Security of Data using Reversible Texture Synthesis in Steganography with Hybrid Algorithm

Madhugeeta Verma, Poonam Dhamal
Savitribai Phule Pune University, GH Raisoni College of Engineering and Management, Pune, India
mmadhugeeta.verma@gmail.com, poonam.dhamal@raisoni.net

Abstract*— *In this paper, we are using reversible texture synthesis method for steganography with AES algorithm and SHA3-512 algorithm. The texture synthesis method generates a new synthetic texture. Here instead of using original cover image to hide data, we are using synthesized texture for hiding source texture with encrypted secret message.

Keywords— Steganography, Reversible Texture Synthesis, AES, SHA3-512.

I. Introduction

With the exponential growth and secret communication over the internet. we need a kind of security which could not be break by any unauthorized person. For this we use different methods like cryptography, steganography etc. steganography is the only method for information hiding. It conceal the message existence inside other medium like text, image, audio video and network [1]. There are large number of image steganography algorithm. Most of them use original cover image to hide the data. But if we embed more data into the image then it will result in distortion which can perceived by human eye. So in that case we have to compromise between capacity and quality. Texture synthesis is a method which will avoid this. In this paper, we are using reversible texture synthesis method for steganography. It generates a new synthetic texture which is same as source texure in local appearance but of arbitrary size. This synthetic texture can be generated in less time and also it is easy to implement. The key advantage of this technique is the quality and speed. A texture differ from an image [2]. As shown in Fig. 1, if a viewer is allowed to see image through movable window then he can observe the different part of image. But as window move through texture, the observable portion will appear similar. This characteristic is exploited in order to hide data in synthetic texture. In this paper, We use the texture synthesis process in steganography for concealing secret messages along with source texture.

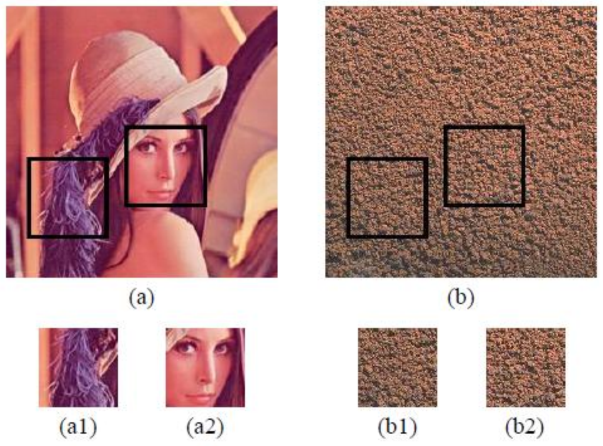


Fig. 1 How texture differ from image

II. Literature Survey

Texture synthesis is the latest technology in steganography. In this, source texture is synthesized using pixel or patch based algorithm in order to produce a synthesized texture.

In [2], author has present an algorithm for texture synthesis. It generates synthetic textures with quality same or better than the textures produced by earlier techniques. It runs 2 orders of magnitude faster. This algorithm generates textures with a deterministic searching process. In this synthesis process is done through tree-structured vector quantization.

In [3], author used a texture characterization method which is based on Fuzzy Texture Unit (FTU) for texture synthesis. The fuzzy texture characterization approach uses the vagueness introduced by different caption, noise and digitization processes to define texture unit so that texture synthesis can obtained.

In [4], author has used *image quilting* method for synthesizing a new image by combining together small patches of existing images. This method is very simple but even then it works good when applied to texture synthesis. Author has also extended this method to texture transfer with some very promising results.

In [5], author has used AES algorithm for encryption and hide it in skin region of image. Skin tone detection is done by using HSV (Hue, saturation, value) color space on input image. Embedding process is done by DWT (Discrete wavelet Transform).

