ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

TO PROVIDE IMAGE DATA SECURITY USING EFFICIENT ENCRYPTION THEN COMPRESSION TECHNIQUE

Salunke Vaishali, Varpe Amit , Hinge Ashwini Co-author Prof.Gavale.S.S.
Department Of Computer Engg,Jaihind Collage Of Engg, India

ABSTRACT

The image encryption has to be happened before the compression of image it shown after performing number of practical approaches. But there is a problem that how to design an image encryption and then compression algorithms so that the encrypted image can be efficiently compress. The encryption-then-compression (ETC) systems of image are designed in this paper and in that there are considered both lossless and lossy compression. There use an AES algorithm in proposed system in that image encryption scheme operated which is to be able to provide high level of security. And with the AES, for better compression of encrypted images arithmetic approach is more efficient. The Huffman algorithm based approach is somewhat inconvenient In terms of compression efficiency, than the advanced lossless/lossy coders of image, which take intrinsic or unencrypted images as inputs. Maximum of the existing ETC results dispose affect penalty on the compression efficiency in comparison.

KEYWORDS: Compression of encrypted image, encrypted domain signal processing.

INTRODUCTION

Let us consider an example in which, a content owner or sender Alice wants to securely and efficiently send image I via an untrusted *channel* provider Charlie, to a recipient Bob. This could be happen as follows. Alice first compress the image, then he encrypt into with the help of Encryption Function, where is the secrete key, which is shown in Figure 1. After performing encryption, the encrypted data is send to Charlie. Then Charlie simply forwards this data to Bob. Then Bob first decrypt data by decryption and then decompress the decrypted data using decompression which get original image.

The sequence of applying the compression and encryption required to be reversed in some other conditions Even if the above Compression-then-Encryption (CTE) example satisfies the requirements in many secure transmission situations. Even if the data owner, Alice is every time interested in protecting the secrecy of the image data through encryption method. Then, Alice has no need to compress her data, and hence, before encrypting the data, to run a compression algorithm, will not use her limited computational resources. This will be true for the use of resource-deprived mobile device. The channel provider Charlie

Compress the data if load on the channel increases to increase the network utilization. So that data which is already compressed which again compress by channel provider. So that it will be better if the operation of compression performed by the channel provider who has copious computational resources. The compression has to be performed on the encrypted data is the big challenge with the Encryption-then-Compression (ETC) framework so that network provider Charlie does not provide cannot access the secrete key.

PROBLEM STATEMENT

Designing an efficient Encryption then Compression (ETC) system using Steganography and NSP.

MOTIVATION

A digital image obtained by sampling and quantizing a continuous tone picture requires an enormous storage. It would take almost 4 minutes to transmit such an image over a 28.8 Kbps modem. To reduce the amount of data

Required for representing sampled digital images is the purpose for image compression and therefore reduce the cost for storage and transmission.

LITERATURE SURVEY

In recent year, the possibility has been retrieving increasing consideration to deal with encrypted signals directly in the encrypted domain [2]-[6]. For Charlie it seems to be impractical to compress the encrypted data at the first stage, whereas to enable a traditional compressor, no signal structure can be

exploited. Without compromising either the compression efficiency or the information-theoretic security [7] showed by Johnson et, through the use of coding the stream cipher encrypted data is compressible. As well as theoretical findings, [7] also proposed practical algorithms to lossless compress the encrypted binary images. After that when the statistics of basic source is anonymous and the memory have to sources [8], [9], on encrypted images compression explored the problem about the encrypted images compression. In various bit-planes by assigning LDPC codes and the inter/intra correlation There are presented different methods for encrypted grayscale/color images lossless compression [11]. Then, Makur and Kumar applied the approach of [7] to the domain of prediction error and obtained improved performance of lossless compression on the grayscale/color images which are encrypted in [12]. For the compression of images which are lossless encrypted grayscale/color stream cipher images Liu et. al developed a onward method in [13]. More recently, on with extended block ciphers encryption data compression describe the framework for the compression of block cipher encrypted data [10].

