ADAPTIVE CODE EMBEDDING FOR REVERSIBLE DATA HIDING IN ENCRYPTED IMAGES

Shuang Yi, Yicong Zhou[*]

Department of Computer and Information Science, University of Macau, Macau 999078, China

ABSTRACT

In this paper, we propose an adaptive code embedding method for reversible data hiding in encrypted images (ACE-RDHEI). It encrypts the original image into a noise-like one by block permutation and modulation, and no reserving room process is needed before image encryption. The secret data is then embedded into the encrypted image using adaptive code embedding (ACE). At the receiver side, using different security keys, users are able to extract the secret data and recover the original image separately and losslessly. Compared with existing RDHEI methods, ACE-RDHEI significantly improves the embedding rate. Experimental results are provided to show the excellent performance of our proposed algorithm.

***Index Terms*—** Reversible data hiding, encrypted image, adaptive code embedding, privacy protection

1. INTRODUCTION

Cryptography and data hiding are two techniques used for digital image privacy protection. The former one aims to change the image into a noise-like one to prevent unauthorized access, while the later one is to embed the secret data into a cover image in an unnoticeable way. Reversible data hiding (RDH) [1] is able to perfectly recover the cover image after extracting the secret data. In recent years, due to the development of the cloud computing, RDH in encrypted images (RDHEI) has caught many researchers' attention. It can be utilized in many applications, such as medical images, law forensics and military applications [2,3]. For example, in order to keep the patient privacy, the doctor encrypts medical images before sending them to the cloud. To manage these images easily, the cloud provider or other administrator may hope to add some notations (or additional secret data), such as the image source and categories, to the image without knowing its content. At the receiver side, we hope the authorized user can extract secret data and recover the original image separately and losslessly.

This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/016/2015/A1 and by the Research Committee at University of Macau under Grants MYRG2014-00003-FST and MYRG2016-00123-FST.

Corresponding author. Tel.: +853 83978458; Fax: +853 88222426, E-mail address: yicongzhou@umac.mo

A lot of RDHEI methods have been proposed in recent years. They can be divided into two categories, namely vacating room before encryption (VRBE) and vacating room after encryption (VRAE) [4]. The former one needs a preprocessing before image encryption. Although it can achieve a higher embedding rate and better performance than that of VRAE, this may not be applicable because the content-owner may have difficulty to do such complex operation [5]. Therefore, many researchers show their interests on developing VRAE methods.

The early published VRAE methods [3, 6–8] divide the original image into blocks and encrypt each block by stream cipher. Secret data is then embedded into the encrypted image by flipping a number of least significant bits (LSBs) within the block. In [9], secret data is embedded by using the public key modulation mechanism. In these methods, data extraction and image recovery can be jointly applied. In addition, they may result in incorrect extraction of secret data and recovery of original image when block size is small. Later, a number of separable RDHEI methods have been proposed so that data extraction and image recovery can be accomplished independently. Methods in [10–13] compress several LSB planes in the encrypted image to accommodate secret data. However, the embedding rate is limited due to the difficulty for reserving room in an encrypted image. In Yin *et al.*'s methods, image are encrypted by permutation [14] and modulation [15], secret data is embedded by histogram shifting method [1]. However, they also suffer from low embedding rate. In addition, method in [15] requires the original image contains no saturation values. In Wu *et al.*'s separable method [16], image is encrypted by stream cipher and secret data is embedded by modifying the two most significant bits in the encrypted image.

In order to improve the embedding rate while keeping the merits of VRAE and separability, we propose a RDHEI method using adaptive code embedding (ACE-RDHEI). It encrypts the original image into noise-like one while keeping redundancy within small image blocks, so that secret data can be embedded into the encrypted image by exploiting its spatial correlations within each block.

