A NOVEL VIDEO ENCRYPTION TECHNIQUE BASED ON SECRET SHARING

Narsimha Raju C, UmaDevi Ganugula, Kannan Srinathan, C. V. Jawahar

International Institute of Information Technology-Hyderabad,
India, Hyderabad.

ABSTRACT

The rapid growth of the Internet and digitized content made video distribution simpler. Hence the need for video data protection is on the rise. In this paper, we propose a secure and computationally feasible video encryption algorithm based on the principle of Secret Sharing. The strength of the DC is distributed among the AC values based on *Shamir's Secret Sharing (SSS) scheme*. The proposed algorithm guarantees security, fastness and error tolerance with a small increase in video size.

***Index Terms*—** Multimedia security, MPEG, Secret Sharing, Cryptography, Discrete Cosine Transform (DCT)

1. INTRODUCTION

Increase in popularity of the Internet catalyzed the emergence of applications involving multimedia transfer. Many of these applications require the secure transfer of videos or images, like video-on-demand, pay-TV to name a few. The Internet in its current form, does not provide security for multimedia communication. Therefore multimedia encryption is needed for complete confidentiality. Classic encryption algorithms like DES, RSA etc. are not feasible for video data because of its large size and high computational requirements. As a countermeasure, earlier encryption algorithms used simple scrambling mechanisms for huge video data. Evidently they are insecure. Moreover, these algorithms typically neglected the data format of the videos [1], thus burdening compression phase of codec.

Owing to the large size and real-time applications of video, two types of encryption algorithms are generally in vogue. One method is *selective* encryption. In this method, only parts of video content are encrypted, thereby reducing the computational requirement. This method is suitable because the full content of the video is not critical. Another method is the *light weight* encryption which trades off security for computational time.

Tang [1] proposed a video encryption algorithm which incorporated encryption in the compression phase. He used a random permutation list to replace the zig-zag ordered DCT coefficients of a block to a 1×64 vector. Zeng and Lie [2] extended this permutation to a segment of macroblocks. In each segment, DCT coefficients of the same frequency are randomly shuffled within band. Chen [3] further modified this idea by extending the permutation range from a segment to a frame. In this method, each macroblock DCT coefficients are divided into 64 groups according to their positions, and scrambled within each group. In addition, the authors permute the motion vectors of P and B frames. However, scramble-only methods do not provide sufficient security [4].

Qiao and Klara [5] proved that Tang's encryption suffers from the known-plaintext and ciphertext-only attack and they proposed a new encryption algorithm called VEA. In this algorithm, a chunk of I frame is divided into two halves. Both the halves are XORed and stored in one half. The other half is encrypted with a standard encryption algorithm DES. Though, this algorithm provides good security, it entails an additional computational load of 47% (which is inappropriate for real-time videos). Shi and Bhargava [6] proposed a *light weight* encryption algorithm in which a secret key randomly changes the sign bits of all the DCT coefficients of an I-frame in MPEG video. The authors further extend the algorithm and encrypt the sign bits of motion vectors of P and B frames. However, as with most of the light weight algorithms known so far, the algorithm of [6] is also susceptible to known-plaintext and ciphertext-only attacks [4].

Note that the selective encryption algorithms typically uses a heavy weight encryption algorithms (DES and AES). Consequently, the time taken for encryption using them is high and make them unsuitable for real-time applications. On the other hand, as mentioned earlier, most of the known light weight algorithms are susceptible to known-plaintext and ciphertext-only attacks [4]. Therefore there is a need for a new secure video encryption algorithm perhaps designed by combining these two classes, which retains their respective advantages, whilst minimizing the dis-advantages. Apparently a naive combination is worse compared to each one of them individually — if one wants to reduce the time, then one may opt for selective light weight encryption, which degrades the security; if one attempts to improve security by the Shannon's principle of selective permute-then-encrypt, it increases the computational time.

In this paper, we propose a selective cum light weight encryption algorithm which is fast enough for real-time applications, yet possessing practically acceptable levels of secu-

rity. A key ingredient of our solution is a suitably adapted and randomized version of Shamir Secret Sharing scheme [7]. Further more, our algorithm, unlike most of the video encryption algorithms, has an inbuilt error-tolerance, which could be handy in many practical applications.

