10-2007

Flexible access control to JPEG 2000 image code-streams

Yongdong WU
Institute for Infocomm Research

Di MA
University of California, Irvine

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1109/TMM.2007.902865

Published in IEEE Transactions on Multimedia, Volume 9, Issue 6, October 2007, Pages 1314-1324.
https://doi.org/10.1109/TMM.2007.902865

Flexible Access Control to JPEG 2000 Image Code-Streams

Yongdong Wu, Di Ma, and Robert H. Deng

***Abstract*—JPEG 2000 is an international standard for still image compression in the 21st century. Part 8 of the standard, named JPSEC, is concerned with all the security aspects, in particular to access control and authentication. This paper presents a novel access control scheme for JPEG 2000 image code-streams. The proposed scheme is secure against collusion attacks and highly efficient. The scheme is also very flexible, allowing access control to JPEG 2000 image code-streams according to any combination of resolution, quality layer and region of interest. The "encrypt once, decrypt many ways" property of our scheme is designed to work seamlessly with the "*compress once, decompress many ways*" feature of the JPEG 2000 image code-streams. Our prototype implementation shows that the scheme is practical and is completely compatible with the core part of the JPEG 2000 standard.**

I. 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-1980s 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 known as JPEG 2000 has been formed recently by ISO/IEC JTC/SC29/WG1 (the working group charged with the development of JPEG 2000 standard). The major intention of JPEG 2000 is twofold [1], [2]. First, 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 JPEG 2000 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. This functionality allows applications to manipulate or disclose only the required image data of a code-stream for any target users based on their privileges or capabilities. JPEG 2000 achieves "*compress once, decompress many ways*" by encoding and organizing code-streams in a complicated but systematic way.

Y. Wu is with Institute for Infocomm Research, Singapore, 119613 (e-mail: wydong@i2r.a-star.edu.sg).

D. Ma is with Donald Bren School of Information and Computer Sciences, University of California, Irvine, CA 92697-3425 USA (e-mail: dma1@ics.uci.edu).

R. H. Deng is with School of Information Systems, Singapore Management University, Singapore 188065 (e-mail: robertdeng@smu.edu.sg).

JPEG 2000 refers to all parts of the standard: Part 1 (the core) [3] specifies the coding technology and is now published as an international standard, while other parts extends Part 1. In particular, Part 8 of the standard, named JPSEC [4], is concerned with all the security aspects of JPEG 2000 image code-streams. It includes a normative part and an informative part. The normative part defines the protection syntax such as encryption template, authentication template and protected region called as ZOI (Zone of Influence) for inter-operability, while the informative part presents ten tools which are compliant with the normative part.

As one of the informative tools in JPSEC, the technique presented in this paper is an access control scheme for JPEG 2000 image code-streams based on cryptographic techniques. The challenge in the design of the access control scheme is to strike a delicate balance among security, efficiency and flexibility. Motivated by the work of Sandhu [5], we construct a novel rooted tree based on the inherent structure and property of JPEG 2000 image code-streams. We use the tree to generate a family of keys to encrypt code-stream packets in such a manner that only users with the right security clearance can decrypt the encrypted packets corresponding to the requested/granted image. The proposed scheme is secure against unauthorized access as well as collusion attacks. It is also efficient since only one-way hash function and symmetric key encryption are used. More importantly, the scheme is extremely flexible, allowing access control to JPEG 2000 image code-streams according to any combination of resolution, quality layer and WOI (Window-of-Interest).[1] This "**encrypt once, decrypt many ways**" property of our scheme is designed to be completely compatible with the "*compress once, decompress many ways*" feature of the JPEG 2000 image code-streams.

A. Related Work

Access control involves authorizing legitimate users with appropriate privileges to access a certain resource while denying access from illegal users [6], [7]. Solutions for authorization fall into two categories, access control models and cryptographic techniques. An access control model mediates access to resources by checking with access control rules established in conformance with a security policy. A cryptographic method for access control manages authorization by encrypting information items such that only authorized users have the right

[1]According to JPEG 2000 specification [3], ROI (Region-of-Interest) in a code-stream is defined explicitly by the image owner. Thus, we use WOI to indicate the region which the client is interested in.

keys to decrypt the scrambled data. A number of schemes [8]–[12] relating to access control based on cryptographic techniques have been proposed. All of these schemes assume that information items as well as users are classified into a certain type of hierarchy and there is a relationship between the encryption key assigned to a node and those assigned to its children. The related work differs mostly in the different cryptographic techniques employed for key generation. Most of them employ complex and computational expensive cryptographic operations, such as RSA [13] or large integer modular exponentiations. Employing these schemes in access control to JPEG 2000 image code-streams is not feasible since user devices (such as PDAs and cell phones) in JPEG 2000 applications can be very resource constrained.

Since a JPEG 2000 packet is identified uniquely by a given resolution-increment (or resolution), a layer-increment (or layer) a precinct, a component and a tile, a straightforward solution is to encrypt each packet with an independent key. This trivial access solution provides great flexibility at the price of a large overhead for key communication and storage. For instance, the JPEG 2000 standard allows a code-stream to support up to $n_R = 33$ resolutions, $n_L = 65535$ layers and large number n_P of precincts. Consider a 1024×1024 image generated by a digital camera and assume that it is processed with 5-resolutions, 16 layers, and 16 precincts. For access to a tile-component, the number of required keys is $5 \times 16 \times 16 = 1280$ in the trial solution.

Grosbois *et al.* [14] proposed two access control schemes for JPEG 2000 image code-streams. The first scheme allows for a preview of low resolution image while preventing the correct display of its higher resolutions by scrambling the sign bits of the wavelet coefficients of the high resolutions code-block by code-block based on pseudo-random sequences. The second scheme provides access control to image quality layers by introducing pseudo-random noise in the higher quality layers of the image. Random seeds for generating the pseudo-random sequences are encrypted and appended to image code blocks. However, the two schemes in [14] have several drawbacks. First, they are not flexible, providing either resolution based access control or quality layer based access control but not both at the same time. Secondly, they introduce considerable overhead to the image code-stream. For instance, assume that a 1024×1024 medium size image with 256 (64×64) code-blocks, the payload for the encrypted seeds amounts to $256 \times 16 = 4096$ bytes assuming that each seed consists of 16 bytes (128 bits). Thirdly, the two schemes lead to decreased compression ratio because wavelet coefficients are randomized before compression.

The Secure Scalable Streaming (SSS) technique proposed in [15] supports low complexity, high quality transcoding at intermediate, possibly untrusted, network nodes without compromising the end-to-end security of the system. SSS encodes, encrypts, and packetizes video into secure scalable packets in a manner that allows transcoders to perform transcoding operations (e.g., bit rate reduction and spatial down sampling) by simply truncating or discarding packets, and without decrypting the data. Secure scalable packets have unencrypted headers that provide side information, such as optimal truncation points, to downstream transcoders. However, SSS scheme does not consider key management which is the major topic of this paper.

Wu *et al.* [16] proposed a progressive protection method for JPEG 2000 code-streams. This progressive method is efficient but is not satisfactory in flexibility. In order to provide compliant encryption of a JPEG 2000 code-stream which has no emulation marker in the protected packets, the packet body is repeatedly encrypted with a standard stream cipher [17]. The advantage in [17] is that the size of the protected code-stream is the same as that of the original code-stream.

