Partial Video Encryption Based on Alternating Transforms

Siu-Kei Au Yeung, Shuyuan Zhu, *Student Member, IEEE*, and Bing Zeng, *Member, IEEE*

***Abstract*—In this letter, we propose a novel video encryption technique that is used to achieve *partial* encryption where an annoying video can still be reconstructed even without the security key. In contrast to the existing methods where the encryption usually takes place at the entropy-coding stage or the bit-stream level, our proposed scheme embeds the encryption at the *transform* stage during the encoding process. To this end, we develop a number of new unitary transforms that are demonstrated to be equally efficient as the well-known DCT and thus used as alternates to DCT during the encoding process. Partial encryption is achieved through alternately applying these transforms to individual blocks according to a pre-designed secret key. Analysis on the security level of this partial encryption scheme is carried out against various common attacks and some experimental results based on H.264/AVC are presented.**

***Index Terms*—H264/AVC, partial encryption, unitary transforms, video encryption.**

I. Introduction

With the increase of multimedia applications over the Internet, security of digital video information becomes more and more important. In practice, digital video signals are often compressed before transmission and several video coding standards such as MPEG2, MPEG4, and H.264/AVC can be chosen for this purpose. These coding standards themselves do not provide any data encryption schemes. One early method is to apply an authentication control mechanism such that users are required to provide security information (e.g., password) before they can access the data. Clearly, more secured method is to encrypt the whole video content by some encryption technique such as Data Encryption Standard (DES) or Advanced Encryption Standard (AES) [1]. The major problem here is that these encryption algorithms often require a long processing time.

On the other hand, we understand that the value of video data broadcasted over the Internets is usually far less than that of other types of data such as financial or military information. Therefore, it is usually not worthy of running a complicated attacking algorithm on the huge amount of a video sequence. Furthermore, due to the use of motion-compensation in most video coding schemes, any mistakes during the attacking will bring a huge negative impact onto all the following P-frames. Because of these, full video encryption seems to be unnecessary in some applications, whereas a partial encryption at very low cost would be highly preferable in which users can still reconstruct "annoying" video frames to perceive the contents but at a very bad quality.[1]

A number of partial video encryption techniques have been proposed in literature. For instance, an encryption algorithm named SECMPEG is proposed in [2], which selectively encrypts the header information of each frame. However, since the header data usually follows a standard format, it is still possible to guess it or decode the video even when the header is missing. Algorithms are then proposed in [3] to change the sign bits of some DCT coefficients (most likely the DC component in each block) and/or encrypt motion vectors accordingly. Other but similar algorithms are proposed in [4]–[6], which are applied either at the entropy coding stage, in the compressed domain, or on some DCT coefficients.

In this letter, we propose to carry out the partial video encryption at a stage that has never been considered before: the *transform* stage where a number of (unitary) transforms are employed alternately according to a security key. To this goal, we need more transforms rather than just DCT and these new transforms should be equally efficient as DCT in encoding all residual frames that are obtained after the motion compensation. Clearly, this encryption mechanism can be combined with all existing ones (on the DCT coefficients, during the entropy coding, in the bit-stream domain, on motion vectors, etc.) so that the encryption space has been expanded by a new *dimension*! Consequently, this may raise the difficulty of attacking by one degree of magnitude or even more.

II. New Unitary Transforms for Video Encoding

In this work, we choose the H.264/AVC standard [7] and thus focus on image/video blocks of size 4×4. When DCT is used, each 1-D transform can be implemented through the flow graph as shown in Fig. 1. It is seen from Fig. 1 that the whole graph consists of three stages: 1) two butterflies that are both equivalent to the plane-based rotation of $\pi/4$; 2) two extra plane-based rotations of $\pi/4$ and $3\pi/8$, respectively; and 3) a permutation. Based on this interpretation, it is easy to know that, while keeping Stages 1 and 3 unchanged, a class of new unitary transforms (of 4-points) can be generated by using different rotation angles in Stage 2. For example, by changing both $\pi/4$ and $3\pi/8$ to $29\pi/90(\sim 58^\circ)$, we obtain the popular discrete sine transform (DST) of type-I, whereas the DST of type-II will be obtained by swapping $\pi/4$ with $3\pi/8$. Clearly, more alternates can be generated by selecting different rotation angles.

