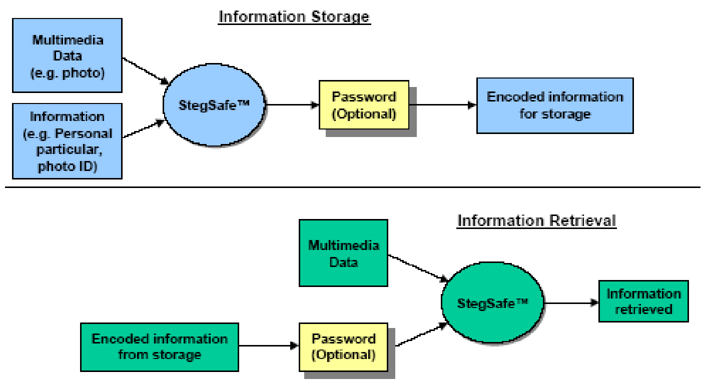

Information Storage
Multimedia Data (e.g. photo)
Information (e.g. Personal particular, photo ID)
StegSafe™
Password (Optional)
Encoded information for storage
Information Retrieval
Multimedia Data
StegSafe™
Information retrieved
Encoded information from storage
Password (Optional)

Figure1.1Steganography systems through data flow diagram

III. IMPLEMENTATION

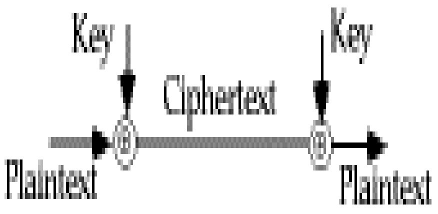
MODERN PRIVATE KEY CIPHERS

3.1. Introduction

- ➢ Now want to concentrate on modern encryption systems
- ➢ These usually consider the message as a sequence of bits (eg as
- ➢ a series of ASCII characters concatenated)
- ➢ Have two broad families of methods Stream ciphers and block
- ➢ ciphers

3.2. Stream Ciphers and the Vernam cipher

- ➢ Process the message bit by bit (as a stream)
- ➢ The most famous of these is the **vernam cipher** (also known as the **one- time pad**)
- ➢ Invented by vernam, working for AT&T, in 1917
- ➢ Simply add bits of message to random key bits
- ➢ Need as many key bits as message, difficult in practice (ie distribute on a mag-tape or cdrom)
- ➢ Is unconditionally secure provided key is truly random



- ➢ Since difficult to distribute so much key, why not generate key
- ➢ stream from a smaller (base) key
- ➢ Use some pseudo-random function to do this

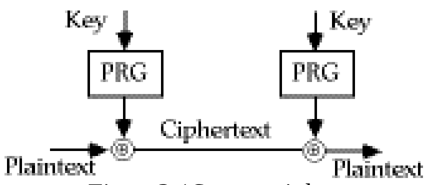


Figure3.1Stream cipher

- Although this looks very attractive, it proves to be very difficult in practice to find a good pseudo-random function that is cryptographically strong
- This is still an area of much research

3.3. Block Ciphers

- In a block cipher the message is broken into blocks, each of which is then encrypted (ie like a substitution on very big characters –64-bits or more)
- Most modern ciphers we will study are of this form

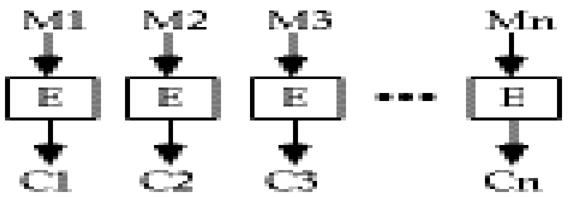


Figure3.2.Modern block cipher

3.4. Substitution-Permutation Ciphers

- In his 1949 paper Shannon also introduced the idea of substitution-permutation (S-P) networks, which now form the basis of modern block ciphers
- An S-P network is the modern form of a substitution-transposition product cipher
- S-P networks are based on the two primitive cryptographic Operations we have seen before.