For the purpose of lossy compression achieving, encrypted data's higher compression ratios was also prepared [14]–[20]. Zhang et. Al proposed the encrypted image's a scalable lossy coding framework via a multi-resolution construction [14] is proposed by. To encrypted images compression emerge from linear encryption, a compressive sensing (CS) mechanism was utilized in [15]. From the compressed and encrypted data for evaluating the authentic image, a modified basis quest algorithm can be applied then. For encrypting compressed images another CS-based approach was reported in [16]. Moreover, in the transform domain, via pixel-domain permutation, Zhang designed an image encryption scheme and demonstrated that by eliminating

The extremely rough and fine content of coefficients [17], there is efficiently compressed the encrypted file. Recently, for encrypted images through multi-layer decomposition a new compression approach advised by Zhang et. al. [18]. The blind compression expansions of encrypted videos were advanced in [19], [20].
After recent years studied, with the comparison of state-of-the-art lossless/lossy image and video coders that need inputs which are unencrypted, still there decline the existing ETC systems significantly short in the performance of compression. The dominant target of this work is the practical design of a couple encryption and compression schemes of image, in such a way that the efficiency of encrypted images compression with the compressing their original counterparts which are unencrypted is essentially same, Meanwhile, reasonably high level of security needs to be ensured. If not otherwise specified, for that 8-bit grayscale images are assumed. Both lossless and lossy compression of encrypted images will be considered. Emphatically there present, over the prediction error domain an image encryption approach conducted which is depends on permutation. For compressing the encrypted information efficiently, then a context-adaptive arithmetic coding (AC) is shown. Due to the i.i.d property of the sequence of prediction error, there is cost of compression is very negligible (< 0.1% coding loss for lossless case) is popularized. Moreover, honestly there obtained high level of security, due to the high prediction error sequence's high sensitivity opposite to the disruptions.

EXISTING SYSTEM

In many secure transmission methods above stated example of Compression-then-Encryption (CTE) serves the requisites, the order of obtaining the compression and encryption required to be reversed in some other order of conditions. As the owner of data, Alice is always concerned with preserving the consistency of the image data by using module of encryption. After that, Alice has not requirement of compressing her data, and due to that, to execute a compression method before data encrypting, will not use her limited usable resources.

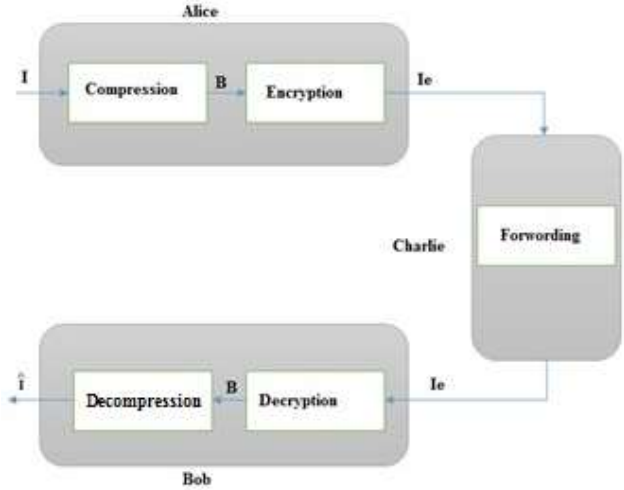

Figure 1. Traditional Compression-then-Encryption (CTE) system.

This will be acceptable for making the use of resource- disadvantaged mobile device. If pressure on the system increases the service provider Charlie

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

conduce the data which may increase the network application. Due to that information which is already once compressed which double being compress by service provider. So that it will be good if the process of computation done by the service provider who has considerable computational resources. he compression has to be processed on the encrypted data is the big issue with the Encryption-then-Compression (ETC) methodology is that t, so that network service provider Charlie will not serve and cannot operate the secrete key. In Figure 2 describe the ETC system is.