The rest of this paper is organized as follows: Section 1 introduces the proposed ACE-RDHEI in detail. Section 3 shows the experimental results and comparisons. Section 4 draws a

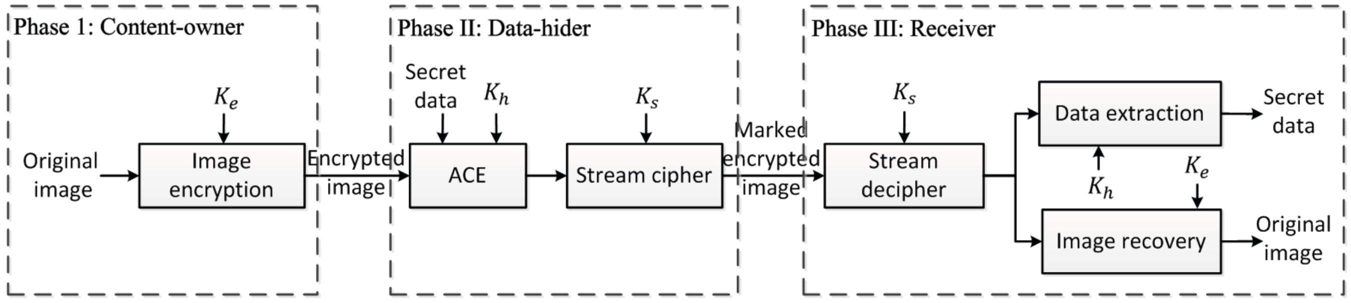

Fig. 1: Framework of the proposed ACE-RDHEI.

conclusion.

2. ACE-RDHEI

Fig. 1 shows the framework of the proposed ACE-RDHEI, which consists of three phases: (1) image encryption; (2) data embedding and (3) data extraction and image recovery. These three phases are accomplished by the content-owner, data-hider and receiver, respectively. Next, we will introduce each phase in detail.

2.1. Image encryption

Suppose an $M \times N$ original image $\mathbb{I}$ is with a depth of 8-bits. Firstly, the content-owner divides $\mathbb{I}$ into a series of 2×2 non-overlapped blocks. Thus, k blocks are obtained, where $k = MN/4$. According to the image encryption key K_e, blocks are permutated within the image, and the obtained image is denoted as $\dot{\mathbb{I}}$. We denote the i^{th} block in $\dot{\mathbb{I}}$ as $\dot{\mathbb{I}}_i$, where $i = 1, 2, ..., k$. Then, pixel in each block is modified by

$$\mathbb{E}_i^j = (\dot{\mathbb{I}}_i^j + \mathbf{R}_i) \mod 256, \quad (j = 1, 2, 3, 4) \tag{1}$$

where $\dot{\mathbb{I}}_i^j$ is the j^{th} pixel in the i^{th} block of image $\dot{\mathbb{I}}$ in raster scan order. $\mathbf{R}_i \in [0, 255]$ is a random value generated by K_e. Thus, the encrypted image $\mathbb{E}$ is obtained. Because the same value $\mathbf{R}_i$ is added to each pixel of $\dot{\mathbb{I}}_i$ for modulation, the resulted block $\mathbb{E}_i$ will mostly keep the same similarity as it is in the original block $\dot{\mathbb{I}}_i$. Thus, we can embed secret data into the encrypted image by exploiting the spatial redundancy within image blocks.

2.2. Data embedding

2.2.1. Pixel classification

After obtaining the encrypted image $\mathbb{E}$, the data-hider fist divides it into k blocks as it is in the image encryption phase. Then, all pixels within the image are classified into 4 categories, namely (1) reference pixel (RP); (2) specific pixel (SP); (3) embeddable pixel (EP) and (4) non-embeddable pixel (NP). RP denotes the set of the first pixel $\mathbb{E}_i^1$ in each block. Thus, RP consists of k pixels and all of them will be kept unmodified during the whole data embedding phase. SP denotes the second pixel $\mathbb{E}_1^2$ in the first block, and it is utilized to store parameter that will be generated later. For each of the rest $(3k - 1)$ pixels, we first calculate its difference value e by

$$e = \mathbb{E}_i^j - \mathbb{E}_i^1, \quad (j = 2, 3, 4) \tag{2}$$

Given a parameter α, if e satisfies Eq. (3), the pixel is classified into EP; otherwise, it belongs to NP.

$$\lceil -2^{\alpha-2} \rceil \leqslant e \leqslant \lceil 2^{\alpha-2} - 1 \rceil, \quad (1 \leqslant \alpha \leqslant 7) \tag{3}$$

Suppose EP and NP consist of n_1 and n_2 pixels, respectively. Thus, $n_1 + n_2 + 1 + k = MN$.

