EFFICIENT AND SECURE ENCRYPTION SCHEMES FOR JPEG2000

Hongjun Wu, Di Ma

Institute for Infocomm Research
21, Heng Mui Keng terrace, Singapore 119613
{hongjun, madi}@i2r.a-star.edu.sg

ABSTRACT

The JPEG2000 syntax requires that any two consecutive bytes in the encrypted packet body should not be larger than 0xFF8F. This stringent requirement has plagued researchers for a few years and no satisfactory solution has been proposed. In this paper, we successfully developed efficient, secure and format-compliant encryption schemes for JPEG 2000. Any secure encryption algorithm (stream cipher or block cipher) can be used in our schemes. The new schemes are remarkably efficient and introduce only extremely small amount of extra computation. The new schemes are highly secure and it is proved that they perfectly protect 99.15% of the information of an image. The encryption tools described in this paper have been proposed to the JPSEC (JPEG 2000 Security) standardization group.

1. INTRODUCTION

One of the well-known image compression standards is JPEG. JPEG stands for Joint Photographic Experts Group, a community of experts that was formed under the auspices of the ISO in the mid 1980's to develop a standard for still image coding. Since then, an evolution of image compression technology has taken place and a much more superior image compression standard called JPEG2000 has been formed recently by ISO/IEC JTC/SC29/WG1 (the working group charged with the development of JPEG2000 standard). The major intention of JPEG2000 is twofold [1]. Firstly, it is designed to address most of the limitations of the original JPEG standard. Secondly, it intends to cater for the widening of application areas for JPEG technology. In addition to excellent coding performance and good error resilience, a remarkable merit of JPEG2000 is its "*compress once, decompress many ways*" functionality, i. e., it supports extraction of images with different resolutions, quality layers and regions-of-interest (ROIs), all from the same compressed code-stream.

JPEG2000 refers to all parts of the standard: Part 1 (the core) [2] is now published as an international standard, five more parts (Parts 2-6) are complete or near complete, and four new parts (Parts 8-11) are under development. In particular, Part 8 of this standard, named JPSEC, is concerned with all the security aspects of JPEG2000 code-streams and files. This paper presents our encryption proposal submitted to JPSEC.

1.1. Related work on multimedia encryption

Due to the popularity of Internet and digital library applications, the intellectual property right (IPR) protection is becoming increasingly important. Encryption is frequently used to protect the multimedia content. A big challenge for multimedia encryption (in the compressed domain) is to maintain syntax compliance. In the bit-streams of multimedia data, there are special bit patterns, such as headers and markers, being used for synchronization purposes. We know that the direct encryption with the existing ciphers gives completely random outputs. It results in false markers and leads to the loss of synchronization information. So the multimedia encryption requires special encryption scheme to achieve syntax compliance.

Two methods for MPEG-4 video encryption in the compressed domain [3] have been adopted into the MPEG-4 Intellectual Property Management and Protection (IPMP) Final Proposed Draft Amendment. However, due to the completely different syntax of MPEG and JPEG2000, no encryption scheme proposed for MPEG can be applied to JPEG2000. The encryption proposal from Canon [4] also addresses the issue of syntax compliance.

1.2. Outline of the paper

The paper is organized as follows. Section 2 illustrates the JPEG2000 code-stream structure. Section 3 introduces our encryption scheme for stream cipher and gives the security and performance analysis. Section 4 introduces the encryption scheme for block cipher. Section 5 shows our experiment results. Section 6 concludes this paper.

2. JPEG2000 CODE-STREAM STRUCTURES

We provide here a brief description of the JPEG2000 code-stream structures. The details can be obtained from [1-3]. The JPEG2000 coded image data is referred to as code-stream, instead of bit-stream as in JPEG and MPEG. The umbrella term "code-stream" refers to both the coded image data and the signaling markers and marker segments which are used to locate and describe coding parameters and auxiliary information.

In the simplest case, a JPEG2000 code-stream is structured as a main header followed by a sequence of tile-streams. The code-stream is terminated by a two-byte marker, EOC (end of code-stream). This is depicted graphically in Figure 1.1. The main header consists of markers and marker segments containing global information necessary for decompression of the entire code-stream. Each tile-stream consists of a tile header followed by the compressed *pack-stream* data for a single tile. Each tile header consists of markers and marker segments containing the information necessary for decompressing the pack-stream of its associated tile. Finally, the pack-stream of a tile consists of a sequence of *packets*.