PROPOSED ETC SYSTEM

To encrypt the data we are using Advanced Encryption Standard (AES) algorithm and then by using Huffman algorithm this data is compressed. A Permutation based image encryption approach for secure and efficient image security. Primary focus is on practical design of a pair of encryption and compression methods in that the encrypted image compression and compressing the unencrypted original image is equally efficient. Due to high awareness of AES algorithm, reasonably high level of security could be retained. Following figure shows the entire architecture of the proposed system.

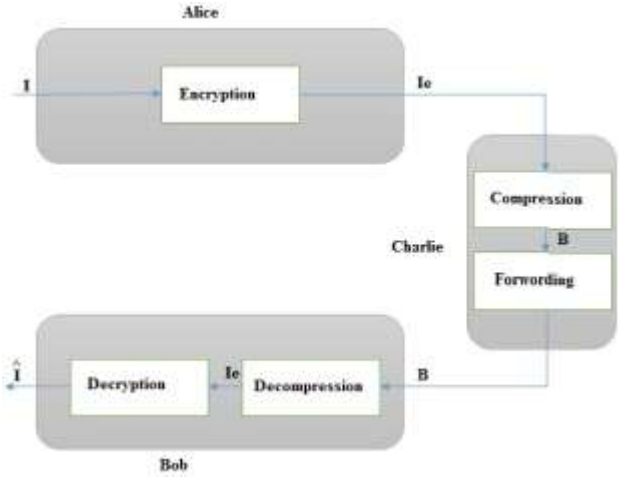

Figure 2. Encryption-then-Compression (ETC) system.

SYSTEM ARCHITECTURE

Proposed system architecture consist of steganography in which text is hide in image and image encryption at the sender side and data compression at the network side. Same process of data decompression and data decryption are performed by the receiver side. This will achieve the security and also low data uses for the sending that data over the internet.

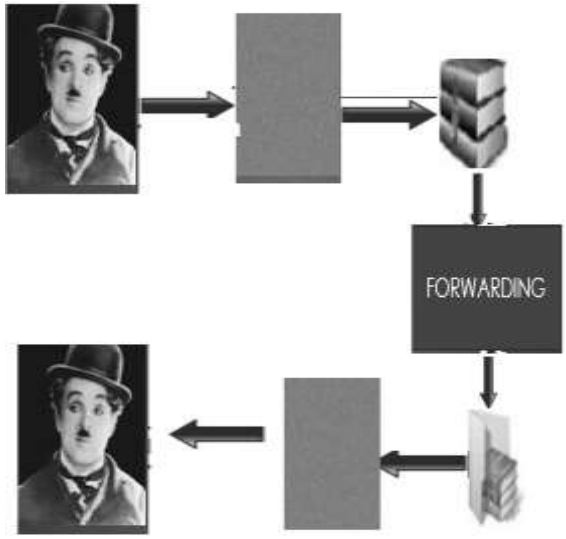

Figure 3. Overview of the system.

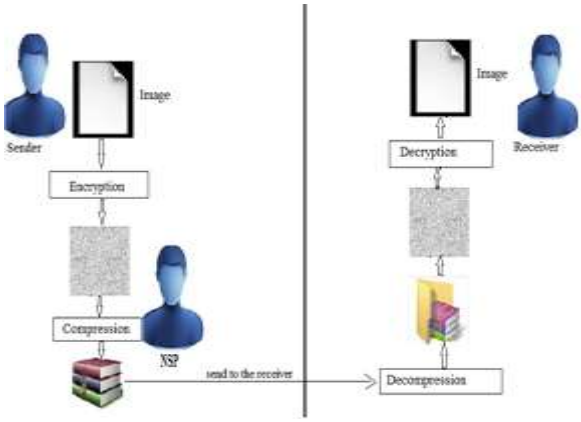

Figure 4 System Architecture

The proposed system architecture shown in Figure 4. It consist of sender, channel provider and receiver. In this system for hiding the data in the image I use the concept of Steganography. If sender want to send some data to the receiver then sender first hide the data in image file and then encrypt this file by using AES algorithm. After encrypting data successfully sender send this data to the network provider. The mediator between the sender and receiver is the network provider. If the traffic on the channel increases or the data send by sender is large then channel provider compress the data which is sent by sender by using Huffman coding algorithm, for minimizing the load of the channel. The channel provider send this compressed data to the receiver after performing compression operation on the encrypted data. If the traffic on the channel is low or data send by sender is minimum, then channel provider simply send this data to the receiver? At the