Recently, Zhu *et al.* [18] proposed a key management scheme for protecting JPEG 2000 code-streams. They created a diagram whose nodes are used to represent resolutions, layers, precincts and tiles. Although their scheme is flexible in terms of truncating, each JPEG 2000 packet has to piggyback at least one node value whose length is identical to that of hash value. Hence, their scheme is more suitable to protection of motion JPEG 2000 stream than protection of JPEG 2000 code-stream.

B. *Outline of the Paper*

The paper is arranged as follows. Section II illustrates the basic concepts and characteristics of JPEG 2000 image code-streams. Section III introduces the access control system setup and several security classes related to JPEG 2000 image code-streams. Section IV presents a naïve access control scheme and its analysis. Section V is the main contribution of the paper where we describe our secure access control scheme. Section VI shows our experiment results. Section VII contains some concluding remarks.

C. *Notations*

We list below important notations used throughout the paper for ease of reference. Terminology such as precinct, resolution, resolution-increment, quality layer (or layer in short) and layer-increment will become clear in the next section. We will refer a JPEG 2000 image code-stream simply as JPEG 2000 code-stream or just code-stream

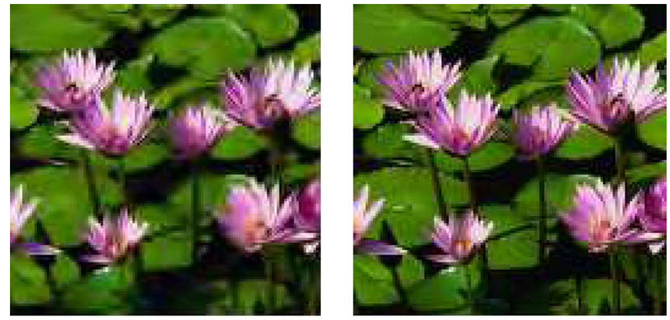
n_C	number of components in a code-stream;
C_c	cth component in a code-stream, $c = 0, 1, \ldots, n_C - 1$;
n_R	number of resolutions/resolution-increments in a code-stream;
R_r	resolution-increment r in a code-stream, $r = 0, 1, \ldots, n_R - 1$;
n_L	number of layers/layer-increments in a code-stream;
L_l	layer-increment l in a code-stream, $l = 0, 1, \ldots, n_L - 1$;
n_P	number of precincts in a resolution. W.l.o.g, assume every resolution has the same number of precincts;
P_p	pth precinct in a resolution, $p = 0, 1, \ldots, n_P - 1$;
$\mathcal{H}(\cdot)$	cryptographic one-way hash function;
$x\|y$	concatenation of x and y.

II. Overview of JPEG 2000 Code-Streams

In what follows, we provide a brief description of the concepts and terminology related to JPEG 2000 code-streams. Our goal is to illuminate the main concepts at a sufficient level to impart an understanding of our access control scheme without

(a) (b) (c)

Fig. 1. $n_R = 3$ resolutions of an image. (a) 128 × 128; (b) 64 × 64; and (c) 32 × 32.



(a) (b)

Fig. 2. Two images of different qualities. Note: bpp (bit per pixel). (a) 0.05 bpp and (b) 0.5 bpp.

dwelling into too much on the details. Those interested in the details are referred to [1]–[3].

A. Basic Concepts and Terminology

Tiles: JPEG 2000 divides an image into rectangular non-overlapping regions known as tiles, which are compressed independently, as though they were entirely distinct images. Tiling reduces memory requirements during compression and decompression. For the sake of simplicity and without loss of generality, we will only consider single tile code-streams.

Components: An image is comprised of one or more components; each consists of a rectangular array of samples. For example, a RGB color image has three components representing the red, green and blue color planes.

Resolution-Increments and Resolutions: Given a tile-component, a $(n_R - 1)$-level dyadic wavelet transform is performed. The first wavelet transform decomposes a component into four frequency subbands LL_1 (horizontally lowpass and vertically lowpass), LH_1 (horizontally lowpass and vertically highpass), HL_1 (horizontally highpass and vertically lowpass) and HH_1 (horizontally highpass and vertically highpass). The second wavelet transform further decomposes LL_1 into another four subbands LL_2, LH_2, HL_2, and HH_2. Finally, the $(n_R - 1)$th wavelet transform decomposes LL_{n_R-2} into four subbands LL_{n_R-1}, LH_{n_R-1}, HL_{n_R-1}, and HH_{n_R-1}. Therefore, a $(n_R - 1)$-level dyadic wavelet transform generates n_R sets of subbands, denoted as $R_0 = \{LL_{n_R-1}\}$, $R_1 = \{LH_{n_R-1}, HL_{n_R-1}, HH_{n_R-1}\}, \ldots, R_{n_R-1} = \{LH_1, HL_1, HH_1\}$. We refer to R_i as resolution-increment i of a code-stream. The n_R resolution-increments above correspond to n_R image sizes or resolutions. The resolution 0 image is constructed from resolution-increment 0, $\{R_0\}$, and is a small "thumbnail" of the original image. The resolution 1 image is constructed from resolution-increments 0 and 1, $\{R_0, R_1\}$, and is a bigger "thumbnail" of the original image. In general, the resolution r image is constructed from resolution-increments 0 to r, $\{R_0, R_1, \ldots, R_r\}$. Note that the resolution $n_R - 1$ image is the original image. Fig. 1 shows $n_R = 3$ resolutions of a code-stream. The original image is of size 128 × 128, resolution 1 is of size 64 × 64, and the lowest resolution 0 is of size 32 × 32.

Layer-Increments and Layers: Following the wavelet decomposition, wavelet coefficients are quantized and each quantized subband is partitioned into small rectangular blocks, referred to as code-blocks. Each code-block is independently entropy encoded to create a compressed bit-stream which is distributed across n_L quality layers. Layers determine the quality or signal-to-noise ratio of the reconstructed image. Roughly speaking, a higher quality image needs more bits for each pixel representation than a lower quality image. Let L_0 denote the code-stream data needed to form a layer 0 image. Let L_l be the additional code-stream data to form a layer l image given $L_0, L_1, \ldots, L_{l-1}$, $l = 1, 2, \ldots, n_L - 1$. That is, a layer l image is formed from $\{L_0, L_1, \ldots, L_{l-1}, L_l\}$. Note that the image of layer $n_L - 1$ is the original image. We refer to L_l as layer-increment l, $l = 0, 1, 2, \ldots, n_L - 1$. Fig. 2 illustrates two images of different qualities.

Precincts: In order to provide locality for accessing certain portions (e.g., WOI) of an image, an intermediate space-frequency structure known as precinct is provided in JEPG2000. Unlike the tile and code-block partitions, the precinct partition does not affect the transformation or coding of sample data; it instead plays an important role in organizing compressed data within a code-stream. Specifically, a precinct is a collection of spatially contiguous code-blocks from all subbands at a particular resolution. In Fig. 3(a), the original image of size 512 × 512 is divided into four precincts of size 256 × 256 each. In Fig. 3(b), each smaller resolution includes four precincts with one-to-one correspondence to the 4 precincts in Fig. 3(a). For instance, the gray blocks marked A, B and C form a precinct in resolutions 2, 1, and 0, respectively, and they all correspond to the gray precinct in Fig. 3(a). In other words, the data in precincts A, B and C can be used to reconstruct the gray region in the original image.

Packets: Packets are the fundamental building blocks in a JPEG 2000 code-stream. It comprises the compressed bit-stream from code-blocks belonging to a specific component, resolution, layer, and precinct. Fig. 4 shows the process for generating packets. The original image is first decomposed into components. Then, the dyadic wavelet transform is applied to each component to produce the subbands corresponding to various resolution-increments. Each subband is quantized and divided into code-blocks. Certain spatially contiguous code-blocks in subbands of a resolution form a precinct.

