Remainder with Threshold Substitution Data Hiding Scheme: Counterexamples and Modification

Alexander G. Chefranov, Gürcü Öz[1]

Abstract— **A well-known data hiding scheme, Remainder with Threshold substitution (RT), is analyzed. RT uses a threshold value, *T*, and two moduli numbers, m_u, and m_l, m_l<m_u, to embed secret data into a cover image. RT does not impose any divisibility constraints on the selection of the parameters, *T*, m_u, and m_l, and its correctness (i.e., the data extracted are always the same as the data embedded) is not proved. By counterexamples constructing, we show that RT works for them incorrectly. Also, RT scheme uses pseudo-random number generator (PRNG) to define a pixel for embedding of the next secret bit portion. PRNG can produce repeated values leading to the repeated embedding into the same pixel, thus overwriting previously embedded secret and preventing its correct extraction. We modify RT scheme (by imposing divisibility constraints on the threshold, moduli values, and PRNG), and prove correctness of the modification. Note that in the reported experiments on RT, its parameters used, *T*=160, and (m_l, m_u) from {(4, 8), (16, 32)}, exactly satisfy our constraints, and, thus, the scheme may be correct for such settings.**

Index Terms— **Data hiding scheme, Remainder with Threshold substitution, Cover Image, Pixel, Secret Embedding, Secret Extraction**

I. Introduction

Data hiding is important due to the need to protect private information. In [1], a well-known (e.g., [2]-[35] refer to it) data hiding scheme named herein Remainder with Threshold substitution (RT) with adaptation to a pixel value is proposed. It uses modulus operator to hide the secret data in a host (cover) image pixel by replacing the remainder of the pixel by the secret data similar to least-significant-bit substitution (LSB, see equations (4), (5) in [36]). Contrary to LSB, RT uses a threshold value, *T*, and two moduli values, m_u, and m_l, m_l<m_u, defining the number of bits to be embedded per pixel. For the pixels with values not less (less) than the threshold, *T*, the number of secret bits to be embedded is $\lfloor\log_2 m_u\rfloor$ ($\lfloor\log_2 m_l\rfloor$), where $\lfloor x\rfloor$ denotes the floor function returning the maximal integer not exceeding x.

Despite proposed in 2005, RT is still used for comparison or as a reference scheme (see, e.g. [2-35]; note that eight of the references are published before 2010, 21 papers from 2010 to 2019, and five papers from 2020 to 2022). RT is used for comparison, e.g., in [2] (see Table 2), [3] (see Table 4), [4] (see Table 2), [23] (see Table 1) and [26] (see Table 1). However, [1] does not define how parameters, *T*, m_l, and m_u, are selected, no divisibility constraints are imposed on them. Correctness of RT is not proved in [1]. We show by counterexamples that RT works incorrectly (the data extracted are not the same as the data embedded) if the parameters do not meet divisibility constraints. It was found out that RT parameters, used in [1] for its experiments, exactly meet these constraints, and, hence, for such settings, it is correct. We fix the problem by imposing constraints on the RT parameters; threshold and moduli values. Also, RT uses a pseudo-random number generator (PRNG) to define a pixel used for embedding of the next secret bit portion. PRNG can produce repeated values that in the case of RT leads to the repetitive one and the same pixel using for secret embedding thus overwriting previously embedded secret that prevents its correct extraction. To fix the problem, it is necessary providing one-to-one random mapping (using randomly generated permutations). We prove that under the imposed constraints such modified scheme, RT-M, works correctly.

The contribution of the paper is as follows:

- inconsistence of the known RT method is found out and proved by counterexamples;
- constraints on RT parameters to fix the inconsistence are specified, thus RT modification, RT-M, is defined;
- consistence of RT-M is proved theoretically.

The rest of the paper is organized as follows. In Section 2, RT scheme is described. In Section 3, counterexamples for RT scheme are constructed, RT-M is proposed, and its correctness is stated. Section 4 concludes the paper. Appendix A illustrates RT correct embedding/extraction by a numerical example. Appendix B contains full proof of RT-M correctness.

II. RT Scheme Description

In RT scheme [1], secret data hiding procedure is divided into three phases. In the first phase, the secret message is represented as a bit-string and encrypted. In the third phase, an extracted bit-string is decrypted and reshaped to the form of the original secret message. We do not touch these transformations, and consider just the second phase (secret bit-string embedding), and a part of the third phase related to the bit-string extraction. In the second phase, a pixel for embedding of the next portion of the secret bits is selected using a PRNG with a specified seed value, and depending on the selected pixel's value, the number of the secret bits to be embedded is defined, followed by their embedding into the pixel. The range of the possible pixel

[1]Submitted on July 13, 2023.

A.G. Chefranov is with the Department of Computer Engineering, Eastern Mediterranean University, Famagusta, North Cyprus (e-mail: alexander.chefranov@emu.edu.tr).

G. Öz is with the Department of Computer Engineering, Eastern Mediterranean University, Famagusta, North Cyprus (e-mail: gurcu.oz@emu.edu.tr).

values, [0,255], is split by T into two subranges, less than T, and not less than T. Embedding/extraction in each sub-ranges uses own modulus. In the mid of each range embedding is done with an optimization equivalent to optimal adjustment pixel procedure (OPAP) introduced in [36] for LSB. OPAP minimizes distance between the cover and respective stego pixel values by adding/subtracting the modulus value that can lead to crossing the border values. That is why, in [1], OPAP-like optimization procedure is not applied near the borders shown in Fig. 1 as filled boxes of the size of the half of the respective modulus value.

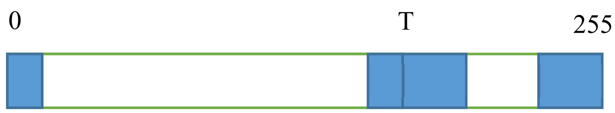

Fig. 1. The range [0,255] of the pixel values split by T into two sub-ranges in the mid of which (empty space) optimization is applied, and with near-border values shown by filling where optimization is not applied.