receiver side, the receivers perform either one or two operations which are depending on the data which is came from channel provider. If the data sent by channel provider is compressed data then receiver decompress this data by Huffman coding algorithm. After decompressing the data, the receiver perform the decryption operation on the decompressed data. After performing the decryption operation, the hidden text from image file is extracted and as a result get the original data which is sent by sender. But, if the channel provider send data to the receiver without compressing it to the receiver, then at receiver side, receiver directly decrypt it without performing the decompression operation and get the original data as a result which is same as data at sender side which is before encryption.

D. Overview of algorithms

i) AES Algorithm

AES is a new cryptographic algorithm. To protect electronic data this algorithm is very helpful. AES is a block cipher of symmetric-key which are use the keys of 128, 192, and 256 bits, and encrypts as well as decrypts contents in blocks of 128 bits. AES use a keys pair, the same key use by the symmetric-key ciphers to encryption and decryption of data. The same number of bits have the data which encrypted which obtained by block ciphers that the input data had. A loop structure use by Iterative ciphers that permutations as well as substitutions of the input data performs repeatedly.

The AES algorithm is depends on permutations and substitutions. The rearrangements of data is Permutations, and substitutions is the replacement of the data i.e. replace one unit of data with another. Using several different techniques, AES performs permutations and substitutions.

The number of repetitions of transformation rounds are represents by The AES cipher key size which performs conversion the input, which is known as the plaintext, into the final output, which is known as the cipher text. Following there are shows the number of cycles of repetition:

- For 128-bit keys 10 cycles of repetition
- For 192-bit keys 12 cycles of repetition.
- For 256-bit keys 14 cycles of repetition.

Several processing steps are consist by each round, each containing four similar but which are different stages. In those, one that based on the key encryption itself. To transform cipher text back into the original plaintext, a set of reverse rounds are applied using the same encryption key.

Description

1. Key Expansions

For each round AES needs a different 128-bit block of round key also one more.

2. Initial Round

AddRoundKey—with a block of the round key, each byte of the state is combined using bitwise xor.

3. Rounds

- Sub Bytes—in this step each byte is replaced with another byte.
- Shift Rows— for a certain number of steps, the state's last three rows are moved cyclically.
- Mix Columns— on the columns of the state a mixing operation operates, in each column combining the four bytes.
- AddRoundKey

4. Final Round (no Mix Columns)

- Sub Bytes
- Shift Rows
- AddRoundKey.

The Sub Bytes step

In the Sub Bytes step, with a Sub Byte each byte in the state matrix is replaced using an 8-bit substitution box. In the cipher, the nonlinearity provided by this operation. From the multiplicative inverse over GF (28) the S-box used is derived, known to have good non-linearity properties. By combining the inverse function The S-box is constructed with an inverse transformation to avoid attacks which is depends on properties of simple algebraic. The S-box and also any opposite fixed points is also chosen for to avoid of any fixed points, for performing the decryption, Sub Bytes step is used inversely, for that, before obtaining the inverse of multiplication, first taking the affine transformation.

The Shift Rows step

On the rows of the state, the Shift Rows step operates; in each row it shifts the bytes cyclically by a certain offset. for AES The first row is left unchanged. Each byte is shifted one to the left of the second row. Similarly, by the offsets of two and three the third and fourth rows are shifted. The shifting pattern is the same, for blocks of sizes 128 bits and 192 bits, by n-1 bytes, circularly row n is shifted left. In this way, from each column of the input stat, the output state's each column of the Shift Rows step is composed of bytes e. The first row is unchanged, for a 256-bit block and 1 byte, 3 bytes and 4 bytes needs for the shifting for the second, third and fourth row respectively. When used with a 256-bit block, as AES doesn't use 256-bit blocks, for the Rijndael cipher this change applies.The case like to avoid the columns being linearly independent, is the importance of this step. The degeneration of AES into four individual ciphers of block.