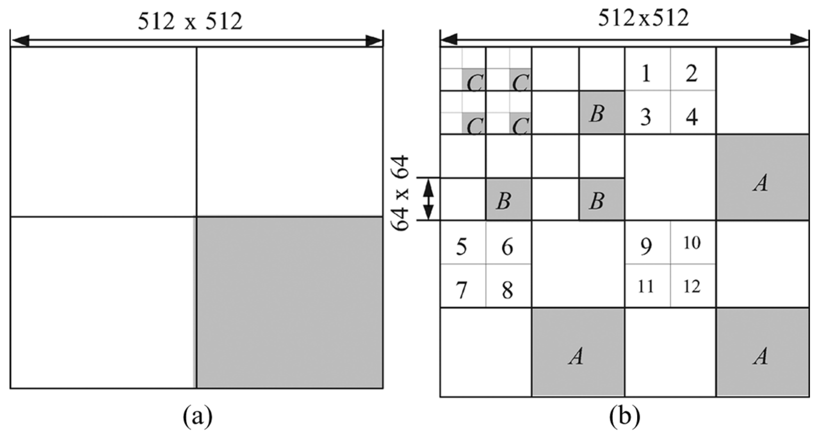


Fig. 3. Precinct partition. (a) Precincts of the original image and (b) precincts of lower resolutions.

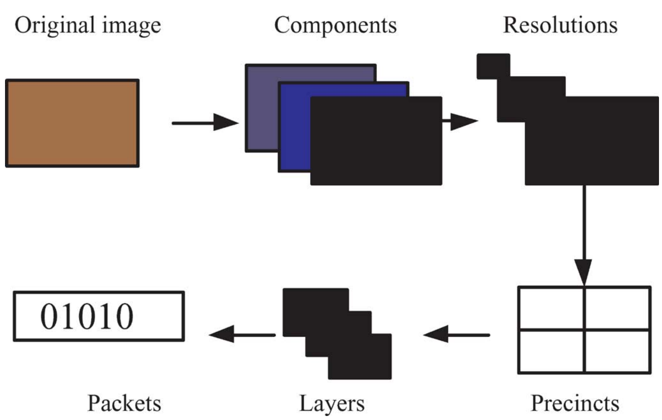


Fig. 4. Packet generation process.

TABLE I
ABSTRACT STRUCTURE OF A JPEG 2000 CODE-STREAM

Header	D_1	D_2	$\ldots$	D_N	EOC

```
for l = 0, 1, ..., n_L - 1
  for r = 0, 1, ..., n_R - 1
    for c = 0, 1, ..., n_C - 1
      for p = 0, 1, ..., n_P - 1
        Arrange packet D^{lrcp}
```

Fig. 5. Arrangement of packets in a code-stream following progression order LRCP, where D^{lrcp} is a certain packet D_i in Table I.

Each code-block is encoded independently into compressed bit-stream which is distributed into quality layer increments. Finally, compressed bits from the same component, resolution-increment, precinct and layer-increment are encapsulated into a packet.

B. Progression Orders

Progressive display allows images to be reconstructed with increasing pixel quality or resolution, as needed or desired, for different target devices. JPEG 2000 supports progression in four dimensions: quality layer, resolution, spatial location and component [1]–[3].

- In quality layer progression, image quality is improved when more layer-increments are received. For example, the image with the lowest quality is reconstructed from decoding L_0. The image with the next quality layer is obtained by decoding L_0 and L_1. Improving quality is then a simple matter of decoding more bits.
- In resolution progression, the first few bytes, i.e., R_0, is used to form a small "thumbnail" of the image. As more resolution-increments R_r are received, $r = 1, 2, \ldots, n_R - 1$, image width and height double.
- In spatial location progression, image is displayed in approximately raster fashion, i.e., from left to right and from top to down, or displayed by WOI.
- Component progression controls the order in which the data corresponding to different components is decoded.

These dimensions of progression can be "mixed and matched" within a single compressed code-stream and this has been referred to as the "*compress once, decompress many ways*" property of JPEG 2000 [1]. Table I shows a typical JPEG 2000 code-stream which consists of a header, data packets $D_1, D_2, \ldots, D_N$ arranged in a particular progression order, and an end-of-code-stream marker EOC. Fig. 5 shows the pseudo-code for generating a code-stream of progression order LRCP (layer-resolution-component-precinct).

III. Overview of the Access Control Schemes

Before proceeding to the presentation of our access control schemes, we first describe the system setup and operation, security classes related to JEPG2000 code-streams and access control rules.

A. System Setup and Operation

Part 9 of the JEPG 2000 standard, JPEG 2000 Interactive Protocol (JPIP) [19] specifies how to respond to user requests of images with various progression orders. JPIP is mainly intended for interactive on-line client/server type of applications. Protected with access control, JPIP can also be adapted for non-interactive as well as off-line distributions. The system setup for access control to JPEG 2000 code-streams is shown in Fig. 6.

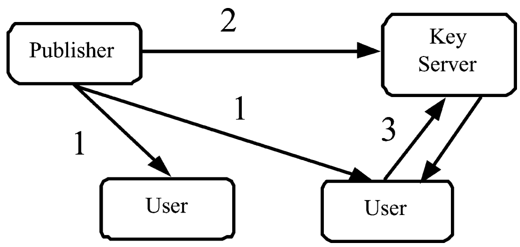


Fig. 6. Access control system setup and operation.

High level operation of the system consists of the following three steps.

1) JPEG 2000 code-streams are first encrypted by the publisher and then distributed to users. Since the code-streams are protected by encryption, all conceivable ways of content data distribution can now be enabled, including for examples Internet, digital cable TV, satellite broadcast and CD-ROM publishing. This concept, called "super-distribution" [20], provides the publisher a very flexible way to use the most appropriate distribution channel.
2) The control data, i.e., keys for decrypting the content data, is forwarded securely from the publisher to an on-line key server.
3) A user who desires to access portions of a code-stream sends his/her request together with authentication information or payment data to the key server. The key server, in turn, responds with appropriate decryption keys according to user's privilege or amount of payment.

B. Security Classes and Access Control Rules

From Section II-A, a code-stream can be reviewed as a collection of bitstreams generated from components $\{C_c, c = 0, 1, \ldots, n_C - 1\}$, precincts $\{P_p, p = 0, 1, \ldots, n_P - 1\}$, resolution-increments $\{R_r, r = 0, 1, \ldots, n_R - 1\}$, and layer-increments $\{L_l, l = 0, 1, \ldots, n_L - 1\}$. We note that components are introduced in JPEG 2000 mainly for improving compression efficiency and for progressive display of images when data is sent over slow channels; they are not very meaningful as far as access control is concerned. Hence, we will not address access control to components during the rest of the paper. In other words, we assume that all components are accessed at the same security level. It should be noted that our access control schemes can be easily extended to enforce access control to components whenever it is needed. Consider the situation where users and data can be classified into a hierarchy of security classes [21]. If a security class A is the predecessor of another security class B, we say that A strictly dominates B and denote this relation as $A > B$. Similarly, we say that A dominates B, denoted as $A \geq B$, if either $A > B$ or $A = B$. We say that A and B are comparable if $A \geq B$ or $B \geq A$; otherwise A and B are incomparable. From the progression properties of code-streams presented in Section II-B, we define the following security classes related to a JPEG 2000 code-stream:

- The security classes of resolution-increments R_r, $r = 0, 1, \ldots, n_R - 1$, are total ordering [21], with

$$R_{n_R-1} > \ldots > R_1 > R_0. \qquad (1)$$

- The security classes of layer-increments L_l, $l = 0, 1, \ldots, n_L - 1$, are total ordering, with

$$L_{n_L-1} > \ldots > L_1 > L_0. \qquad (2)$$

- The security classes of the precincts $P_p, p = 0, 1, \ldots, n_P - 1$, are isolated classes [21]. That is, P_p and $P_{p'}$ are incomparable for all $p \neq p'$.