Below, we describe the second and the part of the third phases of RT scheme using notation of [1]. Note that Case I corresponds to embedding into the lower sub-range with sub-cases I.1 and I.3 corresponding to the near-border pixels, and sub-case I.2 to the mid of the sub-range where the OPAP-like optimization is applied as illustrated by Fig. 1. Similarly, Case II, has respective three sub-cases.

Phase II: [*Secret bit-string, B_s, embedding into the host image, C, resulting in the stego-image, S*]

Input: The host image, *C*; bit-string, B_s; seed key, *SK*.

Output: The stego-image, *S*.

Step 1: Randomly choose a pixel, $P_c(i)$, in *C* using a PRNG with *SK*, where $P_c(i)$ denotes the intensity of the i^{th} pixel with the linear order of top-to-down and left-to-right in *C*.

Step 2: Set the threshold value, *T*, and the two moduli values, m_u, m_l. Then compute the residue, *RES*, and the possible embedding capacity, *EC*:

IF $P_c(i) \geq T$

$$EC = \lfloor \log_2 m_u \rfloor, \quad (1)$$

$$RES = P_c(i) \bmod m_u. \quad (2)$$

ELSE $P_c(i) < T$

$$EC = \lfloor \log_2 m_l \rfloor, \quad (3)$$

$$RES = P_c(i) \bmod m_l. \quad (4)$$

Step 3: $D = |RES - DEC|$, (5)

where *DEC* is the decimal value of *EC* bit-length string fetched from B_s.

Step 4: Embed *DEC* into the pixel, $P_c(i)$, by performing the following process (here, $P_s(i)$ is the intensity of the i^{th} pixel of the stego-image, *S*, after embedding of *DEC*).

Case I: $P_c(i) < T$:

I.1. IF $P_c(i) < \frac{m_l}{2}$

$$P_s(i) = DEC. \quad (6)$$

I.2. ELSE IF $\frac{m_l}{2} \leq P_c(i) < T - \frac{m_l}{2}$

I.2.1. IF $D > \frac{m_l}{2}$

$$AV = m_l - D. \quad (7)$$

I.2.1.1. IF $RES > DEC$

$$P_s(i) = P_c(i) + AV. \quad (8)$$

I.2.1.2. ELSE RES<=DEC

$$P_s(i) = P_c(i) - AV. \quad (9)$$

I.2.2. ELSE $D \leq \frac{m_l}{2}$

$$AV = D. \quad (10)$$

I.2.2.1. IF $RES > DEC$

$$P_s(i) = P_c(i) - AV. \quad (11)$$

I.2.2.2. ELSE $RES \leq DEC$

$$P_s(i) = P_c(i) + AV. \quad (12)$$

I.3. ELSE $T - \frac{m_l}{2} \leq P_c(i) < T$

$$P_s(i) = P_c(i) - RES + DEC. \quad (13)$$

Case II: $P_c(i) \geq T$:

II.1. IF $P_c(i) > 255 - \frac{m_u}{2} + 1$

$$P_s(i) = 255 - m_u + 1 + DEC. \quad (14)$$

II.2. ELSE IF $T + \frac{m_u}{2} < P_c(i) \leq 255 - \frac{m_u}{2} + 1$

II.2.1. IF $D > \frac{m_u}{2}$

$$AV = m_u - D. \quad (15)$$

II.2.1.1. IF $RES > DEC$

$$P_s(i) = P_c(i) + AV. \quad (16)$$

II.2.1.2. ELSE $RES \leq DEC$

$$P_s(i) = P_c(i) - AV. \quad (17)$$

II.2.2. ELSE $D \leq \frac{m_u}{2}$

$$AV = D. \quad (18)$$

II.2.2.1. IF $RES > DEC$

$$P_s(i) = P_c(i) - AV. \quad (19)$$

II.2.2.2. ELSE RES<=DEC

$$P_s(i) = P_c(i) + AV. \quad (20)$$

II.3. ELSE $T \leq P_c(i) \leq T + \frac{m_u}{2}$

$$P_s(i) = P_c(i) - RES + DEC. \quad (21)$$

Step 5: Output the stego-image, *S*, containing pixels, $P_s(i)$, for all i, with embedded secret, B_s.

Flowcharts of Phase II and Phase III are provided in Figs 2-4.

Next is the last phase, Phase III, extracting the secret bit-string from the stego-image, *S*. Here, also similar cases illustrated by Fig. 1 are considered.

Phase III: [*Bit-string extraction*]

Input: The stego-image, *S*; the seed key, *SK*; the threshold value, *T*; the two moduli, m_u and m_l.

Output: The extracted bit-string, B_s'

Step 1: Find the secret embedding pixel, $P_s(i)$, in *S* by using the PRNG with seed, *SK*.

Step 2: Compute *RES'* and *EC'* according to the following two cases.

Case I: $P_s(i) < T$:

$$RES' = P_s(i) \bmod m_l, \quad (22)$$

$$EC' = \lfloor \log_2 m_l \rfloor.$$

Case II: $P_s(i) \geq T$:

$$RES' = P_s(i) \bmod m_u, \quad (23)$$

$$EC' = \lfloor \log_2 m_u \rfloor.$$

Step 3: Translate *RES'* into the bit-string representation with *EC'* bits, *RESBS*. Append *RESBS* to B_s' (B_s' has to be initialized as empty bit-string).

Step 4: Repeat Steps 1–3 until all bits of B_s' are recovered from *S*.

