SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2016, Vol. 59 032105:1–032105:15
doi: 10.1007/s11432-015-5367-x

Stego key searching for LSB steganography on JPEG decompressed image

Jiufen LIU[1,2,3], Yuguo TIAN[1,2], Tao HAN[1,2]*, Junchao WANG[1,2] & Xiangyang LUO[1,2]

[1]*Zhengzhou Information Science and Technology Institute, Zhengzhou* 450001*, China;*
[2]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou* 450001*, China;*
[3]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093*, China*

Received March 7, 2015; accepted May 8, 2015; published online January 22, 2016

Abstract At present, steganalysis research focuses on detecting the existence of a hidden message. However, extracting the hidden information, i.e., an extracting attack, is crucial in obtaining effective evidence in computer forensics. Due to the difficulty of an extracting attack, research in this field is limited. In steganography with a stego key, an extracting attack is equivalent to recovering the stego key. In this paper we study a method for recovering the stego key in least significant bit (LSB) steganography with a decompressed JPEG image as the cover image. Firstly, the recovery of the stego key is translated into a cryptanalysis problem for a sequential cipher. The method for recovering the stego key is based on estimating the modification positions. The minimum size of the data used to recover the stego key successfully is discussed. Secondly, when a decompressed JPEG image is used as the cover image, the probability of recovering the cover pixels using recompression is discussed. Recompression is used to compute the error of the estimated sequence. Finally, an algorithm to recover the stego key in LSB steganography with a decompressed JPEG image as the cover image is proposed. The experimental results for the steganographic software, Hide and Seek 4.1 and its variant, which is a typical representative of LSB steganography, show that the proposed method can successfully recover the stego key in LSB replacement and LSB matching, i.e., the extracting attack is successful, and it outperforms three previous methods in terms of computational complexity.

Keywords steganalysis, extracting attack, LSB steganography, decompressed JPEG image, recompression, stego key

Citation Liu J F, Tian Y G, Han T, et al. Stego key searching for LSB steganography on JPEG decompressed image. Sci China Inf Sci, 2016, 59(3): 032105, doi: 10.1007/s11432-015-5367-x

1 Introduction

Image steganography mainly focuses on how secret messages are embedded into public digital images to realize covert communication. LSB-R (least significant bit replacement) and LSB-M (LSB matching) are two widely used algorithms in steganography. For instance, among the steganography software listed by Dr. Neil F. Johnson of Johnson & Johnson Technology Consultants, LLC (JJTC)[1)], more than half of the programs were developed based on least significant bit (LSB) steganography.

* Corresponding author (email: lhstslhsts@163.com)
1) There are 146 steganography programs listed at http://www.jjtc.com/Steganography/toolmatrix.htm.

An attack against steganography, called steganalysis, mainly focuses on how to detect, extract, and restore or damage a hidden secret message. Currently, research into steganalysis mainly focuses on the detection of the hidden information, which under some circumstances, is enough to solve some problems. However, extracting the hidden information, i.e., an extracting attack, is crucial in obtaining effective evidence in computer forensics. Moreover, research into this problem is also very meaningful for detection. This is because current detection methods decide whether there is hidden information based on whether the cover has been modified, though a modified cover does not always carry any information. Finally, the reliability of the detection may be called into question. It is well known that an extracting attack is a very difficult problem. On one hand, the target of detection is to extract one bit from the existing resources, i.e., to decide whether the image is a stego image or not, while an extracting attack aims to get all the hidden bits, which is far more difficult than detection. On the other hand, recovering the stego key is important in an extracting attack, which is essentially a form of cryptanalysis. However, an extracting attack is very different from traditional cryptanalysis, where the relation between the stego image and stego key is weaker than that between the ciphertext and encryption key. Therefore, an extracting attack faces bigger challenges than traditional cryptanalysis.

So far, to the best of our knowledge, there have been only a few research works on the theory of extracting attacks. Using the information-theoretic approach, we preliminarily studied the relationships among the extraction difficulty of the hidden message, concealment, embedding capacity and key rate [1]. Based on the information-theoretic model, Regalia researched the extraction difficulty for a hidden message embedded by matrix embedding for a stego-only attack [2]. We deduced the theoretical bound of key equivocation and message equivocation of matrix embedding for a carrier-known attack [3]. For a chosen-stego attack, we gave the unicity distance of the stego key. Moreover, we studied the security measurements of matrix embedding for a carrier-known attack and proposed a differential attack method for a chosen-stego attack [4].

Fridrich et al. [5] pointed out that for steganography using a stego key, an extracting attack is equivalent to the recovery of the stego key. At present, there are also only a few research works on recovering the stego key. Fridrich et al. [5, 6] studied the chi-square test, which was used to distinguish the cover image from the stego image in steganalysis and to distinguish the true key from the false key. This method was used to recover the stego key of LSB steganography based on a JPEG image (e.g., F5, Outguess) and generalized to LSB-R and LSB-M based on a spatial image. If the carrier is known, we observed a portion of embedding positions by comparing the carrier and the stego, and recovered the stego key for LSB-R and LSB-M using a single-key collision attack [7]. Besides, when reusing the carrier, we recorded the difference of two stego images, and recovered the stego key for LSB-R and LSB-M using a double-key collision attack and a divide-and-conquer attack [7]. For the stego-only condition, we estimated a portion of the embedding positions of LSB-R according to the noise values of the pixels, and restored the stego key using a single-key collision attack [8]. For the stego-only condition, Liu et al. defined a middle pixel set M, and then calculated the difference value between each pixel and its shifted pixel belonging to the set M on the embedding path. Finally, the stego key corresponding to the largest difference value is identified as the true key [9].

There are also only a few research works on the quick recovery of a stego key. We analyzed the equivalent key number of a steganographic algorithm based on the binary image, CPT (Chen-Pan-Tseng), and gave a method of quickly obtaining the equivalent key for the chosen-stego condition [10]. Aiming at selection mechanisms for embedding positions in accordance with an oriented cycle model, we used the dichotomy and mutation detection technology to decrease the computational complexity of recovering the stego key for LSB-R in the JPEG domain [11].

The embedding capacity of a spatial image is larger than for a JPEG image. Therefore, for a long message, unprofessional steganographers may select a spatial image as the carrier and many steganographic systems on the Internet output a spatial image. For example, among the 146 steganography programs listed by Dr. Neil F. Johnson of Johnson, 85 were developed for digital images. More than 60 programs can output spatial stego images and more than 30 were developed for spatial LSB steganography. In every day life, JPEG images are the most widely used images. For instance, various kinds of digital cameras usually export JPEG images under the default setup, most image editing software such as Adobe Photoshop supports JPEG compression, and JPEG images are the most widely used images in network transmission. Thus, unprofessional steganographers are likely to take a JPEG image as the input, then use popular LSB steganography software downloaded from the Internet to decompress it and embed the secret message into the spatial image obtained, which is referred to as a decompressed JPEG image. Moreover, some efficient detection methods [12–15], aiming at steganographic algorithms with a decompressed JPEG image used as the cover image, have been proposed in state-of-the-art papers. Hence, researching the theory and methods of extracting a hidden message embedded in a decompressed JPEG image has theoretical value and practical significance.