Based on the above security classes, our aim is to enforce the following access control rules for a JPEG 2000 code-stream.

- A user who is allowed to access security class R_r also have access to $R_{r'}$ for all $r' < r$ but not to $R_{r''}$ for all $r'' > r$.
- A user who is allowed to access security class L_l can also access $L_{l'}$ for all $l' < l$ but not $L_{l''}$ for all $l'' > l$.
- A user who is allowed to access a subset of $\{P_p, p = 0, 1, \ldots, n_P - 1\}$ can not have access to any other subsets outside of the granted subset.
- Any "mix and match" of the above regardless of the progression order of the code-stream.

In order to realize access control to JPEG 2000 code-streams which meets our access control rules above, it is natural to form a combined hierarchy of security classes from the isolated precinct security classes, the total ordered resolution-increment security classes and layer-increment security classes. Unfortunately, the resulting hierarchy of security classes is a Directed Acyclic Graph (DAG), not a rooted tree. There are cryptographic solutions available in the literature for key generation and implementing access controls in DAGs (see for examples [8]–[12], [22]). All of them, however, are based on public key cryptosystems and are extremely complex and computationally expensive for our applications.

IV. A Naive Access Control Schemes

In this section we present a simple-minded approach to realize access control to JPEG 2000 code-streams. Our objective here is twofold. The first is to illustrate the essence of the access control problem and the second is to show some pitfalls a robust design must avoid.

A. Packet Key Generation and Code-Stream Encryption

Fig. 7 illustrates how to generate precinct, resolution and layer keys from a master key, combine them to produce packet keys, and then use packet keys to encrypt packets in a code-stream. Specifically, we have the following.

- Given a code-stream, generate a random master key K which is then diversified into resolution keys, layer keys and precinct keys as below.
- Generate resolution keys iteratively from the hash chain

$$k_{R,r} = \mathcal{H}(k_{R,r+1}) \qquad (3)$$

for $r = n_R - 2, n_R - 3, \ldots, 1, 0$ where $k_{R,n_R-1} = \mathcal{H}(K|\text{“}R\text{”})$ and where "R" denotes the ASCII code of the letter R.

- Generate layer keys iteratively from the hash chain

$$k_{L,l} = \mathcal{H}(k_{L,l+1}) \qquad (4)$$

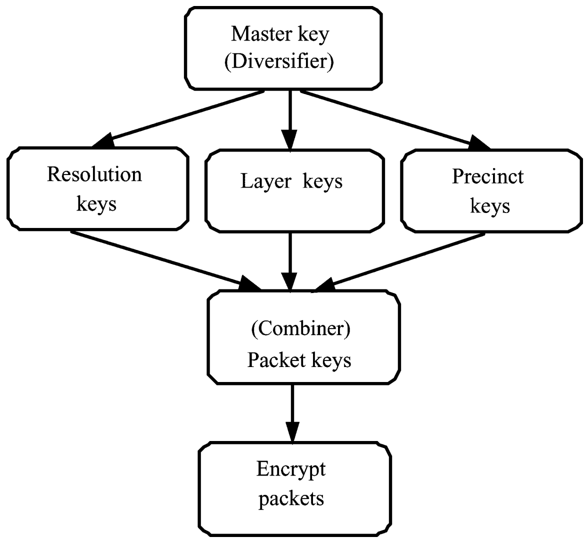


Fig. 7. Packet key generation for the second scheme.

for $l = n_L - 2, n_L - 3, \ldots, 1, 0$, where $k_{L,n_L-1} = \mathcal{H}(K|\text{“}L\text{”})$ and where “L” denotes the ASCII code of the letter L.

- Generate precinct keys from

$$k_{P,p} = \mathcal{H}(K|\text{“}P\text{”}|p) \quad (5)$$

for $p = 0, 1, \ldots, n_P - 2, n_P - 1$, where “$P$” denotes the ASCII code of the letter P.

- Combine keys to generate the packet key

$$k^{rlp} = \mathcal{H}(k_{R,r}|k_{L,r}|k_{P,p}) \quad (6)$$

and encrypt packet D^{rlp} using k^{rlp} for $r = 0, 1, \ldots, n_R - 1$, $l = 0, 1, \ldots, n_L - 1$, and $p = 0, 1, \ldots, n_R - 1$.

B. Access Encrypted Code-Stream

Assume that a user sends a request to the key server in Fig. 6 asking for the right to access an image with resolution r', layer l' and m precincts P_p, $p = p_1, p_2, \ldots, p_m$. The key server replies with the resolution key $k_{R,r'}$, the layer key $k_{L,l'}$, and the precinct keys $k_{P,p}$, $p = p_1, p_2, \ldots, p_m$. To access packets associated with the requested image, the user proceeds as follows.

- Compute, from $k_{R,r'}$, resolution keys $k_{R,r'-1}, k_{R,r'-2}, \ldots, k_{R,1}, k_{R,0}$, iteratively using (3).
- Compute, from $k_{L,l'}$, layer keys $k_{L,l'-1}, k_{L,l'-2}, \ldots, k_{L,1}, k_{L,0}$, iteratively using (4).
- Compute, from the r' resolution keys, l' layer keys, and m precinct keys, packet key k^{rlp} using (6), and then decrypt packet D^{rlp}, for $r = 0, 1, \ldots, r'$; $l = 0, 1, \ldots, l'$; and $p = p_1, p_2, \ldots, p_m$.

C. Discussion

First, we note that this scheme is flexible. It allows a user to access images with any resolution, layer and precinct combination, as long as the user possessing the required privilege or having made sufficient payment. Second, it is quite efficient. For a user to request access to an image with resolution r', layer l' and m precincts, the key server transfers only $m + 2$ keys, 1 resolution key, 1 layer key and m precinct keys. Third, given the $m + 2$ keys, the user is not able to access images with resolutions higher than r' or with layers higher than l'. This is because, given the resolution key $k_{R,r'}$, it is infeasible to compute the resolution key $k_{R,r}$ for any $r > r'$ due to the one-wayness of the hash function $\mathcal{H}(\cdot)$. Similarly, given the layer key $k_{L,l'}$, it is infeasible to compute the layer key $k_{L,l}$ for any $l > l'$. Unfortunately, this scheme is subject to the following collusion attack. Assume that a user first pays to access an image of resolution r_1 and layer l_1. In this case, the key server replies with keys k_{R,r_1} and k_{L,l_1}. The user then pays to access another image from the same code-stream with resolution r_2 and layer l_2. The key server this time supplies the user with keys k_{R,r_2} and k_{L,l_2}. Then without further payment, the user is able to access the image with resolution $r = \max(r_1, r_2)$ and layer $l = \max(l_1, l_2)$. The proof of this fact is straightforward using (3), (4) and (6) and hence is omitted here. As an extreme case of this attack, a user just needs to purchase two images, one with the smallest resolution but the highest quality layer, and the other with the full resolution but the lowest quality layer, then the user can acquire the original image (i.e., the image with full resolution and the highest quality) for free. This is clearly unacceptable to most applications. Our access control scheme given in the next section is secure against this type of collusion attacks.