In [6], author has used DES, AES and RSA algorithm for encryption along with steganographic algorithm. And at the end, their performance has been compared. Based on the result it is proved that AES is quite better than DES and RSA algorithm since it take less time for encryption and decryption and also buffer usage is less as compared to DES and RSA algorithm. RSA take more encryption time. So it is proved that AES is much better then DES and RSA algorithm.

III. Existing System

In the existing system, steganography is done through reversible texture synthesis. It generates a new synthetic texture which is same as source texture in local appearance but of different size. We use the texture synthesis method in steganography to hide secret data. Instead of using cover image for embedding data, algorithm hide the source texture image and secret data using the process of texture synthesis. Due to this we can take out the secret data and source texture from a stego synthetic texture. This approach has 3 different merits. First, this method provides the embedding capacity which is proportional to the size of stego texture image. Second, it is difficult to know for unauthorized person about this steganographic approach. Third, through reversible capability, it provides a method to recover source texture. Existing algorithm increases the embedding capacity and allows less distortion in cover image.

IV. Proposed System

In the proposed system three algorithms are combined in order to increase the capacity, quality and security. In the first stage we will encrypt the secret message with AES algorithm and also we will generate hash value using SHA3-512 algorithm. In the second stage we will use Reversible texture synthesis method. Our system will provide better security then previous techniques.

A) *AES*

Advanced Encryption Standard (AES) algorithm is not only for security but also for great speed. Both hardware and software implementation are faster still. It encrypts data blocks of 128 bits in 10,12 and 14 round depending on key size.

B) *SHA3-512*

SHA-3 is a family of sponge functions characterized by two parameters, the bit rate r and capacity c. The sum, r + c find the width of the SHA-3 function permutation that is used in the sponge construction and is restricted to a maximum value of 1600. Selection of r and c depends on the desired hash output value. Ex.: for a 256-bit hash output r = 1088 and c = 512 and for 512-bit hash output r = 576 and c = 1024 is selected. We will use hash output of 512 bits.

C) *Texture Synthesis*

The process of texture synthesis can be done through pixel or patch based algorithm. Pixel based method can result in blurring and garbage growing. Other problem with pixel based method is error propagation because one wrong synthesized pixel will affect the consecutive synthesis of other pixel. To remove this drawback, patch based method is introduced. It synthesizes one patch at a time so it is faster than pixel based[8].

In the proposed system basic unit for texture synthesis is patch. In Fig. 2, patch is shown. Suppose size of patch width is P_w and patch height is P_h. A patch contain 2 parts i.e. central (known as kernel region) and outer part (known as boundary region). Lets size of kernel width be K_w and Kernel height be K_h.

Suppose there is a source texture with size S_w and S_h. Now we can divide this source texture into number of kernel block of size K_w and K_h. We can use the indexing for kernel block as KB_i. Collection of kernel block can be represented as KB. Now we can expand kernel block with P_d from all side to generate a source patch. This expanding process of kernel block will overlap its neighboring block. Lets collection of source patches is denoted as SP and number of element in the SP is denoted as SP_n.

$$SP_n = \frac{S_w}{K_w} \times \frac{S_h}{K_h} \qquad \ldots\ldots(1)$$

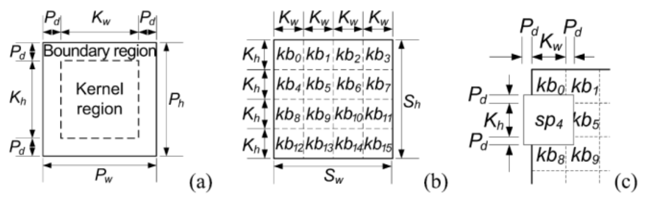


Fig. 2. Patch, Kernel block and source patch

When a patch is eligible to put next to already synthesized patch then that patch is called candidate patch. Candidate patch should be unique. If not then we can extract wrong secret data. In our system, there is a flag mechanism. If source texture has any same candidate patches then for the first patch flag is on and for rest flag is off. It defines the uniqueness in the candidate list.

Message Embedding procedure

Fig. 3 shows the 3 processes of message embedding procedure.