It is well-known that DCT outperforms other (fixed) unitary transforms when the correlation among pixels is very high.

Manuscript received May 04, 2009, revised June 12, 2009. First published June 23, 2009; current version published July 29, 2009. This work was supported in part by an RGC research grant of the HKSAR government. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. H. Vicky Zhao.

The authors are with the ECE Department, Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong SAR, China (e-mail: jeffay@ust.hk; eezhshy@ust.hk, eezeng@ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LSP.2009.2026109

[1]From a business point of view, it is a good idea if such a video of poor quality can be provided so that potential customers can get some impression of the video content before deciding to subscribe or not.

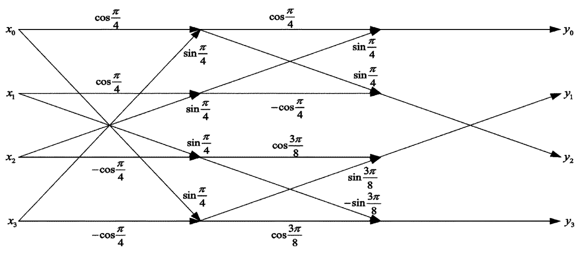


Fig. 1. Flow graph of 4-point DCT (1-D).

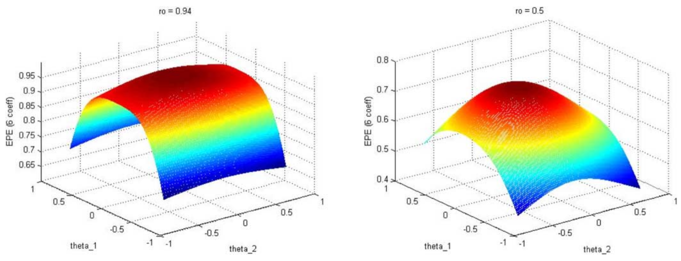


Fig. 2. EPE's for different rotation angles ($\rho = 0.94$ and 0.5).

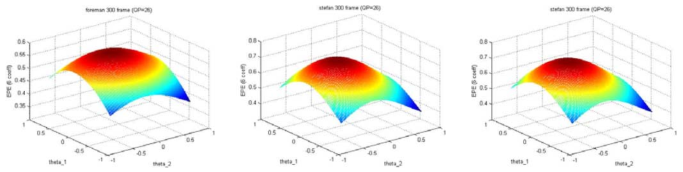


Fig. 3. EPE's for different rotation angles with different test video sequences.

However, this superiority is questionable when DCT is applied on the residual data in a motion-compensation video encoder as such residual data has much weaker correlation among pixels [8], as demonstrated in the following.

A. *Theoretical Analysis on Coding Efficiency*

We follow the commonly-used circular model to assume that the correlation between two samples is calculated as

$$E\{p_{i,j}, p_{m,n}\} = \rho^{\sqrt{(m-i)^2+(n-j)^2}} \tag{1}$$

where $0 < \rho < 1$. Let $\{S_{u,v}\}$ be the set of transform coefficients and $\{c_{i,j}\}$ the $N \times N$ transform matrix ($N = 4$ in our case). Then, the variance of each transform coefficient can be calculated as

$$\begin{aligned}\sigma_{u,v}^2 &= E\{S_{u,v} \cdot S_{u,v}\}\\ &= \sum_{i,j,m,n=0}^{N-1} E\{p_{i,j}p_{m,n}\} \cdot (c_{u,i}c_{u,m}c_{v,j}c_{v,n}).\end{aligned} \tag{2}$$

