ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/304646549

Hash and Salt based Steganographic Approach with Modified LSB Encoding

Article *in* International Journal of Innovative Research in Computer and Communication Engineering · June 2016

DOI: 10.15680/IJIRCCE.2016.0406054

CITATIONS
11

READS
1,605

5 authors, including:

Pramod George Jose
Vrije Universiteit Amsterdam
5 PUBLICATIONS **31** CITATIONS
SEE PROFILE

Soumick Chatterjee
Human Technopole
83 PUBLICATIONS **699** CITATIONS
SEE PROFILE

Mayank Patodia
People's Education Society
1 PUBLICATION **11** CITATIONS
SEE PROFILE

Asoke Nath
St. Xavier's College, Kolkata
283 PUBLICATIONS **2,475** CITATIONS
SEE PROFILE

All content following this page was uploaded by Asoke Nath on 01 July 2016.

The user has requested enhancement of the downloaded file.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Hash and Salt based Steganographic Approach with Modified LSB Encoding

Pramod George Jose[1], Soumick Chatterjee[1], Mayank Patodia[2], Sneha Kabra[3], Asoke Nath[4]

M.Sc. Student, Dept. of Computer Science, St. Xavier's College (Autonomous), Kolkata, India[1]

MCA Student, Dept. of Computer Applications, PES University, Bengaluru, India[2]

MCA Student, Dept. of Computer Applications, VIT University, Vellore, India[3]

Associate Professor, Dept. of Computer Science, St. Xavier's College (Autonomous), Kolkata, India[4]

ABSTRACT: In today's world, secure communication is a major concern. Classical cryptography does not conceal the existence of a hidden message; it merely scrambles the message. Steganography aims to obscure the very existence of any hidden message by hiding the message or file within a cover medium, known as a steganogram, that can be text, image, audio, video etc. Steganography is a very important research area and many algorithms have already been developed. For hiding secret message inside different types of cover files, Nath et al. have already proposed various methods. In the present study, the authors propose a modified approach to one of the most popular types of steganography – LSB Encoding, as the traditional approach is quite predictable. In this solution, a cover medium – which can be an image, audio or video, a password and the secret message or file is taken. Here the secret message can be a plain text message or it can be a file of absolutely any format. Then a salt is generated, which is a cryptographically secure random number, which is concatenated with the password to make it unique; as people tend to choose short, real words as passwords, which are easy to remember and use them over and over. To increase the entropy of this system, SHA-256 of the user given password along with the generated salt is calculated and using this hash as key, the message is encrypted using AES. Then the salt, length of the encrypted message or file, extension of the encrypted message or file, calculated hash as well as the encrypted message is uniformly distributed in the cover medium to obtain the steganogram. The way of distribution is not just encoding in each LSB, but in a modified way based on the password, salt and hash. While decoding, a steganogram and the password is taken. Based on the password, the salt is retrieved and SHA-256 of the password-salt combination is calculated and matched with the hash encoded in the steganogram. If matched, based on this hash, the encrypted message is retrieved and further decrypted to obtain the original message or file.

KEYWORDS: Cryptography, Hash, LSB Encoding, MSB-LSB bit selection, Salt, Steganography

I. INTRODUCTION

A. *What is Steganography?*

Cryptography is the art and study of various techniques employed in communicating securely over an unsecure channel. Encryption is the process of converting a normal message (called plaintext or clear text) and scrambling it based on a key into seemingly garbage text (called cipher text). Decryption is the reverse of this process where this unintelligible text is unscrambled back to the original message based on a key.

Steganography is the art of hiding a secret message (or a file) inside a cover medium which can be an innocuous, innocent looking multimedia file or even an IP packet, without any perceptible change. The secret message is called the payload and is scattered over the bytes of the cover medium uniformly using a key. The resulting, modified cover medium is called the steganogram. The key is necessary to decode the hidden message from the steganogram.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

B. *Why use Steganography?*

Cryptography has four main objectives [1] – confidentiality (message cannot be read by an unintended recipient or an adversary), integrity (any changes made in the cipher text during transmission can be detected easily), authentication (the communicating parties can verify each other's identity and also the origin and destination or the message) and non-repudiation (the sender cannot deny creating the message at a later stage).

Steganography aims to conceal the very existence of communication between communicating parties. To quote Wikipedia [2], "the advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages—no matter how unbreakable—arouse interest, and may in themselves be incriminating in countries where encryption is illegal [3]. Thus, whereas cryptography is the practice of protecting the contents of a message alone, steganography is concerned with concealing the fact that a secret message is being sent, as well as concealing the contents of the message."