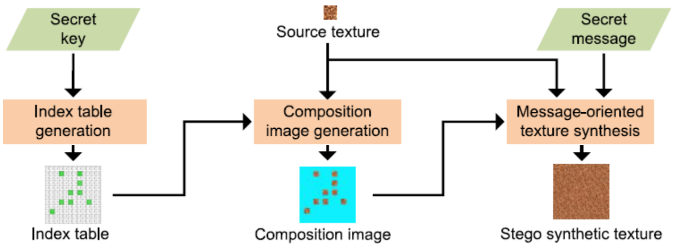


Fig. 3. Message embedding process

Index table generation process

In this process we generate an index table so that we can record the position of source patch in synthetic texture. Through this index table, we can access the synthetic texture and easily retrieve the source texture at the time of extraction.

Suppose dimension of index table be T_{pw} x T_{ph}. T_w and T_h are the dimension of synthetic texture. We can find the number of entries in the index table with formula

$$TP_n = T_{pw} \times T_{ph} = \left\lfloor \frac{(T_w - P_w)}{(P_w - P_d)} + 1 \right\rfloor \times \left\lfloor \frac{(T_h - P_h)}{(P_h - P_d)} + 1 \right\rfloor \qquad \ldots(2$$

Where TP_n is the total number of patches in stego synthetic texture. For the distribution of patches we avoid the positioning on border of synthetic texture. These border area are used by message oriented texture synthesis.

Fig. 4 shows the composition image in which 9 source patches are placed.

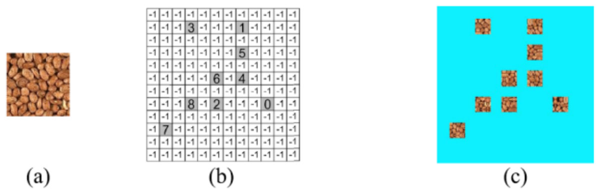


Fig. 4. An illustration of composition image; (a) the source texture (96 × 96), (b) the index table after patch distribution, (c) the composition image (488 × 488) by referring (a) and (b).

Initial value in the index table is -1 means table is blank. We can assign the value while distributing the source patch ID in synthetic texture. In Fig.4 b, we have 9 source patches and 135 locations are blank i.e. their value is -1. If there is non-negative value in the index table then it means it has some source patch ID. In the entries of -1, we can embed secret data in the message oriented texture synthesis.

Patch Composition Process

The second process of message embedding is to paste the source patches into a workbench in order to make a composition image. First, we take a blank image as a workbench where the size of the workbench is same as synthetic texture. By referring to the source patch IDs in the index table, we can paste the source patches into the workbench. During this process, if no overlapping of the source patches is met then directly we can paste the source patches into the workbench, as shown in Fig. 3(c). However, if overlaping occurs then we can use image quilting technique[4] to reduce the effect of overlapped area.

Message-Oriented Texture Synthesis Process

Now we have an index table and a composition image. We will hide secret message through the message-oriented texture synthesis to generate final stego synthetic texture. There are some differences between the conventional patch based texture synthesis and our proposed message-oriented texture synthesis. First, the result of the conventional texture synthesis is a pure synthetic texture where as our algorithm generates different synthetic texture. In our system, the source texture is converted into a number of source patches that is pasted in the large synthetic texture. And the large texture that has been generated is embedded with the secret message. The conventional texture synthesis algorithm has one "L-shape" overlapped area, whereas in our algorithm there are four different shapes of the overlapped area, as shown in Fig. 5.