The Mix Columns step
With a fixed polynomial in the Mix Columns step each column of the state is multiplied. In the Mix Columns step, combined Using an invertible linear transformation, every column of the state are have four bytes. As input needs for The Mix Columns function is four bytes and there also four bytes of output, where all four output bytes affects by each input byte. Mix Columns provides diffusion in the cipher, together with Shift Rows.

The AddRoundKey step
In this step, with the state there are combination of the sub key is combined. From the main key a sub key is derived, for every round with the help of Rijndael's key schedule; there are same size for every sub key as the state. Using bitwise XOR, by combining every byte of the state with the corresponding byte of the sub key, the sub key is added.

i) Huffman Algorithm
Compression
This technique works by creating a binary tree of nodes. The binary tree can be stored in a regular array, the size of which is based on the number of symbols. There will be either a leaf node or an internal node, a node. Initially, all nodes are leaf nodes the symbol itself, the weight of the symbol and a link to a parent node which is optional, for the reading of the code starting from a leaf node which made it easy. The symbol weight, links contains internal nodes to two child nodes and to a parent node there are the optional link. As a commonly, bit '0' represents the left child and bit '1' represents the right child. A complete tree has up to leaf nodes and internal nodes. The most optimal code lengths by omitting unused symbols produces A Huffman tree.
The probabilities of the symbol Containing are represented by them, the process begins with the leaf nodes, then a new node whose children are the 2 nodes is created with smallest probability, such that to the addition of the children's probability, the new node's probability is equal. The previous 2 nodes merged into one node, then new node being now generated. Until only one node remains this procedure is continuously performs, which produce Huffman tree.

Decompression
The process of decompression is simply by bisecting the Huffman tree node by node as each bit is read from the input stream, conversion of the stream of prefix codes to separate byte values. Before this can take place, the Huffman tree must be reconstructed. The character frequencies are fairly predictable, in the simplest case. On each compression cycle, at the rate of at least some measure of efficiency of compression, the tree can be reconstructed and statistically adjusted and thus reused every time. Otherwise, to reconstruct the tree the information must be sent a priori. The frequency count of each character might be to prepend by naive approach to the compression stream.
Development of Huffman Compression and Decompression
Algorithm
Step1- Read the image.
Step2- Conversion into grey level image from given color image.
Step3- After that, for finding symbols, calling a function (i.e. non-repeated pixel value).
Step4- Calling a function for calculating the each symbol‘s probability.
Step5- In decreasing order the probability of symbols are arranged and there combine the lower probabilities and continued this step until there left only two probabilities and according to rule codes are assigned that: there will shorter length code for the highest probable symbol.
Step6- Then there performed Huffman compression
Step7- The reconstruction of original image i.e. by using Huffman decoding decompression is done.
Step8- Equivalent to the encoding tree there generate a tree.
Step9- Then Read the input character wise
Step10-Then, in the leaf output the character encode and return to the root, and until all the codes of corresponding symbols are known extend the step9.

ADVANTAGES

- The sender does only encryption and thereby keeps the task of compression over to channel provider.
- This makes the system efficient and robust.
- System complexity reduces as the work of Encryption and Compression is distributed to two ends for maximum resource utilization.
- High level of Security need is ensured.