C. *Types of steganography*

Depending on the cover medium used, there are various types of steganography – for example, image steganography, audio steganography, text steganography, etc. There are also various techniques to implement each type of steganography – for example, to implement text steganography, line-shift coding, word-shift coding, feature coding, etc. can be used[4]. Another form of steganography is the creation of covert channels in commonly used systems, like selectively fragmenting certain files in a file system where the difference between consecutive file blocks would represent a byte of information [5]. The basic idea behind steganographic systems is to embed data in such places where the existence of data is not at all expected, for example, in the firmware of hard disk microcontrollers [6]. One of the most popular and simplest techniques used in image, audio and video steganography is the LSB (Least Significant Bit) encoding scheme.

D. *Least Significant Bit Encoding*

Any digital data is stored in bytes and each byte contains 8 bits. The bit with the lowest numerical value (bit weight = 1) is called the least significant bit (LSB) – in simple words, the right most bit – which is highlighted in Fig. 1. The second least significant bit (LSB+1) is the bit having the second lowest numerical value (bit weight = 2). Similarly, the most significant bit (MSB) is the bit having the highest numerical value (bit weight = 128), which is the left most bit.

1	0	0	1	0	1	0	1

Fig. 1 Least Significant Bit of a given Byte

Image, audio and video files have a high amount of detail and contain a reasonable amount of redundancy. Making very slight changes to the data bytes of such type of files will not cause a difference that would be perceptible to the human sensory system. For example, in a 24 byte colour image system, there are more 16 million values that each pixel can attain. Suppose there is a pixel with the Red component value as 100, Green component as 200 and Blue component as 150. Incrementing each of the component values by 1, of this particular pixel, would make an ever so slight change in the overall image – indistinguishable to the human eye.

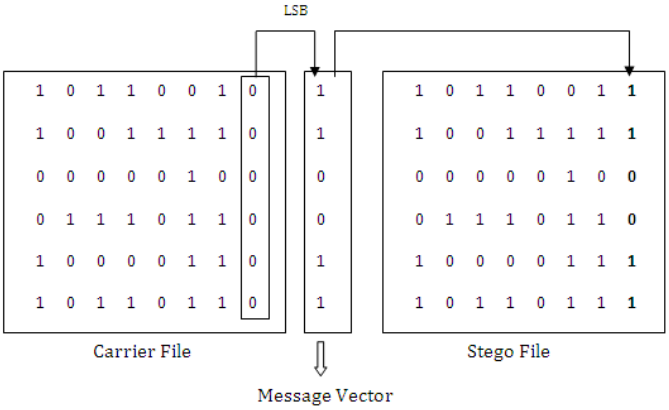
In LSB encoding, it is required to alter the least significant bits of the data bytes of the cover medium according to the bit values of the message to be hidden. Hence, LSB of the first byte of the cover medium should be made same as the first bit (MSB) of the message to be hidden; the LSB of the second byte of the cover medium the same as the second bit (second MSB) of the message to be hidden, the LSB of the third byte of cover medium the same as the third bit (third MSB) of the message and so on and so forth as shown in Fig. 2. Now, to decode the hidden message, simply read the least significant bits of each byte of the cover medium; put 8 bits together to form a single byte of the message. This process is repeated to get the whole hidden message.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016



E. *What is Hash?*

A cryptographic hash function is a one-way mathematical function which transforms an input message of arbitrary length into a unique hash value of fixed length in such a way that it is infeasible to compute the input message, given the hash value. The resulting hash value is also called a message digest or checksum which serves as a "signature" of the input message. In essence, a particular message will generate a unique message digest; which can be generated by only that message. This is particularly useful in case where it is required to authenticate some message or an individual. For example, when a user signs up for an online account on Facebook, the hash value of the provided password is stored in Facebook's database, and when the user tries to login at a later time, the hash of the entered password is compared with the stored hash (Second point of Technical Details under "Keeping Passwords Secure" [7]). If both hashes are the same, the user is granted access.

SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed by the National Security Agency (NSA) [8]. SHA-2 includes significant changes from its predecessor, SHA-1. The SHA-2 family consists of six hash functions with digests (hash values) that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 [9]. In this algorithm, the concept of SHA-256 is used to generate a random pattern, explained later; using which the message bits are scattered in the cover medium.

F. *The concept of Salt*

People tend to choose short, real words as passwords, which are easy to remember and use them over and over which enables for an attack. Since this algorithm uses the hash of the entered password to generate the pattern for scattering the message bits over the cover medium, the same password would result in the same pattern. A salt is a random data which is concatenated with the password to make it unique. This way, if the user uses the same password, concatenating it with the salt makes it unique, resulting in a unique pattern.

