AN EFFICIENT, PARALLEL MULTI-KEY ENCRYPTION OF COMPRESSED VIDEO STREAMS

Alexander Wong and William Bishop
Department of Electrical and Computer Engineering, University of Waterloo
Waterloo, Ontario, Canada
a28wong@engmail.uwaterloo.ca, wdbishop@uwaterloo.ca

ABSTRACT
The popularity of digital video storage and delivery systems has given rise to the need for video encryption to preserve secrecy and digital rights. Conventional encryption techniques are not appropriate for securing multimedia systems given the need to encrypt a large amount of data in real-time. This paper presents a selective video encryption algorithm that utilizes multiple stream ciphers and a unique multi-key mechanism to improve security while maintaining efficiency and format compliance. Furthermore, the algorithm is well suited for parallel hardware implementation. Finally, experimental results from the encryption of various test video sequences demonstrate the effectiveness of the video encryption scheme.

KEY WORDS
multi-key encryption, video compression, parallel processing

1. Introduction

Video compression is an important research topic given the increasing need for the efficient storage and distribution of videos in applications such as Video-on-Demand (VOD), video archival, digital video broadcasting, and digital movie playback. The recent popularity of digital video storage and delivery systems has given rise to a need for video encryption to manage digital rights.

A number of methods have been proposed to provide security for digital video streams. These video encryption methods can be classified into two categories:

1. Naive encryption [1-2]
2. Selective encryption [3-9]

Conventional encryption techniques used by existing transmission and storage systems are not well suited for multimedia content for several reasons. First of all, conventional encryption techniques encrypt all multimedia content at the bit-stream level. Such techniques are often computationally expensive since they ensure a high level of security for all encrypted data. Multimedia content is typically very large in size so the computational cost of such techniques is high. Also, many digital video applications (such as VOD) have real-time processing requirements that are difficult to achieve cost effectively using conventional encryption techniques. This point is particularly relevant for consumer devices where computational complexity must be avoided to ensure manufacturability and system reliability.

An ideal video encryption algorithm for real-time video applications in consumer devices must possess the following characteristics:

1. Provide an acceptable level of security
2. Minimize computational and storage overhead
3. Comply with standard video compression formats

To achieve these goals, it is important to exploit the characteristics of the video compression process itself. The most popular video compression schemes are based on block-transform coding, such as MPEG [10-11] and more recently H.264/AVC [12]. Such techniques are popular because they are computationally efficient and have low storage requirements, making them ideal for low-end consumer devices such as set-top boxes and media players. In general, these compression algorithms utilize transforms that have high energy-compaction properties to group important information into a small amount of space. When using these algorithms, certain components of the compressed data stream have greater influence on the decompressed video output than others. This fact can be used to minimize the amount of data that needs to be encrypted to deliver an acceptable level of security. Furthermore, data elements in compressed video content can be divided such that they can be trivially encrypted in parallel using different keys, thereby improving the level of security while maintaining a high level of performance.

The main contribution of this paper is an efficient, parallel video encryption algorithm that utilizes multiple stream ciphers and a unique multi-key mechanism to provide an acceptable level of security for consumer devices while maintaining efficiency. In this paper, the proposed algorithm is described and explained in detail in Section 2. Experimental results demonstrating the

effectiveness of the video encryption scheme are presented in Section 3. Finally, conclusions are drawn in Section 4.

2. Proposed Video Encryption Algorithm

This section introduces the proposed video encryption algorithm. The theory behind partial video encryption is reviewed and a multi-key video encryption technique for parallel hardware implementation is described in detail. An outline of the proposed video encryption algorithm is provided. The parallel architecture for the algorithm and a discussion of the security implications are presented.

2.1 Partial Video Encryption

Video compression algorithms attempt to pack information into as little space as possible. Certain data elements have a greater influence on the decompressed video output than others. This characteristic can be exploited to reduce the amount of video data that needs to be encrypted while maintaining an acceptable level of security. For this reason, it is important to discuss the different data components that are common to modern block-transform based video compression algorithms. It is crucial to find a balance between computational performance and security.