V. Secure Access Control Scheme

Sandhu [5] introduces a cryptographic implementation for access control in a situation where users and information items are classified into a rooted tree of security classes, with the most privileged security class at the root. The idea is that keys for security classes are generated from their parent class using a parameterized family of one-way functions. In this section, we seek to adapt Sandhu's approach to arrive at a flexible, efficient and secure access control scheme for JPEG 2000 code-streams.

A. Sandhu's Key Generation Scheme

In Sandu's scheme, encryption keys associated with a tree of security classes are generated as follows.

1) For the security class at the root assign an arbitrary key.
2) If a security class Y is an immediate child of X in the tree, let $k_Y = \mathcal{H}(k_X|name(Y))$, where k_X and k_Y are the keys associated with X and Y, respectively, and $name(Y)$ is the name of Y.

The keys k_X and k_Y are used to encrypt information items of security classes X and Y, respectively. A user at a security level, say X, needs to know k_X. Since one-way hash function is public, keys for security classes dominated by X can be generated from k_X as needed. However, it is computationally infeasible to compute keys for any predecessors or any siblings of X due to the one-way nature of the hash function. We show the construction and application of Sandhu trees with a simple example. Consider the security classes A, B_0, B_1, C_0, C_1, C_2, and C_3, where $A > B_0$ and $A > B_1$; $B_0 > C_0$ and $B_0 > C_1$;

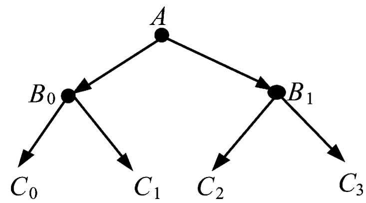


Fig. 8. Example Sandhu tree.

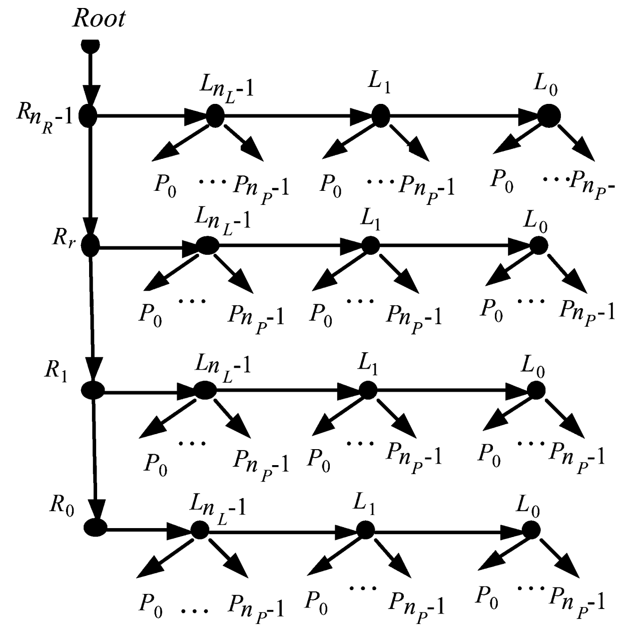


Fig. 9. Rooted tree for a code-stream with progression order resolution-layer-precinct.

and $B_1 > C_2$ and $B_1 > C_3$. The Sandhu tree for this security class hierarchy is given in Fig. 8.

We assign a random key k_A to the root. The keys for B_0 and B_1 are $k_{B_0} = \mathcal{H}(k_A|name(B_0))$ and $k_{B_1} = \mathcal{H}(k_A|name(B_1))$, respectively. The keys for C_0, C_1, C_2, and C_3 are $k_{C_0} = \mathcal{H}(k_{B_0}|name(C_0))$, $k_{C_1} = \mathcal{H}(k_{B_0}|name(C_1))$, $k_{C_2} = \mathcal{H}(k_{B_1}|name(C_2))$ and $k_{C_3} = \mathcal{H}(k_{B_1}|name(C_3))$, respectively. A user with security clearance B_1 is given k_{B_1}. He can easily compute keys for C_2 and C_3, but not for any other nodes in the tree. It is interesting to note that Sandhu tree is, in certain sense, the opposite of Merkle tree [24]. The former is used for access control while the latter is used for authentication.

B. Tree-Based Scheme

Recall the three hierarchies of security classes defined in [1]–[3]—the total ordering security classes of resolution-increments, the total ordering security classes of layer-increments and the isolated classes of precincts. Using the techniques in [21], we can obtain the overall composite hierarchy for a JPEG 2000 code-stream by combining the three simpler hierarchies. As mentioned before, the resulting hierarchy is a DAG and therefore is not amenable to efficient solutions.

Sandhu tree is an efficient, scalable and flexible method of generating cryptographic keys for tree hierarchies, not for DAGs in general. Our challenge here is to bring the advantages of Sandhu tree to the access control of JPEG 2000 code-streams. The key in meeting this challenge is to construct a rooted tree of security classes for JPEG 2000 code-steams. It turns out that this can be done in a systematic way. The trick is to specify a preferred progression order when constructing the overall hierarchy for a JPEG 2000 code-stream.

We use an example to illustrate our idea. Assume that the preferred progression order is resolution-layer-precinct. The rooted tree hierarchy for a JPEG 2000 code-stream is shown in Fig. 9. Observe how the tree is constructed based on the preferred progression order resolution-layer-precinct: the resolution-increments form the trunk of the tree, the layer-increments form the branches and finally the precincts form the leaves. Also observe that the tree preserves the hierarchies of individual security classes, e.g., the trunk formed from the resolution-increments is still a total ordering.

We remark that there are a number of subtle differences between our tree and a Sandhu tree. First, a given class, say P_0, is assigned to multiple nodes in our tree while this is not allowed in a Sandhu tree. Second, keys associated with non-leaf nodes in a Sandhu tree are used for encrypting information items associated with the nodes, while in our tree only keys associated with the leaf-nodes will be used to encrypt packets in the code-stream.

1) Packet Key Generation and Code-Stream Encryption: Key generation in our rooted tree follows the same approach as that in the Sandhu tree.

- Given a code-stream, generate a random master key K.
- Generate keys for the resolution nodes iteratively from the hash chain
$$k^r = \mathcal{H}(k^{r+1}) \tag{7}$$
for $r = n_R - 2, n_R - 3, \ldots, 1, 0$, where $k^{n_R-1} = \mathcal{H}(K|\text{“}R\text{”})$ and where “R” denotes the ASCII code of the letter R.
- For a given r, generate keys for the layer nodes iteratively from the hash chain
$$k^{rl} = \mathcal{H}\left(k^{r(l+1)}\right) \tag{8}$$
for $r = n_R - 1, n_R - 2, \ldots, 1, 0$, $l = n_L - 2, n_L - 3, \ldots, 1, 0$, where $k^{r(n_L-1)} = \mathcal{H}(k^r|\text{“}L\text{”})$ and where “L” denotes the ASCII code of the letter L.
- For a given r and l, generate keys for the precinct nodes from
$$k^{rlp} = \mathcal{H}(k^{rl}|\text{“}P\text{”}|p) \tag{9}$$
for $r = n_R - 1, n_R - 2, \ldots, 1, 0$, $l = n_L - 1, n_L - 2, \ldots, 1, 0$ and $p = 0, 1, \ldots, n_P - 2, n_P - 1$, where “$P$” denotes the ASCII code of the letter P.
- Encrypt the packet D^{rlp} using the key k^{rlp} under a symmetric key algorithm for $r = 0, 1, \ldots, n_R - 1$, $l = 0, 1, \ldots, n_L - 1$, and $p = 0, 1, \ldots, n_P - 1$.