II. ENCODING

A. *Algorithm*

Step 1: Input a Cover Medium
Step 2: Goto Step 3 if Cover Medium is compatible, if not, show error and go back to Step 1.
Step 3: Input hidden file or message
Step 4: Input Password
Step 5: Generate Salt
Step 6: Calculate Hash of Password & Salt
Step 7: Encrypt Message using the Hash to get the Cipher Text
Step 8: Calculate the Length of the Encrypted Message
Step 9: Get File Extension of the encrypted message/file.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Step 10: Encode the salt using the bit pattern of the password
Step 11: Encode the Hash-Length-Extension using the bit pattern of the salt.
Step 12: Encode the Cipher Text using the bit pattern of the Hash.
Step 13: Check whether the whole Cipher Text is encoded, if not, show error, else encoding is successful and the steganogram is obtained.

B. *Explanation with Example*

First, take a cover medium as input. This algorithm works on uncompressed cover data. If compressed data is provided, it needs to be de-compressed first. In case of lossless compression, this can be done easily; whereas for lossy compression, it may be difficult as some redundant cover data samples may be eliminated. Next, take a password as input.

Bytes of the cover sample are selectively chosen to hide the data, depending on the bit pattern of the entered password. That would mean, if same password is given, it would have the same encoding pattern. To overcome this, generate a cryptographically secure random number, also called as a salt which changes the encoding pattern even if the password and cover is the same. The length of the salt is constant which, in this implementation is 4 bytes. Now, append the salt with the password and then run it through SHA-256 to get a 32-byte hash. In the meantime, encrypt the message or file using AES with the hash as key. It is also required to encode the extension of the encrypted data, because it can be of any type. The length of extension is also constant at 4 bytes. Along with this, it is required to store the length of the encrypted data, otherwise it won't be understood how many bytes need to be extracted while decoding. The number of bytes required to store the length of the encrypted message is also constant at 4 bytes.

Depending on the bit pattern of the entered password, selectively chose bytes from the cover medium to encode the salt; and depending on the bit pattern of the salt, chose bytes to encode the computed hash, length and the extension of the encrypted message or file. Further, the encrypted message is stored following the bit pattern of the hash. The reason for storing these data has been explained later, in the decode phase. These bit patterns are also referred to as masks. If at a given time, the length of the mask is lesser than the length of the payload (data to be encoded), then the mask is looped over until the whole payload has been encoded.

To explain this algorithm further, here is an example. Say, that the password entered by the user is: 1001 (for the sake of simplicity), the random salt generated is: 0101 and the following are the first few bytes from the cover medium:

Byte-1	Byte-2	Byte-3	Byte-4	Byte-5	Byte-6	Byte-7	Byte-8
10001110	00001010	10101011	00101000	10101010	00101011	00001101	00101011
Byte-9	Byte-10	Byte-11	Byte-12	Byte-13	Byte-14	Byte-15	Byte-16
10101010	01010100	01010111	10101010	01011101	01110110	11010011	10101011

……. And continues

Byte Selection: It is mentioned earlier that the bit pattern (mask) is responsible for choosing the byte which will be used to encode. Here, the byte is selected based on the index number of 1's in the mask. In this case, password is 1001 so byte number 1 and 4, followed by byte number 5 and 8 will be selected to encode the salt 0101. The mask is lopped over as long as the whole payload has not been encoded, as in this case, first, 1 and 4 is taken based on 1001, then the same mask is repeated to get 5 and 8.

Bit Selection: For bit selection, the classical Least Significant Bit (LSB) encoding scheme has been modified here. In classical approach, payload is encoded in the LSB. But in this approach, the LSB is selected if the MSB of the same byte is 0; else the LSB+1 is chosen for encoding the message. Based on this example, byte numbers 1, 4, 5, 8 are already selected. For byte 1, since the MSB is 1, encode the first bit of the salt in LSB+1 of this byte. Similarly, for byte 4, since the MSB is 0, encode the second bit of the salt in the LSB. This process continues for all other selected bytes.