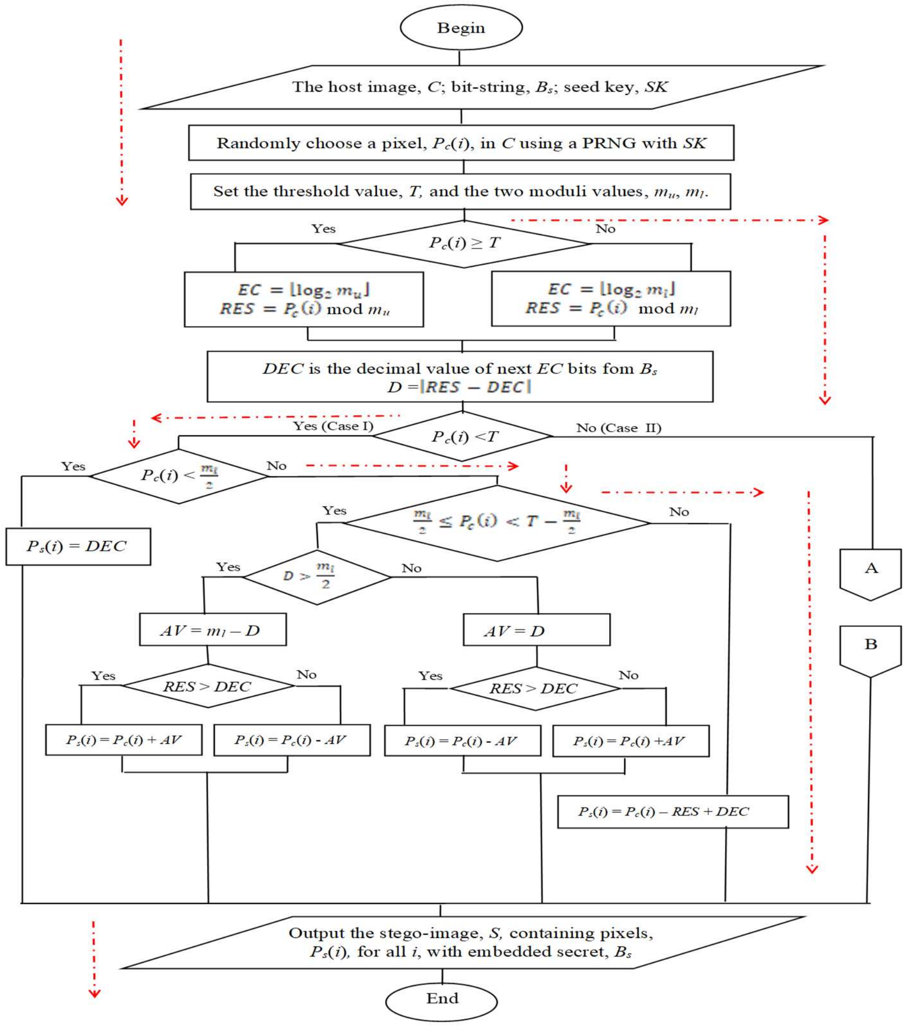

Fig. 2. The algorithm of the Phase II (bit-string embedding) with the control flow for Counterexample 1.

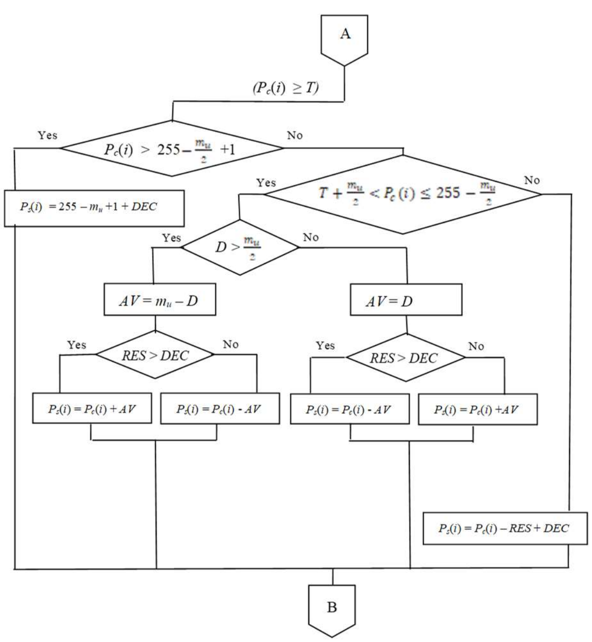

Fig. 3. The algorithm of the Case II of Phase II.

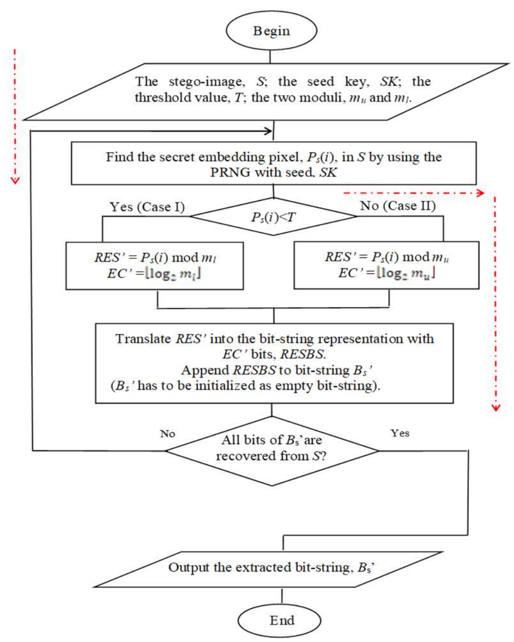

Fig. 4. The algorithm of Phase III (bit-string extraction) with the control flow for Counterexample 1.

Numerical examples of RT scheme embedding/extraction are not provided in [1], but results of experiments with this scheme are reported (see [1, p. 107-108]) for T=160, and (m_u, m_l) from the set {(32, 16), (8, 4)}. That is why, to illustrate the work of RT, we give a numerical example for RT scheme embedding/extraction with T =160, m_u=8, m_l =4, in Appendix A.