In this paper, we study a recovery method for a stego key for LSB steganography with a decompressed JPEG image as the cover image. First, the recovery of the stego key is translated into a problem of analyzing a sequential cipher with errors. A general framework for recovering a stego key based on estimating modification positions is proposed. Moreover, we discuss the possibility of recovering the original cover image (the decompressed JPEG image) using recompression. Finally, an algorithm for recovering the stego key of LSB steganography with a decompressed JPEG image as the cover image is proposed. Relative experimental results for an attack against the steganographic software Hide and Seek 4.1 and its variant show that, when a decompressed JPEG image is used as the cover image, for different quality factors, the sample size for successfully recovering the stego key using the proposed method is much less than the contrasting methods, and the scope of the proposed method for different embedding rates is wider than the contrasting methods. Moreover, both the theoretical analysis and the experimental results for the searching time for the stego key show that the computational complexity of the proposed method is lower than that of the contrasting methods.

The remainder of this paper is organized as follows. In Section 2, the random LSB steganography model is described. In the next section, the method for recovering the stego key based on estimating modification positions is proposed. In Section 4, we discuss the possibility of recovering the original cover image (the decompressed JPEG image) using recompression. The algorithm for recovering the stego key of LSB steganography with a decompressed JPEG image as the carrier is proposed and the experimental results are shown in Section 5. This paper is concluded in Section 6.

2 Random LSB steganography model

In this paper, k, r and L denote the stego key selected from the key space K, the embedding rate and the length of the embedded message, respectively. The embedding process for random LSB steganographic algorithms is as follows.

Firstly, a random sequence $\boldsymbol{y}_k^L = \{y_1, y_2, \ldots, y_L\}$ is generated by a pseudorandom number generator (PRNG) G with the stego key k, and y_i $(1 \leqslant i \leqslant L)$ needed to satisfy $1 \leqslant y_i \leqslant d$, where $d = \lfloor 1/r \rfloor$ is the maximum random interval, to ensure that the following embedding positions x_i $(1 \leqslant i \leqslant L)$ do not exceed the available positions in the cover image. Then we generate the random embedding positions for carrying the payload, $\boldsymbol{x}_k^L = \{x_1, x_2, \ldots, x_L\}$, where

$$x_1 = y_1; \quad x_i = x_{i-1} + y_i, \quad 2 \leqslant i \leqslant L. \tag{1}$$

Finally, the message sequence $\boldsymbol{m}^L = \{m_1, m_2, \ldots, m_L\}$ is embedded in the LSBs of pixels in the positions x_i $(1 \leqslant i \leqslant L)$, and the stego image S is obtained. If the LSB of the pixel in an embedding position matches the corresponding message bit, no further action is needed. Otherwise, we apply LSB-R or LSB-M to modify the pixel and embed the message bit. The receiver who has the stego key k can easily get $\boldsymbol{m}^L$ from S.

We assume the output sequence $Y_1, Y_2, \ldots, Y_L$ of the PRNG G are independent identically distributed random variables, which are integers in $[a+1, a+d]$, where a and d are also both integers, satisfying

$$P\{Y_i = a + i\} = p_i, \quad 1 \leqslant i \leqslant d, \quad \sum_{i=1}^{d} p_i = 1. \tag{2}$$

We only need consider the situation where $a = 0$ (if $a \neq 0$, we can let $Y_i' = Y_i - a$ and consider Y_i'). Note that the output of a PRNG generally follows a uniform distribution. Otherwise, from the perspective of cryptography, it is weak, since a cryptanalyst can recover the seed or construct the equivalent generator under this circumstance. Hence, we only pay attention to the situation that $p_i = 1/d$ $(1 \leqslant i \leqslant d)$, where d is the maximum random interval.

In this paper, we assume that the attacker already knows the details of the steganographic algorithm, but does not know the stego key k used by the sender. Our purpose is to study how to recover the stego key with only the stego image and extract the message under this circumstance.

3 Method for recovering the stego key based on estimating modification positions

When the attacker gets the stego image, they can obtain an estimate of the cover image using methods such as image restoration. Compared with the original carrier, this estimated carrier may contain some errors. Then the attacker can obtain an estimated sequence of modification positions in the carrier by comparing the stego image with the estimated carrier. Obviously, this sequence is a sample of the output sequence of a PRNG with some errors. This is because the LSB of a cover pixel that is equal to the corresponding message bit, is not modified, and the estimation accuracy cannot achieve 100%. The purpose of the extracting attack is to recover the stego key using the estimated sequence and extracting the message. This is equivalent to obtaining a sample sequence of the output of the PRNG with some errors, and there is a need to recover the seed of the PRNG under this circumstance, which is a cryptanalysis problem. The attack methods in cryptanalysis can be used to analyze a specific PRNG, and attacks against a PRNG are not considered in this paper.

3.1 Collision attack based on estimating modification positions

The essential question in recovering the stego key is to distinguish the true key from the false key. If we want to identify the stego key after obtaining a proportion of the correct embedding positions, we can use a collision attack of a sequence cipher. That is, we generate the corresponding random embedding position $\{x_i\}$ by exhausting all the keys k. When the number of positions where the sequence generated by a key matches the observed modification positions, is a maximum, the key can be considered as the true key. Otherwise, it is considered as a false key and is discarded. This kind of collision attack is called a collision attack based on estimating modification positions. Now we analyze the feasibility of this attack and estimate the data size for success.

Under this model of random LSB steganography, we assume $Y_1, Y_2, \ldots, Y_j, \ldots$ is an independent sequence of random variables with the same distribution, where $P\,(Y_1 = i) = 1/d$ $(1 \leqslant i \leqslant d)$. The sequence of random variables $\{X_j, j \geqslant 1\}$ is defined as

$$X_1 = Y_1; \quad X_j = X_{j-1} + Y_j, \quad j \geqslant 2. \tag{3}$$

Obviously, $\{X_j, j \geqslant 1\}$ is a Markov chain and we have the following lemma [7].

Lemma 1. For any given n $(n \geqslant 1)$ and positive integers $x_1 < x_2 < \cdots < x_n$, we have the following inequality:

$$\text{Prob}(\exists \text{ positive integers, } i_1 < i_2 < \cdots < i_n, \text{ s.t. } X_{i_1} = x_1,\, X_{i_2} = x_2, \ldots, X_{i_n} = x_n) \leqslant \left(\frac{(d+1)^{d-1}}{d^d}\right)^n. \tag{4}$$

Lemma 1 illustrates that, with the given key space, when there are enough observed modification positions of the carrier, we can identify the stego key. The required number of modification positions for identifying the stego key is given in the following theorem when the error is P_e and the embedding rate is r.

