IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 4, April 2015
ISSN: 2394 - 6598
www.ijetie.org

Privacy-Assured Outsourcing of Image Reconstruction Service in Cloud

Dheeraj Bhalerao
IT Department,
GSMCOE,University of pune
dheeraj.bhalerao@gmail.com

Ganesh Mhetre
IT Department,
GSMCOE,University of pune
ganeshmhetre21@gmail.com

Hitesh Salunkhe
IT Department,
GSMCOE,University of pune
hitesh.salunkhe1@gmail.com

Anuja Divekar
Assistant Professor
IT Department,
GSMCOE,University of pune
anujadivekar13@gmail.com

ABSTRACT

Large-scale image datasets are being exponentially generated today. Along with such data explosion is the fastgrowing trend to outsource the image management systems to the cloud for its abundant computing resources and benefits. However, how to protect the sensitive data while enabling outsourced image services becomes a major concern. To address these challenges, we propose OIRS[1], a novel outsourced image recovery service architecture, which exploits different domain technologies and takes security, efficiency, and design complexity into consideration from the very beginning of the service flow. Specifically, we choose to design OIRS under the PCIF framework, it is an efficient algorithm for compressing true color images is proposed.The technique uses a combination of simple and computationally cheap operations. The three main steps consist of predictive image filtering, decomposition of data, and data compression through the use of run length encoding, Huffman coding and grouping the values into polyominoes.The result is a practical scheme that achieves good compression while providing fast decompression. Besides, in OIRS, data users can harness the cloud to securely reconstruct images without revealing information from either the compressed image samples or the underlying image content. We use AES algorithm for encryption of image to securely store the image on the cloud.

General Terms

PCIF, OIRS, AES, cloud computing, RLE and Huffman codes

.Keywords

Standard filtering, color filtering, remapping,

1. INTRODUCTION

The Polyomino Compressed Image Format (PCIF, formerly PCF) [2] is a new lossless compressed image format developed by Stefano Brocchi. The goal of the development of the PCIF algorithm is to create an efficient image compressing engine using techniques that do not compromise time performances. Even if most image formats actually imply lossy compression, there can be cases where lossless image compression can be recommandable or even mandatory, as explained here. Actually, the most popular format for lossless image compression is the Portable Network Graphics (PNG) format. Other formats for general-purpose compression such as ZIP, but also higher-performance algorithms based on PPMD or BWT (used for example in the newest winzip versions or in the open-source 7 zip program) usually give worst results as they are not specialyzed for image data. An exception is the RAR algorithm that gives results comparable the ones of PNG. Another format that is rising is the Jpeg2000 standard, offering also a high-performance lossless compression mode. Finally, a new proposal by the Jpeg commitee for lossless image compression is the JPEG-LS standard.The new Polyomino Compressed Image Format achieves globally the **best compression results**in respect to all formats mentioned, with results very close to the JPEG-LS format; benchmarks have been done on two popular image sets (the Waterloo color image set and the Kodak image set), on another image set proposed by the author and on a fourth set proposed inimagecompression.info. A more accurate analysis of results is in the benchmarks

section. Shortly, in comparison to the JP2 lossless files, the PCIF compressed images are slightly bigger regarding photographical images but much smaller in computer generated or high edge images. The PNG files are smaller for very 'regular' images with wide uniform color areas, but most of the PNG files result to be from 25% to 40% bigger than PCIF files.The PCIF algorithm uses a combination of techniques to obtain a good lossless image compression ratio without compromising time performances, as reversible filtering and huffman coding. A new tecnique is also intruduced to group points in convenently codable shapes calles polyominoes. A complete description of the algorithm with an example can be found in thealgorithm section. The PCIF algorithm is addressed to be used for lossless compression of **true color** images. This is important because often this is not considered in research algorithms. Often algorithms that compress true color images just do a color transform of the three color layers and then compress the three obtained channels indipendently. The PCIF algorithm obtains a major advantage from color correlation because instead of applying one only color tranformation to all the image it searches a good tranform for every single 8x8 pixel zone, seeking a tranform close to optimality on a **local basis**.The software realizing the compression and decompression algorithm is available on thedownload section. Actually the software is **free for personal use**; anyway before downloading any file you must carefully read and accept the agreement. All the program has been realized in the Java language, allowing a very easy web integration, as demostrated by the PCIF applet.