III. RT Scheme Counterexamples and its Modification, RT-M

Five counterexamples are given below proving that RT has problems related to its parameters (threshold, moduli values) and to the use of PRNG. In the correct embedding/extraction example (see Appendix A), threshold value $T = 160$ is a multiple of m_u =8 and m_l =4, which are the powers of 2. As we shall see in Appendix A, in these conditions, RT scheme works correctly. We found out that the borders of the both sub-ranges shall be divisible by respective modulus value: m_l shall divide 0 and T, and m_u shall divide T and 256. Since 0 is divided by any number, for m_l , there is only one condition, but for m_u , the both conditions are necessary, and since $256 = 2^8$, m_u shall be a power of 2. Counterexamples 1-4 show that if any of these conditions is violated, there are examples when result of extraction is not equal to the originally embedded secret. Counterexample 1 concerns indivisibility of T by m_l that is denoted by $m_l \nmid T$. Counterexamples 2-4 consider cases when $(m_u \nmid T)\&(m_u \nmid 256)$ in Counterexample 2, $(m_u|T)\&(m_u \nmid 256)$ in Counterexample 3, and $(m_u \nmid T)\&(m_u|256)$ in Counterexample 4, where $a|b$ denotes that an integer, a, divides an integer, b. Counterexample 5 concerns proving of the repetition of the pixel numbers if using just a PRNG. Then RT modification, RT-M, is proposed fixing the problems revealed.

Counterexample 1: It is related with the Case I.3 of Phase II. But in this counterexample, we consider the case when T is not a multiple of m_l. Let the threshold value, T=160, cover pixel, $P_c(i)$ =159 (close to the threshold from below), m_l =9, m_u=16, and a secret message, B_s is $(111)_2$. Since $P_c(i) < T$, using (3), $EC = \lfloor \log_2 9 \rfloor$=3, then read 3 bits from B_s, $(111)_2$, convert it to decimal, DEC = 7, and using (4), $RES = 159 \bmod 9 = 6$. From Case I.3, we have $T - \frac{m_l}{2} = 160 - \frac{9}{2} \le 159 < 160 = T$ is true, and using (13), $P_s(\mathrm{i}) = P_c(\mathrm{i}) - RES + DEC = 159 - 6 + 7 = 160$, which is equal to the threshold, T=160.

Thus, the value of stego-pixel, $P_s(i)$, is 160 that is not less than T=160, whereas the original cover pixel value, $P_c(i)$ =159, is less than T. Hence, when extracting the secret bit-string from the stego-pixel, in Phase III, Case II is used, equation (23), and RES' = $P_s(i)$ mod m_u = 160 mod 16 = 0, EC'=$\lfloor \log_2 m_u \rfloor$=$\lfloor \log_2 16 \rfloor$=4, and B_s'= $(0000)_2$, that is not equal to the original bit-string, B_s=$(111)_2$. Thus, embedding in the Case I.3 may lead to an incorrect extracted value.

Note that in the Counterexample 1, the threshold value, T=160, is not a multiple of m_l = 9.

To clarify the counterexample, tracing of the Phases II (Embedding) and III (Extraction) is provided in Table I and Table II showing the states of the variables after termination of the respective operator (referred to by its equation number). Other counterexamples can easily be traced similarly.

TABLE I
TRACE OF PHASE II (EMBEDDING), STEPS 2-4, FOR T=160, M_L =9, M_U =16, B_s='111' FOR COUNTEREXAMPLE 1

#	Operator	EC	RES	DEC	D	$P_s(i)$
1	(3)	3				
2	(4)		6			
3	(5)			7	1	
4	(13)					160

TABLE II
TRACE OF PHASE III (EXTRACTION), STEPS 2-3, FOR T=160, M_L =9, M_U =16, $P_s(I)$ =160 FOR COUNTEREXAMPLE 1

#	Operator	EC'	RES'	$RESBS$
1	(23)	4	0	0000

The Execution flow of Counterexample 1 is shown in Figs. 2-4.

Counterexample 2: It is related with the Case II.3 of Phase II. We also consider the case when cover pixel, $P_c(i)$ =161 (close to the threshold, T=160, from above), with m_l =8, m_u=15, and a secret message, B_s is $(111)_2$. Since $P_c(i) \ge T$, using (1), $EC = \lfloor \log_2 15 \rfloor$=3, then read 3 bits from B_s, $(111)_2$, convert it to decimal, DEC = 7, and using (2), $RES = 161 \bmod 15 = 11$. From Case II.3, we have $T = 160 \le Pc(i) = 161 \le T + \frac{m_u}{2} = 160 + \frac{15}{2} = 167.5$ is true, and using (21), $P_s(\mathrm{i}) = P_c(\mathrm{i}) - RES + DEC = 161 - 11 + 7 = 157$, which is less than the threshold, T=160.

Thus, the value of stego-pixel, $P_s(i)$, is 157 that is less than T=160, whereas the original cover pixel value, $P_c(i)$ =161, is greater than T. Hence, when extracting the secret bit-string from the stego-pixel, in Phase III, Case I is used, equation (22), and RES' = $P_s(i)$ mod m_l = 157 mod 8 = 5, EC'=$\lfloor \log_2 m_l \rfloor$=$\lfloor \log_2 8 \rfloor$=3, and B_s'= $(101)_2$, that is not equal to the original bit-string, B_s=$(111)_2$. Thus, embedding in the Case II.3 may lead to an incorrect extracted value.

Note that in the Counterexample 2, the threshold value, T=160, is not a multiple of m_u = 15.

To clarify the counterexample, tracing of the Phases II (Embedding) and III (Extraction) is provided in Table III and Table IV.