2.2.2. Adaptive code embedding (ACE)

Then, we use ACE to embed secret data into the encrypted image. Because the parameter α will be used for data extraction and image recovery at the receiver side, and it can be represented by 3 bits, we embed it into the first 3 bits in SP by bit replacement. For each pixel in NP, we replace its first 1 bit by a special bit '1', and keep the remaining 7 bits unmodified. The original 3 bits in SP and first 1 bit of each pixel in NP will be selected and stored with secret data. For each pixel in EP, we replace its first α bits by a predefined code and embed secret data into the rest $(8 - \alpha)$ bits by bit replacement. We denote the encrypted image after ACE as $\dot{\mathbb{E}}$.

Fig. 2 shows an example of codes when $\alpha = 1, 2, 3$ and 4. Taking $\alpha = 3$ for example, according to Eq. (2), $e \in \{-2, -1, 0, 1\}$. Thus, $2^{\alpha-1}$ bits ('00', '01', '10', '11') can denote the 4 cases of e by one-to-one mapping. In order to distinguish EP from NP, we add a '0' to the front of each case of codes. Therefore, $\{000, 001, 010, 011\}$ are one-to-one mapped to 4 cases of e, respectively.

Given an α, we can calculate the effective embedding rate (ER$_\alpha$) by

$$\text{ER}_\alpha = \frac{n_1 * (8 - \alpha) - n_2 - 3}{MN} \quad \text{(bpp)} \tag{4}$$

The maximum embedding rate (ER$_{max}$) is then obtained by

$$\text{ER}_{max} = \max\{\text{ER}_\alpha\}_{\alpha=1}^{7} \quad \text{(bpp)} \tag{5}$$

(a)

	NP	EP
$\alpha = 1$	1	$e = 0$
		0

(b)

	NP	EP	
$\alpha = 2$	1	$e = -1$	$e = 0$
		00	01

(c)

	NP	EP			
$\alpha = 3$	1	$e = -2$	$e = -1$	$e = 0$	$e = 1$
		000	001	010	011

(d)

	NP	EP							
$\alpha = 4$	1	$e = -4$	$e = -3$	$e = -2$	$e = -1$	$e = 0$	$e = 1$	$e = 2$	$e = 3$
		0000	0001	0010	0011	0100	0101	0110	0111

Fig. 2: Example of codes when (a) $\alpha = 1$; (b) $\alpha = 2$; (c) $\alpha = 3$ and (d) $\alpha = 4$.

Thus, the payload $\mathbf{P}$ consists of three parts: (1) the first three bits in SP ($\mathbb{E}_1^2$); (2) the first 1 bit for each pixel in NP and (3) the encrypted secret data. Note that, in order to achieve a higher security, the secret data is encrypted by K_h before embedding.

2.2.3. *Stream cipher*

After ACE, using K_s, pixels in SP, EP and NP are modified by a stream cipher to obtain the final marked encrypted image $\mathbb{D}$ as shown in Eq. (6), and pixels in RP will be kept unmodified. Here, $\dot{\mathbb{E}}_{(z)}$ means the z^{th} pixel in set {SP, EP, NP}; $\mathbf{T}_z$ is a random integer within the range of [0, 255] and generated by K_s; '$\oplus$' is the bit-level X-OR operation.

$$\mathbb{D}_{(z)} = \dot{\mathbb{E}}_{(z)} \oplus \mathbf{T}_z, \quad (z = 1, 2, ..., 3k) \tag{6}$$

2.3. Data extraction and image recovery

At the receiver side, using different combination of security keys, the secret data and original image can be extracted and recovered separately.

2.3.1. *Data extraction*

When holding keys K_s and K_h, users are able to extract the secret data. Firstly, the marked encrypted image $\mathbb{D}$ are divided into k non-overlapped blocks as in encryption phase. Then, all pixels except in RP are stream deciphered by

$$\dot{\mathbb{E}}_{(z)} = \mathbb{D}_{(z)} \oplus \mathbf{T}_z, \quad (z = 1, 2, ..., 3k) \tag{7}$$

where $\mathbf{T}_z$ is generated by the same way that in data embedding phase.