While each block-transform based video compression technique varies in terms of design and implementation details, they are all based on the same fundamental concepts. The encoded video data produced by all techniques share common data components that are present in nearly all such compression standards ranging from MPEG-1 to H.264/AVC. The common data components are the following:

1. Motion vectors
2. DC coefficients
3. AC coefficients

Motion vectors represent the movement of blocks of pixels from one frame to the next and they are used for motion estimation and motion compensation. The DC coefficients represent the average energy of the image. The AC coefficients represent the details of the image. The DC coefficients have a more significant impact on the overall reconstruction of the image than the AC coefficients. This is an important point as the AC coefficients make up the majority of the encoded video data. Computational overhead can be reduced substantially by only encrypting the DC coefficients. Encryption of the DC coefficients yields an acceptable level of security for some applications. It has been shown that it is impossible to derive the DC coefficients given the AC coefficients for a DCT of size 8x8 DCT [13]. Furthermore, research has shown that the encryption of motion vectors is more significant than the encryption of DCT coefficients in B and P frames [14]. Therefore, it is clear that the encryption of important structural information such as DC coefficients and motion vectors is most important.

Partial video encryption can be utilized to deliver different levels of security for each of the common data components. Applications requiring a low level of security need not encrypt all data components. For example, a digital TV service may not require the best possible encryption of its content to achieve its goal of making the video stream unusable. In such a situation, it is desirable to favor low computational complexity over high security. The proposed encryption algorithm targets consumer devices such as commodity set-top boxes and multimedia playback devices. In this context, it is more important to exploit the way information is encoded to deliver real-time performance while providing an acceptable level of security. Therefore, only the sign bits of the DC coefficients and the sign bits of the AC coefficients at the three lowest frequencies are encrypted in the proposed algorithm.

2.2 Parallel Multi-Key Video Encryption

The proposed video encryption algorithm extends the Video Encryption Algorithm (VEA) proposed in [7] by utilizing multi-key encryption and multiple stream ciphers in parallel to provide fast, real-time encryption/decryption performance at a level of security that is sufficient for some consumer digital video applications. Multi-key techniques can be used to significantly improve the security of VEA while maintaining encryption efficiency. In VEA, a single key-stream is used to encrypt the selected video components. Factors such as transmission errors (which can lead to lost video data) and multimedia playback functionality (such as rewind and fast forward) require the ability to resume the decryption and playback process at a specified point in time in a video stream. There must be no dependencies on content prior to the specified point in time. This is accomplished in VEA by the use of resynchronization points, where the secret key is reused from the first bit for encryption at each resynchronization point. The user can start the video stream at any resynchronization point. The main drawback of resynchronization is that it makes VEA vulnerable to plain-text attacks, since the key-stream can be easily determined if an original image and its corresponding encrypted image are known. Since only one key-stream is used, knowledge of the key-stream allows for the decryption of the entire video stream. VEA is also vulnerable to a reused key attack. While Shi and Bhargava [7] suggest that this type of attack can be prevented by requesting new keys periodically, this approach wastes bandwidth and computing cycles. Block ciphers such as DES and AES can also been used to avoid security attacks of this nature, as done in the RVEA

algorithm [6] but the added complexity of block ciphers makes them undesirable for use in some low-cost consumer devices, particularly if a parallel implementation is desired.

The proposed multi-key encryption scheme utilizes multiple encryption keys in such a way that it is less vulnerable to attack. The algorithm only uses the number of keys required to deliver an acceptable level of security. Keys have a longer lifespan in this context but they are used in such a way that security is maintained at an acceptable level. Thus, the proposed multi-key encryption scheme is suitable for parallel implementation as it supports cost-effective, real-time video encryption and decryption.