To measure the coding efficiency of a transform in the case that some quantization is carried out on transform coefficients, we can measure the energy packing efficiency (EPE) that is defined as the energy portion contained in the first M_0 transform coefficients (along the zig-zag scanning order) compared with the total energy. In our evaluation, we set $M_0 = 6$:

$$EPE = \sum_{t=0}^{5} \frac{E\{S_t^2\}}{\sum\limits_{t=0}^{15} E\{S_t^2\}}. \tag{3}$$

In order to evaluate various transforms with different rotation angles as described above, we set two rotation angles in Stage 2 of Fig. 1 to be $\pi/4 + \theta_1$ and $3\pi/8 + \theta_2$. We compute EPE's via adjusting both θ_1 and θ_2 from $-\pi/4$ to $\pi/4$. Fig. 2 shows the results with $\rho = 0.94$ and 0.5, respectively. In the case with a very high correlation, we observed that EPE achieves its maximum when both θ_1 and θ_2 are set to 0—which is the original DCT case. However, when the correlation gets weaker—which is the situation in encoding of residue data, two rotation angles θ_1 and θ_2 that achieve the maximum also shift. From the right part of Fig. 2, we observed that θ_1 tends to be positive (between 0 and $\pi/8$) whereas θ_2 goes negative (between 0 and $-\pi/8$). To further verify these results, we performed experiments on real video sequences (*Foreman*, *Mobile*, and *Stefan*, 300 frames in the CIF format). Fig. 3 shows the results with $\text{QP} = 26$ (in H.264). It is clear that the results in Fig. 3 match very well to those shown in Fig. 2 ($\rho = 0.5$). Based on our simulation results, it is worthy to be pointed out that the optimal rotation angles seem to be independent of the QP factor.

B. *Practical Coding Efficiency of New Transforms*

In order to evaluate the actual performance of the new transforms (of different rotation angles), we tested the following two cases: 1) $\theta_1 = \pi/24$, $\theta_2 = -\pi/24$; and 2) $\theta_1 = -\pi/24$, $\theta_2 = \pi/24$, and compared them with two traditional transforms: DCT and DST-II, using the H.264 encoder. Two new cases give rotation angles $(7\pi/24, 8\pi/24)$ and $(8\pi/24, 7\pi/24)$, respectively in Stage 2 of Fig. 1, which are actually the equal-space samples between two original angles $\pi/4$ and $3\pi/8$. All transforms are implemented using the floating-point arithmetic. Fig. 4 reports the results for the same 3 test sequences from which we observe that the two new transforms indeed work similarly as or sometimes even better than DCT and/or DST-II.

Notice that we have chosen the floating-point arithmetic in conducting the tests mentioned above, whereas H.264 runs on integer arithmetic. However, following the similar principle, we can also develop new integer-based transforms and then use them together with H.264's integer DCT. For instance, integer DST has been designed and used together with integer DCT in an alternative fashion [8]. More results will be reported in our future works.

III. Partial Encryption by Alternating Transforms

Our partial video encryption scheme is divided into two parts: 1) random (secret) key generation and 2) alternating transforms according to the secrete key.

A. *Random Key Generation*

Almost all video encryption algorithms rely on a random key generator to produce a sequence of pseudo-random codes. To this end, RC4 turns to be the most commonly-used random key generator [9], and it has been adopted in popular protocols such as SSL and WEP. The permutation is initialized with a variable length key which has typically 40 to 256 bits (we set it to 128 bits in our experiments) and two 8-bit random pointers. The key-scheduling algorithm (KSA) is first applied. The key-stream is then generated using the pseudo-random generation algorithm (PRGA).