TABLE III
TRACE OF PHASE II (EMBEDDING), STEPS 2-4, FOR T=160, ML=8, MU=15, BS='111' FOR COUNTEREXAMPLE 2

#	Operator	EC	RES	DEC	D	P_s(i)
1	(1)	3				
2	(2)		11			
3	(5)			7	4	
4	(21)					157

TABLE IV
TRACE OF PHASE III (EXTRACTION), STEPS 2-3, FOR T=160, M_L=8, M_U=15, P_s(I)= 157 FOR COUNTEREXAMPLE 2

#	Operator	EC'	RES'	$RESBS$
1	(22)	3	5	101

Counterexample 3: It is related with the Case II.1 of Phase II. In this counterexample, we consider the case when m_u is a multiple of T and it is not a power of 2. Let $P_c(i) = 253$, $T = 252, m_u = 18$, m_l=8, and secret message, B_s=$(0111)_2$. Since $P_c(i)$=253 ≥ T=252 , using (1), $EC = \lfloor \log_2 18 \rfloor = 4$, then read 4 bits from B_s, $(0111)_2$, convert it to decimal, DEC = 7.

From Case II.1, since 253=P_c(i) >255-m_u/2+1=255$-\frac{18}{2}$+1=247, so using (14), $P_s(i) = 255 - m_u + 1 + DEC = 255 - 18 + 1 + 7 = 245$ which is less than the threshold, *T=252*. Hence, $P_s(i)$ is less than *T*, on the other hand, original $P_c(i)$ = 253 > *T*. Applying Phase III, Step 2, Case I, since $P_s(i)$=245<T=252, by equation (22), *RES'*=$P_s(i)$ mod m_l = 245 mod 8 = 5, EC'=$\lfloor \log_2 m_l \rfloor$=$\lfloor \log_2 8 \rfloor$=3. In *Step 3*, we get binary representation of *RES'*= $(101)_2$ = B_s' that is not equal to the original $B_s = (0111)_2$. Thus, embedding in the Case II.1 may lead to incorrect extracted value.

Note that in the Counterexample 3, the threshold value, *T*=252, is divisible by m_u=18, but m_u is not a power of 2.

To clarify the counterexample, tracing of the Phases II (Embedding) and III (Extraction) is provided in Table V and Table VI.

TABLE V
TRACE OF PHASE II (EMBEDDING), STEPS 2-4, FOR *T*=252, *ML*=8, *MU*=18, *BS*='0111' FOR COUNTEREXAMPLE 3.

#	Operator	*EC*	*RES*	*DEC*	*D*	$P_s(i)$
1	(1)	4				
2	(2)		9			
3	(5)			7	2	
4	(21)					245

TABLE VI
TRACE OF PHASE III (EXTRACTION), STEPS 2-3, FOR *T*=252, *ML*=8, *MU*=18, *PS(I)*=245 FOR COUNTEREXAMPLE 3.

#	Operator	*EC'*	*RES'*	*RESBS*
1	(22)	3	5	101

Counterexample 4: It is, as Counterexample 3, related to Case II.1 of Phase II. In this counterexample, we consider the case where m_u is not a multiple of *T* and it is a power of 2. Let $T = 252$, m_u = 16, m_l=8, $P_c(i)$=253, $B_s=(1011)_2$. Since $P_c(i)$ = 253 ≥ *T*=252, using (1), $EC = \lfloor \log_2 16 \rfloor$ = 4, then read *EC* bits from B_s, $(1011)_2$, convert it to decimal, *DEC* = 11. From Case II.1, we have, P_c(i)=253 >255-m_u/2+1= 255$-\frac{16}{2}$+1=248, so using (14), $P_s(i) = 255 - m_u + 1 + DEC = 255 - 16 + 1 + 11 = 251$, which is less than the threshold, *T*=252. Hence, $P_s(i) = 251$ is less than *T*=252, on the other hand, original $P_c(i)$ = 253 > *T*=252. Applying Phase III, Step 2, Case I, since $P_s(i)$=251<T=252, by equation (22), *RES'*=$P_s(i)$ mod m_l = 251 mod 8 = 3, EC'=$\lfloor \log_2 m_l \rfloor$=$\lfloor \log_2 8 \rfloor$=3. In *Step 3*, we get 3-bit binary representation of *RES'* = $(011)_2$=B_s' that is not equal to the original $B_s=(1011)_2$. Thus, embedding in the Case II.1 may lead to incorrect extracted value.

Note that in Counterexample 4, the threshold value, *T*=252, is not a multiple of m_u=16, but m_u is a power of 2.

To clarify the counterexample, tracing of the Phases II (Embedding) and III (Extraction) is provided in Table VII and

TABLE **VIII**.

TABLE VII
TRACE OF PHASE II (EMBEDDING), STEPS 2-4, FOR *T*=252, *ML*=8, *MU*=16, *BS*='1011' FOR COUNTEREXAMPLE 4

#	Operator	*EC*	*RES*	*DEC*	*D*	$P_s(i)$
1	(1)	4				
2	(2)		9			
3	(5)			11	2	
4	(21)					251

TABLE VIII
TRACE OF PHASE III (EXTRACTION), STEPS 2-3, FOR *T*=252, *ML*=8, *MU*=16, *PS(I)*=251 FOR COUNTEREXAMPLE 4

#	Operator	*EC'*	*RES'*	*RESBS*
1	(22)	3	3	011

Counterexample 5: It is related to the use of PRNG in Phase II, Step 1. In that step, a next pixel for embedding is selected randomly that may result in the reuse of one and the same pixel for embedding of several secret bit-string portions. For example, if a cover image contains *N* pixels, and PRNG selects the next pixel uniformly randomly, then the probability of the choices without repetition is $\frac{1}{N} \cdot \frac{1}{N-1} \cdot \ldots \cdot \frac{1}{2} \cdot 1 = \frac{1}{N!}$, and the probability of repeating at least of two choices is $PR(N) = 1 - \frac{1}{N!}$ that tends to 1 with the growth of *N*. For an image with 512x512 pixels, *N*=2^{18}=262144, and $PR(N) = 1 - \frac{1}{N!} = 1 - \frac{1}{262144!} \approx 1$. A sample of 10 out of 100 uniformly pseudo-randomly generated numbers in Maple is shown in Fig. 2.

```
roll := rand(1 ..100) :
a := Array(1 ..100)
for i from 1 to 100 do a(i) := roll( ); end:
a(61 ..70)
    [ 73  22  32  98  9  53  3  98  69  3 ]
```

Fig. 5. Maple commands to generate 100 uniformly distributed pseudo-random numbers and display 10 of them, *a*(61..70).