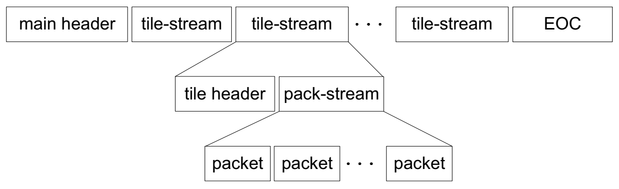


FIGURE 1.1 THE JPEG2000 CODE-STREAM.

The anatomy of a packet is depicted in Figure 1.2. A packet consists of a packet header followed by a packet body. To note, the Standard ensures none of the code-stream's delimiting marker codes (these all lie in the range 0xFF90 through 0xFFFF) can appear in the pack-stream except marker segment SOP (start of packet) and marker EPH (end of packet header).

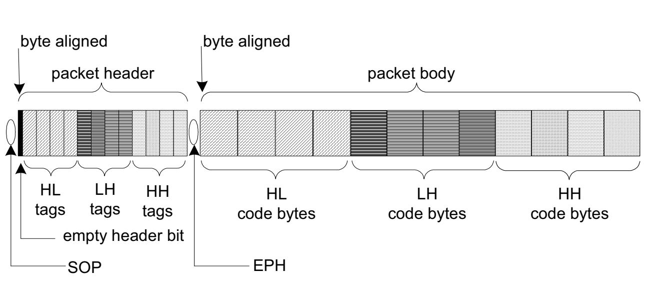


Figure 1.2 JPEG 2000 Packet Structure

3. ENCRYPTION SCHEME WITH STREAM CIPHER

The encryption scheme is to encrypt the packet body. According to the JPEG2000 code-stream structures, the marker codes (these all lie in the range 0xFF90 through 0xFFFF) should not appear in the packet body. We propose here a very simple and efficient encryption scheme for protecting JPEG2000 code-streams and files with the stream cipher. Our scheme protects 99.15% of the information of the JPEG2000 coded image data. It is syntax compliant and keeps the length of the coded image data unchanged. In the next section, we present a variant of this scheme based on the block cipher.

3.1. The encryption/decryption algorithms

Denote a packet body as $M = m_1 \| m_2 \| \cdots \| m_n$, where $\|$ denotes concatenation and each m_i denotes one byte. Denote the ciphertext as $C = c_1 \| c_2 \| c_3 \| \cdots \| c_n$, where each c_i denotes one byte.

<u>Encryption Algorithm</u>

Use a stream cipher to generate a key stream from the secret key and the initialization vector. Consider the key stream as a byte sequence. Discard those bytes with value 0*x*FF. Denote the resulting key stream as $S = s_1 \| s_2 \| \cdots \| s_n$, where each s_i denotes one byte and $0 \le s_i < 0\text{xFF}$

```
if $m_1 = 0\text{xFF}$, then $c_1 = m_1$;
if $m_1 \ne 0\text{xFF}$, then $c_1 = (m_1 + s_1) \bmod 0\text{xFF}$;
for $i = 2$ to $n$,
    if $m_i = 0\text{xFF}$ or $m_{i-1} = 0\text{xFF}$, then $c_i = m_i$;
    if $m_{i-1} \ne 0\text{xFF}$ and $m_i \ne 0\text{xFF}$,
        then $c_i = (m_i + s_i) \bmod 0\text{xFF}$;
```

<u>Decryption Algorithm</u>

Generate the key stream S same as that generated in the encryption algorithm

```
if $c_1 = 0\text{xFF}$, then $m_1 = c_1$;
if $c_1 \ne 0\text{xFF}$, then $m_1 = (c_1 - s_1) \bmod 0\text{xFF}$;
for $i = 2$ to $n$,
    if $c_i = 0\text{xFF}$ or $c_{i-1} = 0\text{xFF}$, then $m_i = c_i$;
    if $c_{i-1} \ne 0\text{xFF}$ and $c_i \ne 0\text{xFF}$,
        then $m_i = (c_i - s_i) \bmod 0\text{xFF}$
```

3.2. Format compliance

In the encryption scheme above, no additional 0xFF is introduced into the encrypted message. And every byte following the original 0xFF is not modified. It thus guarantees that the value of any two consecutive bytes would not be larger than 0xFF8F.