To implement one 128-bit random generation and two 8-bit random generations in the RC4 key generator, we roughly need 12 additions and 18 shifts (including 4-bit, 16-bit, and 32-bit

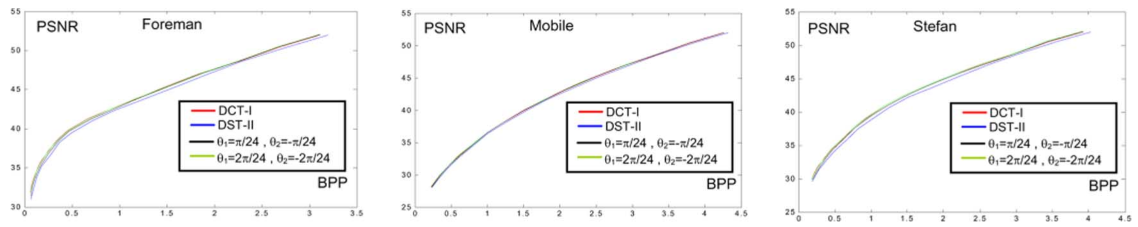


Fig. 4. R-D performances of two new unitary transforms (with different rotation angles) in comparison with DCT and DST-II, tested on three video sequences (*Foreman*, *Mobile*, and *Stefan*).

shifts). In addition, we also need 1024 additions, 512 16-bit shifts, and 512 4-bit shifts for KSA (2 shifts and 1 addition are needed for each mod operation). On the other hand, during the decoding of one video frame of the CIF format, let's assume that 50% of blocks (i.e., 3168 blocks) have non-zero quantized coefficients. It is estimated that the total operations include 101 376 additions, 12 672 1-bit shifts, and 50 688 K-bit shifts (K depends on the QP value): each 4×4 block require 32 additions, four 1-bit shifts, and 16 K-bit shifts. Clearly, the computational complexity from the key generation is negligible ($\sim 1\%$) when compared with that from the decoding of one video frame.

B. Alternating DCT/DST Encoding Based on the Random Key

Our proposed video encryption procedure is stated as follows:

```
------------Encryption Algorithm------------
Step 1: $2^{M-1}$ transforms are designed by using different rotation angles
Step 2: Initialize the RC4 key generator by a random 128-bit key
Step 3: For an input residual block of size 4×4, do
    Step 3.1: Get $M$ bits from the random generator
    Step 3.2: The first $M-1$ bits are marked as Dec; a transform is selected according to Dec
    Step 3.3: The $M^{th}$ bit is marked as Sign; change the DC component's sign if Sign = "1"
Step 4: Go back to Step 3
Step 5: Go back to Step 2 after finishing one frame
```

At the decoder side, the pseudo-random key will be reproduced so that the decoder can make all decisions correctly. It is determined by the user and clearly it controls the security level—more alternating transforms provides a higher security level. To further increase the security level at a given M, we can apply different transforms (alternately according to the random key) along various columns and rows within each block. To accommodate this case, Step 3 needs to be modified to

```
Step 3: For an input residual block of size 4×4, do
    Step 3.1: Get $8\times(M-1)+1$ bits from the random generator
    Step 3.2: The first $8\times(M-1)$ bits are marked as Dec; 4 vertical transforms and 4 horizontal transforms are determined
              according to Dec - each segment of $M-1$ bits determines the transform used for each column or row …
```

In Step 3.3, we performed a sign-flip on the DC component, whereas the decoder can do the back-flip when the key is available.

IV. Experimental Results and Security Analysis

It is always not an easy task to evaluate the effectiveness of a video encryption scheme because there are many possible attacks. In this section, we try to provide some analysis of our proposed algorithm under a few common attacks during the decoding [10], [11].

