

Comparison and Analysis of Selected Video Encryption Algorithms Implemented for MPEG-2 Streams

Daniel Socek

*Department of Computer Science and Engineering
Florida Atlantic University
777 Glades Rd
Boca Raton, FL 33431-0991, USA
dsocek@fau.edu*

Abstract

Communication security of video streams can be achieved by applying some conventional encryption technique on the entire video bit-stream. When a video is delivered over a non-dedicated channel, such as the Internet, encrypting entire bit-stream creates a major bottleneck. Thus, a lot of research was dedicated to finding good selective video encryption techniques, which encrypt only a part of the bit-stream. Many such techniques were proposed, however a comprehensive comparative study was never made. In this paper, I present the comparative analysis of few selected video encryption algorithms, based on the experiments implemented for MPEG-2 codec.

1. Introduction

Multimedia security is in general provided by a method, or a set of methods, used to protect the multimedia content. These methods are heavily based on cryptography and they enable either communication security, or security against piracy (Digital Rights Management and watermarking), or both. Communication security of digital images and textual digital media can be accomplished by means of standard symmetric key cryptography. Such media can be treated as binary sequence and the whole data can be encrypted using a cryptosystem such as AES or DES. In general, when the multimedia data is static (not a real-time streaming) we can treat it as a regular binary data and use the conventional encryption techniques. Encrypting the entire multimedia stream using standard encryption methods is often referred to as the naive approach.

However, due to variety of constraints (such as the real-time processing, etc.), communication security for

streaming audio and video media over the Internet, or similar non-dedicated channel, is harder to accomplish. The expensive overhead introduced by applying encryption to the entire bit-stream is not acceptable solution for such applications. On the other hand, using fast yet insecure scrambling methods is also not acceptable in many situations. Thus, some of the current research is focused on modifying and optimizing the existing cryptosystems for real-time audio/video. It is also oriented towards exploiting the format specific properties of many standard multimedia formats by encrypting only selected bits from the multimedia bit-stream in order to achieve the desired speed and enable real-time streaming. This is referred to as the *selective encryption*.

Some of the first research initiatives that introduced the idea of selective encryption were introduced in mid-1990s, and since then, there are continuous research efforts to improve existing as well as to create new approaches. Some comprehensive surveys were written that classify the proposed selective video, image and audio encryption techniques (such as [1], [4], and [5]) but none of them compares the actual performances since implementations are not available for many of these schemes. Not only that only few of these techniques were ever implemented, but also the implementations were made for the old MPEG-1 codec (by modifying the Berkeley MPEG-1 implementation). Not a whole lot of comparative studies exist for the proposed video selective encryption algorithms, and that is one of the reasons I am interested in comparing their performances in this project. By [1], some of the most important research accomplishments in this area are video encryption algorithm by Qiao and Nahrstedt [2], and four video encryption algorithms by Bhargava, Shi and Wang [3]. In this paper, I implemented these algorithms for MPEG-2 codec, and compared their performances in terms of security, time expansion, and

different types of videos (high motion vs. low motion videos.)

2. Video Encryption Algorithms and their Security Analysis

The *Video Encryption Algorithm (VEA)* by Qiao and Nahrstedt [2] is constructed with the goal to exploit the statistical properties of the MPEG video standard. The algorithm consists of the following four steps:

- *Step 1:* Let the $2n$ byte sequence, denoted by $a_1a_2\dots a_{2n}$, represent the chunk of an I-frame
- *Step 2:* Create two lists, one with odd indexed bytes $a_1a_3\dots a_{2n-1}$, and the other with even indexed bytes $a_2a_4\dots a_{2n}$.
- *Step 3:* Xor the two lists into an n -byte sequence denoted with $c_1c_2\dots c_n$
- *Step 4:* Apply the chosen symmetric cryptosystem E (for example DES or AES) with the secret key $KeyE$ on either odd list or even list, and thus create the ciphertext sequence $c_1c_2\dots c_nE_{KeyE}(a_1a_3\dots a_{2n-1})$ or $c_1c_2\dots c_nE_{KeyE}(a_2a_4\dots a_{2n})$ respectively.¹

Clearly, the decryption mechanism at the other end of the communication channel consists of two easy steps: Apply the symmetric cryptosystem E with the appropriate key to the second half of the ciphertext to obtain the first half of the original sequence, and xor this result with the first half of the ciphertext to obtain the other half of the original sequence.