3.3. Performance analysis

The algorithm uses a stream cipher to generate a key stream for encryption and decryption. The “modulo 0*x*FF” can be effectively implemented, so the encryption and decryption algorithms introduce only a few extra operations for each byte and will not incur delay in real-time delivery.

3.4. Security analysis

There are three types of information leakage for a byte in the plaintext. The first type of information leakage is that one byte is not encrypted if its value is $0x\text{FF}$. Denote this type of information leakage as Δs_1. Then $\Delta s_1 = 8$. The second type of information leakage is that if its immediate previous byte is $0x\text{FF}$, then that byte is not encrypted. Denote this type of information leakage as Δs_2. Then $\Delta s_2 = 8$. The last type of information leakage is that if one byte and its previous byte is not equal to $0x\text{FF}$, that byte is encrypted as $(m_i + s_i) \bmod 0\text{xFF}$. Denote this type of information leakage as Δs_3. Let $p' = \frac{1}{255}$. Then $\Delta s_3 = 8 - \sum_{i=0}^{0\text{xFE}} (-p' \times \log_2 p')$. Denote p_{255} as the probability that the value $0x\text{FF}$ appears. Rigorous analysis shows that $p_{255} = \frac{1}{257}$. In average, the total information leakage for one byte is $\Delta s = p_{255} \times \Delta s_1 + p_{255} \times \Delta s_2 + (1 - 2 \times p_{255}) \times \Delta s_3 = 2^{-3.881}$.So about $\Delta s / 8 = 0.848\%$ of the information is leaked. In other words, our encryption method protects 99.15% of the coded image data.

3.5. Implementation related issues

We recommend the use of RC4 [5] (with the first 256-byte output being discarded) as the encryption algorithm illustrated above. RC4 is a stream cipher designed more than 10 years ago and today it is the most widely used stream cipher (for example, the Internet banking system normally uses RC4 to encrypt the transaction information). The security of RC4 has been extensively studied. Although some weaknesses of RC4 have been found [6-8], the RC4 with the first 256-byte output being discarded is secure enough for image protection.

We note that a stream cipher can be obtained by running a block cipher in the Output-Feedback (OFB) mode. So any block cipher, such as AES [9] and Triple-DES [10], in the OFB mode can be used in this encryption scheme.

As for every stream cipher, an initialization vector (IV) is needed when the secret key is used more than once to foil plain-text attack. The IV does not have to be secret. For the same key, the initialization vectors should be different. It can be obtained from a secure random number source, or a counter, or the current time as accurate as millisecond plus the information related to the image (such as the image index and the hash of the image before encryption). If the IV is generated in a random way, we recommend that the length of IV is at least 128 bits to avoid collision. The initialization vector should be stored together with the encrypted image for decryption purpose.

For a JPEG2000 image which is subjected to transcoding, progression order changing, bit-rate relocation and etc., each packet need a different key to generate the key stream for encryption unless the stream cipher is self-synchronous. This can be achieved by assigning each image with a different master key and all the packet keys are generated from this master key. The key generation method is out of scope of this paper. Those who are interested in the key generation technique are referred to [11].

4. ENCRYPTION SCHEME WITH BLOCK CIPHER

The basic idea in Section 3 can be applied to design an encryption scheme using block cipher. The encryption algorithm is given below. The decryption algorithm can be derived from the encryption algorithm easily and is omitted here.

Encryption algorithm

```
1.  Select the bytes in the packet body as follows
    if $m_1 \neq 0\text{xFF}$, then $m_1$ is selected,
    for $i = 2$ to $n$,
        if $m_{i-1} \neq 0\text{xFF}$ and $m_i \neq 0\text{xFF}$,
        then $m_i$ is selected.

    From those selected bytes, form a new message
    $M' = M'_1 \| M'_2 \| M'_3 \| \cdots \| M'_\alpha$, where each
    $M'_i$ is one block.

2.  for $i = 1$ to $\alpha$,
        encrypt $M'_i$ repeatedly until all the bytes in the
        ciphertext block are not equal to 0xFF, then let
        $C'_i$ be the resulting ciphertext block.

3.  Replace the selected bytes in $M$ with those
    $C'_i$ ($1 \le i \le \alpha$) and obtain the ciphertext $C$.
```