Theorem 1. Assume that the length of the stego key is l bits, i.e., the size of the key space is 2^l, and the collision attack needs M_0 modification positions to make the expectation less than 1 and further identify the stego key. M_0 is the minimum of the values M satisfying the following inequality:

$$2^l \sum_{i=M-[MP_e]}^{M} C_M^i \left(\frac{(d+1)^{d-1}}{d^d}\right)^i \leqslant 1, \tag{5}$$

where C_M^i denotes that the combination of i things taken M at a time, and $[x]$ denotes rounding of x.

Proof. Assume that the attacker obtains M modification positions. For any stego key, P_N denotes the probability that the sequence generated by the key matches at least N modification positions. According to Lemma 1, we have

$$P_N \leqslant \sum_{i=N}^{M} C_M^i \left(\frac{(d+1)^{d-1}}{d^d}\right)^i. \tag{6}$$

When the key k is the true key, the sequences generated by it can match $M-[MP_e]$ modification positions. In a collision attack, any false key that produces a sequence that matches more than $M-[MP_e]$ modification positions will mistakenly be taken as a true key (a false-alarm key). $P(M,d,P_e)$ denotes the probability of getting a false-alarm key. Putting $N = M-[MP_e]$ into (6), $P(M,d,P_e)$ satisfies the following inequality:

$$P(M,d,P_e) \leqslant \sum_{i=M-[MP_e]}^{M} C_M^i \left(\frac{(d+1)^{d-1}}{d^d}\right)^i. \tag{7}$$

If the stego key length is l bits, the size of the key space is 2^l, where there are $2^l - 1$ false keys obviously. Therefore, the mathematical expectation of the false-alarm keys obtained from a collision attack is $(2^l-1)P(M,d,P_e)$, and with (7), we get

$$(2^l-1) \times P(M,d,P_e) < 2^l \sum_{i=M-[MP_e]}^{M} C_M^i \left(\frac{(d+1)^{d-1}}{d^d}\right)^i. \tag{8}$$

Hence, the number of modification positions, M, satisfies the following inequality

$$2^l \sum_{i=M-[MP_e]}^{M} C_M^i \left(\frac{(d+1)^{d-1}}{d^d}\right)^i \leqslant 1. \tag{9}$$

Then Eq. (5) can ensure the expectation of the number of false keys is less than 1 and we can identify the true key.

However, it is not easy to calculate the minimum M satisfying (5) directly, so we can search for M_0 by exhausting M ($M \leqslant W$) in order ranging from small to large, where W is the total number of observed modification positions. Now we analyze the required data size M_0 for the attack. As is shown in [1], an attack is infeasible when the key space is too large or the embedding rate is too small or large. Obviously, large errors P_e also make the attack infeasible. Thus is also illustrated in Table 1, which shows the minimum integer M_0 satisfying (5) with different embedding rates (i.e., different d) and different errors P_e when $l = 16$ and 24. The symbol $*$ in Table 1 means that Eq. (5) is false for all positive integers.

3.2 Algorithm for recovering a stego key

Assume that the estimated sequence of modification positions, $\boldsymbol{x}^W$, has already been obtained using some method, and the error P_e has been calculated.

Step 1: Determine the data size M. Substitute the length of the stego key, l, the interval, d, and the error, P_e, into 5. If M_0 can be obtained by searching in order ranging from small positive integers to large positive integers, let $M = M_0$; otherwise, let $M = W$.

Table 1 Minimum data size M_0

P_e (%)	$l=16$					$l=24$				
	$d=2$	$d=4$	$d=6$	$d=8$	$d=9$	$d=2$	$d=4$	$d=6$	$d=8$	$d=9$
0	39	16	11	9	9	58	24	17	14	13
3	82	21	15	12	11	117	29	21	17	16
5	135	26	15	12	11	213	35	24	17	16
7	614	31	18	12	11	1001	39	25	20	19
10	$*$	35	19	15	14	$*$	50	28	20	19
15	$*$	57	25	18	16	$*$	84	37	26	24
20	$*$	125	35	23	19	$*$	204	49	30	29
25	$*$	$*$	51	30	23	$*$	$*$	72	40	35
30	$*$	$*$	93	41	32	$*$	$*$	143	59	46
35	$*$	$*$	534	57	40	$*$	$*$	860	88	65
40	$*$	$*$	$*$	110	65	$*$	$*$	$*$	182	105
45	$*$	$*$	$*$	$*$	160	$*$	$*$	$*$	$*$	255
49	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$

Step 2: The sample $\boldsymbol{x}^M=\{x_1,x_2,\ldots,x_M\}$ is constructed from the first M elements of the sequence of $\boldsymbol{x}^W$. Let

$$Z=\left\lceil\frac{2x_M}{d+1}\right\rceil$$

and exhaust the key space to distinguish the true key from the false key. (Note that, in this way, we can ensure the PRNG covers the range near the last pixel x_M, after it runs Z times.) k denotes the test key, N_k denotes the number of modification positions corresponding to k, *key* denotes the suspicious key, and N_{key} denotes the number of modification positions corresponding to *key*, where the initial value of N_{key} is 0. The specific process is as follows.

(1) Select the test key k from the key space in order. Generate the first Z random positions $\boldsymbol{x}_Z^k=\{x_1^k,x_2^k,\ldots,x_Z^k\}$ using the PRNG with k as the key.

(2) Calculate N_k, which is the number of positions where the sequence $\boldsymbol{x}_Z^k$ matches $\boldsymbol{x}^M$.

(3) If $N_k\geqslant N_{\text{key}}$, then let key $=k$ and $N_{\text{key}}=N_k$.

(4) If the key space is exhausted, go to step 3. Otherwise, goto step 1.

Step 3: If the number of keys where the number of modification positions is equal to the maximum N_{key}, is equal to 1, then take the key as the true key; otherwise, the attack fails.

4 Probability of successfully recovering the original cover image using recompression

First of all, note that, if not specified, the carriers referred to in this article are decompressed JPEG images, the block size is 8×8 and the steganographic algorithms used in this paper are LSB-R or LSB-M, also known as LSB steganography. Almost all the operations are based on an 8×8 block as a unit.

4.1 Notation for JPEG decompression and unsaturated stego blocks

The symbol J denotes a JPEG image, and its quality factor is q. The quantization matrix corresponding to q is Q, whose coefficients are denoted as $Q(i)$ $(0\leqslant i\leqslant 63)$. Let D be a quantized discrete cosine transform (DCT) coefficient matrix of a block in J. A gray value matrix B is obtained after D is decompressed. Then we have

$$B=\text{int}_8\left(B'\right),\quad B'=\text{Dct}^{-1}\left(QD\right),\tag{10}$$

where QD denotes the multiplication of the elements in the corresponding positions of two 8×8 matrices Q and D, and $\text{Dct}^{-1}(\cdot)$ denotes the inverse DCT. In addition, $\text{int}_8(\cdot)$ denotes the operation rounding a value to an 8-bit integer, i.e., values less than 0 become 0, values greater than 255 become 255, and values between 0 and 255 are rounded to the nearest integer. Note that the first two are overflow operations, and the last one is a rounding operation. If there are no overflow operations but only rounding operations when $\mathrm{int}_8(\cdot)$ operates on the block B', the block B is called an **unsaturated block**, otherwise it is called a **saturated block**.