The experiments that were originally performed in 1997 were using DES and IDEA cryptosystems; however, the newer choice for the underlying cryptosystem E would probably be the recently standardized AES cryptosystem (Rijndael). The security of method proposed by Qiao and Nahrstedt is very close to the security of encryption scheme E that is internally used.

In [3], Shi Wang and Bhargava classified their previous work into four different video encryption algorithms: *Algorithm I*, *Algorithm II (VEA)*, *Algorithm III (MVEA)*, and *Algorithm IV (RVEA)*.

The first algorithm, denoted simply by the *Algorithm I*, uses the permutation of Huffman codewords in the I-frames. This method incorporates encryption and compression in one step. The secret part of the algorithm is a permutation π , which is used to permute standard JPEG/MPEG Huffman codeword list. In order to save compression ratio, the permutation π must be such that it only permutes the codewords with the same number of bits. In addition,

the distance between the original and the permuted codeword list must be greater than the encryption quality parameter β , which is chosen by the user.

The security of Algorithm I is not particularly good. In [6], it is shown that the Algorithm I is highly vulnerable to both known-plaintext attack, and ciphertext-only attack. If some of the video frames are known in advance (such as standard introductory jingles and similar), one can reconstruct the secret permutation π by comparing the original and encrypted frames. Vulnerability to this type of attack was also discussed in [3]. However, Algorithm I is also subject to ciphertext-only attack. The *low-frequency error attack* can be applied on ciphertext produced by Algorithm I [6]. Basically, since permutation π is of the special form; i.e., it only shuffles codewords with the same length, the most security comes from shuffling the 16-bit codewords in the AC coefficient entropy table. However, since there is a very limited number of codewords with length of less than 16 bits, it is very easy to reconstruct all of the DC coefficients and most frequent AC coefficients (since these will be encoded with less than 16-bit codewords). In other words, the only hard part would be to figure out how does the permutation π shuffle the 16-bit codewords. But these are appearing extremely rare, and the reconstructed video may be of almost the same quality as the original. Furthermore, the rare pixels that do contain 16-bit codewords can easily be interpolated to get rid of the noise effect and to create the reconstruction image that, at least to the human visual system, appears flawless as the original.

The *Algorithm II (VEA)* uses the following selective encryption observation: it is sufficient to encrypt only the sign bits of the DCT coefficients in an MPEG video. The Algorithm II simply xors the sign bits of the DCT coefficients with a secret m -bit binary key $k = k_1k_2\dots k_m$. The effect of this scheme is that the encryption function randomly changes the sign bits of the DCT coefficients depending on the corresponding bit value of k . Let us denote the sign bits of the DCT coefficients in an MPEG stream that belong to the same GOP (Group Of Pictures) by $s_1s_2\dots s_n$. Then, a bit s_i will not be changed if the value of the key bit $k_{i \pmod m}$ is 0, and it will be flipped if the value of the key bit $k_{i \pmod m}$ is 1. The next GOP would simply reuse the secret key, which serves for the resynchronization purposes. The ability to resynchronize the video stream is important in the case of the unreliable network, transmission errors, and VCR-like functionality such as fast forwarding or rewinding.

The security of Algorithm II depends on the length of the key. The authors encourage the use of a long

¹ The authors' choice was to encrypt the even sequence creating the ciphertext $c_1c_2\dots c_nE_{KeyE}(a_2a_4\dots a_{2n})$

binary key; however, too long key may be infeasible and impractical. On the other hand, if the key is short, the system can be easily broken. If the key is as long as the video stream and it is unique and used only once that would correspond to Vernam cipher (also referred to the one-time pad), which is known to be absolutely secure. However, this is highly impractical for mass applications such as VOD (Video on Demand) and similar. On the other hand, if the key is too short, the whole method simply becomes the Vigenère-like cipher, a well-studied classical cipher for which there are many attacks developed (for example, the Kasiski analysis) [6]. In addition, Vigenère is highly sustainable to the known-plaintext attack (one can literally read off the secret key). Authors in [3] suggest the use of the pseudo-random generator that generates a stream of pseudo-random bits to be the key k of arbitrary length. The trouble with this approach is the security, randomness, and speed of this P-RNG (Pseudo-Random Number Generator). The additional issues include the synchronization of the P-RNG and the secure transmission of the seed (the secret key) for the P-RNG.