2. RELATED WORK

Compressed sensing [3], [4], [5] is a recent data sensing and reconstruction framework well-known for its simplicity of unifying the traditional sampling and compression for data acquisition. Along that line of research, one recent work [6] by Divekar et al. proposed to leverage compressed sensing to compress the storage of correlated image datasets. The idea is to store the compressed image samples instead of the whole image, either in compressed or uncompressed format, on storage servers. Their results show that storing compressed samples offers about 50% storage reduction compared to storing the original image in uncompressed format or other data application scenarios where data compression may not be done. But their work does not consider security in mind, which is an indispensable design requirement in OIRS. In fact, compared to that only focuses on storage reduction, our proposed OIRS aims to achieve a much more ambitious goal, which is an outsourced image service platform and takes into consideration of security, efficiency, effectiveness, and complexity from the very beginning of the service flow.This privacy-preserving image recovery service in OIRS that we propose to explore is also akin to the literature of secure computation outsourcing [7], [8], which aims to protects both input and output privacy of the outsourced computations. Another existing list of work that loosely relates to (but is also significantly different from) our work is secure multiparty computation (SMC). Firstly introduced by Yao [32] and later extended by Goldreich et al. [19] and others. SMC allows two or more parties to jointly compute some general function while hiding their inputs to each other. However, schemes in the context of SMC usually impose comparable computation burden on each involved parties, which is undesirable when applied to OIRS model. In short, practically efficient mechanisms with immediate practices for secure image recovery service outsourcing in cloud are still missing.

3. PROPSED SYSTEM

The basic service model in the OIRS architecture includes the following: At first, data owner acquires raw image data, in the form of compressed image samples, from the physical world under different imaging application contexts. To reduce the local storage and maintenance overhead, data owner later outsources the raw image samples to the cloud for storage and processing. The cloud will on-demand reconstruct the images from those samples upon receiving the requests from the users. In our model, data users are assumed to possess mobile devices with only limited computational resources. Owner Process Sample y Encrypted Image Recovery over {y*} Encrypt Encrypted y* Recovered f Decrypt Output f* Secret K Cloud Servers Compressed Sparse signal f Sensing User Process Metadata FIGURE 1. The OIRS architecture in public cloud. Fig. 1 demonstrates the basic message flow in OIRS. Let f and y be the signal and its compressed samples to be captured by the data owner . For privacy protection, data owner in OIRS will not outsource y directly. Instead, he outsources an encrypted version $y*$ of y and some associated metadata to cloud. Next, the cloud reconstructs an output $f*$ directly over the encrypted $y*$ and sends $f*$ to data users. Finally, the user obtains f by decrypting $f*$. We leave the management and sharing of the secret keying material K between the data owner and users in our detailed decryption of OIRS design. In Fig. 1, each block module is considered as the process of a program taking input and producing output.The cloud is assumed to honestly perform the image reconstruction service as

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 4, April 2015
ISSN: 2394 - 6598
www.ijetie.org

specified, but be curious in learning owner/user's data content. Because the images samples captured by data owners usually contain data specific/sensitive information, we have to make sure no data outside the data owner/user's process is in unprotected format.

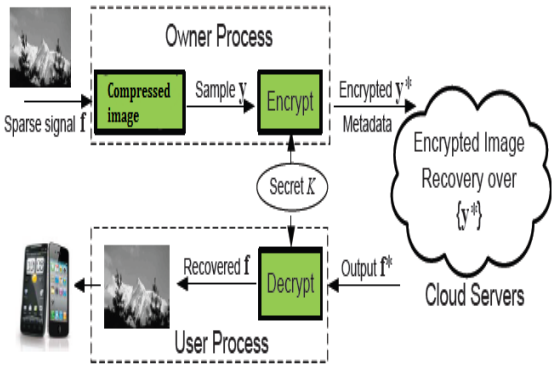
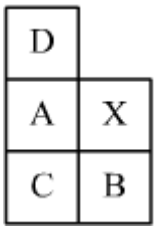


FIGURE 1. The OIRS Architecture in public cloud.