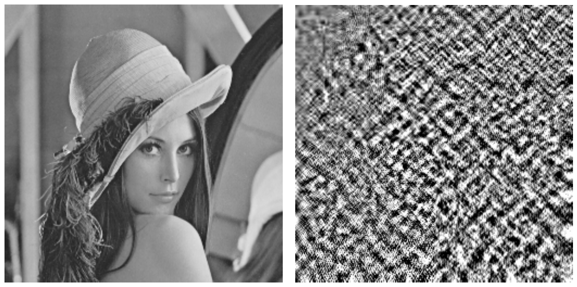
The block size is 16 bytes for AES and 8 bytes for Triple-DES. Note that the last block M'_α in the encryption algorithm may not be a full block. In that case, ciphertext stealing technique [12] can be applied to preserve the length of the message. If ciphertext stealing is used in the encryption process, the decryption of the last two blocks $C'_{\alpha-1}$ and C'_{α} should be adjusted accordingly.

In average, the probability that all the bytes in a ciphertext block are not equal to $0x\mathrm{FF}$ is 96.7% for 64-bit block size and 93.9% for 128-bit block size. The scheme above requires 3.18% extra computation for block cipher with 64-bit block size in the Electronic Codebook (ECB) mode, and 6.46% for 128-bit block size.

The security analysis is the same as that given in Subsection 3.4 and we can prove that this scheme also protects 99.15% of the information of the image.

5. EXPERIMENTS

We have implemented the encryption schemes presented in Section 3.1 and 4.1. The RC4 and AES are used in the experiments. Both schemes provide the same security protection. Figure 5.1 (a) and (b) show the granted and un-granted view of the Lenna image, respectively. The un-granted view of the image is totally unintelligible to human being. It is obvious that our encryption schemes successfully protect the JPEG2000 image.



(a) granted image (b) un-granted image

Figure 5.1 Experiment Result

6. CONCLUSION

This paper is our response to the call for proposal of the JPEG2000 Security (JPSEC) Working Group. The schemes are very computationally efficient. The schemes are also very secure since they protect 99.15% of the information of an image. Furthermore the schemes are standard compliant which guarantee the inheritance of many nice properties of the un-protected code-streams that have been well built and studied. Since code-streams are protected by encryption, all conceivable ways of content distribution, such as broadcast and CD-ROM publishing, can be used. We have implemented the encryption schemes which demonstrated the practical feasibility and the high efficiency of the proposed schemes.

REFERENCES

1. D. S. Taubman, M. W. Marcellin, “JPEG2000 Image Compression Fundamentals, Standard and Practice”, Kluwer Academic Publishers, 2000
2. “Information Technology – JPEG 2000 image coding system”, ISO/ISC International Standard 15444-1, ITU Recommendation T.800, 2000
3. J. Wen, M. Severa, W. Zeng, et al., “A format-compliant configurable encryption framework for access control of video”, IEEE Trans. Circuits & Systems for Vido Technology, Special Issue on Wireless Video, pp. 545-557, June 2002
4. Canon, “Encryption tool for JPEG 2000 access control”, ISO/IEC JTC 1/SC 29/WG1 N2893
5. R.L. Rivest, “The RC4 Encryption Algorithm”. RSA Data Security, Inc., March 12, 1992.
6. J. Golic, “Linear Statistical Weakness of Alleged RC4 Keystream Generator”, in Advances in Cryptology -- Eurocrypt'97, Springer-Verlag.
7. L. Knudsen, W. Meier, B. Preneel, V. Rijmen and S. Verdoolaege, “Analysis Methods for (Alleged) RC4”, in Advances in Cryptology -- Asiacrypt'98, LNCS 1514, pp. 327-341, Springer-Verlag.
8. I. Mantin and A. Shamir, “A Practical Attack on Broadcast RC4”, in Fast Software Encryption (FSE'01), LNCS 2355, pp. 152-164, 2002.
9. National Institute of Standards and Technology, FIPS-197, Advanced Encryption Standard.
10. “Banking – Key management (wholesale)”, ISO 8732, International Organization for Standardization, Geneva, Switzerland, 1988.
11. Y. Wu, D. Ma and R. Deng, “Flexible access control to JPEG2000 code-streams”, submitted to IEEE Tran. Multimedia.
12. B. Schneier, “Applied Cryptography: Protocols, Algorthms, and Source Code in C”, second edition, John Wiley & Sons, Inc., 1996.