Prior to explaining the theory, several terms must be introduced. A resynchronization group is defined as a group of frames starting from one resynchronization point to the frame immediately prior to the next resynchronization point, as shown in Figure 1. A common-key group is a sequence of resynchronization groups that make use of the same multi-key set.

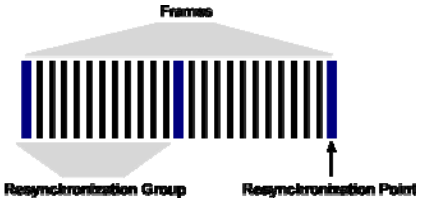


Figure 1. Example of a resynchronization group

The data components that need to be encrypted in a video frame can be divided into n partitions. Using n encryption keys of length m, each partition can be encrypted by a stream cipher using a different key. These keys are referred to as partition encryption keys. The mapping between a partition and its encryption key is determined using a permutation key of length $k\log_2(n!)$ where k is the number of resynchronization groups per common-key group. This is illustrated in Figure 2 for a case where $n = 4$ and $k = 2$.

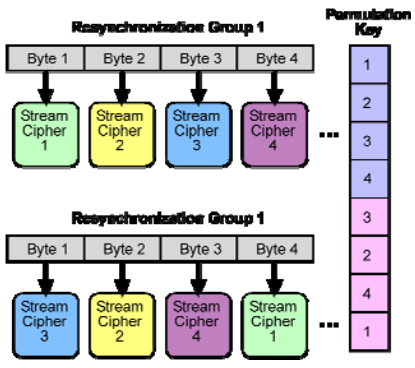


Figure 2. Parallel multi-key video encryption using a permutation key

Consider a scenario where a set of four 256-bit partition keys and a 64-bit permutation key are used for each common-key group. Each video frame in a resynchronization group is divided into four partitions and encrypted using one of the available stream ciphers with one of the four partition encryption keys. The mapping between the partitions and the keys is encoded in 5 bits since there can be a total of 24 permutations of key mappings. Since the permutation key is 64 bits in length, there can be a maximum of 12 resynchronization groups for each common-key group to avoid reusing the permutation key. If each resynchronization group is 1 second long, this algorithm provides acceptable security for 12 seconds of video.

The approach introduced so far can reduce the number of keys needed but it does not solve the real problem. Video streams typically last much longer than 12 seconds. A practical work-around for this problem using the multi-key concept is proposed in the following manner. Let us define a video stream as a sequence of j common-key groups. For each video stream, a set of n encryption keys and nonces is used to generate $n \times j$ encryption keys using a stream cipher as a pseudo-random number generator, as each common-key group requires a set of n encryption keys. These generated keys can then be stored in memory and retrieved depending on the frame that needs to be decrypted. In the case of pre-recorded video, the set of initial keys and nonces can be stored in encrypted form on the physical media. In the case of real-time streaming broadcasts and VOD services, they can be sent in encrypted form through the network. An example of this approach can be illustrated using High Definition DVD (HD-DVD) playback devices. When a disc is inserted into the player, a set of four 256-bit keys and nonces are decrypted and used to generate encryption keys for decrypting the video content. Assuming 10,800 seconds (three hours) of video is held on the disc and each common-key group is 12 seconds long, a total of 900 x 4 = 3,600 256-bit encryption keys are generated and stored in memory. These encryption keys can be easily generated using a modern stream cipher that has a period significantly greater than what is required for this approach. For example, the average period of a key-stream generated by the HC-256 stream cipher is estimated to be 2^{65546} [15]. The 3,600 256-bit keys require less than 115 kB of memory to store. A player can retrieve the appropriate keys to decrypt a video frame based on the common-key group to which it belongs.

While this workaround approach is by no means equivalent to using randomly generated keys, it nevertheless provides the level of security required by some consumer devices. Therefore, the described parallel multi-key extensions to VEA serve as the basis for the proposed encryption algorithm.