3.1 Standard Filtering

In the standard filtering [2] phase the algorithm attempts to obtain a good distribution of values by making an estimate of the value of each sample based on points next to it; every value will be replaced by its difference with the computed estimate. An ordering on the samples is defined: a point is previous to another if it is on a column to its left or if it is on the same column but has a minor row index. This ordering is important to guarantee the complete reversibility of the transformation, as we will use only samples *previous* than a given one to compute the estimate. In this way if we apply the filter from the last value to the first, when we will have to remove it we can scan the image from the first value to the last obtaining exactly the same estimates and re-obtaining the initial values by adding these values to the filtered samples. This technique is well-known and is used, for example, in the PNG format.To maintain constant the range of the values, we will do all operations in modulo 256. The matrix representing a color will be represented before and after filtering by a matrix of bytes. Since it is logical to consider an error of -1 on the estimate close to 0, we will now consider the range of values to go from -128 to 127. Estimates on the values of the image are done using assumptions of continuity. Even if a raster image represents a discrete signal, this signal should be in most of its part similar to a sampling of a continuous and quite regular function. In other words we expect that great parts of images are smooth and do not have sudden changes. An edge is a part of an image that goes clearly against these assumptions, and so we expect that borders or particularly irregular parts of an image cause the filtering estimate to be not so good. Anyway this is unavoidable for any compression algorithm, as an edge is a part of the image rich of information and difficult to compress. In the benchmarks section it is shown how the PCIF algorithm deals with edges and borders better than other algorithm as the Jpeg2000 algorithm.

Since characteristics of images vary from zone to zone, to get better results the algorithm applies different filtering functions to different zones. The image is divided in 8x8 zones, and for each of the zone a filtering function that minimizes the sum of the absolute values in that zone is chosen. In simple terms, the function that gets values more



close to zero is considered to be the best. All filtering functions uses for the sample x at most the four points a, b, c and d, shown in figure, to compute the estimate. This is useful because in this way the memory occupation of the filtering phase can be minimal (only two columns at a time must be memorized to reverse the filtering) and because using only close values we take advantage of the locality principle.

3.1.1 Used Filtering Functins

The filtering functions tested for every zone are the following; the numbering of the functions will be explained further. For each filter there will be a short justification for its extimate. Each division is intended as an integer division, and can so be computed efficiently with a bit shift.

1. 1. f(x) = 0

No filter; usefull for very chaotic zones

1. 2. f(x) = a
2. 4. f(x) = b
3. 5. f(x) = c
4. 7. f(x) = d

A value immediately next to x is used for its extimate. Usefull in very regular zones where points next to each other have similar values. One of these filters is best

IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 4, April 2015
ISSN: 2394 - 6598
www.ijetie.org

than the others if the correlation is stronger in the relative direction.

1. 8. f(x) = a + (b-c) / 2
2. 9. f(x) = b + (a-c) / 2

One of the points a, b and c is used as a 'base' extimate. The other two are used to guess the image variation in the point.

1. 3. f(x) = (a + b) / 2
2. 10. f(x) = (a + b + c + d + 1) / 4
3. 11. f(x) = (a + d) / 2

Usefull when an image has a zone with little variations that can be efficiently guessed through an average of the adiacent points. The '+1' in filter 10 is used to obtain an efficient rounding of the division on a, b, c and d.

1. 6. f(x) is defined by the following pseudo-code:
2. if (a + b - c < 0) {
3. f(x) = 0;
4. } else if (a + b - c > 255) {
5. f(x) = 255;
6. } else {
7. f(x) = a + b - c;
8. }

Linear filter; the point coplanar with a, b and c is chosen as an extimate for x. The value is limited to the sample's domain.

1. 12. f(x) is the Path filter, defined by the following pseudo-code:
2. p = a + b - c;
3. pa = abs(p - a);
4. pb = abs(p - b);
5. pc = abs(p - c);
6. if (pa <= pb and pa <= pc) then f(x) = a
7. else if pb <= pc then f(x) = b
8. else f(x) = c;

The Paeth filter choses a value between a, b and c than minimizes the gradient of the image in the point x.