FLOW CHART

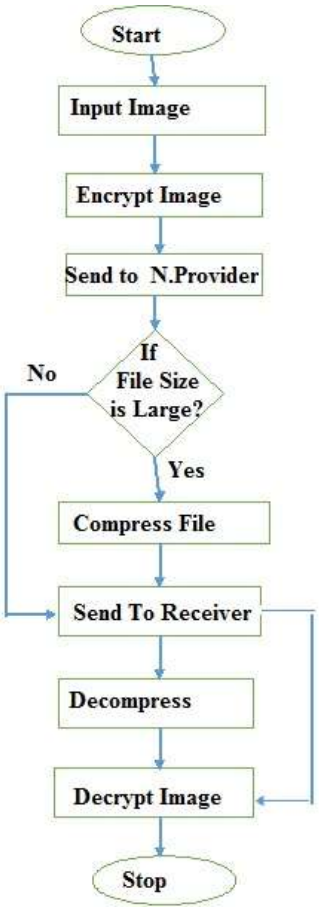

Figure 5. System flow chart

SYSTEM DESIGN

Mathematical Models

i) User Module:
Set (P) = {p0, p1, p2, p3, p4, p5, p6, p7}
p0=Select data for send.
p1=Select Image for send.
p2=Apply Steganography encryption.
p3=Apply Encryption algorithm.
p4=Send encrypted image
p5=receive image
p6=apply decompression
p7=apply steganography decryption

ii) Sending Image Module:
Set (K) = {p4, k0, k1, k2, k3}
k0= take encrypted image.
k1= apply compression algorithm
k2=.forward the compressed to receiver.
k3= update the status of message.

iii) Key Module:
Set (D) = {p2, p3, p4, d0, d1, d2, d3}
d0=.select encryption algorithm.
d1=select key parameters.
d2=encrypt the input image.
d3= decrypt the input image.

iv) Receive Image Module:
Set(C) = {p5, p6, d0, d1, d3, p7, c0}
C0= get confidential data from image.

Union and Intersection of project:-
Set (P) = {p0, p1, p2, p3, p4, p5, p6, p7}
Set (K) = {p4, k0, k1, k2, k3}
Set (D) = {p2, p3, p4, d0, d1, d2, d3}
Set (C) = {p5, p6, d0, d1, d3, p7, c0}

Vein Diagram:-
i) (P ∩ D)= {p2, p3, p4}

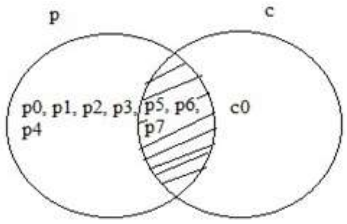

Figure: P ∩ D

ii) (P∩C) = {p5, p6, p7}

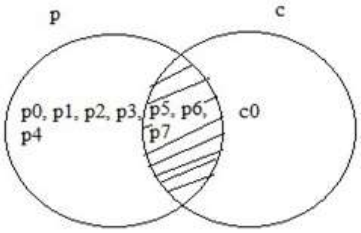

Figure: P ∩C

iii) (D ∩ C) = {d0, d1, d3}

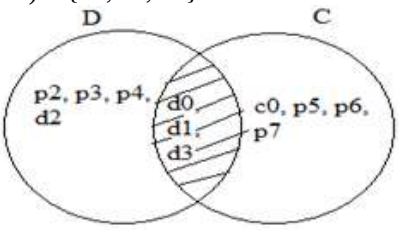

Figure: D ∩ C

COMPARISON OF EXISTING AND PROPOSED SYSTEM

Existing System	Proposed System
In this system, the sender performs first compression then encryption operation.	In this system, the sender performs only encryption operation.
The service provider simply forward the data to the receiver.	The service provider compress the encrypted data.
Receiver first decrypt the encrypted file then decompress it and get the original data.	The receiver first decompress the compressed data then decrypt it and get the original data.
Image Encryption and Decryption via prediction Error Clustering and Random Permutation	Image Encryption and Decryption via Advanced Encryption Standard (AES) Algorithm.
Compression and Decompression of Encrypted Image via Adaptive AC	Compression and Decompression of Encrypted Image Via Huffman Algorithm.

CONCLUSION

We can conclude that, we have designed an efficient image Encryption-then-Compression (ETC) system. The image encryption and decryption has been achieved via AES algorithm within the proposed framework. Highly efficient compression and decompression of the data which is encrypted has then been realized by using Huffman Algorithm. Both theoretical and experimental results have shown that reasonably high level of security has been retained.