Let's continue this example – first column shows the byte number, second column shows the byte before encoding the salt, and the last column shows the byte after encoding the salt.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Table I. Changing LSB or LSB+1 to encode the salt

Byte No	Original Cover Bytes	After Encoding Salt	Remarks
1	10001110	10001100	LSB+1 is selected, bit changed
2	00001010	00001010	No change
3	10101011	10101011	No change
4	00101000	00101001	LSB is selected, bit changed
5	10101010	10101000	LSB+1 is selected, bit changed
6	00101011	00101011	No change
7	00001101	00001101	No change
8	00101011	00101011	LSB is selected, bit not changed
9	10101010	10101010	No change
10	01010100	01010100	No change
11	01010111	01010111	No change
12	10101010	10101010	No change
13	01011101	01011101	No change
14	01110110	01110110	No change
15	11010011	11010011	No change
16	10101011	10101011	No change
……	……. And continues		

After it is selected in which byte the data is to be encoded and whether the encoding will be done in LSB or LSB+1, it is checked whether the bit in the LSB or LSB+1 matches with the bit to be encoded. If matched, the bit remains unchanged, or else the bit is changed.

Next task is to encode the hash, followed by length and extension based on the bit pattern of the salt. Let's assume the hash is 110011. For the sake of simplicity, let's assume the bit pattern of length is 1 and the bit pattern of extension is 0. So, as a whole, the payload becomes 11001110. Let's now select the bytes where the hash-length-extension will be encoded. As the bit pattern of salt is 0101, byte numbers 2, 4, 6, 8, 10, 12, 14, 16 are selected. As it is seen here there is a collision occurring at byte number 4 and 8 as they have already been used to encode the salt itself. To prevent collision, the authors introduce the concept of regions and limits. The last byte used to encode the salt, in this case, byte number 8, is the limit of region 1. Similarly, the last byte used to encode the hash-length-extension, will be the limit of region 2. The remaining cover medium is region 3 where the rest of the encrypted message will be encoded.

The default mask of region 1 is the bit pattern of the password, which in this case is 1001, for region 2 it is bit pattern of the salt, in this case 0101 and for region 3, it is the bit pattern of the hash, which is in this case 110011. The cumulative mask of a region is a bit pattern where the 1's represents the bytes which have already been used to encode the payload in that region. Effective mask is also a bit pattern which is generated using the formula $\bar{A}.B$, where A represents each bit of the cumulative mask, and B represents the corresponding bit of the default mask. If a region is being used for the first time, then the cumulative mask and effective mask would be same as the default mask.

Step 1: Encoding Salt, using password (1001).

Table II. Masks during encoding the salt

	Region1	Region2	Region3
Default Mask	1001		
Cumulative Mask	1001		
Effective Mask	1001		
Limit	----		

Now follow the effective mask for encoding the salt. After encoding, the limit of the first region turns out to be 8.

Step 2: Encoding hash-length-extension (HLE) using Salt (0101).

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Table III. Masks during encoding the hash-length-extension

	Region1	Region2	Region3
Default Mask	1001	0101	
Cumulative Mask	1001	0101	
Effective Mask	0100	0101	
Limit	8	----	

Region 2 is being used for the first time, so all the three masks have the same bit pattern as the salt. The default mask of region 1 remains the same.

Now generate the effective mask by following the formula $\bar{A}.B$

Old cumulative Mask (A) = 1001
Bit pattern of Salt (B) = 0101
Effective Mask ($\bar{A}.B$) = 0100

The new cumulative mask is the bitwise OR of old cumulative mask and the effective mask.

Old cumulative Mask (A) = 1001
Effective Mask (C) = 0100
New Cumulative Mask ($A + C$) = 1101

The default mask of region 1 is the bit pattern of the password, which in this case is 1001, for region 2 it is bit pattern of the salt, in this case 0101 and for region 3, it is the bit pattern of the hash, which is in this case 110011

Table IV. Changing LSB or LSB+1 to encode HLE

Byte No	Original Cover Bytes	After Encoding HLE	Remarks
1	10001110	10001100	Salt is encoded
2	00001010	00001011	LSB is selected, bit changed
3	10101011	10101011	No change
4	00101000	00101001	Salt is encoded
5	10101010	10101000	Salt is encoded
6	00101011	00101011	LSB is selected, bit not changed
7	00001101	00001101	No change
8	00101011	00101011	Salt is encoded
9	10101010	10101010	No change
10	01010100	01010100	LSB is selected, bit not changed
11	01010111	01010111	No change
12	10101010	10101000	LSB+1 is selected, bit changed
13	01011101	01011101	No change
14	01110110	01110111	LSB is selected, bit changed
15	11010011	11010011	No change
16	10101011	10101011	LSB+1 is selected, bit not changed
17	11100011	11100011	No change
18	11000000	11000010	LSB+1 is selected, bit changed
19	10111000	10111000	No change
20	00011100	00011100	LSB is selected, bit not changed
……	……. And continues		

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

After encoding, the limit of the second region turns out to be 20.