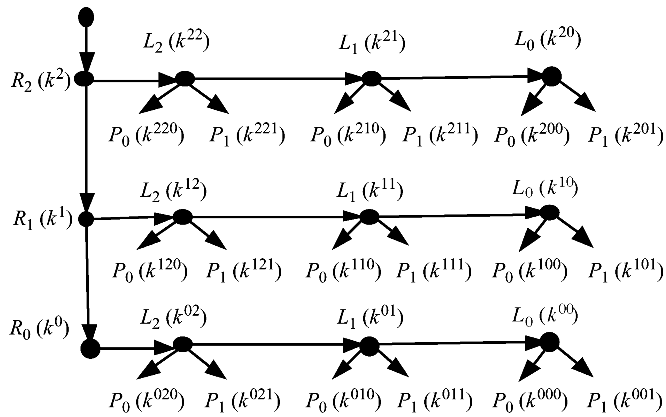


Fig. 10. An example rooted tree for a code-stream with $n_R = 3, n_L = 3$, and $n_P = 2$ and preferred progression order resolution-layer-precinct.

2) Access Encrypted Code-Stream: There are three cases to consider here depending on the user access requirements.

Case 1) A user requests access to the image of resolution r' (i.e., the image of resolution r' with the highest quality layer and all the precincts). The key server in Fig. 6 replies with $k^{r'}$. To obtain the packets corresponding to the requested image, the user proceeds as follows.

(I) Compute, from $k^{r'}$, keys $k^{r'-1}, k^{r'-2}, \ldots, k^1, k^0$, iteratively using (7).

(II) Compute, from the keys obtained in step (I), keys k^{rl}, iteratively using (8), for $r = r', \ldots, 1, 0$ and $l = n_L - 1, \ldots, 1, 0$.

(III) Compute, from the keys obtained in step (II), the packet key k^{rlp} using (9), and then decrypt the packet D^{rlp}, for $r = 0, 1, \ldots, r'$; $l = 0, 1, \ldots, n_L - 1$ and $p = 0, 1, \ldots, n_P - 1$.

Case 2) A user requests access to the image of resolution r' and layer l' (and with all the precincts). The key server supplies the user with keys $k^{rl'}$, $r = 0, 1, \ldots, r'$. The user then computes all the necessary packet keys using (8) and (9).

Case 3) A user desires access to the image of resolution r', layer l' and $m \leq n_P$ precincts P_p, $p = p_1, p_2, \ldots, p_m$. The key server replies with the keys k^{rlp} for $r = 0, 1, \ldots, r'$, $l = 0, 1, \ldots, l'$ and $p = p_1, p_2, \ldots, p_m$. The user simply uses these keys to decrypt the corresponding packets in order to obtain the desired image.

Fig. 10 depicts an example rooted tree for a code-stream with $n_R = 3$, $n_L = 3$, and $n_P = 2$. The key associated with each node is shown in the parentheses next to the node. To access the image with resolution 1, a user only needs to know k^1; to access the image with resolution 1 and layer 1, the user needs to know k^{11} and k^{01}; to access the image with resolution 1, layer 1 and precinct 0, the user needs to know keys k^{110}, k^{100}, k^{010}, and k^{000}.

3) Discussion: This scheme allows privileged users to access images with any resolution, layer and precinct as well as their combinations; therefore, it is very flexible and maintains the "*compress once, decompress many ways*" merit of the JPEG 2000 standard. The collusion attack to the second scheme shown in Section IV is due to the key combining operation in (6), but the scheme in this section is protected from the collusion attack since its key generation process is strictly sequential and is free from the combining operation as in the second scheme. The overhead for key transmission from the key server to a user depends on the type of images requested and on the way the rooted tree is constructed. For the tree in Fig. 9, to access the image with resolution r', only one key $k_{r'}$ is required; to access the image with resolution r' and layer l', $(r' + 1)$ keys, $k^{rl'}$, $r = 0, 1, \ldots, r'$, need to be sent to the user; however, to access the image of resolution r', layer l' and m precincts P_p, $p = p_1, p_2, \ldots, p_m$, the key server has to transmit $(r' + 1)(l' + 1)m$ keys, k^{rlp} for $r = 0, 1, \ldots, r'$, $l = 0, 1, \ldots, l'$, and $p = p_1, p_2, \ldots, p_m$ to the user. Therefore, the tree in Fig. 9 is the most efficient in accommodating resolution requests and the least efficient for handling precinct requests.

In general, we can easily adapt our rooted tree construction according to user request patterns. For example, assume that most of the user requests are precinct requests, followed by resolution requests and then followed by layer requests, our tree should be constructed based on the preferred progression order precinct-resolution-layer. The resulting tree, shown in Fig. 11, is the most efficient for precinct requests but the least efficient for layer requests. To access an image with m precincts (i.e., an image with m precincts, full resolution and the highest layer), only m keys are needed. In practical applications, users are normally not interested in individual precincts, but rather on WOI of a code-stream. For such applications, we can modify our key generation process by assigning one key to precincts corresponding to the WOI and another to all the precincts outside of the WOI. This reduces the number of keys for precincts to just two regardless of the number of precincts in a code-stream.

VI. Experiments

We have implemented our third scheme presented in Section V-B for access control to JPEG 2000 code-streams in C++. The demo has been shown in the ISO/IEC JPSEC meeting. The software comprises three modules, a publisher module, a key server module and a user module. The publisher module accepts a JPEG 2000 code-stream. Note that information such as n_R, n_L, n_P and the preferred progression order of the code-stream are contained in the code-stream header. The publisher module first constructs a rooted-tree based on the preferred progression order of the code-stream, assigns a random master key K to the root, computes packet keys and encrypts all the packets in the code-stream according to the process given in Section V-B. We use RC4 [25] to encrypt packets since stream cipher RC4 requires no padding in packet encryption. The publisher module deposits the encrypted code-stream along with its file name, cs_{name}, into a public directory, and forwards $\{cs_{name}, n_R, n_L, n_P, K\}$ to the key server module over a secure channel. The user module fetches the encrypted code-stream from the public directory and interacts with the key server to obtain appropriate decryption keys. The user module allows a user to request for any resolution,

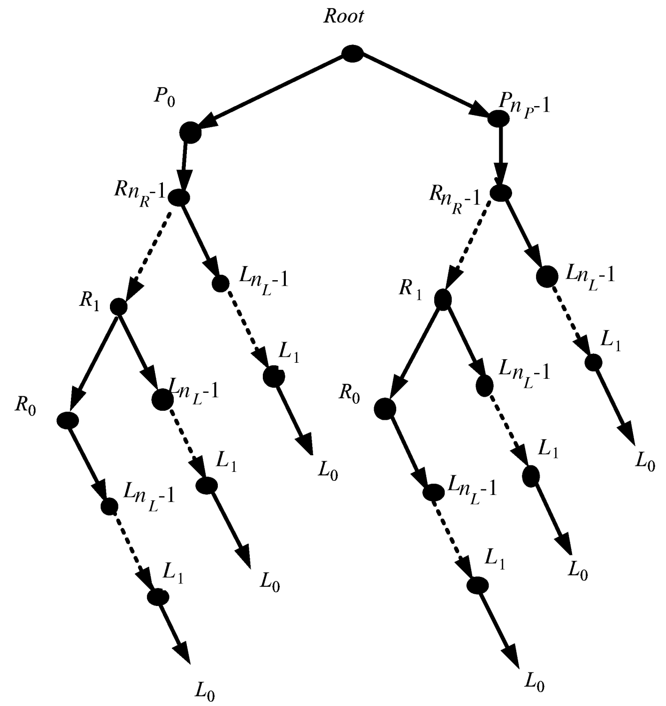
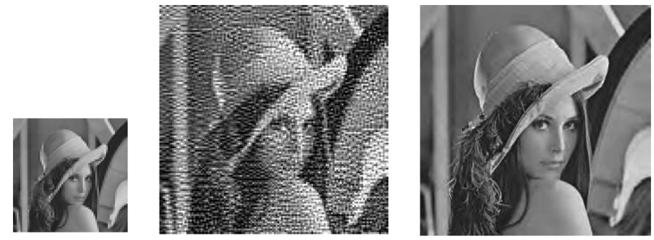


Fig. 11. Rooted tree for a code-stream with preferred progression order precinct-resolution-layer.