ACKNOWLEDGEMENTS

Author would like to take this opportunity to express our profound gratitude and deep regard to my (Project Guide name), for his exemplary guidance, valuable feedback and constant encouragement throughout the duration of the project. His valuable suggestions were of immense help throughout my project work. His perceptive criticism kept me working to make this project in a much better way. Working under him was an extremely knowledgeable experience for me.

REFERENCES

1. J. Zhou, X. Liu, and O. C. Au, "On the design of an efficient encryptionthen-compression system," in *Proc. ICASSP*, 2013, pp. 2872–2876.
2. T. Bianchi, A. Piva, and M. Barni, "On the implementation of the discrete Fourier transform in the encrypted domain," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 1, pp. 86–97, Mar. 2009.
3. T. Bianchi, A. Piva, and M. Barni, "Encrypted domain DCT based on homomorphic cryptosystems," *EURASIP J. Inf. Security*, 2009 Article ID 716357.
4. T. Bianchi, A. Piva, and M. Barni, "Composite signal representation for fast and storage-efficient processing of encrypted signals," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 180–187, Mar. 2010.
5. M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-preserving ECG classification with branching programs and neural networks," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 2, pp. 452–468, Jun. 2011.
6. Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 1053–1066, Jun. 2012.
7. M. Johnson, P. Ishwar, V. M. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2992–3006, Oct. 2004.
8. D. Schonberg, S. C. Draper, and K. Ramchandran, "On blind compression of encrypted correlated data approaching the source entropy rate," in *Proc. 43rd Annu. Allerton Conf.*, 2005, pp. 1–3.
9. D. Schonberg, S. C. Draper, and K. Ramchandran, "On compression of encrypted images," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2006, pp. 269–272.
10. D. Klinc, C. Hazay, A. Jagmohan, H. Krawczyk, and T. Rabin, "On compression of data encrypted with block ciphers," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6989–7001, Nov. 2012.
11. R. Lazzeretti and M. Barni, "Lossless compression of encrypted greylevel and

color images," in *Proc. 16th Eur. Signal Process. Conf.*, Aug. 2008, pp. 1–5
12. A. Kumar and A. Makur, "Distributed source coding based encryption and lossless compression of gray scale and color images," in *Proc. MMSP*, 2008, pp. 760–764.
13. W. Liu, W. J. Zeng, L. Dong, and Q. M. Yao, "Efficient compression of encrypted grayscale images," *IEEE Trans. Imag. Process.*, vol. 19, no. 4, pp. 1097–1102, Apr. 2010.
14. X. Zhang, G. Feng, Y. Ren, and Z. Qian, "Scalable coding of encrypted images," *IEEE Trans. Imag. Process.*, vol. 21, no. 6, pp. 3108–3114, Jun. 2012
15. A. Kumar and A. Makur, "Lossy compression of encrypted image by compressing sensing technique," in *Proc. IEEE Region 10 Conf. TENCON*, Jan. 2009, pp. 1–6.
16. X. Zhang, Y. L. Ren, G. R. Feng, and Z. X. Qian, "Compressing encrypted image using compressive sensing," in *Proc. IEEE 7th IIHMSP*, Oct. 2011, pp. 222–225.
17. X. Zhang, "Lossy compression and iterative recobstruction for encrypted image," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 1, pp. 53–58, Mar. 2011
18. X. Zhang, G. Sun, L. Shen, and C. Qin, "Compression of encrypted images with multilayer decomposition," *Multimed. Tools Appl.*, vol. 78, no. 3, pp. 1–13, Feb. 2013.
19. D. Schonberg, S. C. Draper, C. Yeo, and K. Ramchandran, "Toward compression of encrypted images and video sequences," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 4, pp. 749–762, Dec. 2008.
20. Q. M. Yao, W. J. Zeng, and W. Liu, "Multi-resolution based hybrid spatiotemporal compression of encrypted videos," in *Proc. ICASSP*, Apr. 2009, pp. 725–728.
21. X. Wu and N. Memon, "Context-based, adaptive, lossless image codec," *IEEE Trans. Commun.*, vol. 45, no. 4, pp. 437–444, Apr. 1997.
22. M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Trans. Imag. Process.*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.