The *Algorithm III (MVEA)* is an improvement to the Algorithm II (VEA) described above. It includes the following additions: the sign bits of differential values of motion vectors in P- and B-frames can also be randomly changed. This type of improvement makes the video playback more random and more non-viewable. When the sign bits of differential values of motion vectors are changed, the directions of motion vectors change as well. In addition, the magnitudes of motion vectors change, making the whole video very chaotic. The authors found that the encrypting of sign bits of motion vectors makes the encryption of sign bits of DCT coefficients in B- and P-frames unnecessary.

Furthermore, the original Algorithm III (MVEA) was designed to encrypt only the sign bits of DC coefficients in the I-frames of MPEG video sequence, while leaving the AC coefficients unencrypted. This significantly reduces the computational overhead, but it opens up new security risks. Namely, because the DC coefficient and the sum of all AC coefficients within the block are related, an adversary may use the unencrypted AC coefficients to derive the unknown (encrypted) DC coefficients [3]. For that reason, the authors recommend encrypting all DCT coefficients in the I-frames for applications that need higher level of security.

Just like the Algorithm II (VEA), the algorithm III (MVEA) relies on the secret m -bit key k . Also, the resynchronization is done at the beginning of a GOP. Unfortunately, the fundamental security issues and

problems that are applicable to VEA are also applicable to MVEA.

Finally, the *Algorithm IV (RVEA)* is significantly more secure approach than the previous three algorithms. This approach is considered to be robust under both ciphertext-only attack and known-plaintext attack. The difference between RVEA and MVEA/VEA algorithms is that RVEA uses conventional symmetric key cryptography to encrypt the sign bits of DCT coefficients and the sign bits of motion vectors. The conventional cryptosystems are well mathematically understood, and thoroughly tested by the experts in the field, which definitely adds on to the security aspect of RVEA. The selective approach significantly speeds up the process of conventional encryption by only encrypting certain sign bits in the MPEG stream. The experiments show that the encryption quality is quite good considering the amount of information changed.

In the Algorithm IV (RVEA), the sign bits of DCT coefficients and motion vectors are simply extracted from the MPEG video sequence, encrypted using a fast conventional cryptosystem such as AES, and then restored back to their original position in the encrypted form. The effect of this is similar to VEA/MVEA where the sign bits are either flipped or left unchanged. The authors limited the number of bits for encryption to at most 64 for each MPEG stream macroblock, for the purposes of reducing and bounding the computation time. Next, I describe exactly how these sign bits are selected for encryption.

Each MPEG video slice consists of one or more *macroblocks*. The macroblock corresponds to a 16x16 pixel square, which is composed of four 8x8 pixel luminance blocks denoted by Y_1, Y_2, Y_3 and Y_4 , and two 8x8 chrominance blocks Cb and Cr . Each of these six 8x8 blocks can be expressed as a sequence $ba_1a_2...a_n$, where n is at most 64 since b is the code for DC coefficient and a_i is the code for a non-zero AC coefficient. Then, for each such 8x8 block, we can define the sequence $\beta\alpha_1\alpha_2... \alpha_n$, that corresponds to the sign bits of the matching DCT coefficients b, a_1, a_2, \dots , and a_n . Figure 1 shows the order of selection of the sign bits for each macroblock. The obvious reason for this selection order is that the DC and higher order AC coefficients are carrying much more information than the low-order AC coefficients.

In conclusion, RVEA only encrypts the fraction (typically about 10%) of the whole MPEG video by using the conventional secure cryptographic schemes such as DES, IDEA, AES, etc. Therefore, the Algorithm IV (RVEA) is a much better method than the previous three algorithms in terms of security.

Furthermore, it saves up to 90% of the computation time comparing to the naive approach.