3.4.1. Substitution Operation

- A binary word is replaced by some other binary word
- The whole substitution function forms the key
- If use n bit words, the key is 2^(n)!bits, grows rapidly

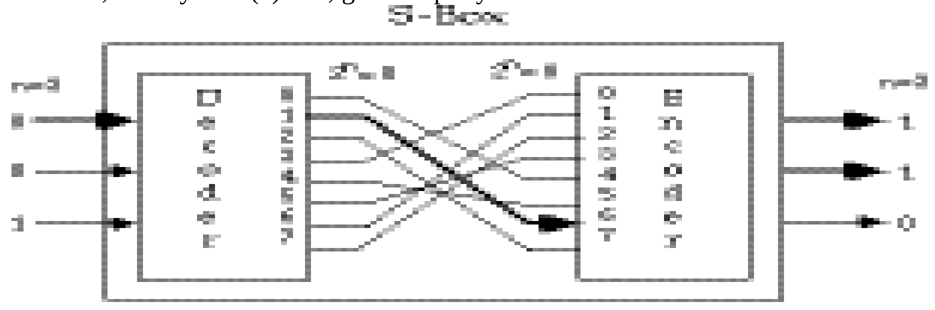


Fig 2.1 Substitution Operation

Can also think of this as a large lookup table, with n address lines

- (hence 2^(n) addresses), each n bits wide being the output value
- will call them **S-boxes**

3.4.2. Permutation Operation

- A binary word has its bits reordered (permuted)
- The re-ordering forms the key
- If use n bit words, the key is n! bits, which grows more slowly, and hence is less secure than substitution

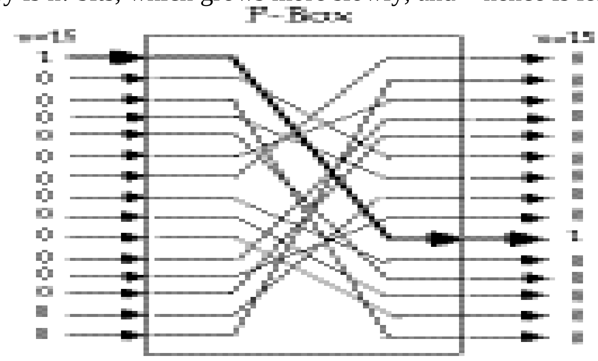


Fig 2.2 - Permutation or Transposition Function

This is equivalent to a wire-crossing in practice (though is much harder to do in software)
will call these **P-boxe**

3.4.3. Avalanche effect

where changing **one** input bit results in changes of approx **half** the output bits More formally, a function f has a good **avalanche** effect if for each bit i,0<=i<m, if the 2^(m) plaintext vectors are divided into 2^(m-1) pairs X and X_(i) with each pair differing only in bit i; and if the 2^(m-1) exclusive-or sums, termed avalanche vectors V_(i) = f(X) (+) f(X_(i)) are compared, then about half of these sums should be found to be 1.

3.4.4. Practical Substitution-Permutation Networks

In practice we need to be able to decrypt messages, as well as to encrypt them, hence either:

Have to define inverses for each of our S & P-boxes, but this doubles the code/hardware needed, or Define a structure that is easy to reverse, so can use basically the same code or hardware for both encryption and decryption Horst Feistel, working at IBM Thomas J Watson Research Labs devised just such a structure in early 70's, which we now call a **feistel cipher** The idea is to partition the input block into two halves, L(i-1)and R(i-1), and use only R(i-1)in each round i (part) of the cipher