After obtaining the encrypted image $\dot{\mathbb{E}}$ with data embedded in, we first extract parameter α from the pixel $\dot{\mathbb{E}}_1^2$. For the rest $(3k-1)$ pixels, we check them one-by-one to see whether they belong to EP or NP. For each pixel in EP, we then extract $(8 - \alpha)$ bits of the payload. Finally, we extract the encrypted secret data from the payload and decrypt it using K_h to obtain the plaintext secret data.

2.3.2. *Image recovery*

To recover the original image, we first obtain the image $\dot{\mathbb{E}}$ and payload $\mathbf{P}$ by the same way that in data extraction phase. Then, we recover the first 3 bits of SP and the first 1 bit of each pixel in NP using $\mathbf{P}$. For each pixel in EP, according to the first α bits code, we first obtain its corresponding difference value e based on the one-to-one mapping rule. Then, we recover pixels in EP by

$$\mathbb{E}_i^j = \dot{\mathbb{E}}_i^j + e, \quad (j = 2, 3, 4) \tag{8}$$

Thus, the encrypted image $\mathbb{E}$ is obtained. Next, we decrypt it by

$$\dot{\mathbb{I}}_i^j = (\mathbb{E}_i^j - \mathbf{R}_i) \mod 256, \quad (j = 2, 3, 4) \tag{9}$$

where $\mathbf{R}_i$ is generated by the same way as it is in the image encryption phase. Finally, we inversely permutate these k blocks using K_e to obtain the final recovered image $\mathbb{I}$.

3. EXPERIMENTAL RESULTS

In this section, we show the experimental results of the proposed ACE-RDHEI and comparisons with several existing related works. All test images used in this paper are with a size of 512×512 and selected from the Miscelaneous[1] database.

Fig. 3 shows the simulation of ACE-RDHEI using *Lena* image. When $\alpha = 5$, it reaches the maximum embedding rate of 1.579 bpp. The encrypted image (Fig. 3(b)) and marked encrypted image (Fig. 3(c)) are both noise-like ones to prevent unauthorized access. The recovered image (Fig. 3(d)) is exactly the same with the original one (Fig. 3(a)), no distortion will occur.

The detailed pixel value distribution of the original, encrypted and marked encrypted *Lena* images are shown in Fig. 4. From the results we can observe that pixel values in the encrypted image (Fig. 4(b)) and marked encrypted image

[1] http://decsai.ugr.es/cvg/dbimagenes/g512.php.

Table 1: Effective embedding rate of different images under various α

ER_α (.bpp)	α						
	1	2	3	4	5	6	7
Lena	—	0.16	0.70	1.32	**1.58**	1.27	0.68
Airplane	0.14	0.62	1.28	**1.72**	1.68	1.26	0.66
Peppers	—	0.07	0.55	1.15	**1.53**	1.29	0.68
Barbara	—	—	0.35	0.85	**1.09**	0.94	0.54
Boat	—	0.20	0.77	1.32	**1.43**	1.16	0.64
Lake	—	—	0.30	0.79	**1.15**	1.09	0.62

Table 2: Comparisons of the maximum embedding rate when the original image can be lossless recovered.

ER_{max} (.bpp)	Zhang [3]	Hong [6]	Zhang [17]	Li [8]	Zhang [10]	Wu [16]	Yin [15]	Yin [14]	Zhou [9]	ACE-RDHEI
Lena	0.004	0.004	0.005	0.010	0.036	0.060	0.12	0.13	0.15	**1.58**
Airplane	0.001	0.001	0.005	0.013	0.036	0.039	0.19	0.21	0.19	**1.72**
Barbara	0.001	0.001	0.001	0.005	0.036	0.001	0.09	0.10	0.15	**1.09**
Peppers	0.004	0.004	0.005	0.006	0.036	0.009	0.11	0.12	0.19	**1.53**
Boat	0.001	0.004	0.005	0.009	0.036	0.009	0.14	0.15	0.15	**1.43**
Lake	0.001	0.004	0.005	0.005	0.036	0.007	0.09	0.10	0.01	**1.15**