From Fig. 5, we see that value 98 is repeated twice, resulting in overwriting of the embedded secret, and, hence, losing information in the Phase III of the secret extraction.

Conditions (24)-(27) together with the requirement of non-repeating pixel numbers generated by PRNG in Phase II, Step 1 of RT, define the RT modification, RT-M. Thus, RT-M differs from RT by the choice of its parameters defined by (24)-(27). Correctness of RT-M is stated in the theorem below.

RT Modification, RT-M, Correctness Theorem: If the following conditions (24)-(27) hold

$$m_l | T, \tag{24}$$
$$m_u | 2^8, \tag{25}$$
$$m_u | T, \tag{26}$$
$$m_l < m_u < T < 2^8 , \tag{27}$$

and if the PRNG used in Phase II, Step 1, guarantees non-repeating sequence of pixels selected for embedding/extraction, then RT scheme works correctly, i.e. extraction by Phase III of RT scheme of the secret from a stego-image obtained by embedding of a secret into the cover image by Phase II of RT scheme is equal to the original secret for any original secret.

Proof of the RT Modification correctness is provided in Appendix B. Note that Counterexamples 1-4 violate respectively conditions (24)-(26), and Counterexample 5 violates the condition on the pseudo-random pixel sequence generation stated in the RT Modification, RT-M. Also note, that the parameters values, T=160, m_l is from {4, 16}, m_u is from {8, 32}, used in experiments [1], meet our conditions (24)-(26), and hence RT might work correctly provided a proper PRNG was actually used. It shall be also noted that violation of the conditions (24)-(26) does not imply incorrectness of embedding/extraction. So, in the conditions of Counterexample 1, if the secret bit-string would be '110' instead of '111' used, then stego value will be not 160, but 159, and in the extraction process, it will be extracted as $159 \, mod \, 9 = 6 = '110'$, that is, correctly.

IV. CONCLUSION

In this paper, incorrectness of the RT scheme [1] using adaptation to the pixel value where the next secret portion is embedded with the help of three parameters, threshold, and two moduli values defining how many secret bits shall be embedded, is proved by counterexamples for which the data extracted is not the same as the embedded. Numerical example of RT scheme correct embedding/extraction is given in Appendix A for the parameters mentioned in [1] as used for their experiments. Counterexamples 1-4 are constructed to violate introduced for RT-M scheme conditions (24)-(26). Counterexample 5 uses weakness of the RT scheme which is the consequence of the use of a PRNG for the next for embedding pixel defining. Such randomness can lead to the pixel repetition, i.e. several times using one and the same pixel for the secret embedding, and each next writing destroying previously written there secret data. The problems are fixed by imposing constraints (24)-(27) on the threshold and two moduli values, and also on the PRNG so that it shall provide a one-to-one random mapping, which guarantees non-repeating sequence of pixels selected for embedding/extraction. As a result, the modification of RT scheme, RT-M, is defined, and the proof of its correctness is provided in Appendix B. Note that RT scheme parameters used in the experiments [1] meet the conditions (24)-(27).

Appendix A: RT scheme correct embedding/extraction example

Consider a numerical example for RT scheme with threshold value T=160, m_u=8, m_l=4. Let the cover image, C, pixel values are $(160\ 200\ 255\ 150)_{10}$, the secret bit-string B_s is $(01110111101)_2$. For simplicity, we do not use a PRNG in the example to determine a pixel for embedding; just the pixels are used one by one, in natural order, for embedding. Thus, we consider embedding of the secret, B_s, into the cover image, C, followed by extraction of the embedded secret from the stego-image, S.

Embedding of the secret into the first four bytes of the cover image:

Phase II: [*Secret embedding*]

1. Embed into the pixel, $P_c(i)$, i = 1:

Step 1: Chosen pixel is $P_c(i) = P_c(1) = 160$.

Step 2: T=160, m_u=8, m_l=4.

Since $P_c(i) \geq T$; $(160 \geq 160)$ is true, compute EC using (1) $EC = \lfloor \log_2 m_u \rfloor = \lfloor \log_2 8 \rfloor = 3$, then read 3 bits from B_s, $(011)_2$ and convert it to decimal, DEC = 3. Compute RES using (2). $RES = P_c(i) \, mod \, m_u$.=160 mod 8 = 0.

Step 3: Compute D using (5), $D = |RES - DEC| = |0 - 3| = 3$.

Step 4: Embed DEC=3 into the pixel $P_c(i)$ =160. Here,

- **Case II**: $P_c(i) \geq T$; $(160 \geq 160)$ is true and

  II.3. $T \leq P_c(i) \leq (T + \frac{m_u}{2})$; $(160 \leq 160 \leq 160 + \frac{8}{2})$ is true; compute $P_s(i)$ using (21)

  $P_s(1)=P_s(i) = P_c(i) - RES + DEC$=160 – 0 + 3=163.

2. Embed into the pixel, $P_c(i)$, i = 2:

Step 1: Chosen pixel is $P_c(i) = Pc(2) = 200$.

Step 2: T=160, m_u=8, m_l=4.