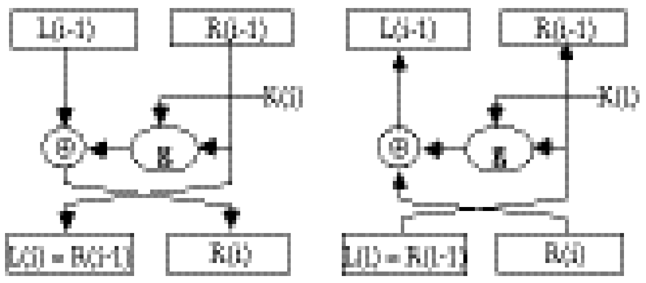


Fig 2.4 - A Round of a Feistel Cipher

- The function g incorporates one stage of the S-P network,controlled by part of the key K(i)known as the ith sub key
- This can be described functionally as: L(i) = R(i-1)
  R(i) = L(i-1) (+) g(K(i), R(i-1))
- This can easily be reversed as seen in the above diagram,working backwards through the rounds
- In practice link a number of these stages together (typically 16 rounds) to form the full cipher

3.5. Blowfish

Blowfish is a variable-length key block cipher. It does not meet all the requirements for a new cryptographic standard discussed above: it is only suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than DES when implemented on 32-bit microprocessors with large data caches, such as the Pentium and the PowerPC.

3.5.1. Description of the algorithm

Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two parts: a key-expansion part and a data- encryption part. Key expansion converts a key of at most 448 bits into several sub key arrays totaling 4168 bytes. Data encryption occurs via a 16-round Feistel network. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

Sub keys:

Blowfish uses a large number of sub keys. These keys must be precomputed before any data encryption or decryption.

1. The P-array consists of 18 32-bit sub keys: P1, P2,..., P18.
2. There are four 32-bit S-boxes with 256 entries each: S1,0, S1,1,..., S1,255; S2,0, S2,1,..,, S2,255; S3,0, S3,1,..., S3,255; S4,0, S4,1,..,, S4,255.

The exact method used to calculate these sub keys will be described later.

Encryption: Blowfish is a Feistel network consisting of 16 rounds. The input is a 64-bit data element, x.

```
Divide x into two 32-bit halves: xL, xR
For i = 1 to 16:
xL = xL XOR Pi
xR = F(xL) XOR xR
Swap xL and xR
Next i
Swap xL and xR (Undo the last swap.)
xR = xR XOR P17
xL = xL XOR P18
Recombine xL and xR
```

Function F
Divide xL into four eight-bit quarters: a, b, c, and d
F(xL) = ((S1,a + S2,b mod 2^{32}) XOR S3,c) + S4,d mod 2^{32}

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order. Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all sub keys are stored in cache. Generating the Sub keys. The sub keys are calculated using the Blowfish algorithm. The exact method is as follows:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3). For example:

P1 = 0x243f6a88
P2 = 0x85a308d3
P3 = 0x13198a2e
P4 = 0x03707344

2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)
3. Encrypt the all-zero string with the Blowfish algorithm, using the sub keys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified sub keys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P- array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations are required to generate all required sub keys. Applications can store the sub keys rather than execute this derivation process multiple times.

3.5.2 Byte Substitution

- A simple substitution of each byte
- Uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
- Eeach byte of state is replaced by byte in row (left 4-bits) & column (right 4-bits)
  eg. byte {95} is replaced by row 9 col 5 byte which is the value
  {2A}
- S-box is constructed using a defined transformation of the values in GF(2^8) designed to be resistant to all known attacks

Shift Rows

- A circular byte shift in each each 1st row is unchanged
  2nd row does 1 byte circular shift to left
  3rd row does 2 byte circular shift to left
  4th row does 3 byte circular shift to left
- Decrypt does shifts to right
- Since state is processed by columns, this step permutes bytes between the columns