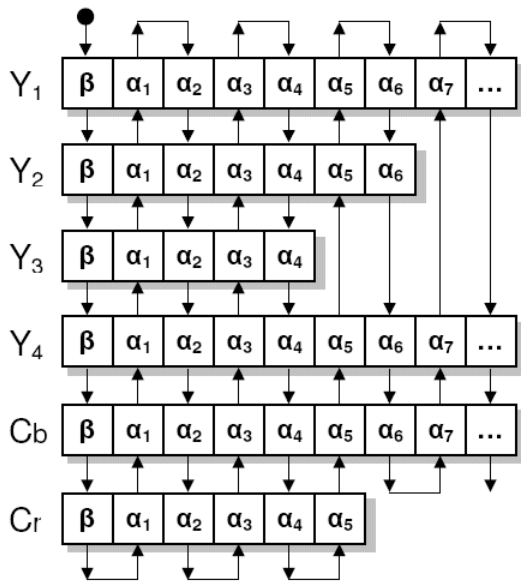


Figure 1. RVEA selection of DCT sign bits to encrypt

3. MPEG-2 Implementation

I have implemented the following 4 methods: VEA by Qiao and Nahrstedt, Algorithm II (VEA), Algorithm III (MVEA) and Algorithm IV (RVEA) by modifying the available MPEG-2 source code. The MPEG-2 code I have used is MPEG-2 Encoder / Decoder, Version 1.2, July 19, 1996, developed by MPEG Software Simulation Group. In addition, for VEA and Algorithm IV (RVEA), which use conventional symmetric key cryptography, I decided to use the Rijndael cryptosystem, which is currently an official Advanced Encryption Standard (AES) originally developed by Joan Daemen and Vincent Rijmen. I have used one of the fastest implementation of AES developed by Szymon Stefanek which can be downloaded from Vincent Rijmen's AES webpage [7]. It is a C++ optimized version of the previous ANSI-C fast implementation of AES by Vincent Rijmen and K.U. Leuven.

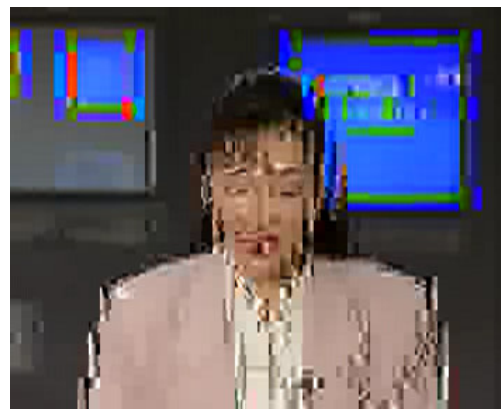
The following screen-shots illustrate the effects of the selective encryption using the aforementioned four algorithms viewable from the ordinary decoder. The special decoder is needed to recover the actual video, since the decoder has to be able to decrypt the selected encrypted bits before playback.



Regular MPEG-2 video (AKIYO Frame#101)



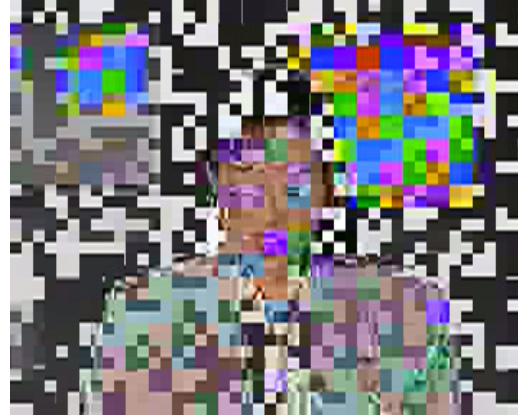
Encrypted sign bits of motion vectors only (AKIYO Frame#101)



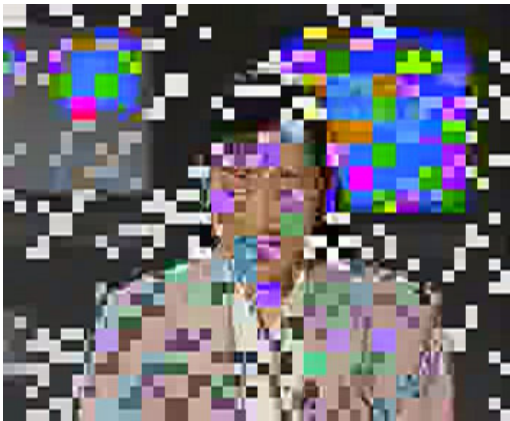
Algorithm II (VEA) - Encrypted sign bits of DCT coefficients (AKIYO Frame#101) with fixed 64-bit key