Since $P_c(i) \geq T$; $(200 \geq 160)$ is true, compute EC using (1): $EC = \lfloor \log_2 m_u \rfloor = \lfloor \log_2 8 \rfloor = 3$, then read next 3 bits from B_s, $(101)_2$ and convert it to decimal, DEC = 5. Compute RES using (2). $RES = P_c(i) \, mod \, m_u$.=200 mod 8 = 0.

Step 3: Compute D using (5), $D = |RES - DEC| = |0 - 5| = 5$.

Step 4: Embed DEC=5 into the pixel $P_c(2)$ =200. Here,

- **Case II**: $P_c(i) \geq T$; $(200 \geq 160)$ is true and

  II.2. $(T + \frac{m_u}{2}) < P_c(2) \leq (255 - \frac{m_u}{2} + 1)$; $(160 + \frac{8}{2}) < 200 \leq (255 - \frac{8}{2} + 1)$, is true then,

  II.2.1 $D > \frac{m_u}{2}$; $5 > \frac{8}{2}$ is also true, so, compute AV using (15) $AV = m_u - D$ = 8 – 5 = 3. From Case **II.2.1.2** $RES \leq DEC$; $0 \leq 5$ *is* true, compute $P_s(2)$ using (17)

  $P_s(i)=P_s(2) = P_c(2) - AV$ = 200 – 3 =197.

3. Embed into the pixel, $P_c(i)$, i = 3:

Step 1: Chosen pixel is $P_c(i) = P_c(3) = 255$.

Step 2: T=160, m_u=8, m_l=4.

Since $P_c(i) = P_c(3) \geq T$; $255 \geq 160$ is true, compute EC using (1)

$EC = \lfloor \log_2 m_u \rfloor = \lfloor \log_2 8 \rfloor = 3$, then read 3 bits from B_s, $(111)_2$ and convert it to decimal, DEC = 7. Compute RES using (2). $RES = P_c(3) \, mod \, m_u$.=255 mod 8 = 7.

Step 3: Compute D using (5), $D = |RES - DEC| = |7 - 5| = 2$.

Step 4: Embed DEC=7 into the pixel $P_c(3)$ =255. Here,

- **Case II**: $P_c(3) \geq T$; $(255 \geq 160)$ is true and

  II.1. Pc(i)=$P_c(3)$=255 $> 255 - \frac{m_u}{2} + 1 = 255 - \frac{8}{2} + 1$=252 is true. Compute $P_s(3)$ using (14)

  $P_s(i)=P_s(3) = 255 - m_u + 1 + DEC$ = 255 – 8 +1 + 7 =255.

4. Embed into the pixel, $P_c(i)$, i = 4:

Step 1: Chosen pixel is $P_c(i) = P_c(4) = 150$.

Step 2: T=160, m_u=8, m_l=4.

Since $P_c(4) < T$; $(150 < 160)$, is true, compute EC using (3) $EC = \lfloor \log_2 m_l \rfloor = \lfloor \log_2 4 \rfloor$=2, then read 2 bits from B_s, $(01)_2$ and convert it to decimal, DEC = 1. Compute RES using (4). $RES = P_c(4) \, mod \, m_l$.=150 mod 4 = 2.

Step 3: Compute D using (5), $D = |RES - DEC| = |2 - 1| = 1$.

Step 4: Embed DEC=1 into the pixel $P_c(4)$ =150. Here,

- **Case I**: $P_c(4)$=150 <T=160 is true, then

  I.2. $\frac{m_l}{2} \leq P_c(i) < T - \frac{m_l}{2}$; $\frac{4}{2} \leq 150 < 160 - \frac{4}{2}$ is true, then

  I.2.2. $D \leq \frac{m_l}{2}$; $1 \leq \frac{4}{2}$ is true, compute AV using (10), $AV = D$ = 1. Here **I.2.2.1.** $RES=2 > DEC=1$ is true, compute $P_s(4)$ using (11)

  $P_s(i) = P_s(4) = P_c(i) - AV$ =150 – 1= 149.

Step 5: The stego-image, *S*, containing the $P_s(i)$, $i=1...4$, of embedded secret, B_s, is (163 197 255 149).

Consider now extraction from the four consecutive bytes of the cover image.

Phase III: [*Bit-string extraction*]

1. B_s'={}; Extract from the pixel, $P_c(i)$, $i = 1$;

Step 1: $P_s(1) = 163$.

Step 2: Compute *RES'* and *EC'* according to the following case.

• **Case II**: $P_s(i)=P_s(1)=163 \geq T=160$ is true; compute *RES'* and *EC'* using (23)

$RES' = P_s(1) \; mod \; m_u = 163 \; mod \; 8 = 3$ and $EC' = \lfloor \log_2 8 \rfloor = 3$.

Step 3: Translate the *RES'* = 3 into the bit representations with *EC'*=3 bit-length: *RESBS*=$(011)_2$. So the secret message, B_s'= $(011)_2$ is restored.

2. Extract from the pixel, $P_c(i), i = 2$;

Step 1: $P_s(i)=P_s(2) = 197$.

Step 2: Compute *RES'* and *EC'* according to the following case.

• **Case II**: $P_s(i)=P_s(2)=197 \geq T=160$ is true; compute *RES'* and *EC'* using (23)

$RES' = P_s(2) \; mod \; m_u = 197 \; mod \; 8 = 5$ and $EC' = \lfloor \log_2 8 \rfloor = 3$.

Step 3: Translate the *RES'* = 5 into the bit representations with *EC'*=3 bit-length: *RESBS*=$(101)_2$. So the secret message, B_s'=$(011\ 101)_2$ is restored.

3. Extract from the pixel, $P_c(i)$, $i = 3$;

Step 1: $P_s(i)=P_s(3) = 255$.

Step 2: Compute *RES'* and *EC'* according to the following case.

• **Case II**: $P_s(i)=P_s(3)=255 \geq T=160$ is true; compute *RES'* and *EC'* using (23)

$RES' = P_s(3) \; mod \; m_u = 255 \; mod \; 8 = 7$ and $EC' = \lfloor \log_2 8 \rfloor = 3$.