Step 3: Encoding the encrypted message using Hash (110011). Let's assume first few bits of the encrypted message to be 101110111…..

Table V. Masks during encoding the message

	Region1	Region2	Region3
Default Mask	1001	0101	110011
Cumulative Mask	11011111	110111110111	110011
Effective Mask	00000010	100010100010	110011
Limit	8	20	----

Region 3 is being used for the first time, so all the three masks have the same bit pattern as the hash. The default mask of region 1 and 2 remains the same.

Now as it is seen, the length of the cumulative mask is lesser than the length of the hash, the authors have to introduce a concept of Mask Rolling. In this concept, repeat the bit pattern of cumulative mask and the hash, so that it matches the length of the region.

Old cumulative Mask (A) = 11011101
Bit pattern of Hash (B) = 11001111
Effective Mask ($\bar{A}.B$) = 00000010

The new cumulative mask is the bitwise OR of old cumulative mask (rolled) and the effective mask.
Old cumulative Mask (A) = 11011101
Effective Mask (C) = 00000010
New Cumulative Mask (A+C) = 11011111

Now have to follow the effective mask of region 1 for encoding a part of encrypted data in region 1. After running out of region 1, move into region 2, and follows the same procedure for generating the effective and cumulative mask. In short, the calculations are as follows: -

Old cumulative Mask (X) = 010101010101
Bit pattern of Hash (Y) = 110011110011
Effective Mask ($\bar{X}.Y$) = 100010100010

Old cumulative Mask (X) = 010101010101
Effective Mask (Z) = 100010100010
New Cumulative Mask (X+Z) = 110111110111
After running out of region 1, move into region 2, where the effective mask of region 2 is used.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Table VI. Changing LSB or LSB+1 to encode the encrypted secret message or file

Byte No	Original Cover Bytes	After Encoding message	Remarks
1	10001110	10001100	Salt is encoded
2	00001010	00001011	HLE is encoded
3	10101011	10101011	No change
4	00101000	00101001	Salt is encoded
5	10101010	10101000	Salt is encoded
6	00101011	00101011	HLE is encoded
7	00001101	00001101	LSB is selected, bit not changed
8	00101011	00101011	Salt is encoded
9	10101010	10101000	LSB+1 is selected, bit changed
10	01010100	01010100	HLE is encoded
11	01010111	01010111	No change
12	10101010	10101000	HLE is encoded
13	01011101	01011101	LSB is selected, bit not changed
14	01110110	01110111	HLE is encoded
15	11010011	11010011	LSB+1 is selected, bit not changed
16	10101011	10101011	HLE is encoded
17	11100011	11100011	No change
18	11000000	11000010	HLE is encoded
19	10111000	10111010	LSB+1 is selected, bit changed
20	00011100	00011100	HLE is encoded
21	00011000	00011000	LSB is selected, bit not changed
22	01111100	01111101	LSB is selected, bit changed
23	11111100	11111100	No change
24	00011101	00011101	No change
25	10111001	10111011	LSB+1 is selected, bit changed
26	11011011	11011011	LSB+1 is selected, bit not changed

III. **DECODING**

A. *Algorithm*

Step 1: Input the Steganogram and password.
Step 2: Go to Step 3 if Steganogram is compatible, if not, show error and go back to Step 1.
Step 3: Input Password.
Step 4: Extract salt from steganogram using the bit pattern of the password.
Step 5: Compute Hash of the extracted salt and entered password
Step 6: Extract the Hash-Length-Extension from steganogram using the bit pattern of the salt
Step 7: Compare extracted hash with the computed hash. Proceed if equal else show error and abort.
Step 8: Extract the Cipher Text using bit pattern of the hash.
Step 9: Decrypt the Cipher Text using the hash

B. *Explanation with Example*

First take input of the cover medium and the password. Based on the bit pattern of the password, extract the salt. As in this example, password is 1001 and first few bytes of cover medium is: -

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

Byte-1	Byte-2	Byte-3	Byte-4	Byte-5
10001100	00001011	10101011	00101001	10101000
Byte-6	**Byte-7**	**Byte-8**	**Byte-9**	**Byte-10**
00101011	00001101	00101011	10101000	01010100

……. And continues

Bytes 1, 4, 5, and 8 are selected based on the bit pattern of the password. It is quite clear that the length of the salt is constant. In this example it is 4 bits, in real-life, it would be of 4 bytes or 32 bits.