Let R be the rounding error caused by the operation $\mathrm{int}_8(\cdot)$, and then we have

$$R = B - B' = \mathrm{int}_8(B') - B'. \tag{11}$$

Definition 1. If a cover block B is an unsaturated block, its corresponding stego block is called an **unsaturated stego block**, otherwise it is called a **saturated stego block**.

It is indicated in [16] that if there is no value of 0 or 255 in an 8×8 cover block, the block must be an unsaturated block. It can be seen that the number of unsaturated blocks in an image will be very large and increase with an increment of the quality factor [16]. Since LSB steganography may change 0 into 1, or 254 into 255, some saturated stego blocks may be determined as unsaturated stego blocks. We present a **determination method** that can exclude all the saturated stego blocks: if an 8×8 stego block S does not contain any pixels with values of 0, 1, 254 or 255, S must be an unsaturated stego block. Note that the determination method will exclude all the saturated stego blocks but a proportion of unsaturated stego blocks may also be excluded. However, as is illustrated in [17], the number of unsaturated stego blocks excluded by the determination method is very small and will not affect the validity of the proposed method.

4.2 Notation for recompression

Definition 2. Let I be J's decompressed image and I_S be the stego image corresponding to I. I_S is compressed with the quantization matrix Q and then decompressed into the spatial image I'. This procedure is called **JPEG recompression**, referred to as **recompression**.

Here we assume that we have the original quantization matrix Q. This is because, for the decompressed JPEG image, using the method in [13], we can estimate Q very accurately from the stego image I_S. Below, we describe the recompression with an 8×8 block as the unit.

Let S denote the stego block of B, and C_r denote the additive noise matrix introduced by embedding data in B with the embedding rate r. Then, combining with (11), we have

$$S = B + C_r = B' + R + C_r. \tag{12}$$

S is recompressed as follows.

(1) DCT. According to the additive property of DCT and (10), we have

$$\mathrm{Dct}(S) = \mathrm{Dct}(B') + \mathrm{Dct}(R) + \mathrm{Dct}(C_r) = QD + \varepsilon + X_r, \tag{13}$$

where $\mathrm{Dct}(\cdot)$ denotes the DCT of an 8×8 image block, $\varepsilon = \mathrm{Dct}(R)$ denotes the frequency domain noise matrix caused by rounding errors in the spatial domain, and $X_r = \mathrm{Dct}(C_r)$ denotes the frequency domain noise matrix caused by the embedding process of spatial image steganography.

(2) Quantization. Let D_S be the DCT coefficient matrix after quantizing $\mathrm{Dct}(S)$. According to (13), we have

$$D_S(i) = \mathrm{round}\left(\frac{Q(i)\,D(i) + \varepsilon(i) + X_r(i)}{Q(i)}\right) = D(i) + \mathrm{round}\left(\frac{\varepsilon(i) + X_r(i)}{Q(i)}\right), \quad 0 \leqslant i \leqslant 63, \tag{14}$$

where $\mathrm{round}(\cdot)$ denotes the rounding operation.

(3) Saving the image data in an uncompressed form. Let B_S be the 8×8 spatial pixel block after recompressing S. Then we have

$$B_S = \mathrm{int}_8\left(\mathrm{Dct}^{-1}(QD_S)\right). \tag{15}$$

4.3 Probability of successfully recovering the cover blocks

We regain the original cover image by recompressing the stego image. When the saturated blocks are rounded by $\mathrm{int}_8(\cdot)$, truncation errors will appear, which are always large, such as -30 and 25 [16]. Consequently, the frequency domain noise introduced by the truncation errors will also be very large. When the frequency domain noise in a certain position exceeds half of the quantization step, the quantization of the recompression cannot eliminate this noise, i.e., the cover block cannot be recovered using recompression. Therefore, we obtain the corresponding cover block by recompressing the unsaturated block. Below, Theorem 2 from [17] gives the probability of successfully recovering a cover block by recompressing the unsaturated blocks.

Theorem 2. Let D be an 8×8 quantized DCT coefficient matrix of J, B be the gray value matrix after D is decompressed, and S be the stego block corresponding to B. In addition, after S is recompressed, D_S and B_S denote the DCT coefficient block and spatial pixel block, respectively. The sufficient condition for $B_S = B$ is

$$-\frac{1}{2} \leqslant \frac{Y_r(i)}{Q(i)} < \frac{1}{2}, \quad 0 \leqslant i \leqslant 63, \tag{16}$$

where $Y_r(i)$ is the sum of the frequency domain noise values caused by the rounding error and the embedding process, i.e.,

$$Y_r(i) = \varepsilon(i) + X_r(i), \quad 0 \leqslant i \leqslant 63. \tag{17}$$

Theorem 2 illustrates that, as long as each $Y_r(i)$ is between $-Q(i)/2$ and $Q(i)/2$ $(0 \leqslant i \leqslant 63)$, the original cover block B can be obtained by recompressing S, which is the stego block of the unsaturated block B. At the moment, if we know the probability distribution of $Y_r(i)$ and its probability density function (PDF), we can calculate the probability that $Y_r(i)$ is between $-Q(i)/2$ and $Q(i)/2$. Theorem 3 from [17] gives the probability distribution of $Y_r(i)$.

Theorem 3. Some binary random data are embedded in the decompressed JPEG image I using LS-B steganography with embedding rate r. If the embedding is independent of the cover image I, in terms of each unsaturated stego block S, then $Y_r(i)$ approximately follows a Gaussian distribution with expectation 0 and variance $1/12 + r/2$, i.e.,

$$Y_r(i) = \varepsilon(i) + X_r(i) \sim N(0, 1/12 + r/2), \quad 0 \leqslant i \leqslant 63. \tag{18}$$

Let $P(r,q)$ denote the probability of successfully recovering the cover block B by recompressing S. First, we have

$$P(r,q) = P\left\{-\frac{Q(i)}{2} \leqslant Y(i) < \frac{Q(i)}{2},\ 0 \leqslant i \leqslant 63\right\}. \tag{19}$$

According to Theorem 3, each $Y(i)$ follows the Gaussian distribution $N(0, 1/12 + r/2)$, and then we have

$$P\left\{-\frac{Q(i)}{2} \leqslant Y(i) < \frac{Q(i)}{2}\right\} = \int_{-Q(i)/2}^{Q(i)/2} f_r(x)\,dx, \quad 0 \leqslant i \leqslant 63, \tag{20}$$

where $f_r(x)$ is the PDF of $Y_r(i)$.

Since the Karhunen-Loeve transform is a decorrelation transformation and DCT is a good approximation of the Karhunen-Loeve transform, DCT has a certain decorrelation property. Therefore, there is a slight correlation between two coefficients in a DCT coefficient block. It has been pointed out [5] that individual DCT coefficients have few inter-block correlations, i.e., apparently, the two coefficients are not independent. However, as is shown in [17], we find that we can use an approximate value to estimate $P(r,q)$:

$$P(r,q) \approx \prod_{i=0}^{63} P\left\{-\frac{Q(i)}{2} \leqslant Y_r(i) < \frac{Q(i)}{2}\right\} = \prod_{i=0}^{63} \int_{-Q(i)/2}^{Q(i)/2} f_r(x)\,dx, \tag{21}$$

where $f_r(x)$ is the PDF of $Y_r(i)$.