2. ENCRYPTION ALGORITHM

2.1. Principle of Secret Sharing

In this subsection we briefly describe the concept of Shamir Secret Sharing. A (k, n) threshold-based secret sharing describes how a secret is shared among n participants [7]. It constructs a $k-1$ degree polynomial $f(x)$ as

$$f(x) = (d_{k-1}x^{k-1} + \ldots + d_1x + d_0) \ mod \ p$$

where the value d_0 is the secret, $d_1 \ldots d_{k-1}$ are random numbers and p is a prime number. The secret shares are the pair of values (x_i, y_i) where $y_i = f(x_i)$, $1 <= i <= n$. The polynomial function $f(x)$ is destroyed after each shareholder possesses a pair of values (x_i, y_i) so that no single share holder knows the secret value d_0. In fact, no group of $k-1$ or fewer secret shares can discover the secret d_0. On the other hand, when k or more secret shares are available, then we may set at least k linear equations $y_i = f(x_i)$ for the unknown d_i's. The unique solution to these equations shows that the secret value d_0 can be obtained from the Lagrange's interpolation [7].

The SSS is regarded as a perfect secret sharing(PSS) scheme since knowledge of any $(k-1)$ values does not reveal any information about the secret.

2.2. Proposed Encryption Algorithm

Discrete Cosine Tranform (DCT) is used to compress images and videos. It decomposes an image signal into its frequency components. The transform coefficients can be classified into two groups namely, DC and AC coefficients. The DC coefficient is the mean value of the image block and carries most of the energy in the image block. The AC coefficients (ACs) carry energy depending on the amount of detail in the image block. However, most of the energy is compacted in the dc coefficient and a few ac coefficients. In [8, 9], the authors claim that encryption of 3 to 9 ACs is sufficient to hide the details of a given macroblock.

Our algorithm is a selective and light weight encryption algorithm, which takes the DC component of each DCT block and depending on the number of ACs in that block distributes the DC among the ACs and itself, based on the principle of secret sharing. We use Shamir's (k, n) secret sharing scheme, where $n = k + 1$. The value of k varies from 4 to 12 depending on the number of ACs in that block. Though we can share the DC among all the ACs available, it has the drawbacks of increase in video size and decryption time at the receiver. We observed that decryption time is acceptable for real-time video applications when k lies between 4 to 12. Though the algorithm can have any value of k between 4 to 12, in this work we apply algorithm only for k = 4,8,12.

Algorithm 1 Encryption Algorithm

for Each and Every DCT block **do**
  Step 1: Initialize nac = number of non-zero ACs
  Step 2:
  if $nac < 10$ and $nac > 5$ **then**
    perform (4,5) secret sharing with
    $DC, AC_1, \ldots AC_3$ as input and
    store the result in $DC, AC_1, AC_2, AC_3, AC_{nac}$
  end if
  if $nac < 20$ **then**
    perform (8,9) secret sharing with
    $DC, AC_1, \ldots AC_7$ as input and
    store the result in $DC, AC_1, AC_2 \ldots AC_7, AC_{nac}$
  end if
  if $nac > 20$ **then**
    perform (12,13) secret sharing with
    $DC, AC_1, \ldots AC_{11}$ as input and
    store the result in $DC, AC_1, \ldots AC_{11}, AC_{nac}$
  end if
end for

Algorithm 2 Secret Sharing

Step 1: Two values r_1 and r_2 are taken from randomly generated number list
Step 2: A $(k-1)$ degree polynomial is constructed as follows:

$$f(x) = (AC_{k-1}X^{k-1} + \ldots + DC) \ mod \ 251$$

$DC = f(r_1)$, $AC_1 = f(r_2)$, $AC_2 = f(2) \ldots AC_p = f(p)$ and $AC_{nac} = f(nac)$
where p=3,7,11.

Algorithms 1 and 2 describe the video encryption procedure. When algorithm 1 is considered, in step 1, the count of non-zero ACs is stored as nac. In step 2, the secret sharing principle is applied based on the number of ACs in the block. If the number of ACs is less than 5, we simply XOR the DC and ACs of that particular block by a randomly generated number list. If nac lies between 5 and 10, $(4, 5)$ secret sharing is applied. Other wise $(8, 9)$ or $(12, 13)$ secret sharing is applied when nac value is less than 20 and greater than 20, respectively.

The last AC coefficient, AC_{nac} is used to store an extra coefficient. Though, we are losing an AC coefficient because of this, around 95% of the cases it would be one. Even if it is not one the loss in video quality will not be visible to the end user. So the proposed method of encryption has slight loss in Quality Of Service (QOS) but this loss can be neglected, as there is not much visible degradation observed in the video.

Algorithm 2 explains how secret sharing is used to distribute the DC among itself and the ACs. A Pseudo Random

Number Generator (PRNG) is used and a list of random numbers is generated. The PRNG takes the *key (seed)* as the input and produces a list of random numbers within $GF(251)$. Every time two values are taken from this list, denoted as r_1 and r_2. These act as keys to that particular block.