Mix Columns

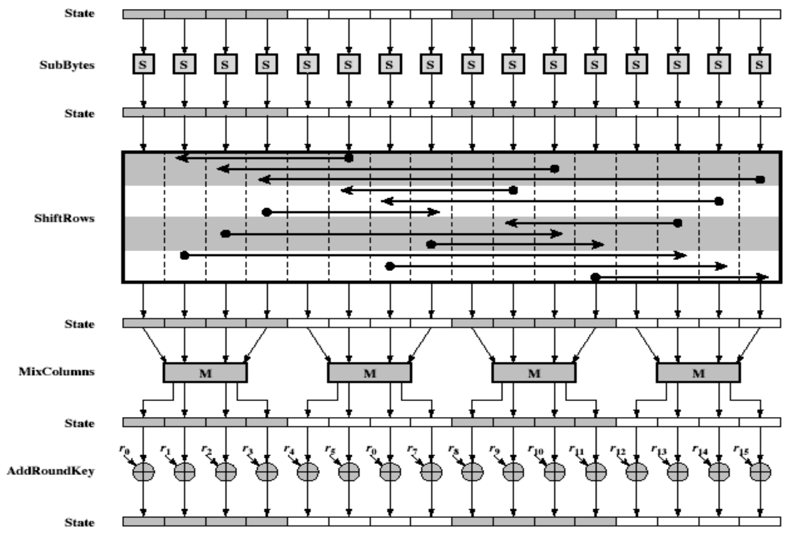
- each column is processed separately
- each byte is replaced by a value dependent on all 4 bytes in the column effectively a matrix multiplication in GF(2^8) using prime poly m(x) =$x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

3.6 Add Round Key

- XOR state with 128-bits of the round key
- Again processed by column (though effectively a series of byte operations)
- Inverse for decryption is identical since XOR is own inverse, just with correct round key designed to be as simple as possible

AES Round


State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

3.6.1. AES Key Expansion

- Takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
- Start by copying key into first 4 words
- Then loop creating words that depend on values in previous & 4 places back in 3 of 4 cases just XOR these together every 4th has S-box + rotate + XOR constant of previous before XOR together designed to resist known attacks

3.6.2. AES Decryption

- AES decryption is not identical to encryption since steps done in reverse
- But can define an equivalent inverse cipher with steps as for encryption but using inverses of each step with a different key schedule
- Works since result is unchanged when swap byte substitution & shift rows swap mix columns & add (tweaked) round key

Implementation Aspects

- Can efficiently implement on 8-bit CPU byte substitution works on bytes using a table of 256 entries shift rows is simple byte shifting add round key works on byte XORs mix columns requires matrix multiply in GF(2^8) which works on byte values, can be simplified to use a table lookup
- Can efficiently implement on 32-bit CPU redefine steps to use 32-bit words can precompute 4 tables of 256-words then each column in each round can be computed using 4 table lookups + 4 XORs at a cost of 16Kb to store tables designers believe this very efficient implementation was a key factor in its selection as the AES cipher

IV. RESULTS AND DISCUSSION

Steganography systems enhancing data security. It uses images for hiding valuable data and information. Protecting Data file in Image (SDI) is a *Steganography* systems program. It is a program that allows to "hide" any kind of file inside standard bitmap pictures. The pictures look like normal images, so people will not suspect they contain hidden data.

You can use a password to hide your files, and only those who know the password use are able to retrieve them - without it, people cannot even be sure there is something hidden in the image.

4.1. Basic concepts

SDI hides data inside a picture by modifying its colors in a way that is almost unnoticeable by the human eye. When a file is hidden, it conceals each part in an area of the picture; the areas used to store each part of the file are chosen by doing several calculations with the password given. When retrieving a file, the same calculations are done to the password and, knowing which area contains each part of the file, it can be reconstructed. If the wrong password is used when retrieving a file, SDI will try to read the file from the wrong areas and will not find anything there.

When a file is hidden in a picture that already has something in it, the new file may be written in areas where the previous data was stored (as SDI has no way of knowing that it was already there), so the old file will be erased or corrupted. Also notice that a large file will need more areas to be fully hidden than a small one, so it may be easier to see the modifications in the picture. If you hide a file that is too large, there will be a lot of "noise" in the picture and it will be easy to notice there is something hidden in the picture, even if other people cannot see what is actually hidden in it.