Table 2 Fitting results for LSB-M

	$q=65$		$q=75$			$q=85$		
	$k=1$	$k=2$	$k=1$	$k=2$	$k=3$	$k=1$	$k=2$	$k=3$
w_3	0	0	0	0	1.4933	0	0	0.0014
w_2	0	25.6235	0	1.5651	−4.4403	0	−0.3916	−0.3944
w_1	0.8741	−50.3605	0.2885	−2.8164	4.4014	0.2565	0.7836	0.7854
w_0	0.1258	25.7369	0.7114	2.2513	−1.4534	0.7617	0.6030	0.6026
MAD	**0.0000208**	0.0000613	0.0001281	**0.0000245**	0.000685	0.00158	**0.00039**	0.000521

4.4 Probability of successfully recovering the cover pixels

Eq. (21) gives the approximate probability $P(r,q)$ of successfully recovering the cover blocks of the decompressed JPEG image. Let $E(r,q)$ be the probability of successfully recovering the cover pixels. Next, as is discussed in [17], we use $P(r,q)$ to estimate $E(r,q)$.

Obviously, there exists a strong positive correlation between $E(r,q)$ and $P(r,q)$. Hence, we use the polynomial fitting method to build the mathematical relation for $E(r,q)$ and $P(r,q)$, and suppose

$$E(r,q) = w_k \times (P(r,q))^k + w_{k-1} \times (P(r,q))^{k-1} + \cdots + w_1 \times P(r,q) + w_0. \tag{22}$$

We need to determine the degree k and the coefficients w_j $(j = 0,1,\ldots,k)$ of the polynomial and employ the polyfit function in the MATLAB toolbox to accomplish the fitting. First, we convert all the images in the BOSSbase 0.92 database[2)] into decompressed JPEG images with quality factors q $(q = 65,75,85)$, where half of the decompressed JPEG images are randomly selected for the training set, which is used to determine the degree k and coefficients of the fitting polynomial, and the other half constitute the test set. Then, with embedding rates r $(r = 0,0.05,0.1,0.15,\ldots,0.95,1.0)$, binary random data are embedded in the images of the training set using LSB-M. Next, we recompress all the unsaturated stego blocks, count the number of successfully recovered cover pixels, and then obtain the observed probability $E_{\text{om}}(r,q)$. We compute $P(r,q)$ using (21). For the given degrees k $(k = 1,2,3)$, the polynomial fitting proceeds over $E_{\text{om}}(r,q)$ and $P(r,q)$, and then we can obtain the coefficients of the fitting polynomials, as illustrated in Table 2. We substitute the obtained degree and coefficients into (22) and work out the mean absolute difference (MAD) between $E_{\text{om}}(r,q)$ and $E(r,q)$. Then we choose the degree and coefficients corresponding to the smallest MAD, and substitute them into (22) to estimate $E(r,q)$. Finally, we execute the test over the test set using the estimated $E(r,q)$. The fitting results can be seen in Figure 1.

It can be seen from Table 2 that, when the quality factor q is 65, the first-order polynomial fitting corresponds to the smallest MAD, i.e., $E(r,q) = 0.8741 \times P(r,q) + 0.1258$. When the quality factor q is 75, the second-order polynomial fitting corresponds to the smallest MAD, i.e., $E(r,q) = 1.5651 \times (P(r,q))^2 - 2.8164 \times P(r,q) + 2.2513$. Finally, when the quality factor q is 85, the second-order polynomial fitting corresponds to the smallest MAD, i.e., $E(r,q) = -0.3916 \times (P(r,q))^2 + 0.7836 \times P(r,q) + 0.6030$. Since the MAD of estimating $E(r,q)$ using polynomial fitting is very small, the fitting is favorable and reliable. As well, Figure 1 intuitively reflects the validity of polynomial fitting, which agrees with the above theoretical analysis.

We conduct a similar experiment using LSB-R and acquire the degrees and coefficients of the fitting polynomials, as shown in Table 3. Finally, for LSB-M and LSB-R, we run the test over the test set and calculate the MADs between $E_{\text{om}}(r,q)$ and $E(r,q)$, which are shown in Table 4. It can be observed from the above experimental results that the polynomial fitting can precisely estimate the probability $E(r,q)$, which obviously satisfies $E(r,q) > 0.5$, as shown in Figure 1.

In terms of LSB steganography, $E(r,q)$ and $P(r,q)$ are related to r and q, except for the cover image, so we can implement the polynomial fitting method, for the quality factors q $(q = 65,66,\ldots,90)$, and select the fitting polynomial corresponding to the smallest MAD for each quality factor. Furthermore,

2) http://boss.gipsa-lab.grenoble-inp.fr.