For a given (k, n) secret-sharing, a $(k-1)$ degree polynomial is constructed as mentioned in the algorithm. $f(r_1)$ becomes DCs share and $f(r_2)$ becomes AC_1's share. The shares of the remaining ACs are calculated based on their respective positions as $AC_p = f(p)$.

During decryption at the receiver end, the authorized user first has to generate a set of random numbers using the *key (seed)* as the input to PRNG and then proceed with constructing k equations from n values then use Legrange interpolation to get back the DC and AC coefficients of each block [7]. Even if he loses $n-k$ values (only one value in the current setting), he can retrieve the DCT coefficients correctly. An unauthorized user cannot get back the secret because of the property that any $k-1$ shares doesn't reveal anything about the secret.

3. PERFORMANCE ANALYSIS

This section addresses the key features of the proposed algorithm.

3.1. Security Analysis

For multimedia data, an encryption algorithm is said to provide *acceptable level* of security if the cost of breaking the proposed method of encryption requires more investment than buying the key. The proposed method provides this acceptable level of security. Also encryption algorithm is *secure*, if it can withstand ciphertext-only and known-plaintext attacks.

In the ciphertext-only attack, the attacker has to find the original values from the encrypted values. According to our algorithm, for a given macroblock the computational cost of breaking would be 251^k where k can be 4,8,12. On an average the cost of breaking would be 251^8. Now for a given video frame with q macroblocks, the computational cost of breaking the ciphertext by an unauthorized user would be 251^{8q}. This makes it practically infeasible. Thus, our algorithm is robust to ciphertext-only attack.

In the known-plaintext attack, unauthorized user has both original and the corresponding encrypted values. Since we use a freshly generated *seed (key)* for each new transfer of the video, it is obvious that our algorithm is secure against known-plaintext attack. The key is transmitted within the MPEG video bitstream in encrypted form using the public-key of the receiver.

3.2. Error Tolerance

Secret Sharing is the key factor that helps in providing error-tolerance. In the proposed algorithm k values are distributed among $k+1$ values. So, lose or change of one value can be tolerated. This feature of the algorithm comes with extra computational overhead which is also tolerable. If a value is lost, rest of the k values can be used to get back the original coefficients according to secret sharing. If a value is modified, the original values can be obtained by trail-and-error in at most $k-1$ iterations. Hence error-tolerance is achieved.

4. EXPERIMENTAL RESULTS

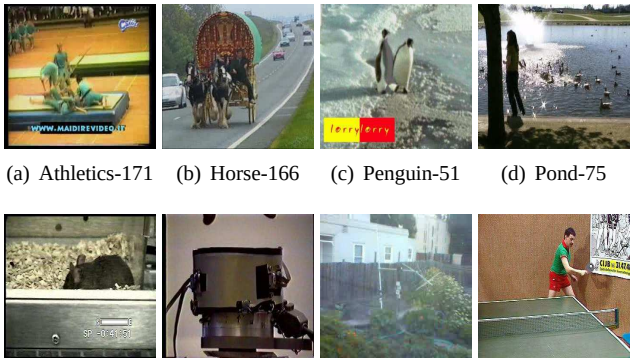
The proposed encryption algorithm is tested using a library from Cornell university named DALI [10], which supports MPEG-1 and MPEG-2 videos. While distributing the DC among the ACs and itself, the proposed encryption is applied considering the DCT coefficients of each and every block as positive.

A series of experiments are conducted to judge the k-value of secret sharing. We found that as the number of ACs is increased, the distribution of DC among more elements provides better security than simple distribution of DC among a fixed number of ACs. We experimented with distribution of DC among the ACs with k value fixed and k value varying depending on number of ACs. In both the cases, k is allowed to vary from 4 to 20. The results show that an optimal solution with acceptable computational overhead to the adversary and less decryption time for an authorized receiver occurs when k is allowed to be 4,8 or 12. Though the algorithm provides good security for any varying k, in this work we consider k= 4,8,12 only.

The algorithm is applied for multiple videos with different contents and motion characteristics. Figure 4 shows various kinds of videos along with their frame numbers. Figure 4 shows the corresponding encrypted video frames. The resultant video can be played by any MPEG decoder. Though motion in the video can be easily identified when the video is played back, it is very difficult to recognize the moving objects.

Sometimes the I-blocks in P and B frames cause partial leakage of the image information. In order to achieve higher security I-blocks of P and B frames need to be encrypted. However, such an encryption increases encryption time intolerably. In order to decrease this overhead in time, motion vectors of P and B frames can be encrypted as substitutes to I-blocks, without loss in security level. But for the real-time applications, encryption of motion vectors is not required because this partial leakage of image information helps a non-paid customer by creating interest to purchase the video. The algorithm cannot be applied to highly sensitive videos where everypart of the video is important, such as military applications. In order to apply this method in the domain of military applications, minor modifications are needed, i.e., the algorithm proposed in [11] should be augmented to this algorithm for the encryption of motion vectors.