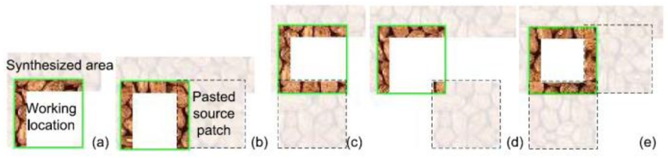
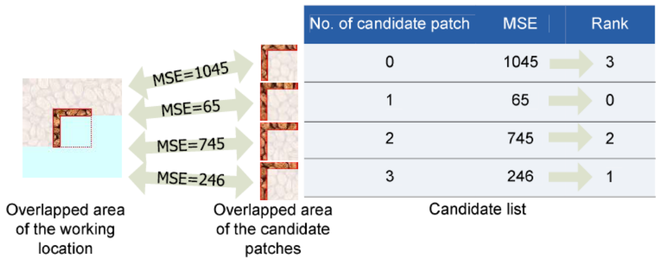


Fig. 5. Five different shapes, (a) to (e), of the overlapped area may occur during our message-oriented texture synthesis algorithm. The blank area inside the green square frame represents the working location which is going to be synthesized.


MSE=1045
MSE=65
MSE=745
MSE=246
Overlapped area of the working location
Overlapped area of the candidate patches
Candidate list

No. of candidate patch	MSE	Rank
0	1045	3
1	65	0
2	745	2
3	246	1

Fig. 6. The MSEs and rank of patches in the candidate list for the working location.

For every candidate patch in candidate list, one of the five shapes of overlapped area will occur. Mean square error

(MSE) of the overlapped region between the candidate patch and the synthesized area can measured. When all MSEs of the patches in the candidate list are calculated, we can rank these candidate patches according to their MSEs. Fig. 6 shows an example of a candidate list where 4 candidate patches have different MSEs. The candidate patch having smallest MSE shows that it is the most similar to the synthesized area in the working location. Once the ranks of all candidate patches are calculated, then we can select the candidate patch where its rank equals the decimal value of an *n*-bit secret data. So, a segment of the *n*-bit secret data can be concealed into the selected patch to be pasted into the working location. In our proposed system, we used a simple image quilting technique [4] to reduce the visual defect encountered in the overlapped area.

Capacity determination

The embedding capacity is important for data embedding scheme. Embedding capacity of the algorithm depend on the capacity of bits that can be concealed in each patch (BPP, bit per patch), and the number of patches which is embedded in the stego synthetic texture (*EPn*). Every patch can hide at least 1 bit of the secret data; thus, the lower bound of BPP will be one, and the maximal capacity in bits that can be concealed at each patch is the upper bound of BPP, as denoted by *BPPmax* . Our algorithm can offer total capacity (TC) which is shown in equation (3)which is the multiplciation of BPP and *EPn*.

$$TC = \text{BPP} \times EP_n = \text{BPP} \times (TP_n - SP_n) \qquad \ldots\ldots(3)$$

The number of the embeddable patches is the difference between the number of patches in the synthetic texture (*TPn)* and the number of source patches subdivided in the source texture (*SPn)*.

Source Texture Recovery, Message Extraction and Capacity determination

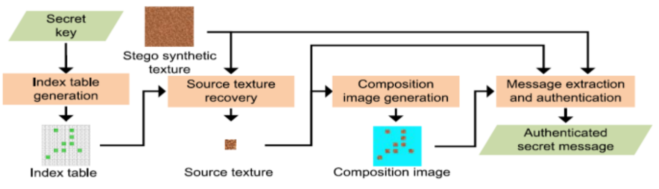


Fig. 7. The flowchart of the four-step message extracting procedure.

The message extraction at the receiver side involves generation of index table, retrieving the source texture, then texture synthesis is performed and lastly extracting and authenticating the secret data concealed in the stego synthetic texture. The extracting procedure has 4 steps, as shown in Fig. 7. With the secret key held in the receiver side, the same index table as in the embedding procedure can be generated. The next step is the recovery of source texture. Each kernel region of size $K_w \times K_h$ and its respective order with respect to source texture of size $S_w \times S_h$ can be retrieved by referring to the index table with dimensions $T_{pw} \times T_{ph}$. We can then arrange kernel blocks according to their order, thus source texture is recovered which will be exactly same as the source texture. In the third step, we generate the composition image to paste the source patches into a workbench to generate a composition image by referring to the index table. The composition image that is generated is same as one produced in the embedding procedure. The last step is the extraction of message and authentication process.