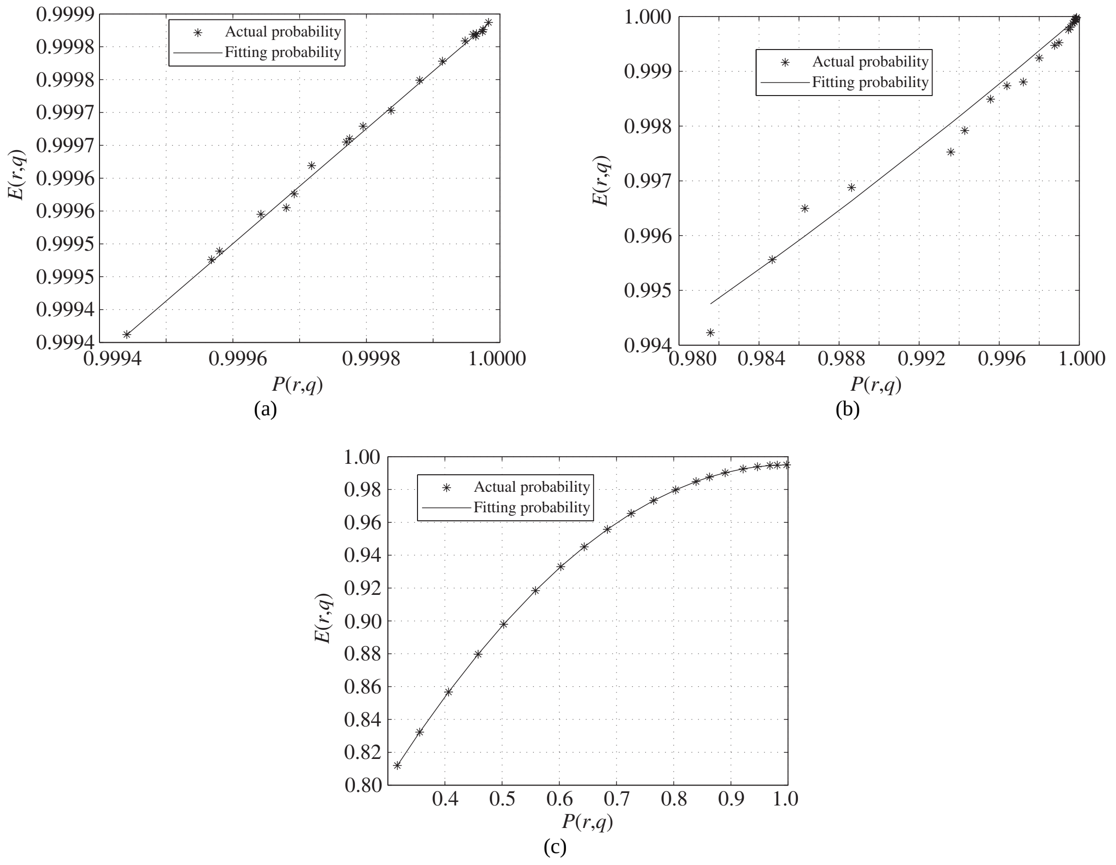

Figure 1 Approximate and experimental probabilities of successfully recovering the cover blocks. (a) $q = 65$; (b) $q = 75$; (c) $q = 85$.

Table 3 Fitting results for LSB-R

	$q = 65$		$q = 75$			$q = 85$		
	$k = 1$	$k = 2$	$k = 1$	$k = 2$	$k = 3$	$k = 1$	$k = 2$	$k = 3$
w_3	0	0	0	0	1.4944	0	0	0.0014
w_2	0	25.6242	0	1.5635	−4.4434	0	−0.3954	−0.3961
w_1	0.8750	−50.3611	0.2886	−2.8126	4.4075	0.2609	0.7863	0.7820
w_0	0.1249	25.7365	0.7112	2.2561	−1.4580	0.7688	0.6061	0.6032
MAD	**0.0000202**	0.0000599	0.0001265	**0.0000242**	0.000669	0.001568	**0.000386**	0.000501

Table 4 MAD in estimating $E\,(r, q)$

LSB-M			LSB-R		
$q = 65$	$q = 75$	$q = 85$	$q = 65$	$q = 75$	$q = 85$
0.0000516	0.0001208	0.000645	0.0000501	0.0001189	0.000623

we retain the fitting results and only need to query the corresponding fitting polynomial according to the quality factor, to estimate $E\,(r, q)$ using $P\,(r, q)$.

5 Proposed method for recovering the stego key of LSB steganography with a decompressed JPEG image

Firstly, the recompression technique is used to recover the carrier and an estimated sequence of modification positions is obtained through comparing the stego image with the recovered carrier. A method of recovering a stego key with a decompressed JPEG image as the carrier is proposed using the relevant theories described in Sections 3 and 4.

The estimated sequence of modification positions is a sample of the output of a PRNG with some errors. Below, we calculate the error P_e of the estimated sequence.

5.1 Estimation of the error P_e

Let W be the length of the estimated sequence. Since the possibility of successfully recovering each cover pixel corresponding to this sequence is $E(r,q)$, the number of unsuccessfully recovered cover pixels is about $W \times (1 - E(r,q))$. An error in recovering the cover pixels leads to an error in estimating the modification positions, and the number of errors in estimating the modification positions is about $W \times (1 - E(r,q))$. Therefore, the error P_e is computed with

$$P_e \approx \frac{W \times (1 - E(r,q))}{W} = 1 - E(r,q). \tag{23}$$

5.2 Algorithm for recovering the stego key of LSB steganography with a decompressed JPEG image

Below, we describe in detail the steps of the algorithm for recovering the stego key. Assume that a stego image I_S containing a secret message, which is embedded using spatial random LSB steganography in a decompressed JPEG image, has been obtained. The quality factor q and the embedding rate r of the stego image can be accurately estimated by the method in [13]. Then $P(r,q)$ can be computed according to (21), $E(r,q)$ can be obtained using the polynomial fitting method of Subsection 4.4, and the error P_e can be obtained from (23). Let W be the total number of observed modification positions. We use the following steps to distinguish between the keys.

Step 1: Divide I_S into some non-overlapping 8×8 blocks and remove the blocks whose length or width is less than 8.

Step 2: All the saturated blocks are excluded using the determination method in Subsection 4.1, and all the unsaturated blocks are obtained.

Step 3: Each unsaturated stego block is recompressed in sequence and the corresponding cover block is obtained. Then an estimate of the modification positions corresponding to this block is obtained. The process is described in detail as follows.

(1) Fetch an unsaturated stego block S in order.

(2) Recompress S according to the quality factor q and obtain the recovered cover block B_S.

(3) Observe the difference between S and B_S. If the difference for a position is 1 or -1, this position is determined to be a modification position. Finally, the estimated sequence of the modification positions in this image, $\boldsymbol{x}^W = \{x_1, x_2, \ldots, x_W\}$, is obtained.

Step 4: Execute the algorithm for recovering the stego key based on estimating the modification positions presented in Section 3.

5.3 Extracting attack against Hide and Seek 4.1

Hide and Seek is typical steganographic program based on LSB-R with a GIF image as the carrier. It uses a PRNG controlled by a key to select the payload-carrying pixels. We show the experimental results of attacking Hide and Seek 4.1[3)] using the proposed method. In Hide and Seek 4.1, only a GIF image with 256 colors of size 320×480 can be used as a cover image. The `random(num)` function in Borland C++ 3.1 is used to generate random numbers and a 16-bit seed is chosen as the initial state. The dynamically changing parameter `num` is used to control the largest offset step size. Moreover, the parameter `num` is related to the length of the remaining message and the number of remaining cover pixels, to ensure the message is uniformly spread in the cover image. Therefore, the key used for choosing the payload-carrying pixels in Hide and Seek 4.1 is composed of the 16-bit seed and the message length.

3) http://www.jjtc.com/security/stegtools.htm.

Table 5 Experimental results when $q = 65$

Images	r (%)	Proposed method				Method in [6]			Method in [8]		Method in [9]	
		$E(r,q)$ (%)	M	LSB-R	LSB-M	M	LSB-R	LSB-M	M	LSB-R	M	LSB-R
Lena	0.03	99.9997	8	×	×	−	×	×	−	×	79	×
Baboon	0.05	99.9996	8	√	√	−	×	×	−	×	132	×
Peppers	1.5	99.9996	8	√	√	−	×	×	729	√	3933	√
Airplane	5.3	99.9995	11	√	√	−	√	×	769	√	13894	√
Goldhill	7.4	99.9989	18	√	√	9900	√	√	823	√	19399	√
Lena	50	99.9992	37	√	√	54400	√	√	3025	√	131072	√
Baboon	66	99.9977	68	√	√	89500	√	√	6561	√	173020	√
Peppers	75	99.9909	101	√	√	−	√	√	12100	√	196608	√
Airplane	90	99.9569	315	√	√	−	√	√	75625	√	235930	√
Goldhill	94.6	99.9533	1134	√	√	−	×	×	229156	√	247990	√
Lena	99	99.9560	∗	√	√	−	×	×	−	×	−	√
Baboon	99.3	99.9552	∗	√	√	−	×	×	−	×	−	×
Peppers	99.5	99.9600	∗	×	×	−	×	×	−	×	−	×

Since $(320 \times 480)/8 = 19200$, the maximum message length is limited to 19000 bytes in the experiments. Note that the message length is in bytes in the parameter setting for the PRNG.