2.3 Proposed Encryption Algorithm

Based on the theory presented, the proposed encryption algorithm can be outlined as follows (where n is the number of partitions):

1. Obtain n+1 initial keys and generate n+1 corresponding nonces.
2. Encrypt the initial keys and nonces, along with the total number of common-key groups, and store them if the video content is distributed in prerecorded media. Otherwise, send the encrypted data over the network.
3. Combine the initial keys and nonces from Step 1 to create n+1 initial seeds for n+1 cryptographically secure pseudo-random number generators (CSPRNGs).
4. At the start of each common-key group, generate n encryption keys and one permutation key in parallel using the CSPRNGs.
5. At the start of each resynchronization group, the n encryption keys are used to initialize n stream ciphers.
6. For each frame in the resynchronization group, the data elements that need to be encrypted are selected and are divided into n partitions (first byte in the 1st partition, second byte in the 2nd partition, and etc.). Each partition is encrypted using the stream cipher specified by the permutation key. The partitions are encrypted in parallel. The data elements to be encrypted are:
   a. Sign bits of all DC coefficients
   b. Sign bits of AC coefficients at the three lowest frequencies
7. Repeat Step 6 until the entire video stream has been encrypted.

The decryption process is as follows:

1. Retrieve and decrypt n+1 random keys and n+1 nonces along with the total number of common-key groups.
2. Use the decrypted information to generate the encryption and permutation keys for all common-key groups and store them into memory.
3. At the start of each common-key group, retrieve its associated encryption keys and permutation key from memory.
4. At the start of each resynchronization group, the n partition encryption keys are used to initialize n stream ciphers.
5. For each frame in the resynchronization group, the data elements that need to be decrypted are selected and are divided into n partitions. Each partition is decrypted using the stream cipher specified by the permutation key. The partitions are decrypted in parallel.
6. Repeat Step 5 until the entire video stream has been decrypted.

2.4 Security Implications

The advantage of this scheme in reducing the effectiveness of a plaintext attack can be shown in the following manner. Like VEA implementations, if an attacker knows the plain-text video frame and the corresponding cipher-text video frame, the overall key-stream for the video frame can easily be determined. However, unlike the basic implementation of VEA, the key-stream in the proposed algorithm is composed of multiple unique key-streams. If the key-stream found is used directly to decrypt frames from the next resynchronization group, there is only a probability of $1/(n!)$ that the decrypted frames are correct due to the permutation of key mappings. To determine if a decrypted frame is correct, the attacker must verify the output manually. Therefore, a video frame from that resynchronization group must be decrypted a maximum of $n!$ times even when the key-streams are known, which is computationally expensive given the added cost of video decompression. This can increase the cost of extracting the original video stream to a point where it is cheaper to acquire the media through legal means. Another way the attacker can use the compromised key-streams of a previous resynchronization group in the same common-key group is to try each key with each partition individually and see if the output video is improved. However, given the fact that DC coefficients are differentially coded for intra-blocks in most modern block-transform video compression formats, it is very hard to determine whether the output video is improved as the effects of encryption on one block is propagated to related blocks and therefore make this type of attack intractable.

It has been shown in Section 2.2 that the proposed algorithm is resistant to plaintext attacks compared to the original VEA, while still allowing for the use of stream ciphers that are less computationally complex than block ciphers. Given the fact that a new set of keys are used for each common-key group and that each key is 256-bit in length, ciphertext-only attacks such as brute force exhausting key searches are not practical for the proposed algorithm. This is especially true given the typically large amount of video data to be processed.

2.5 Computational Costs and Parallel Implications

The proposed algorithm is based on VEA. Since only the sign bits of DC coefficients and selected AC coefficients are encrypted and they make up a small portion of the total video stream, it is significantly more efficient than encrypting the entire video stream. Secondly, modern stream ciphers, in general, require significantly fewer computations than block ciphers. Therefore, the use of

stream ciphers in the proposed algorithm allows for great reduction in computational overhead compared to techniques that use modern block ciphers. Moreover, the use of simple stream ciphers lends itself well to a parallel implementation in hardware.