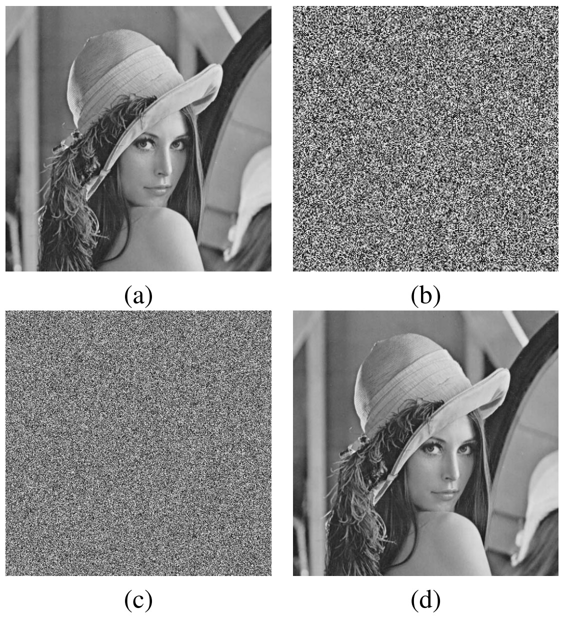

Fig. 3: Simulation results of the proposed algorithm in *Lena* image: (a) the original image; (b) encrypted image; (c) marked encrypted image with $\alpha = 5$, $\text{ER}_{max} = 1.579$ bpp; (d) recovered image, PSNR = $+\infty$ dB.

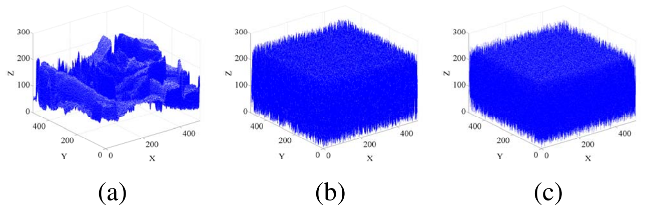

Fig. 4: The distribution of the pixel values in test image *Lena*. (a) The original image; (b) the encrypted image and (c) the marked encrypted image.

Table 3: UACI and NPCR results between the original and encrypted images ($\mathbb{I}, \mathbb{E}$), and original and marked encrypted images ($\mathbb{I}, \mathbb{D}$).

	UACI (%)		NPCR (%)	
	$(\mathbb{I}, \mathbb{E})$	$(\mathbb{I}, \mathbb{D})$	$(\mathbb{I}, \mathbb{E})$	$(\mathbb{I}, \mathbb{D})$
Lena	28.72	28.62	99.62	99.61
Airplane	32.31	32.29	99.62	99.61
Barbara	29.92	29.85	99.59	99.59
Peppers	31.12	31.02	99.58	99.58
Boat	29.56	29.45	99.58	99.59
Lake	31.44	31.33	99.58	99.58

(Fig. 4(c)) are uniform distributed in the whole image. Thus, the security can be ensured.

Table 1 lists the effective embedding rate of different images under various α. From the results we can observe that, different settings of α will result in different embedding rates. Image with less texture is able to achieve a higher embedding rate than that with more texture. When α is small, more textured images may not able to embed secret data (e.g., $\alpha = 1, 2$ for *Barbara, Lake*). The maximum embedding rate can reach as high as 1.72 bpp (*Airplane*, $\alpha = 4$).

Table 2 compares the maximum embedding rate of ACE-RDHEI with several existing VRAE methods. The results are under the same situation that the original image can be lossless recovered. For methods in [3] and [6], we set the image block size to 8×8 for demonstration and for [14] and [15], it is set to 4×4. For Wu *et al.*'s method [16] and Zhou *et al.*'s method [9], each pixel group and image block are embedded with 4 and 3 bits of the secret data, respectively. From the results we can observe that, the proposed algorithm significantly improves the maximum embedding rate under various images.

We use the Unified Changing Intensity (UACI) and Number if Pixel Change Rate (NPCR) to evaluate the differences between the original and encrypted images, and original and marked encrypted images, respectively. The results are listed in Table 3. As we can see, the UACI and NPCR results are close to their theoretical values, 33.464% and 99.609%, which proved in [18]. Thus, the security can be ensured by the encryption effect.