Table 4 shows the result of our technique on various videos along with the percentage increase in their video size. Figure 4 shows a graph comparing our algorithm with tech-



(a) Athletics-171 (b) Horse-166 (c) Penguin-51 (d) Pond-75

(e) Rat-89 (f) Lab-245 (g) House-1977 (h) TT-13

Fig. 1. Example Test Videos Along With Frame Numbers

Fig. 2. Respective Encrypted version of the Test Videos

niques of complete XOR, scrambling with XOR [12] and scrambling of the DCT coefficients [3]. The average increase in video size by our method is 30.23% only and the average time taken is $7.83ms$ only. Increase in space is the overhead on the compression process. Though, it is high when compared to scrambling method, scrambling only methods are insecure. The time taken for the execution is also less compared to the other techniques mentioned, making it fast. Hence the algorithm is secure, fast and error-tolerant.

5. CONCLUSION

We proposed a novel selective and light weight encryption algorithm for the security of video data, based on the principle of sharing the DC coefficients among the ACs and DC. The-

Test Video	No of frames	original length	encrypted length	% increase in video size
Athletics	184	280	396	41.43
Horse	801	1968	2468	25.41
Penguin	86	400	540	35.00
Pond	911	5144	6256	21.62
Rat	315	1592	1972	23.87
Lab	635	2228	2680	20.29
House	4320	6612	9144	38.29
Tennis	150	1228	1664	35.95

Table 1. Percentage increase in size after encryption

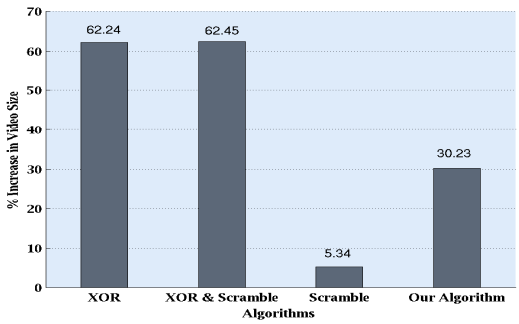


Fig. 3. Encryption Overhead on the Compression

oretical analysis and experimental results show that the proposed encryption scheme is fast, provides good security with error-tolerance and adds very less overhead on the compression of the video which most of the real-time video applications require. This technique can be used for real-time video processing applications such as pay-for-view.

6. REFERENCES

[1] L. Tang, "Methods for encrypting and decrypting MPEG video data efficiently," in *Proc. of ACM Multimedia*, 1996.

[2] Wenjun Zeng and Shawmin Lei, "Efficient frequency domain selective scrambling of digital video," in *Proc. of the IEEE Transactions on Multimedia*, 2002, pp. 118–129.

[3] Zhenyong Chen, Zhang Xiong, and Long Tang, "A novel scrambling scheme for digital video encryption," in *Proc. of Pacific-Rim Symposium on Image and Video Technology(PSIVT)*, 2006, pp. 997–1006.

[4] Borko Furht, Daniel Socek, and Ahmet M. Eskicioglu, "Fundamentals of multimedia encryption techniques," in *Multimedia Security Handbook*.

[5] L. Qiao and Klara Nahrstedt, "A new algorithm for MPEG video encryption," in *Proc. of First International Conference on Imaging Science System and Technology*, 1997, pp. 21–29.

[6] C. Shi and Bharat Bhargava, "A fast MPEG video encryption algorithm," in *Proc. of ACM Multimedia*, 1998, pp. 81–88.

[7] A. Shamir, "How to share a secret," in *Proc. of communications of the ACM*, 1979, vol. II, pp. 612–613.

[8] Zheng Liu, Xue Li, and Zhaoyang Dong, "Enhancing security of frequency domain video encryption," in *Proc. of ACM multimedia*, 2004, pp. 304–307.

[9] J. Meyer and F. Gadegast, "Security mechanisms for multimedia data with the example MPEG-1 video," available at www.gadegast.de/frank/doc/secmeng.pdf.

[10] "DALI library," available at http://www.cs.cornell.edu/dali.

[11] Zheng Liu and Xue Li, "Motion vector encryption in multimedia streaming," in *Proc. of 10Th international Conference of Multimedia Modelling Conference*, 2004, pp. 64–71.

[12] L. S. Choon, A. Samsudin, and R. Budiarto, "Lightweight and cost-effective MPEG video encryption," in *Proc. of Information and Communication Technologies: From Theory to Applications*, 2004, pp. 525–526.