V. Experimental Results

The experimental results are implemented using Visual studio 2010. We test the system using different size of source texture. Total usable patch is small as compare to total number of patch. Also as the number of secret bit increases in the patch, quality of image decrease but it cannot be perceived by human eyes.

Synthetic texture size: 1008 x 1008			
S_w x S_h	Total carrier units	Usable Units	Bit per Patch
96 x 96	798035	29520	5
128 x 128	798025	60893	7
192 x 192	798025	72536	8

Table 1 : Total embedding capacity in bits our algorithm can provide when size of texture is 1008 x 1008.

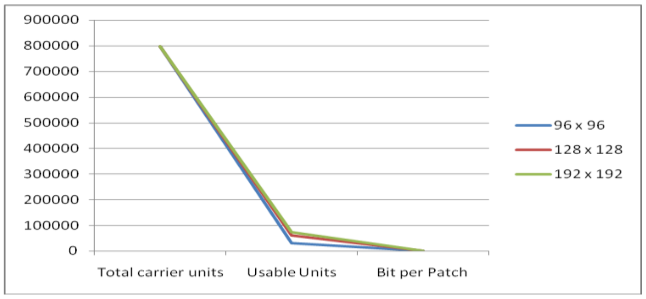
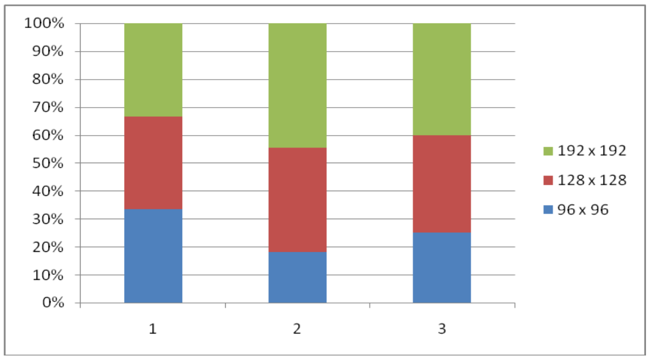


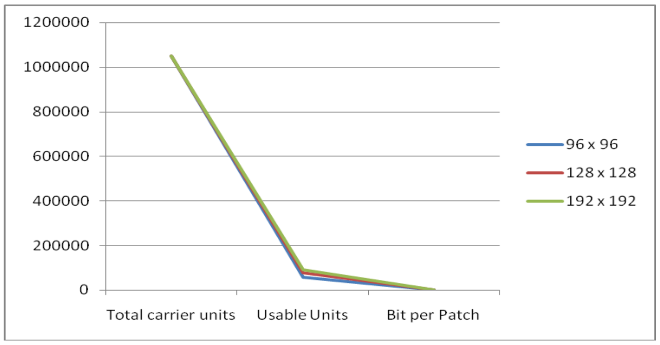
Fig 11.1: Line chart of embedding capacity when size of texture is 1008 x 1008.



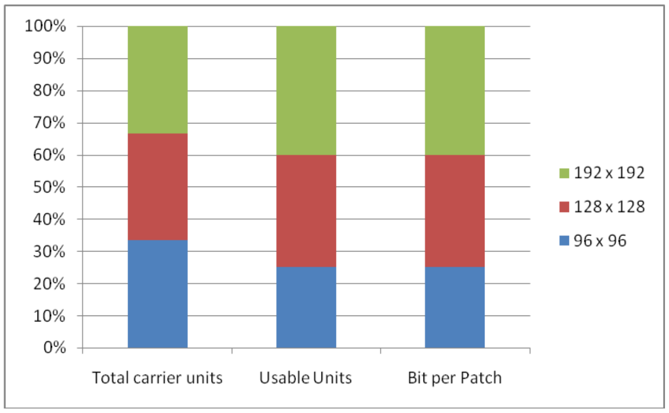
11.2: Column chart of embedding capacity when size of texture is 1008 x 1008.