1. 0. f(x) is defined by the following:

   if a <= c <= b then f(x) is equal to the linear filter
   otherwise, f(x) is equal to the Paeth filter

This filter is a good compromise between the linear filter and the Paeth filter. The condition is a sort of edge detector. In other words, this filter could be expressed: if the image is quite regular, then the linear filter could be ok, if the image is varying then detect the most probable edge ad use it as an extimate.

The program by default uses the filters from 0 to 11; the Paeth filter doesn't seem to give any advantage. Parameters can be varied during compression to use only the first n filters, obtaining a shorter processing time in exchange to a light loss in image compression. For this reason the filters have been numbered from the most usefull to the least, allowing quite good performances also with few filtering functions.

3.1.2 Color Filtering

The second filtering phase consists in color filtering. While we previously have simplified the image using assumptions of smoothness, now we will consider color correlation: we expect that when an image has a color change from a zone to another, often this change will be reflected in more than one of the three color components, so we can use one of them as an extimate for the others.

The problem of color decorrelation is often treated in simplier ways, and the PCIF compression algorithm is the first (to my knowledge) to use a color-based filtering. Another typical approach is to apply a color tranform before compression to allow the three obtained layers to be compressed indipendently. This tecnique does not consider that different parts of an image can be correlated with the other colors in different ways. This is why the PCIF algorithm is addressed to true color images: for black and white images this phase could be skipped and values could be coded probably more efficiently directly after the filtering phase.

Being the color filtering phase applied to already ('standardly') filtered layers, the extimates will not be done on the assumption that colors will have similar values, but that irregularities or particular variations in the original images (that are represented by the values in the filtered layers) reflect themselves in more color components at a time.

Similarly to the standard filtering phase, an ordering is defined between colors: blue, green and red. Again, the image is divided in 8x8 pixel squares and for each zone a different filtering function can be applied. The color filtering functions that can be used are the following. Considering that p[x] is the point that we are extimating and x represents 0, 1 and 2 if the point is relative to the blue, green or red layer respectively.

1. if x=0 then f(p[x]) = 0, otherwise f(p[x]) = f(p[x-1])

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 4, April 2015
ISSN: 2394 - 6598
www.ijetie.org

2. f(p[x]) = 0
3. if x=0 then f(p[x]) = 0, otherwise f(p[x]) = f(p[0])

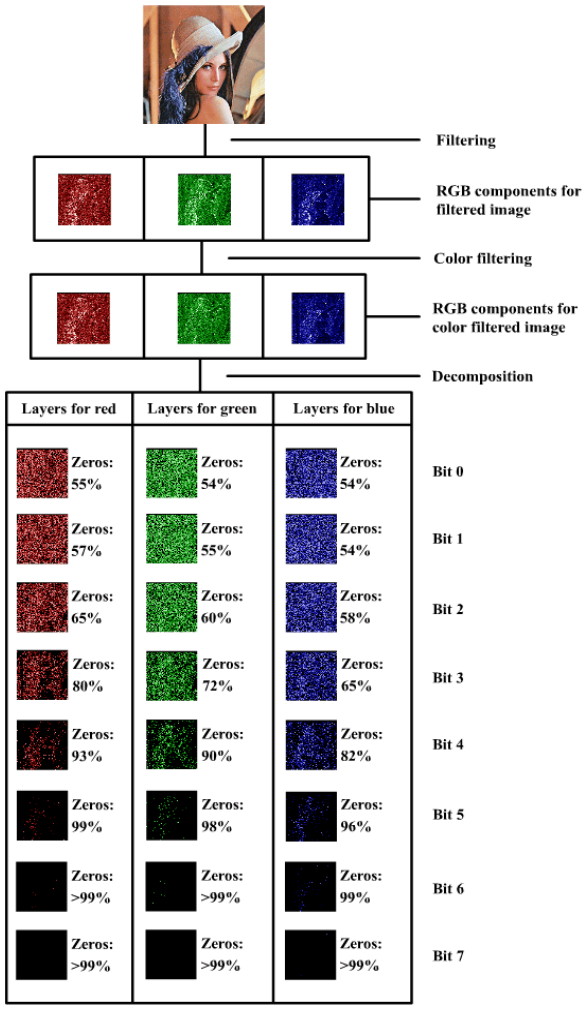
Filter 2 is used in zones that should not be filtered as this would give a bad result. Filter 1 and 3 use the previous or the first color for the extimate of the point. Since the points representing the color 0 (blue) are used for the extimates, they are the first values in our ordering and so to make the reversion of this the operation possible these should not be changed. Again, the information about the used filters will be stored in the compressed file.