From the selected bytes, extract the salt following the same MSB-LSB rule explained before. The extracted salt is 0101.

Now, extract the hash-length-extension using the bit pattern of the salt. For this, back trace the same algorithm. It is also known that the length of hash is 6 bits in this example, and 32 bytes in real-world, extension and length to be 1 bit each in this example and 4 bytes each in real-world. From the value of length, the length of the encrypted data is obtained.

Next step is to generate the hash of entered password and extracted salt using SHA-256. Now if the computed hash matches with the extracted hash, then proceed with the extraction of the encrypted data or else abort. If the password is incorrect, then the bit pattern itself will be wrong which would result in an incorrect salt value, which in turn, will result in an incorrect hash value and it will not match with the computed hash. After the encrypted message is extracted, it is decrypted using AES with the hash as key to get the unencrypted data back.

IV. RESULTS AND DISCUSSION

Case 1: Cover Medium: Ring08.wav, Size 972KB from Windows Default sounds. Secret File: lsb.png, size 10.1KB. Password: password1

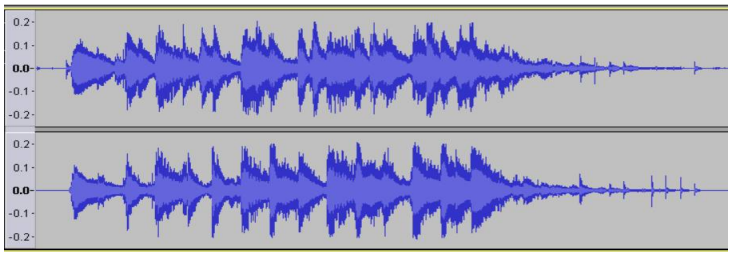


Fig. 3 Audio Graph of Ring08.wav before encoding
Audio graph depicting the various 16-bit stereo audio samples (showing both the channels) of Ring08.wav before encoding, which will be used as the cover medium to hide the secret file.

+

Fig. 4 Secret File lsb.png
Secret file which is to be encoded within the cover medium

=

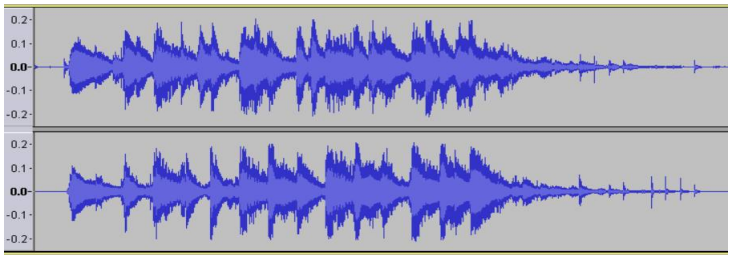


Fig. 5 Audio Graph of Ring082.wav after encoding

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

After Encoding, Ring082.wav is generated, size remains as 972KB. The size and graph of the cover medium remains unchanged. After decoding, the size and the content of the secret message/file remains the same.

Case 2: Cover Medium: BlankMap-World-large.png (Size: 102 KB), Secret Message: This is a Test Message Password: newpass123

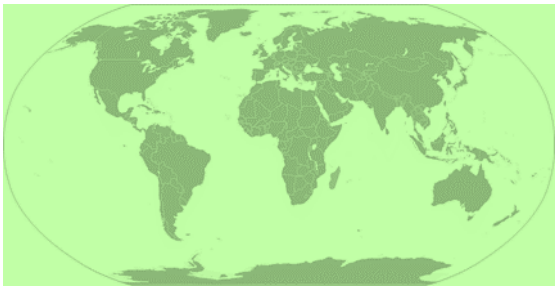


Fig. 6 BlankMap-World-large.png before encoding
A PNG image, which is compressed in lossless manner, used as the cover medium to hide the secret message

+

This is a Test Message

Fig. 7 The Secret Message that is to be encoded

=

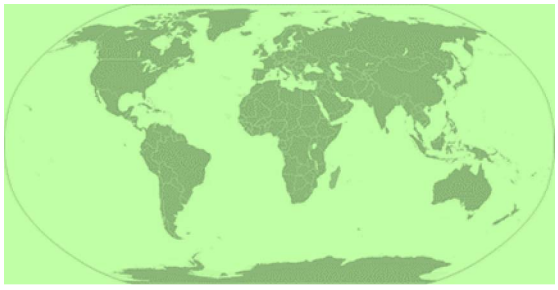


Fig. 8 BlankMap-World-large2.png after encoding
Appearance of the obtained PNG image remains same even after encoding the secret message

After Encoding, BlankMap-World-large2.png is generated; the size and appearance remains the same as 102KB. After decoding, the size and the content of the secret message/file remains the same.