Synthetic texture size: 1024 x 1024			
S_w x S_h	Total carrier units	Usable Units	Bit per Patch
96 x 96	1048576	55530	5
128 x 128	1048576	77742	7
192 x 192	1048576	88848	8

Table 2 : Total embedding capacity in bits our algorithm can provide when size of texture is 1024 x 1024.



11.3: Line chart of embedding capacity when size of texture is 1024 x 1024.



11.4: Column chart of embedding capacity when size of texture is 1024 x 1024.

VI.Conclusion

This paper proposes reversible steganographic algorithm using texture synthesis with AES and SHA3-512 algorithm. These all methods are used in our system in order to increase capacity, quality and security. This algorithm uses original source texture, and produces a large stego synthetic texture concealing encrypted secret messages.

Acknowledgement

This research was supported by prof. Poonam Dhamal. I would like to thank her for help and encouragement. I thank my colleagues from GH Raisoni college of engineering and management who provided insight and expertise that greatly assisted the research

References

i. *Mehdi Hussain and Mureed Hussain, "A survey of image steganography techniques," vol. 54, May 2013.*

ii. *Li-Yi Wei, Marc Levoy "Fast Texture Synthesis using Tree-structured Vector Quantization, Stanford University.*

iii. *G.Venkata Rami Reddy, Dr.V Vijaya Kumar, B. Sujatha, A novel Texture Synthesis Algorithm using patch Matching by Fuzzy Texture Unit, Vol. 4 No. 01 January 2012*

iv. *Alexei A. Efros, William T. Freeman, Image quilting for Texture Synthesis and Transfer, University of California, Berkeley, 2Mitsubishi Electric Research Laboratories*

v. *Manoj gowtham.G.V,Senthur .T, Sivasankaran .M, Vikram .M, Bharatha Sreeja .G, "AES based steganography" , International Journal of Application or Innovation in Engineering & Management (IJAIEM), Volume 2, Issue 1, January 2013.*

vi. *B. Padmavathi, S. Ranjitha Kumari,* **"A** *Survey on Performance Analysis of DES, AES and RSA Algorithm along with LSB Substitution Technique",* ***Volume 2 Issue 4, April 2013***

vii. *Ekta Dagar, Sunny Dagar, "LSB Based Image steganography using X-Box Mapping, 2014 IEEE*

viii. *Kuo-Chen Wu and Chung-Ming Wang, "Steganography Using Reversible Texture Synthesis," IEEE transactions on image processing, vol. 24, no. 1, january 2015*

ix. *Ali Abdulgader, Mahamod Ismail, Nasharuddin Zainal, Tarik Idbeaa "Enhancement of AES algorithm based on chaotic map and shift operation for image encrytion", Journal of Theoretical and Applied Information Technology10th January 2015. Vol.71 No.1*

x. *Muzaffar Rao, Thomas Newe and Ian Grout ,"Secure Hash Algorithm-3(SHA-3) implementation on Xilinx FPGAs, Suitable for IoT Applications", Sep. 2-4, 2014, Liverpool, UK*

Authors bibleography

	Madhugeeta Verma received her Bachelor's degree in Information technology from RITEE, Raipur, Pt Ravi shanker University, Chattisgarh, India .Now, she is pursuing her M.E degree in Computer Engg. from GHRCEM College, Pune University, Pune, India.
	Poonam Dhamal received her Bachelor's degree in Computer Engg. from VPCOE, Baramati, Pune University, Pune, India .Now, she is pursuing her M.E degree in Computer Engg. from M.A.E., Alandi, Pune University, Pune, India. She is working as Lecturer in GHRCEM College, Wagholi, Pune, India. Her research areas are Computer Network, MANET, VANET and Wireless Networks.