3.1.2 Remapping steps

After the filtering phase, the values representing the filtrered image are reorganized to obtain an optimal distribution. What we want to obtain is a maximal frequency for value 0, and higher frequency values assigned ordinately for values from 1 to 255. The algorithm obtains this goal by remapping every possible value of the filtered samples into another, so that the described condition is satisfied. The individuation of values with higher frequency is obtain through a count made during the previous color filtering phase; to guarantee the inversion of this operation a permutation array is created representing the reassignment executed in this phase, and this structure is encoded in the compressed file.

3.2 Decomposition

At this point the image is decomposed in 24 matrixes containing only zeros and ones. Each of these matrixes will refer to one of the 8 bits of the bit representation of one of the 3 primary colors representing the image. What we expect after this phase are some very chaotical layers relatively to the least significative bits and somematrixes with lots of zeros corresponing to the most



significant bit. Another important property that we can find is that the ones in the more ordered layers are often grouped together, offering highly compressible data to the tecniques that will follow this phase. During the decomposition the algorithm also counts the number of zeros and ones in each layer; this will be usefull to extimate the chaos in each matrix and determine with which tecnqique this will be compressed or to determinate, if the layer is too chatic, that the algorithm can save time storing it as it is as there would be a very low gain obtained through its compression. The figure 2 represents how the famous Lenna image is decomposed into layers by the algorithm after the various filtering phases: After the decomposition phase, layers are sufficiently uncorrelate one from another to allow the following compression phase to operate on them separately without losing compression efficience.

IJETIE
International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 4, April 2015
ISSN: 2394 - 6598
www.ijetie.org

3.3 Compression

At the beginning of this phase for each layer a compression method is chosen. This allows the application of different compression techniques for different types of data, as the compression through polyominoes is more effective on layers with few ones while the compression with RLE and Huffman codes has better results with layers with higher enthropy. In table, the different sizes of the various compressed components of the Lenna image.

We consider we can dispose of the percentage *d* of zeros in a layer, as the algorithm counts the number of zero values during the previous decomposition phase. We also consider the index of the layer *b*, as 0 for layers relative to the least sigificative bit up to 7 for the most significative. This is because we expect that, having equal density of zeros, layers that refer to higher bits have more grouped one values, and this lowers the enthropy of the matrix. The function that determines the compression method is the following:

- If d + b > 90 then compress the layer using polyominoes.
- If 52 < d + b <= 90 then compress the layer with RLE and Huffman coding.
- If d + b <= 52 then store the layer 'as is'.

Note that thanks to the remapping phase we expect that b is always >= 50. In the third rule of the above, we save time in compression and decompression at the cost of only a few bits of the final file size, as we expect that very chaotical layers that fall in this category are very weakly compressible.

The values above have been chosen empirically thanks to various experimentations, always keeping in mind the efficiency goal. The limit 90 in the first and second rule gives quite similar results if lowered up to 85, but in this cases the RLE and Huffman compression method has been chosen as it allows a faster elaboration. Further information about the two different compression techniques can be found at the compression through polyominoes section and in the compression through RLE and Huffman codes section. In the following table, we can see the different sizes of the various compressed components of the Lenna image. For every layer the used compression method and its compressed size are shown. The intestation of the image contains mainly the information about the filters applied to the image in the different zones, and also includes the remapping array and various image information.