Case 3: Cover Medium: albert_einstien.jpg. As this algorithm currently works only for either uncompressed media or lossless compressed media, and JPEG being lossy; it has to be converted into either of those. In this example, this image is converted into a BMP file using MS Paint (Size: 148 KB), Secret File: Program.cs (C# Code File, Size: 2KB), Password: pass1234

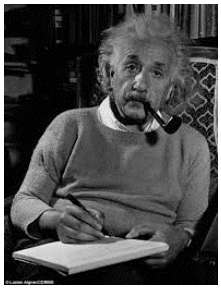


Fig. 9 albert_einstien.bmp before encoding
A BMP Image, an uncompressed greyscale image file, is used as the cover medium to hide the secret file

+

Program.cs
(C# Code File)

Fig. 10 Secret File Program.cs

=

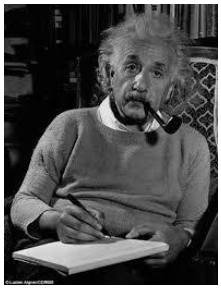


Fig. 11 albert_einstien2.bmp after encoding
Appearance of the obtained BMP image remains same even after encoding the secret file into it.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

After encoding, albert_einstien2.bmp is generated; size remains the same as 148 KB. After encoding, the size and appearance of the cover medium remains unchanged.

It has been observed that, this algorithm works flawlessly for uncompressed media files such as – WAV or AIFF for Audio, BMP for Image or AVI for video. But it may create issues while working with certain compressed media files, as, in compressed media files the exact pixel or audio sample values are not stored, and eventually this algorithm may alter vital information resulting in corrupted output. Compressed media files are of two types – lossy & lossless. Lossless compression is a class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data. By contrast, lossy compression permits reconstruction only of an approximation of the original data, not the complete original data [10]. This algorithm may create problem with lossy compression formats, but works fine with lossless compression formats such as FLAC or ALAC for Audio, PNG or GIF for Image but nothing common for video, though x264 or MSU are good candidates.

The exact payload capacity of this algorithm cannot be determined beforehand as the bytes of the cover medium, which will be used as the carrier bytes, are selected by the hash of the password and the salt. In the best case condition, if all the bytes of the cover medium are selected as carrier bytes, then, the payload capacity would be 12.5% of the cover medium.

But, according to use case statistics, the payload capacity of this algorithm is around 8-9%. This is a side-effect of the additional layer of randomness – the selection of cover medium bytes to be used as the carrier bytes by the hash of the password and the salt, makes it tougher for a potential eavesdropper to retrieve the hidden message. Since the pattern is not deterministic, the payload capacity is not deterministic as well.

The known drawbacks of this algorithm are: -

- If the message to be hidden is very small than the cover medium, then the encrypted message bits tend to get encoded in the first few bytes of the cover medium. This can be overcome by using a concept called permutative straddling [11].
- This algorithm currently works only with cover media which are either uncompressed or use lossless compression.
- This algorithm is not resistant against manipulations done on the cover medium after the message has been encoded. In other words, if an image-steganogram is rotated or cropped or if an audio-steganogram is edited or cut, then there are high chances of data loss.
- This algorithm works on a byte by byte basis. If the bit depth of any of the individual components of the cover medium is less than 8 bits (a byte), the current implementation will not work.
- This algorithm works only on cover media with reasonable redundancy. In other words, this algorithm will not work for text based cover media.

V. CONCLUSION

In this paper, the authors have presented a novel way to hide sensitive data in a cover medium by scattering the message bits over the cover medium based on the hash of a cryptographic salt and a password provided by the user. The salt ensures that a unique distribution pattern is used even if the user chooses the same password over and over again as there are 2^{32} combinations possible for the salt. Message is encoded based on hash, as the output of SHA-256 function has 2^{256} combinations, and finding the exact one used is similar to finding a needle in the haystack – almost impossible to brute force. Here the authors have used AES for encrypting the secret message or file before encoding, but any other encryption algorithm can be used.

The advantage of this algorithm is that it provides multiple level of security. Also this algorithm changes the traditional LSB encoding scheme in such a way, that it becomes more unpredictable.

This algorithm currently does not support media files that employ lossy compression. For example, to support encoding in JPEG, which uses DCT, algorithms such as JSteg or F5 have to be integrated. This algorithm can made be to resist manipulations, such as cropping, by detecting important areas of the image such as faces which are usually never cropped. If the encoded message is very small, it can be encoded several times and some level of redundancy can be created. Apart from these, there are various other areas for improvement which can be addressed in future.