Algorithm III (MVEA) - Encrypted sign bits of DCT coefficients and Motion Vectors (AKIYO Frame#101) with fixed 64-bit key



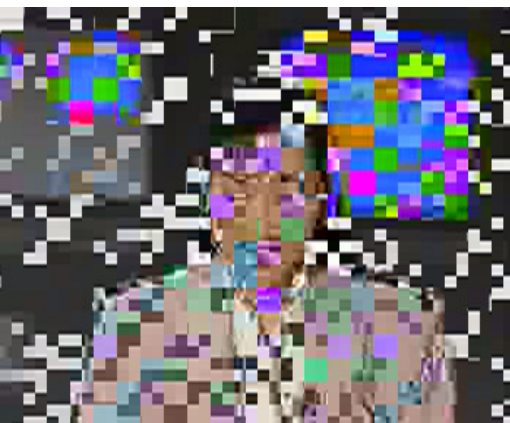
Algorithm IV (RVEA) - Encrypted sign bits of DCT coefficients and Motion Vectors (AKIYO Frame#101) with fixed 256-bit AES



Algorithm II (VEA) - Encrypted sign bits of DCT coefficients (AKIYO Frame#101) with fixed 127-bit key



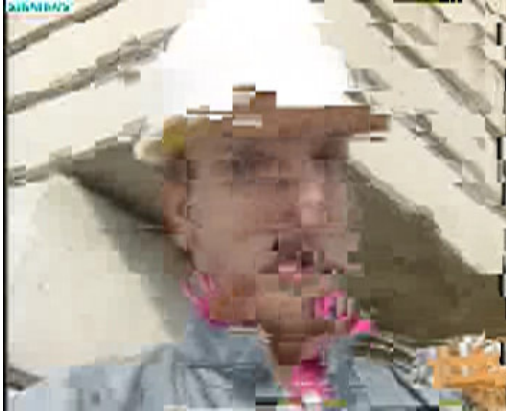
VEA - Encrypted (AKIYO Frame#101) with fixed 256-bit AES



Algorithm III (MVEA) - Encrypted sign bits of DCT coefficients and Motion Vectors (AKIYO Frame#101) with fixed 127-bit key



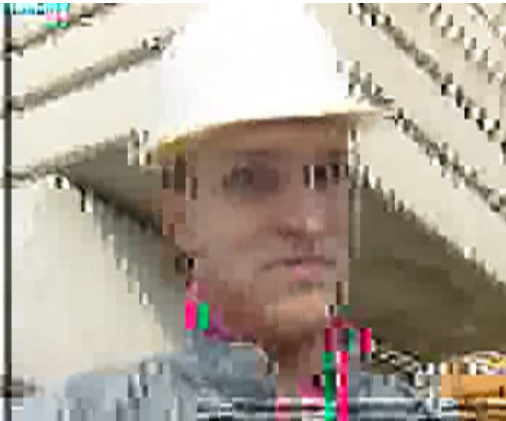
Encrypted sign bits of motion vectors only (FOREMAN Frame#101)



Encrypted sign bits of motion vectors only (FOREMAN Frame#101)



Algorithm II (VEA) - Encrypted sign bits of DCT coefficients (FOREMAN Frame#101) with fixed 127-bit key



Algorithm II (VEA) - Encrypted sign bits of DCT coefficients (FOREMAN Frame#101) with fixed 64-bit key



Algorithm III (MVEA) - Encrypted sign bits of DCT coefficients and Motion Vectors (FOREMAN Frame#101) w/ fixed 127-bit key



Algorithm III (MVEA) - Encrypted sign bits of DCT coefficients and Motion Vectors (FOREMAN Frame#101) with fixed 64-bit key



Algorithm IV (RVEA) - Encrypted sign bits of DCT coefficients and Motion Vectors (FOREMAN Frame#101) w/ fixed 256-bit AES



VEA - Encrypted (FOREMAN Frame#101) with fixed 256-bit AES

4. Performance Analysis

In addition to the theoretical security analysis performed before, by conducting these experiments I was also interested in observing the perceptual quality of encrypted video sequences. Notice that Algorithm II(VEA) and Algorithm III(MVEA) do not have particularly good disguising effect like the other two algorithms. This is especially the case when the fixed secret key is of size 64 (which is also the number of DCT coefficients within the 8x8 block). Taking a secret key that is not a multiple of 64 will further randomize the video picture and the perceptual effect will be better. In addition, Algorithm III(MVEA) performs very similar as Algorithm II (VEA) in respect to the speed and the visual degradation. However, notice that Algorithm III(MVEA) perceptually outperforms Algorithm II(VEA) when it is applied to high-motion video (since there are more motion vectors).