Layer	Bit	Color	Zero percentage	Compressed with	Size
PCF Intestation	-	-	-	-	3.4 KB
0	7	Blue	99.9 %	Polyominoes	124 bytes
1	7	Green	99.9 %	Polyominoes	45 bytes
2	7	Red	99.9 %	Polyominoes	18 bytes
3	6	Blue	99 %	Polyominoes	1.7 KB
4	6	Green	99.8 %	Polyominoes	641 bytes
5	6	Red	99.9 %	Polyominoes	349 bytes
6	5	Blue	96 %	Polyominoes	7.3 KB
7	5	Green	98 %	Polyominoes	3.7 KB
8	5	Red	99 %	Polyominoes	2.4 KB
9	4	Blue	82 %	RLE and Huffman	21.8 KB
10	4	Green	90 %	Polyominoes	15.4 KB
11	4	Red	93 %	Polyominoes	10.5 KB
12	3	Blue	65 %	RLE and Huffman	30.1 KB
13	3	Green	71 %	RLE and Huffman	27.7 KB
14	3	Red	80 %	RLE and Huffman	22.6 KB
15	2	Blue	58 %	RLE and Huffman	31.6 KB
16	2	Green	60 %	RLE and Huffman	31.2 KB
17	2	Red	65 %	RLE and Huffman	30.0 KB
18	1	Blue	54 %	RLE and Huffman	31.96 KB
19	1	Green	55 %	RLE and Huffman	31.88 KB
20	1	Red	57 %	RLE and Huffman	31.68 KB
21	0	Blue	54 %	RLE and Huffman	31.88 KB

IJETIE International Journal of Emerging Technology and Innovative Engineering
Volume I, Issue 4, April 2015
ISSN: 2394 - 6598
www.ijetie.org

22	0	Green	54 %	RLE and Huffman	31.99 KB
23	0	Red	55 %	RLE and Huffman	31.95 KB
Total filesize	-	-	-	-	432 KB

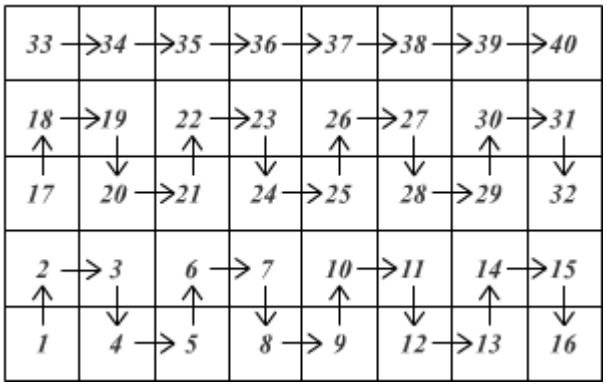
TABLE 1. Remapping array and various image information.

3.3.1 Compression through RLE and Huffman Codes

Layers that are mediumly chaotic are compressed throught a tecnique that combines Run Lenght Encoding (RLE) and Huffman coding. The combination of these two tecniques is quite widely used to obtain a high compression even through Huffman codes even when we have lots of occurencies of an unique simbol (0, in our case). We obtain in this way a compressed size very close to the enthropy of the layer without using computationally-expensive tecniques as arithmetic coding.

This method can be expressed shortly by the following steps:

1. The layer is scanned from left to right and from low to bottom, proceeding in a zig-zag pattern described in figure (an example on an 8x5 matrix). The numer of zeros and ones encountered is counted (RLE).
2. Every couple of sequences of zeros and ones is encoded with an Huffman tree.



The Huffman tree must be encoded along the compressed bitstream. The coupling of the sequences is usefull to capture the information about density of ones in a given zone: if we have found a high number of zeros it is reasanable to expect a low number of ones to follow. Grouping together sequences of ones and zeros we consider this information in the coding obtaining a higher compression ratio. The zig-zag order is used to consider correlation in both dimensions without losing the computational advantage given by the principle of locality, as we consider only two rows of the layer at a time. A more efficient way to consider bidimensional correlation, even if this would reduce the computational efficiency, would be to use a fractal-style visit order.

3.4 Advance Encryption Algorithm

The AES algorithm gains wide application in our daily life, such as smart cards, cell phones, automated teller machines and WWW servers. AES encrypts a plaintext to become a ciphertext, which can be decrypted back to the original plaintext by using common private key, an example is shown in Figure 2, It can be seen the ciphertext is very different from and gives no clue to the original plaintext. Figure 2 shows the Encryption of AES operation using cipher key

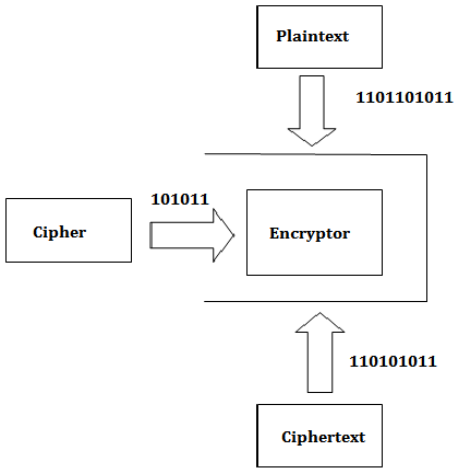


FIGURE 2. : Encryption of AES operation.