We use the encryption scheme IDEA to encrypt the message length, the seed for the PRNG and the version information, and generate 64-bit ciphertexts, which are embedded into the LSBs of the first 64 cover pixels. Then, starting from the 65th pixel, we embed the message into random positions controlled by the message length and the seed. The key for IDEA consists of the passwords with a maximum size of 8 bytes, so the key length of Hide and Seek 4.1 is 64 bits, i.e., the total number of keys is 2^{64}. A receiver who knows the passwords can decrypt the message length and the seed, and then extract the message. It is difficult to recover the 64-bit IDEA key, but we can skip the first 64 pixels and use the remaining pixels to recover the 16-bit seed and the message length directly using the proposed method. Since the method of [15] can estimate the embedding rate with an error ± 0.005 without knowing the actual embedding rate, we only need to test about $19000 \times 0.01 = 190$ possible lengths when searching for the key. In other words, the key length is only about $l = 16 + \log_2 190 \approx 23.6$ bits.

First, using the quality factors 65, 75 and 85, we compress the test images, which are five uncompressed gray BMP images of size 320×480: Lena, Baboon, Peppers, Airplane and Goldhill. And we decompress the JPEG images obtained using the quality factors to generate the spatial GIF images as the cover images. The reason for selecting the quality factors 65, 75 and 85 is that, when $q \leqslant 75$, $P(r,q) \geqslant 99.5\%$ with any embedding rate, i.e., $P_e \leqslant 0.25\%$, and when $q > 80$, $P(r,q)$ decreases quickly. Then we use Hide and Seek 4.1 to embed the message and implement the extracting attack without the embedding rate using the methods in [6], [8] and [9] and the proposed method. Since all the methods for recovering the stego key, for spatial LSB steganography using a stego key, are given in the references [6], [8] and [9], these methods are selected for comparison. Moreover, we replace LSB-R in Hide and Seek 4.1 by LSB-M and repeat the previous attack. All the experimental results are shown in Tables 5–7, where the symbol $*$ denotes that the sample size M is the total number of observed modification positions, the symbol $-$ denotes that the sample size is the total embedding length, $\surd$ denotes that the attack succeeds, and $\times$ denotes that the attack fails.

The experimental results in Tables 5–7 show: (1) both the method in [6] and the proposed method are suitable for both LSB-R and LSB-M, while the methods in [8], and [9] are only suitable for LSB-R. (2) The sample size for successfully recovering the stego key by the proposed method, M, is much less than the methods in [6], [8] and [9]. (3) The range of embedding rates for successfully recovering the stego key by the proposed method is wider than the methods in [6], [8] and [9]. With $q = 65$ or 75, even when the embedding rate is as small as 0.05% or as large as 99.3%, the proposed method can succeed in attacking both LSB-R and LSB-M. With $q = 85$, when the embedding rate is as small as 0.05%, the

Table 6 Experimental results when $q = 75$

Images	r (%)	Proposed method				Method in [6]			Method in [8]		Method in [9]	
		$E\left(r,q\right)$ (%)	M	LSB-R	LSB-M	M	LSB-R	LSB-M	M	LSB-R	M	LSB-R
Lena	0.03	99.9996	8	×	×	–	×	×	–	×	79	×
Baboon	0.05	99.9996	8	√	√	–	×	×	–	×	132	×
Peppers	1.5	99.9988	8	√	√	–	×	×	729	√	3933	√
Airplane	5.6	99.9930	11	√	√	–	√	×	784	√	14680	√
Goldhill	8.7	99.9524	18	√	√	11400	√	√	841	√	22544	√
Lena	50	99.9139	39	√	√	55800	√	√	3025	√	131072	√
Baboon	66	99.8954	71	√	√	91600	√	√	6561	√	173020	√
Peppers	75	99.8699	124	√	√	–	√	√	12100	√	196608	√
Airplane	90	99.8811	629	√	√	–	√	√	75625	√	235930	√
Goldhill	94.1	99.7955	1227	√	√	–	×	×	218089	√	246680	√
Lena	99	99.5877	∗	√	√	–	×	×	–	×	–	√
Baboon	99.3	99.6032	∗	√	√	–	×	×	–	×	–	×
Peppers	99.5	99.4978	∗	×	×	–	×	×	–	×	–	×

Table 7 Experimental results when $q = 85$

Images	r (%)	Proposed method				Method in [6]			Method in [8]		Method in [9]	
		$E\left(r,q\right)$ (%)	M	LSB-R	LSB-M	M	LSB-R	LSB-M	M	LSB-R	M	LSB-R
Lena	0.03	99.9544	8	×	×	–	×	×	–	×	79	×
Baboon	0.05	99.9210	8	√	√	–	×	×	–	×	132	×
Peppers	1.9	99.9054	8	√	√	–	×	×	738	√	4980	√
Airplane	6.8	99.9097	11	√	√	–	√	×	802	√	17826	√
Goldhill	25	98.7896	24	√	√	30700	√	√	1296	√	65536	√
Lena	40	98.2485	273	√	√	43100	√	√	1936	√	104860	√
Baboon	45	97.3931	405	√	√	48300	√	√	2304	√	117960	√
Peppers	50	97.3637	1180	√	√	56600	√	√	3025	√	131072	√
Airplane	55	95.9942	1899	√	√	70500	√	√	3364	√	144180	√
Goldhill	66	94.0175	2334	√	√	93700	√	√	6561	√	173020	√
Lena	83.6	93.8630	∗	√	√	–	√	√	24649	√	219150	√
Baboon	84	93.3359	∗	√	√	–	√	√	25921	√	220200	√
Peppers	90	92.3880	∗	×	×	–	×	×	–	×	–	×

proposed method can succeed, but when the embedding rate exceeds 90%, the proposed method fails because $E\left(r,q\right)$ is too low and P_e too high.

Below, we discuss the computational complexity of the methods in [6], [8] and [9], and the proposed method. The basis of the four methods is that the key is identified according to the difference between the true embedding path and the false embedding path and the key space must be exhausted. Therefore, given the key space, the computational complexity of the attack is determined by the sample size of the test, i.e., a larger sample size leads to higher computational complexity. Assume that the computation time of the attack with the sample size M and the method F is Time$(F\left(M\right))$.