Algorithm IV(RVEA) has descent degrading effect but one can still recognize parts of the video. The video is highly degraded, but we shouldn't be satisfied with this algorithm's performance if we need elevated levels of security, where the picture must be highly distorted. Finally, we can see that Qiao and Nahrstedt's approach yields the best perceptual effect. Even though this is the most expensive method in terms of computation, it is not much slower than the Algorithm IV(RVEA). Thus, I would recommend the use of VEA over the RVEA for more secure applications. For the degradation only, using much faster algorithms Algorithm II(VEA) and Algorithm III(MVEA) is preferred due to high speed, however, a minimal cryptanalysis effort would easily break them since the symmetric key is constant (Vigenère-type).

The following table summarizes observed performance results:

ENCRYPTION ALGORITHM	SEQUENCE	ENCODING TIME
None	Low-motion (Akiya)	111 sec
Algorithm II (VEA) w/ fixed 64-bit key	Low-motion (Akiya)	112 sec
Algorithm III (MVEA) w/ fixed 64-bit key	Low-motion (Akiya)	112 sec
Algorithm II (VEA) w/ fixed 127-bit key	Low-motion (Akiya)	126 sec
Algorithm III (MVEA) w/ fixed 127-bit key	Low-motion (Akiya)	127 sec
Algorithm IV (RVEA) w/ 256-bit AES	Low-motion (Akiya)	155 sec
VEA w/ 256-bit AES	Low-motion (Akiya)	168 sec
None	High-motion (Foreman)	136 sec
Algorithm II (VEA) w/ fixed 64-bit key	High-motion (Foreman)	138 sec
Algorithm III (MVEA) w/ fixed 64-bit key	High-motion (Foreman)	138 sec
Algorithm II (VEA) w/ fixed 127-bit key	High-motion (Foreman)	141 sec
Algorithm III (MVEA) w/ fixed 127-bit key	High-motion (Foreman)	143 sec
Algorithm IV (RVEA) w/ 256-bit AES	High-motion (Foreman)	150 sec
VEA w/ 256-bit AES	High-motion (Foreman)	157 sec

5. Conclusions

It is important to carefully select what level of security needs to be in place and then choose an algorithm. The choice of algorithm will affect the codec speed. I presented some comparative analysis of few selected encryption algorithms that can serve as good criteria for choosing the right encryption algorithm.

6. References

[1] B. Furht and D. Socek, "Multimedia Security: Encryption Techniques," *IEC Comprehensive Report on Information Security*, International Engineering Consortium, Chicago, IL, 2003.

[2] L. Qiao and K. Nahrstedt, "A New Algorithm for MPEG Video Encryption," *Proceedings of the 1st International Conference on Imaging Science, Systems and Technology (CISST '97)*, Las Vegas, NV, July 1997, pp. 21-29.

[3] B. Bhargava, C. Shi, and Y. Wang, "MPEG Video Encryption Algorithms", August 2002, available at <http://raidlab.cs.purdue.edu/papers/mm.ps>

[4] T. Lookabaugh, D. C. Sicker, D. M. Keaton, W. Y. Guo and I. Vedula, "Security Analysis of Selectively Encrypted MPEG-2 Streams," *Multimedia Systems and Applications VI Conference*, Orlando, FL, September 7-11, 2003.

[5] X. Liu and A.M. Eskicioglu "Selective Encryption of Multimedia Content in Distribution Networks: Challenges

and New Directions,” *IASTED International Conference on Communications, Internet and Information Technology* (CIIT 2003), Scottsdale, AZ, November 17-19, 2003.

[6] T. Seidel, D. Socek, and M. Sramka, “Cryptanalysis of Video Encryption Algorithms,” *Proceedings of The 3rd Central European Conference on Cryptology*, TATRACRYPT 2003, Bratislava, Slovak Republic, 2003.

[7] C++ Implementation of AES by Szymon Stefanek available at:
<http://www.esat.kuleuven.ac.be/~rijmen/rijndael/>