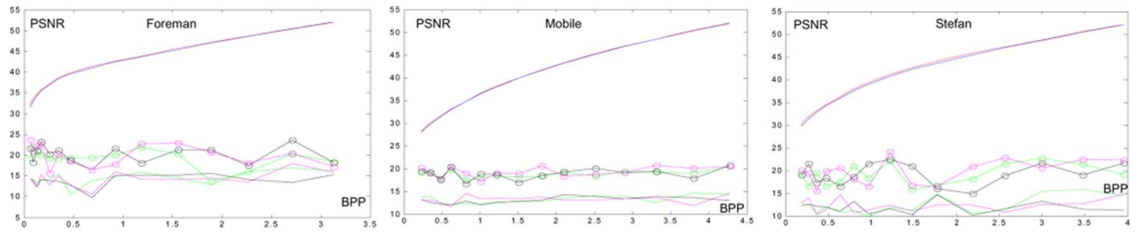
A. Known-Plaintext and Chosen-Plaintext Attacks

For the known-plaintext attack, cryptanalysts have the original video data as well as its encrypted data. For the chosen-plaintext attack, a simple "black-box" is available such that cryptanalysts even be able to obtain any encrypted sequence by providing different pieces of input signal. Under these two attacks, cryptanalysts would be able to obtain the key (for the RC4 key random generator) after analyzing a long enough sequence of data. For known-plaintext, some synchronous issues need to be handled because the attackers need to determine which bits within a byte are used for encryption since they have no idea about the starting point of the encryption. Of course, a more secured key generator can offer a better protection against those attacks; however more computation power is needed to generate the key. One simple and efficient method we applied in our proposed algorithm is to refresh all keys (128-bit keys and two 8-bit pointers) every frame. After refreshing, it becomes useless to analyze all previously-encrypted frames.

B. Ciphertext-Only Attacks

A more realistic attack is the ciphertext-only in which only encrypted data is available. The aim of the security analysis here is to see how much visual information attackers can recover under this scenario. One of the typical approaches for ciphertext-only attacks is to use the so-called brute-force method, i.e., try all possible keys. The key space is 2^{128+16} for the generator used in our algorithm, which definitely is not feasible for anyone to guess. On the other hand, however, attackers can try all possible transforms during the decoding process (suppose they somehow obtain these transforms). In our algorithm as described above, we have eight selections per block (four horizontal and four vertical) and each time can select one from four transforms, which amounts to $4^8 = 65535$ combinations per block. Meanwhile, there are maximally 6336 blocks per frame of CIF format. Clearly, it is not feasible either to try all combinations for all blocks. In addition, any incorrect guess would propagate errors to other frames due to the motion estimation process. As a conclusion, we believe that it is extremely difficult to recover the signal by using the brute-force approach.

Fig. 5 shows some simulation results achieved with our proposed encryption algorithm. Here, we select one transform from four candidates for each column or row within a block according to the generated random key. It is observed from Fig. 5 that there exists no difference on the coding efficiency between the unencrypted signal and the encrypted one if the security key is known. To evaluate its security level, we tried to decode the encrypted video by assuming that 0, 1, or 2 transform(s) adopted at the encoder-side is/are somehow known, while the decoder



—: DCT; —: Decrypt with key; —: Decrypt with random key; —: Decrypt with random key (1 transform known); —: Decrypt with random key (2 transforms known); O: sign of DC known.

Fig. 5. R-D performance for the proposed encryption algorithm with and without the security key.

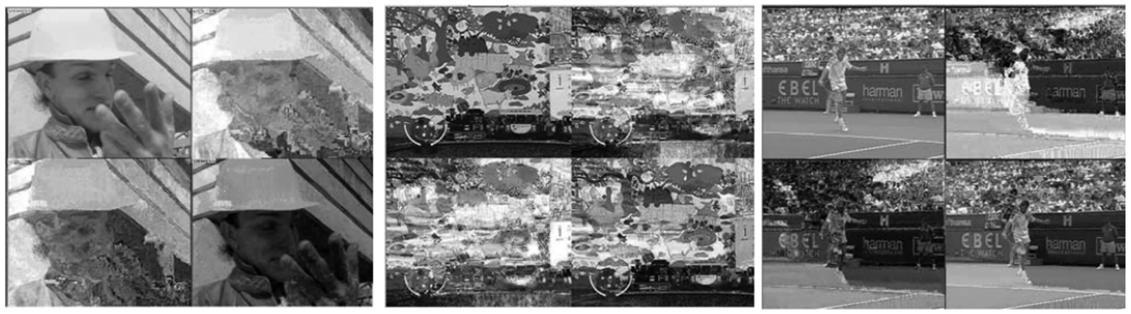


Fig. 6. Performance of our proposed encryption: top-left: decode with key; top-right: random key for all transform and DC sign; bottom-left: two transforms (out of four) known; and bottom-right: two transforms (out of four) and DC sign known.