4.2 Security recommendations

Security was a primary concern in SDI's design. However, if you want to be sure nobody will find out that one of your pictures contains hidden data or, even worse, retrieve the hidden file, you should follow these guidelines:

- Do not use short passwords or passwords that can be easily guessed (such as your name, phone number, or a single word), as an attacker could automatically try all of these passwords. Use different capitalizations, combinations of words, numbers and punctuation marks, and anything else you can think of.
- Do not use pictures available from the Internet or other publicly-available sources. The best source for pictures is scanning photographs - preferably your own. This is because if someone finds the original picture, they will be able to see yours is somewhat different, and may suspect there is hidden data in it - even if they cannot read it.
- Erase the original picture after you have hidden the file. The reason is the same as the one above - if someone finds the original picture, they might suspect something. If possible, use a program that wipes (overwrites) the data when deleting it; PGP is able to do this.
  Do not use computer-generated images. They might have large areas filled with a single color or containing linear fades, and small changes in those areas can be easily noticed. It is also possible to identify changes in other computer-generated images such as fractals.
- Do not write files that are too big, or the noise in the picture will be easily noticed and someone may suspect there is hidden data in it. For 24-bit images, you should not write files larger than about 40% of the picture size. For 8-bit images, the files should be even smaller. These values are only estimates, and are not valid in all cases. After the hiding process, take a close look at the resulting picture - if you think its quality is worse than that of the original, you should use a larger picture.

Even if you follow all of these, remember that no program or algorithm is completely safe. The author does not take responsibility for any problems that may arise from security flaws or errors in the program. You should not trust this program for hiding critical data; if that is the case, get professional advice.

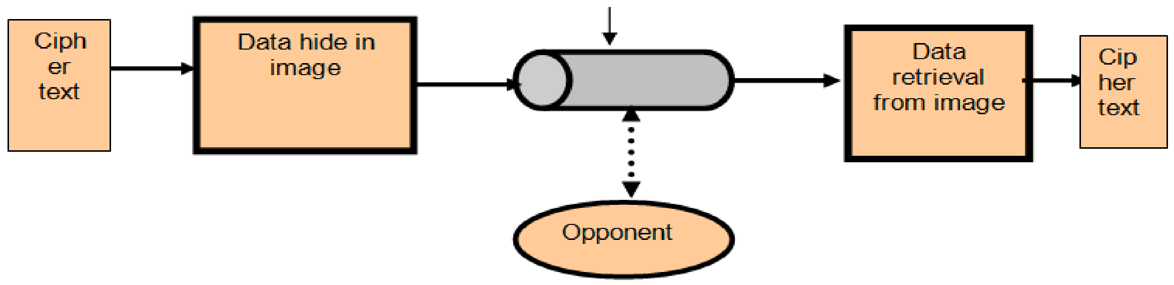


Figure 3.1 Protecting data in image

Cipher text—coded message

Opponent—nothing but hackers

Information channel--- e-mail, web, LAN

V. CONCLUSIONS

Until recently, information hiding techniques received very much less attention from the research community and from industry than cryptography. Steganography has its place in security. It is not intended to replace cryptography but supplement it.

Hiding a message with Steganography methods reduces the chance of a message being detected. However, if that message is also encrypted, if discovered, it must also be cracked (yet another layer of protection). There are an infinite number of Steganography applications.

This article explores a tiny fraction of the art of Steganography. It goes well beyond simply embedding encrypted text in an image. Steganography does not only pertain to digital images but also to other media (files such as voice, other text and binaries; other media such as communication channels, the list can go on and on).

REFERENCES

[1] Herbert S. Zim, Codes and Secret Writing, William Marrow and Company. New York, NY, 1948.
[2] J. Brassil, S. Low, N. Maxemchuk, L. O'Goram, "Hiding Information in Document Images,"
[3] CISS95.
[4] Tuomas Aura, "Invisible Communication," IEEE 1995.
[5] Neil F. Johnson, Zoran Duric, Sushil Jajodia, Information Hiding: Steganography and Watermarking - Attacks and Countermeasures Kluwer Academic Press, Norwrll, MA, New York,
[6] The Hague, London, 2000.
[7] Diffie,W. "The first ten year of cryptography".
[8] Stinsown,D."Cryptography:Theory and practice" .
[9] Uyless black "Internet security protocols".
[10] Ankit Fadia "Network Security".
[11] Cryptography and network security Third edition -William stallings