Let F_1, F_2, F_3 and F_4 be the proposed method, the method in [8], the method in [6] and the method in [9], respectively. Let M_1, M_2, M_3 and M_4 be the sample sizes for F_1, F_2, F_3 and F_4, respectively, for the same stego image. The computation times are $\mathrm{Time}\left(F_1\left(M_1\right)\right)$, $\mathrm{Time}\left(F_2\left(M_2\right)\right)$, $\mathrm{Time}\left(F_3\left(M_3\right)\right)$ and $\mathrm{Time}\left(F_4\left(M_4\right)\right)$. On one hand, for the same sample size M, the proposed method needs only to compare the chosen sample with the estimated sample once, so this needs M comparisons. The method in [8] needs to calculate once for the chosen sample size and this needs M comparisons. The determining results need to be aggregated and this needs M additions. The method in [6] needs to select the maximum value and the minimum value of the sample, and this needs $2M$ comparisons. The sample is divided into 8 classes and this needs M comparisons. Finally, the normalized data are substituted into the chi-square test statistics, and this needs several additions, multiplications and divisions. The method in [9] needs to do 8 subtractions for each pixel and its shifted pixel and it selects the maximum and the minimum of the 8 difference values, and this needs 16 subtractions and 32 comparisons. So the M samples need at least $16M$ subtractions and $32M$ comparisons. Therefore, we have

$$\mathrm{Time}\left(F_1\left(M\right)\right) < \mathrm{Time}\left(F_2\left(M\right)\right) < \mathrm{Time}\left(F_3\left(M\right)\right) < \mathrm{Time}\left(F_4\left(M\right)\right).$$

On the other hand, the experimental results in Tables 5–7 show that $M_1 < M_2 < M_3 < M_4$, for the same stego image. Hence, we have

$$\mathrm{Time}\left(F_1\left(M_1\right)\right) < \mathrm{Time}\left(F_2\left(M_2\right)\right) < \mathrm{Time}\left(F_3\left(M_3\right)\right) < \mathrm{Time}\left(F_4\left(M_4\right)\right),$$

i.e., the computational complexity of the proposed method is the lowest among the four methods. In this experiment, the decompressed gray image Peppers of size 320×480 is the cover image and its corresponding quality factor is 90. The random embedding path is generated by Hide and Seek 4.1. The PC has an Intel Core i5 3.2 GHz CPU and 4 GB DDR3 RAM. We search for the key on this PC, and the sample sizes of the four methods are $M_1 = 1427$, $M_2 = 44634$, $M_3 = 57300$ and $M_4 = 76800$. The speed of searching for the key by F_1 is about 106123 keys per second, the speed of the collision attack in F_2 is 1418 keys per second, the speed of the chi-square test in F_3 is 183 keys per second, and the speed of searching for the key by F_4 is 132 keys per second. In conclusion, the computational complexity of the proposed method is lower than the methods in [6], [8] and [9].

6 Conclusion

In this paper, the problem of recovering a stego key is translated into a problem of cryptanalysis, where the sample sequence of the output generated by the PRNG with the error P_e is utilized to recover the seed of the PRNG. Moreover, we obtain the minimum data size required for the attack using a theoretical analysis. Since the embedded message is the weak noise, the recompression is used to recover most of the cover pixels for the decompressed JPEG image. The attacker not only receives the stego image, but also obtains most of the cover pixels. An estimated sequence of modification positions is obtained by comparing the stego image with the recovered carrier. In this paper, we infer the theoretical probability of recovering the carrier and propose a method for recovering the stego key when a decompressed JPEG image is used as the carrier, which is applicable for both LSB-R and LSB-M. The theoretical analysis and experimental results show the efficiency and reliability of the proposed method, and also show that the computational complexity of the proposed method is lower than that of the methods in [6], [8] and [9]. Since the key point of this paper is not cryptanalysis, we do not consider a specific attack against the PRNG, and just translate the problem of recovering the stego key into a traditional cryptanalysis problem. Our future work is to recover the key quickly using cryptanalysis methods.

Acknowledgements This work was partly supported by National Natural Science Foundation of China (Grant No. 61302159), Excellent Youth Foundation of Henan Province of China (Grant No. 144100510001), Foundation of Science and Technology on Information Assurance Laboratory (Grant No. KJ-14-108), and Doctoral Dissertation Innovation Fund of Zhengzhou Information Science and Technology Institute (Grant No. BSLWCX201309).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Zhang W M, Li S Q. Information-theoretic analysis for the difficulty of extracting hidden information. Wuhan Univ J Nat Sci, 2005, 10: 315–318

2 Regalia P A. Cryptographic secrecy of steganographic matrix embedding. IEEE Trans Inf Forens Secur, 2008, 3: 786–791

3 Liu J F, Chen J Y, Zhang W M, et al. Cryptographic secrecy analysis of matrix embedding. In: Proceedings of the IEEE International Conference on Multimedia Information Networking and Security, Nanjing, 2010. 691–695

4 Chen J Y, Liu J F, Zhang W M, et al. Cryptographic secrecy analysis of matrix embedding. Int J Comput Intell Syst, 2013, 6: 639–647

5 Fridrich J, Goljan M, Soukal D. Searching for the stego-key. In: Proceedings of SPIE 5306, Security, Steganography, and Watermarking of Multimedia Contents VI, San Jose, 2004. 70–82

6 Fridrich J, Goljan M, Soukal D, et al. Forensic steganalysis: determining the stego key in spatial domain steganography. In: Proceedings of SPIE 5681, Security, Steganography, and Watermarking of Multimedia Contents VII, San Jose, 2005. 631–642

7 Zhang W M, Liu J F, Li S Q. Approaches for recovering key of LSB steganography (in Chinese). Acta Sci Nat Univ Sunyatseni, 2005, 44: 29–33

8 Zhang W M, Li S Q, Liu J F. Extracting attack to LSB steganography in spatial domain (in Chinese). Chin J Comput, 2007, 30: 1625–1631

9 Liu J, Tang G M. Stego key estimation in LSB steganography. J Multimedia, 2012, 7: 309–313

10 Zhang W M, Li S Q, Liu J F. Analysis for the equivalent keys of steganographic scheme CPT. Acta Electron Sin, 2007, 35: 2258–2261

11 Chen J Y, Zhu Y F, Zhang W M, et al. Chosen-key extracting attack to random LSB steganography. J Commun, 2010, 31: 73–80

12 Zhang J, Zhang D. Detection of LSB matching steganography in decompressed images. IEEE Signal Process Lett, 2010, 17: 141–144

13 Luo W Q, Wang Y G, Huang J W. Security analysis on spatial ± 1 steganography for JPEG decompressed images. IEEE Signal Process Lett, 2011, 18: 39–42

14 Li X, Zhang T, Zhang Y, et al. A novel blind detector for additive noise steganography in JPEG decompressed images. Multimed Tools Appl, 2014, 68: 1051–1068

15 Li X, Zhang T, Zhang Y, et al. Quantitative steganalysis of spatial ± 1 steganography in JPEG decompressed images. Multimed Tools Appl, 2014, 73: 1487–1506

16 Luo W Q, Huang J W, Qiu G P. JPEG error analysis and its applications to digital image forensics. IEEE Trans Inf Forens Secur, 2010, 5: 480–491

17 Liu J F, Tian Y G, Han T, et al. LSB steganographic payload location for JPEG-decompressed images. Digit Signal Process, 2015, 38: 66–76