randomly selects a transform when the transform adopted at the encoder side is not known. Experimental result shows that there is no significant benefit even when attackers can guess two out of four transforms correctly.

To show the visual performance of our proposed algorithm, we present some snapshots of three test video sequences in Fig. 6. It is clear from Fig. 6 that the visual quality of the decoded video frames is indeed extremely bad if the key is not completely available. In the meantime, however, it is also noticed that some visual information has not been destroyed completely, especially in the case when the DC sign is known.

V. Conclusions

In this letter, we introduced a class of unitary transforms that are generated by some plane-based rotations in a high-dimensional space. It is demonstrated both theoretically and experimentally that some of these new transforms perform as equally well as or even slightly better than DCT in encoding residual frames in typical video coding scenarios. As an application, we selected a few such new transforms as alternates to DCT to be used in the H.264 standard to build a partial video encryption at the transform stage—a place that has never been considered before. Thus, the space in which people can do encryption has been increased by one dimension! Experimental results showed that the proposed algorithm does provide a sufficient level of encryption without affecting the overall video quality and requires only a few overhead-bits.

As the final remark, we would like to re-emphasize that the results presented in this work are obtained by only considering the encryption at the transform stage, whereas other encryptions (at the entropy-coding stage, in the bit-stream level, on motion vectors, etc.) are not included *intentionally*, showing that there does exist a big room for encryption at the transform stage. Although these results do not offer a high security, they do serve our purpose well: *partial* video encryption at a low cost. Clearly, more secured encryptions can be obtained easily by combining our algorithm with the existing ones.

References

[1] D. Stinson, *Cryptography Theory and Practice*. London, U.K.: Chapman & Hall/CRC, 2006.
[2] J. Meyer and F. Gadegest, "Security mechanisms for multimedia data with the example MPEG-1 video," in *Project Description of SECMPEG*, 1995.
[3] B. Bhargava, C. Shi, and S. Wang, "MPEG video encryption algorithms," *Multimedia Tools Applicat.*, vol. 24, pp. 57–79, 2004.
[4] M. Cai, J. Jia, and L. Yan, "An H.264 video encryption algorithm based on entropy coding," in *Int. Conf. Intelligent Information Hiding and Multimedia Signal*, 2007, pp. 41–44.
[5] R. Iqbal, S. Shirmohammadi, and A. Saddik, "Compressed- domain encryption of adapted H.264 video," in *IEEE Int. Symp. Multimedia*, 2006, pp. 979–984.
[6] S. Spinsante, F. Chiaraluce, and E. Gambi, "Masking video information by partial encryption of H.264/AVC coding parameters," *EUSIPCO-2005*, pp. 1338–1441, 2005.
[7] I. Richardson, *H.264 and MPEG-4 Video Compression*. Hoboken, NJ: Wiley, 2003.
[8] S. Lim, D. Kim, and Y. Lee, "Alternative transform based on the correlation of the residual signal," *Int. Congr. Image and Signal Processing*, pp. 389–393, 2008.
[9] K. Kaukonen and R. Thayer, "A stream cipher encryption algorithm "Arcfour"," *IETF Draft*, 1999.
[10] C. Wu and C. Kuo, "Design of integrated multimedia compression and encryption systems," *IEEE Trans. Multimedia*, vol. 6, pp. 828–839, 2005.
[11] D. Xie and C. Kuo, "Multimedia data encryption via random rotation in partitioned bit streams," in *ISCAS-2005*, 2005, pp. 5533–5536.