It is also important to discuss the proposed algorithm in the context of parallel computing. As the proposed algorithm is designed to make use of multiple independent stream ciphers to encrypt approximately equal lengths of information, it is ideal for implementation using parallel hardware. The video stream may be partitioned trivially for the purpose of encryption and decryption. The amount of overhead associated with parallelizing this algorithm is very small so the speedup associated with parallelization is linear. For example, if a content stream is encrypted into *n* different partitions that are encrypted in parallel, the ideal speedup is *n* times that of encrypting the video stream serially. While it is true that the hardware cost of parallelization also increases linearly, it should be noted that the stream cipher is relatively simple to build and therefore can be easily replicated.

3. Experimental Results

To evaluate the video scrambling capabilities of the proposed algorithm, the video encryption system was implemented using the popular RC4 stream cipher, using a set of four 256-bit encryption keys and a 64-bit permutation key for each common-key group. The actual security of the stream cipher used for video scrambling evaluation is unimportant, as any good stream cipher should generate what appear to be random binary sequences and so is sufficient for testing purposes. The actual stream cipher to be used for the proposed algorithm can vary depending on the specific application. Three different MPEG-1 compressed video clips were used as test streams. Aside from the ideal scenario where all the selected data is encrypted, a number of scenarios were also tested to evaluate the output when the video stream is compromised to different extents. Therefore, the following scenarios were tested:

1. No partitions have been compromised
2. ¼ of the partitions have been compromised
3. ½ of the partitions have been compromised
4. ¾ of the partitions have been compromised

The average PSNR for each encrypted video stream is shown in Table 1. A decoded frame under the different scenarios is shown in Figure 3. It can be observed that the output is incomprehensible when all the selected data is encrypted. It can also be observed from the compromised output frames that the decoded video stream remains mostly incomprehensible until ¾ of the video content is compromised. However, the decoded output video stream is still degraded enough under this scenario to be considered unwatchable. In terms of efficiency, the average computational overhead of the proposed algorithm over 10 test trials is found to be 2.16% of the total computational time required to decrypt a video stream. Therefore, it is clear that the proposed algorithm is suitable for securing video content in low-cost consumer devices.

TABLE 1
AVERAGE PSNR OF ENCRYPTED VIDEO STREAMS

Level of Security Compromise	Average PSNR[1] (dB)		
	TENNIS	CARTOON	BIKE
Uncompromised	6.8103	7.1938	6.5662
¼ of partitions	6.0696	5.1229	6.7165
½ of partitions	6.5017	6.9985	6.0235
¾ of partitions	7.1498	7.7693	7.4561

[1] The average PSNR was calculated over a sequence of 10 frames for each video stream. The PSNR for each frame is calculated between the compressed video frame and the encrypted video frame.

4. Conclusions

This paper presents an efficient parallel video encryption algorithm suitable for consumer devices. Partial video encryption techniques are used to significantly reduce the computational overhead associated with encryption while achieving an acceptable level of security. Multi-key encryption and parallel stream ciphers are used to improve both security and computational performance. Experimental results from the encryption of various test video sequences demonstrate the effectiveness of the video encryption scheme. It is our belief that this method can be successfully implemented in low-cost consumer devices such as set-top boxes and digital movie disc players. Future work includes the design and implementation of a parallel video stream encryption processor based on the proposed algorithm.

Acknowledgements

This research has been sponsored in part by Epson Canada and the Natural Sciences and Engineering Research Council of Canada.