4. CONCLUSION

In this paper, we proposed an encryption domain based reversible data hiding method using adaptive code embedding. It encrypts the original image by block permutation and modulation. Secret data is then embedded into the encrypted image by exploiting the spatial correlations of pixels within the block. Experimental results and comparisons have shown the excellent performance of our proposed algorithm.

5. REFERENCES

[1] Zhicheng Ni, Yun-Qing Shi, N. Ansari, and Wei Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354–362, 2006.

[2] Y. Q. Shi, X. Li, X. Zhang, H. Wu, and Ma B., "Reversible data hiding: Advances in the past two decades," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2016, doi: 10.1109/ACCESS.2016.2573308.

[3] Xinpeng Zhang, "Reversible data hiding in encrypted image," *IEEE Signal Processing Letters*, vol. 18, no. 4, pp. 255–258, 2011.

[4] K. Ma, Weiming Zhang, Xianfeng Zhao, Nenghai Yu, and Fenghua Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 553–562, 2013.

[5] Z. Qian and X. Zhang, "Reversible data hiding in encrypted images with distributed source encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 4, pp. 636–646, 2016.

[6] Wien Hong, Tung-Shou Chen, and Han-Yan Wu, "An improved reversible data hiding in encrypted images using side match," *IEEE Signal Processing Letters*, vol. 19, no. 4, pp. 199–202, 2012.

[7] Jie Yu, Guopu Zhu, Xiaolong Li, and Jianquan Yang, "An improved algorithm for reversible data hiding in encrypted image," *Digital Forensics and Watermaking*, vol. 7809, pp. 384–394, 2013.

[8] Ming Li, Di Xiao, Zhongxian Peng, and Hai Nan, "A modified reversible data hiding in encrypted images using random diffusion and accurate prediction," *ETRI Journal*, vol. 36, no. 2, pp. 325–328, 2014.

[9] J. Zhou, W. Sun, L. Dong, X. Liu, O. C. Au, and Y. Y. Tang, "Secure reversible image data hiding over encrypted domain via key modulation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–1, 2015.

[10] Xinpeng Zhang, "Separable reversible data hiding in encrypted image," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 826–832, 2012.

[11] Xinpeng Zhang, Zhenxing Qian, Guorui Feng, and Yanli Ren, "Efficient reversible data hiding in encrypted images," *Journal of Visual Communication and Image Representation*, vol. 25, no. 2, pp. 322–328, 2014.

[12] Shuli Zheng, Dandan Li, Donghui Hu, Dengpan Ye, Lina Wang, and Jinwei Wang, "Lossless data hiding algorithm for encrypted images with high capacity," *Multimedia Tools and Applications*, pp. 1–14, 2015.

[13] Xinpeng Zhang, Chuan Qin, and Guangling Sun, "Reversible data hiding in encrypted images using pseudorandom sequence modulation," *Digital Forensics and Watermaking*, vol. 7809, pp. 358–367, 2013.

[14] Zhaoxia Yin, Bin Luo, and Wien Hong, "Separable and error-free reversible data hiding in encrypted image with high payload," *The Scientific World Journal*, vol. 2014, pp. 8, 2014.

[15] Zhaoxia Yin, Huabin Wang, Haifeng Zhao, Bin Luo, and Xinpeng Zhang, "Complete separable reversible data hiding in encrypted image," *Cloud Computing and Security: First International Conference, ICCCS 2015*, pp. 101–110, 2015.

[16] Xiaotian Wu and Wei Sun, "High-capacity reversible data hiding in encrypted images by prediction error," *Signal Processing*, vol. 104, pp. 387–400, 2014.

[17] Xinpeng Zhang, Chuan Qin, and Guangling, "Reversible data hiding in encrypted images using pseudorandom sequence modulation," *Digital Forensics and Watermaking*, vol. 7809, pp. 358–367, 2013.

[18] Chong Fu, Jun-jie Chen, Hao Zou, Wei-hong Meng, Yong-feng Zhan, and Ya-wen Yu, "A chaos-based digital image encryption scheme with an improved diffusion strategy," *Optics Express*, vol. 20, no. 3, pp. 2363–2378, 2012.