Fig. 12. Original "Lenna" image, with $n_R = 4$, $n_L = 8$, and $n_P = 16$.

layer, precinct and any combinations of the above. A JPEG 2000 code-stream for the source image "Lenna" shown Fig. 12 is used as our test example. The "Lenna" code-stream has $n_R = 4$, $n_L = 8$ and $n_P = 16$ with the preferred progression order resolution-layer-precinct. We divide the 16 precincts into two groups, with the center 4 precincts as the WOI and the rest 12 precincts as the "Boarder Region." For a given resolution and layer, packets in the WOI are encrypted using a single key and packets in the "Boarder Region" are encrypted using another key. This approach to encrypting precincts has two advantages. First, it is practical and intuitive for user access since users normally are not interested in randomly selected precincts, but in either WOI or the entire image; second, it also greatly reduces the number of keys.



(a) (b) (c)

Fig. 13. Results of the first experiment. (a) Granted image, (b) extrapolated image ($\mathrm{PSNR} = 27.9$ dB), and (c) the reference image (resolution 2).

Referring to Fig. 9 and the key generation process in Section V-B, all the keys generated for this code-stream are given in Table II below. Note that there are only two packet encrypt keys for a given resolution-increment r and layer-increment l. For example, for $r = 3$ and $l = 4$, the two packet keys are k^{34a} and k^{34b}, one for encrypting packets in the WOI and the other for packets in the "Boarder Region."

In our first experiment, we assume that a user purchases the image at resolution 1. The user obtains key k^1 and gets the image shown in Fig. 13(a). Since the user is not able to derive k^r, for $r > 1$, she/he is not able to decrypt packets corresponding to higher resolutions. However, the user may still try to construct images of higher resolutions by extrapolating the granted image with encrypted packets corresponding to higher resolutions. Fig. 13(b) shows the extrapolated image of resolution 2. To quantitatively describe the quality degradation of the extrapolated images, we adopt the traditional definition of the PSNR (peak signal-noise-ratio) between a reference image $\mathbf{I}$ and an inspected image $\tilde{\mathbf{I}}$, both of size $W \times H$, as

$$PSNR = 10(\log_{10} 255^2 - \log_{10} \sigma^2) \tag{10}$$

$$\sigma^2 = \frac{1}{WH} \sum_{x=1}^{W} \sum_{y=1}^{H} \Big(\mathbf{I}(x,y) - \tilde{\mathbf{I}}(x,y) \Big)^2. \tag{11}$$

Generally speaking, if $PSNR > 30$ dB [1], an image is regarded as visually acceptable. Using the image in Fig. 13(c) (resolution 2 and the highest layer) as the reference, the PSNR of the image in Fig. 13(b) is 27.9 dB given that the undecrypted bitstream is used directly. Obviously, the extrapolated image is of unacceptable quality.

In the next experiment, we assume that the user requests the image with resolution 2 and layer 4. Accordingly, the key server supplies the user with keys k^{24}, k^{14}, and k^{04}. The image granted access is given in Fig. 14(a). Failing to obtain packets for layers higher than 4, the user tries to get an image at layer 7 by extrapolating the granted image with the encrypted higher layer packets. The resulting image is shown in Fig. 14(b), which ends up has a much lower quality than the granted image.

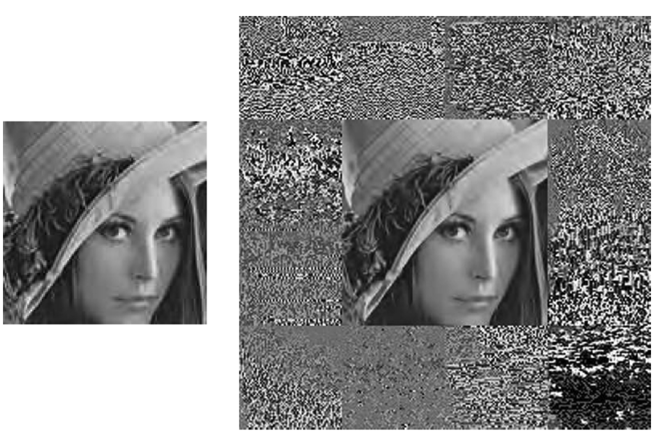
Our third experiment is related to access to WOI. Here the user requests to access the WOI (i.e., the 4 center precincts) with resolution 3 and layer 4. In response, the key server supplies the user with keys k^{rla}, $r = 0, 1, 2, 3$ and $l = 0, 1, 2, 3, 4$. The granted WOI is shown in Fig. 15(a) while the extrapolated image is shown in Fig. 15(b).

TABLE II
KEYS ASSOCIATED WITH THE ROOTED TREE FOR THE "LENNA" CODE-STREAM

k^3	k^{37}	$\cdots$	k^{34}	$\cdots$	k^{31}	k^{30}
	k^{37a}, k^{37b}	$\cdots$	k^{34a}, k^{34b}	$\cdots$	k^{31a}, k^{31b}	k^{30a}, k^{30b}
k^2	k^{27}	$\cdots$	k^{24}	$\cdots$	k^{21}	k^{20}
	k^{27a}, k^{27b}	$\cdots$	k^{24a}, k^{24b}	$\cdots$	k^{21a}, k^{21b}	k^{20a}, k^{20b}
k^1	k^{17}	$\cdots$	k^{14}	$\cdots$	k^{11}	k^{10}
	k^{17a}, k^{17b}	$\cdots$	k^{14a}, k^{14b}	$\cdots$	k^{11a}, k^{11b}	k^{10a}, k^{10b}
k^0	k^{07}	$\cdots$	k^{04}	$\cdots$	k^{01}	k^{00}
	k^{07a}, k^{07b}	$\cdots$	k^{04a}, k^{04b}	$\cdots$	k^{01a}, k^{01b}	k^{00a}, k^{00b}

(a) (b)

Fig. 14. Results of the second experiment (using the image in Fig. 13(c) as the reference). (a) Granted image $(\mathrm{PSNR} = 54\ \mathrm{dB})$ and (b) extrapolated image $(\mathrm{PSNR} = 41.5\ \mathrm{dB})$.



(a) (b)

Fig. 15. Results of the third experiment. (a) Granted WOI and (b) extrapolated image.

VII. CONCLUSION

Based on the state-of-the-art wavelet technology, the JPEG 2000 is an emerging international standard for image compression. Part 8 of the JPEG 2000 standard is concerned with JPEG 2000 code-stream security with particular emphasis on flexible access control and authentication.