ISSN(Online): 2320 - 9801
ISSN (Print) : 2320 - 9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 6, June 2016

REFERENCES

1. "What is cryptography?": http://goo.gl/T0pzTh
2. "Wikipedia – Steganography": https://en.wikipedia.org/wiki/Steganography
3. "How to Protect Your Online Privacy": goo.gl/jCNLGx
4. Dr. Govind N Sarage, "Various Aspects of Steganography", International Journal of Advanced Research in Computer Science and Software Engineering, Vol 4, Issue 8, 2014.
5. Hassan Khan, Mobin Javed, Fauzan Mirza and Syed Ali Khayam, "Evading Disk Investigation and Forensics using a Cluster-Based Covert Channel", ACM Conference on Computer and Communication Security (CCS), 2009.
6. Iain Sutherland, Gareth Davies, Andrew Blyth, "Malware and steganography in hard disk firmware", Journal in Computer Virology, Vol. 7, Issue 3, pp.215-219, 2011.
7. "Keeping Passwords Secure – Facebook": https://goo.gl/Agmg80
8. Wouter Penard and Tim van Werkhoven, "On the Secure Hash Algorithm family", Gerard Tel (ed.) - Cryptography in Context, pp.1-18, 2008
9. "Wikipedia - SHA-2": https://en.wikipedia.org/wiki/SHA-2
10. "Wikipedia - Lossless compression": https://en.wikipedia.org/wiki/Lossless_compression
11. Andreas Westfeld, "F5 - A Steganographic Algorithm - High Capacity Despite Better Steganalysis", 4th International workshop on information hiding, pp. 289–302, 2001.
12. Gary C. Kessler, "Steganography: Hiding Data Within Data", http://www.garykessler.net/library/steganography.html, September 2001.
13. A. Joseph Raphael and Dr. V. Sundaram, "Cryptography and Steganography-A Survey", International Journal of Computer Technology and Applications, Vol. 2, Issue 3, 2011.
14. Asoke Nath, Sankar Das, Samriddhi Joshi, Saulat Daanyaal Alam and Alvin Roetgen, "Steganography Algorithm to Hide Any Secret Message inside an Audio File", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 4, Issue 5, pp.8904-8909, 2016.
15. Joyshree Nath, Sankar Das, Shalabh Agarwal and Asoke Nath, "A Challenge In Hiding Encrypted Message In Lsb And Lsb+1 Bit Positions In Various Cover Files", Journal of Global Research in Computer Science, Vol. 2, Issue 4, pp.180-185, 2011.
16. Joyshree Nath and Asoke Nath, "Advanced Steganography Algorithm using Encrypted secret message", International Journal of Advanced Computer Science and Applications, Vol. 2, Issue 3, pp.19-24, 2011.
17. Joyshree Nath, Sankar Das, Shalabh Agarwal and Asoke Nath, "Advanced Steganographic Approach for Hiding Encrypted Secret Message in LSB, LSB+1, LSB+2 and LSB+3 Bits in Non-standard Cover Files", International Journal of Computer Applications, Vol. 4, Issue 7, pp.31-35, 2011.

BIOGRAPHY

Pramod George Jose is a M.Sc. Computer Science student from Department of Computer Science, St. Xavier's College (Autonomous), Kolkata, has completed 2nd Semester; interested in Cryptography, Steganography, Computer Security and low level working of computer systems.

Soumick Chatterjee is a M.Sc. Computer Science student from Department of Computer Science, St. Xavier's College (Autonomous), Kolkata, has completed 2nd Semester; interested in Web Technologies, Cross-platform development, Cryptography, Steganography, Image Processing with successful experience in Tech Entrepreneurship.

Mayank Patodia is a MCA student from Department of Computer Applications, PES University, Bengaluru, has completed 2nd Semester; interested in Android Development, Web Technologies and UX Development.

Sneha Kabra is a MCA student from Department of Computer Applications, VIT University, Vellore, has completed 2nd Semester; interested in Steganography.

Dr. Asoke Nath is Associate Professor in the Department of Computer Science, St. Xavier's College (Autonomous), Kolkata. Dr. Nath is involved in research work in Cryptography and Network Security, Steganography, Green Computing, Mathematical modelling of social networks, Big data analytics, Cognitive Radio, Data Science, e-learning, MOOCs etc. He has published more than 191 papers in Journals and conference proceedings.

View publication stats