3.4.1 AES ALGORITHM

Rijndael is a block cipher developed by Joan Daemen and Vincent Rijmen. The algorithm is flexible in supporting any combination of data and key size of 128, 192, and 256 bits. However, AES merely allows a 128 bit data length that can be divided into four basic operation blocks. These blocks operate on array of bytes and organized as a 4×4 matrix that is called the state[3]. For full encryption, the data is passed through Nr rounds (Nr = 10, 12, 14) .These rounds are governed by the following transformations:

Subbyte Transformation: Is a non linear byte Substitution, using a substation table (s-box), which is

constructed by multiplicative inverse and Affine Transformation.

Shift rows transformation: Is a simple byte transposition, the bytes in the last three rows of the state are cyclically shifted; the offset of the left shift varies from one to three bytes.

Mix columns transformation: Is equivalent to a matrix multiplication of columns of the states. Each column vector is multiplied by a fixed matrix. It should be noted that the bytes are treated as polynomials rather than numbers.

Add round key transformation: Is a simple XOR between the working state and the round key. This transformation is its own inverse.

Expansion key: With AES encryption, the secret key is known to both the sender and the receiver. The AES algorithm remains secure, the key cannot be determined by any known means, even if an eavesdropper knows the plaintext and the cipher text. The AES algorithm is designed to use one of three key sizes (Nk). AES-128, AES-196 and AES-256 use 128 bit (16 bytes, 4 words), 196 bit (24 bytes, 6 words) and 256 bit (32 bytes, 8 words) key sizes respectively. These keys, unlike DES, have no known weaknesses. All key values are equally secured thus no value will render one encryption more vulnerable than another. The keys are then expanded via a key expansion routine for use in the AES cipher algorithm. This key expansion routine can be performed all at once or „on the fly" calculating words as they are needed.

4. CONCLUSION

In this paper, we have proposed OIRS, an outsourced image recovery service with PCIF algorithm for compression and AES algorithm for encryption with privacy assurance. OIRS exploits techniques from different domains, and aims to take security, design complexity, and efficiency into consideration from the very beginning of the service flow. Data users, on the other hand, can leverage cloud's abundant resources to outsource the image recovery related optimization computation, without revealing either the received compressed samples, or the content of the recovered underlying image. Both extensive security analysis and empirical experiments have been provided to demonstrate the privacy-assurance, efficiency, and the effectiveness of OIRS.

5. REFERENCES

[1] Cong Wang, Bingsheng Zhang, Kui Ren , and Janet M. Roveda, IEEE Transaction on cloud computing vo:1 no:1 year:2013

[2] Elena Barcucci, Srecko Brlek, and Stefano Brocchi,"PCIF: An Algorithm for Lossless true color Image Compression.

[3] E. Candès, J. Romberg, and T. Tao, ''Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information,'' IEEE Trans. Inf. Theory, vol. 52, no. 2, pp. 489–509, Feb. 2006.

[4] E. Candès and T. Tao, ''Near-optimal signal recovery from random projections: Universal encoding strategies,'' IEEE Trans. Inf. Theory, vol. 52, no. 12, pp. 5406–5425, Dec. 2006.

[5] D. Donoho, ''Compressed sensing,'' IEEE Trans. Inf. Theory, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[6] A. Divekar and O. Ersoy, ''Compact storage of correlated data for content based retrieval,'' in Proc. Asilomar Conf. Signals, Syst. Comput., 2009, pp. 109–112.

[7] S. Goldwasser, Y. T. Kalai, and G. Rothblum, ''Delegating computation: Interactive proofs for muggles,'' in Proc. STOC, 2008, pp. 113–122.

[8] S. Hohenberger and A. Lysyanskaya, ''How to securely outsource cryptographic computations,'' in Proc. TCC, 2005, pp. 264–282.

[9] P. Radhadevi, P. Kalpana,"Secure Image Encryption Using AES.