Our access control scheme for JPEG 2000 code-streams uses hash functions and rooted trees for systematic key generation and packet encryption. Since code-streams are protected by encryption, all conceivable ways of content distribution, such as broadcast and CD-ROM publishing, can be used. A user who desires to access any part of the code-stream interacts with a key server for authentication and key acquisition. The scheme is secure and efficient, and very importantly, flexible. That is, our scheme allows access control to JPEG 2000 code-streams according to any combination of resolution, quality layer and window-of-interest. From this point of review, our scheme is designed for "**encrypt once, decrypt many ways**," which matches perfectly with the "*compress once, decompress many ways*" property of the JPEG 2000 code-streams. We have implemented our access control scheme in a prototype which demonstrated the practical feasibility and compatibility of the proposed scheme with JPEG 2000 standard Part 1. The proposed scheme has been incorporated into the JPSEC [4].

REFERENCES

[1] D. S. Taubman and M. W. Marcellin, *JPEG 2000 Image Compression Fundamentals, Standard and Practice*. Norwell, MA: Kluwer, 2000.
[2] M. Rabbani and R. Joshi, "An overview of the JPEG 2000 still image compression standard," *Signal Process.: Image Commun.*, vol. 17, pp. 3–48, 2002.
[3] *ITU-T Rec. T.800*, ISO 154447 [Online]. Available: http://www.jpeg.org
[4] *Information technology—JPEG 2000 image coding system-Part 8: Secure JPEG 2000*, ISO/IEC 15444-8, Apr. 2007.
[5] R. S. Sandhu, "Cryptographic implementation of a tree hierarchy for access control," *Inform. Process. Lett.*, vol. 27, no. 2, pp. 95–98, 1988.
[6] E. Bertino, S. Jajodia, and P. Samarati, "Access controls in object-oriented database systems—Some approaches and issues," *Adv. Database Syst.*, vol. 759, pp. 17–44, 1993.
[7] R. S. Sandhu and P. Samarati, "Access control: Principle and practice," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 40–48, 1994.
[8] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 239–248, 1983.
[9] G. C. Chick and S. E. Tavares, "Flexible access control with master keys," in *Proc. Advances in Cryptology-Crypto'89*, 1990, vol. 435, LNCS, pp. 316–322.

[10] L. Harn and H. Y. Lin, "A cryptographic key generation scheme for multi-level data security," *J. Comput. Secur.*, vol. 9, no. 6, pp. 539–546, 1990.

[11] S. J. MacKinnon, P. D. Taylor, H. Meijer, and S. G. Akl, "An optimal algorithm for assigning cryptographic keys to access control in a hierarchy," *IEEE Trans. Comput.*, vol. C-34, no. 9, pp. 797–802, 1985.

[12] K. Ohta, T. Okamoto, and K. Koyama, "Membership authentication for hierarchical multigroup using the extended Fiat-Shamir scheme," in *Proc. Advances in Cryptology-Eurpcrypt'90*, 1991, vol. 473, pp. 316–322.

[13] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 637–647, 1978.

[14] R. Grosbois, P. Gerbelot, and T. Ebrahimi, "Authentication and access control in the JPEG 2000 compression domain," in *Proc. SPIE 46th Annu. Meeting: Applications of Digital Image Processing XXIV*, 2001, vol. 4472, pp. 95–104.

[15] S. Wee and J. Apostolopoulos, "Secure scalable streaming and secure transcoding with JPEG-2000," in *Proc. IEEE Int. Conf. Image Process.*, 2003, vol. 1, pp. 14–17.

[16] Y. Wu, D. Ma, and R. H. Deng, "Progressive protection of JPEG 2000 code-streams," in *Proc. IEEE Int. Conf. Image Processing*, 2004, pp. 3439–3442.

[17] Y. Wu and R. H. Deng, "Compliant encryption of JPEG 2000 code-streams," in *Proc. IEEE Int. Conf. on Image Processing*, 2004, pp. 3447–3450.

[18] B. B. Zhu, Y. Yang, and S. Li, "An efficient key scheme for multiple access of JPEG 2000 and motion JPEG 2000 enabling truncations," in *Proc. IEEE Consumer Communications and Networking Conference (CCNC)*, Jan. 2006, vol. 2, pp. 1124–1128.

[19] ISO/IEC FDIS 15444-9: Information Technology—JPEG 2000 Image Coding System—Part 9: Interactivity Tools, APIs and Protocols Jul. 1, 2004.

[20] R. Mori and M. Kawahara, "Superdistribution: The concept and the architecture," *Trans. IEIEC* vol. E73, no. 7, Jul. 1990 [Online]. Available: http://www.virtualschool.edu/mon/ElectronicProperty/MoriSuperdist.html

[21] R. S. Sandhu, "Lattice-based access control models," *IEEE Computer*, vol. 26, no. 11, pp. 9–19, 1993.

[22] I. Ray, I. Ray, and N. Narasimhamurthi, "A cryptographic solution to implement access control in a hierarchy and more," in *Proc. 7th ACM Symp. Access Control Models and Technologies*, 2002, pp. 65–73.

[23] National Institute of Standards and Technology, Secure Hash Standard (SHS) FIPS Publ. 180-1, 1995.

[24] R. C. Merkle, "A certified digital signature," in *Proc. Advances in Cryptology-Crypto '89*, 1989, vol. 435, pp. 218–238.

[25] R. L. Rivest, RC4 Encryption Algorithm RSA Data Security, Inc., Mar. 12, 1992.

Yongdong Wu received the B.A and M.S. degrees in automation control from Beijing University of Aeronautics and Astronautics, Beijing, China, in 1991 and 1994, respectively, and the Ph.D. degree in pattern recognition and intelligent control from the Institute of Automation, Chinese Academy of Science, in 1997.

He is currently a Principal Investigator with the Infocomm Security Department, Institute of Infocomm Research, Singapore. His interests includes multimedia security, e-business, digital right management, and network security. He has five patents and 60 technical publications in international conferences and journals.

Dr. Wu won the Tan Kah Kee Young Inventor Award in 2004 and 2005.

Di Ma received the B. Eng. degree in computer science and engineering and the M. Eng degree in electrical engineering from Xi'an Jiaotong University, China, in 1995 and 1998, respectively. She received her second M. Eng degree in computer science from Nanyang Technological University, Singapore, in 2000. She is currently pursuing the a Ph.D. degree at the University of California, Irvine.

She was a Senior Research Engineer with the Infocomm Security Department, Institute for Infocomm Research, Singapore. Her research interests include applied cryptography and security in multimedia, networks, and databases. She has 15 technical publications in international conferences and journals.

Ms. Ma was the recipient of Tan Kah Kee Young Investigator Award in 2004.

Robert H. Deng received the B.S. degree from the National University of Defense Technology, China, and the M.Sc. and Ph.D. degrees from the Illinois Institute of Technology, Chicago.

He has been with Singapore Management University since 2004, and is currently Professor, Associate Dean for Faculty and Research, and Director of the SIS Research Center, School of Information Systems. Prior to this, he was Principal Scientist and Manager of the Infocomm Security Department, Institute for Infocomm Research. He has 26 patents and more than 150 technical publications in international conferences and journals in the areas of computer networks, network security, and information security.

Dr. Deng has served as general chair, program committee chair, and member of numerous international conferences. He received the University Outstanding Researcher Award from the National University of Singapore in 1999 and the Lee Kuan Yew Fellow for Research Excellence from the Singapore Management University in 2006.