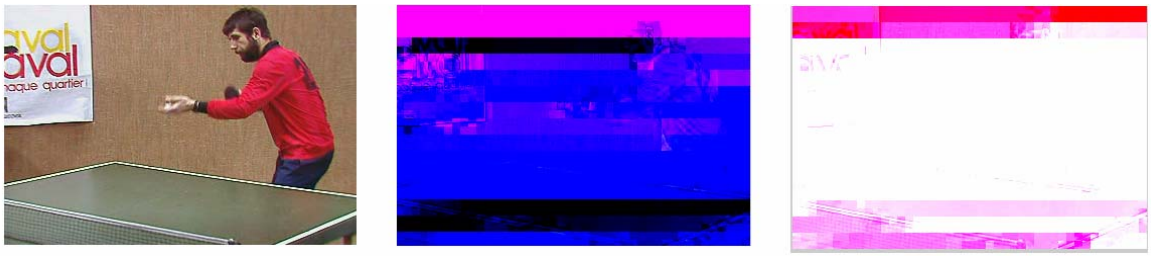
References

[1] I. Agi and L. Gong, An Empirical Study of Secure MPEG Video Transmissions, *Proceedings of the 1996 Symposium on Networks and Distributed System Security*, San Diego, California, 1996, 137–144.

[2] L. Tang, Methods for Encrypting and Decrypting MPEG Video Data Efficiently, *Proceedings of the 4th ACM International Conference on Multimedia*, Boston, Massachussetts, 1996, 219-229.

[3] S. Lian, J. Sun, Z. Wang, and Y. Dai, A Fast Video Encryption Scheme Based on Chaos, *Proceedings of the 8th International Conference on Automation, Robotics, and Vision*, Kunming, China, 2004, 126-131.

[4] S. Lian, Z. Wang, and J. Sun, A Fast Video Encryption Scheme Suitable for Network Applications, *Proceedings of the International Conference on Communications, Circuits, and Systems*, Chengdu, China, 2004, 566-570.
[5] L. Qiao and K. Nahrstedt, A New Algorithm for MPEG Video Encryption, *Proceedings of the First International Conference on Imaging Science, Systems, and Technology*, Las Vegas, Nevada, 1997, 21-29.
[6] C. Shi, S. Wang, and B. Bhargava, "MPEG Video Encryption in Real-Time Using Secret Key Cryptography," *Proceedings of Parallel and Distributed Processing Techniques and Applications*, Las Vegas, Nevada, 1999.
[7] C. Shi and B. Bhargava, A Fast MPEG Video Encryption Algorithm, *Proceedings of 6th ACM International Conference on Multimedia*, Bristol, England, 1998, 81-88.
[8] C. Shi and B. Bhargava, An Efficient MPEG Video Encryption Algorithm, *Proceedings of the 17th Symposium on Reliable Distributed Systems*, West Lafayette, Indiana, 1998, 381-386.
[9] T. B. Maples and G. A. Spanos, Performance Study of a Selective Encryption Scheme for the Security of Networked, Real-time Video, *Proceedings of the 4th International Conference on Computer Communications and Networks*, Las Vegas, Nevada, 1995.
[10] ISO/IEC 11172, Coding of Moving Pictures and Associated Audio for Digital Storage Media Up to About 1.5 Mbits/s – Part 2: Video, 1993.
[11] ISO/IEC 13818-2, Generic Coding of Moving Pictures and Associated Audio Information – Part 2: Video, 1994.
[12] Joint Video Team of ITU-T and ISO/IEC JTC 1, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), 2003.
[13] S. Li and B. Bhargava, A Note on "MPEG Video Encryption Algorithms", available at http://www.hooklee.com/MMTA2004_note.pdf, 2004.
[14] B. Bhargava, C. Shi, and Y. Wang, MPEG Video Encryption Algorithms, *Multimedia Tools and Applications*, *24*(1), 2004, 57-79.
[15] H. Wu, A New Stream Cipher HC-256, *Cryptology ePrint Archive*, 2004, available at http://eprint.iacr.org/2004/.



(a) Original (b) Uncompromised (c) ¼ of partitions compromised

(d) ½ of partitions compromised (e) ¾ of partitions compromised

Figure 3. A frame from the TENNIS video clip