Step 3: Translate the *RES'* = 7 into the bit representations with *EC'*=3 bit-length: *RESBS*=$(111)_2$. So the secret message, B_s' = $(011\ 101\ 111)_2$ is restored.

4. Extract from the pixel, $P_c(i)$, $i = 4$;

Step 1: $P_s(i) = P_s(4) = 149$.

Step 2: Compute *RES'* and *EC'* according to the following case.

• **Case I**: $P_s(i)=P_s(4)=149 < T=160$ is true; compute *RES'* and *EC'* using (22)

$RES' = P_s(4) \; mod \; m_l = 149 \; mod \; 4 = 1$ and $EC' = \lfloor \log_2 4 \rfloor = 2$.

Step 3: Translate the *RES'* = 1 into the bit representations with *EC'*=2 bit-length: *RESBS*=$(01)_2$. So the secret message, B_s'= $(011\ 101\ 111\ 01)_2$ is restored.

Step 4: The embedded bits of B_s'= $(011\ 101\ 111\ 01)_2$ are recovered from the stego-image (163 197 255 149).

As we see, the extracted bit-string, B_s', coincides with the original secret bit-string, B_s. Thus, RT scheme for that example works correctly.

Appendix B: Proof of the RT Modification, RT-M, correctness: We consider the proof for **Case I.** in items 1-6, and **Case II** in items 7-12 below. Items 1 and 6 consider near-border cases and Items 2-5 consider mid cases where OPAP-like optimization is applied for the lower sub-range. Respectively, Items 7 and 12 consider near-border cases and Items 8-11 consider mid cases for the upper sub-range.

1. Proof for **Case I** $P_c(i) < T$ and **Case I.1** $P_c(i) < \frac{m_l}{2}$.

From (3) and *DEC* definition in (5),

$$0 \leq DEC < 2^{EC} \leq m_l, \tag{B1}$$

From (4),

$$0 \leq RES = P_c(i) \; mod \; m_l < m_l. \tag{B2}$$

From **Case I.1** condition, (B1), (6), and (27),

$$P_s(i) = DEC < T. \tag{B3}$$

Thus, $P_s(i)$ and $P_c(i)$ are less than *T*. In the extraction, using (22) and (B3)

$$RES' = P_s(i) mod \; m_l = DEC \; mod \; m_l.$$

Since by (B1), $DEC < m_l$, *RES'* = *DEC*, i.e. extracted and embedded values are the same. Thus, for the **Case I.1**, correctness is proved.

2. Proof for **Case I.2** $\frac{m_l}{2} \leq P_c(i) < T - \frac{m_l}{2}$, **Case I.2.1** $D > \frac{m_l}{2}$, and **Case I.2.1.1** *RES* > *DEC*.

From (5), **Case I.2.1** condition, and **Case 1.2.1.1** condition,

$$D = |RES - DEC| = \text{RES} - \text{DEC} > \frac{m_l}{2}. \tag{B4}$$

From (B4),

$$DEC < RES - \frac{m_l}{2}. \tag{B5}$$

Using (4) and (B5),

$$DEC < P_c(i) mod \; m_l - \frac{m_l}{2}. \tag{B6}$$

From (7) and (B4),

$$AV = m_l - D = m_l - RES + DEC. \tag{B7}$$

Using (B7) in (8),

$$P_s(i) = P_c(i) + AV = P_c(i) + m_l - RES + DEC. \tag{B8}$$

From (4), (B6), and (B8),

$$P_s(i) = P_c(i) + m_l - RES + DEC < Pc(i) + m_l - P_c(i) mod \; m_l + P_c(i) mod \; m_l - \frac{m_l}{2} = P_c(i) + \frac{m_l}{2}. \tag{B9}$$

Then, from (B9) and **Case I.2** condition $\frac{m_l}{2} \leq P_c(i) < T - \frac{m_l}{2}$,

$$P_s(i) < P_c(i) + \frac{m_l}{2} < T. \tag{B10}$$

Thus, we see that $P_s(i)$ and $P_c(i)$ are both less than *T*.

In extraction algorithm, using (4), (22), (B1), and (B8),

$$RES' = P_s(i) \; mod \; m_l = (P_c(i) + m_l - RES + DEC) \; mod \; m_l = (P_c(i) - P_c(i) \; mod \; m_l) \; mod \; m_l + m_l \; mod \; m_l + DEC \; mod \; m_l = 0 + 0 + DEC \; mod \; m_l = DEC.$$

Thus, extracted value, *RES'*, and embedded value, *DEC*, are the same, and their binary representation is also the same, since *EC'*=*EC*, and the correctness is proved for the **Case I.2.1.1**

3. Proof for the **Case I.2.1** $D > \frac{m_l}{2}$ and **Case I.2.1.2** *RES* ≤ *DEC*.

From (5), (7), and **Case I.2.1.2** condition,

$$AV = m_l - D = m_l + RES - DEC. \tag{B11}$$

Use (B11) in (9):

$$P_s(i) = P_c(i) - AV = P_c(i) - m_l - RES + DEC. \tag{B12}$$

Using (B1), (4), and (B12),

$$P_s(i) = P_c(i) - m_l - RES + DEC = P_c(i) - m_l - P_c(i) mod \; m_l + DEC \leq P_c(i) - (m_l - DEC) < P_c(i)\,. \tag{B13}$$

Then, from (B13) and **Case I** condition, $P_c(i) < T$,

$$P_s(i) < P_c(i) < T. \tag{B14}$$

Thus, we see that $P_s(i)$ and $P_c(i)$ are both less than *T*.

In extraction algorithm, using (B1), (B12), (4) and (22),

$$RES' = P_s(i) mod \; m_l = (P_c(i) - P_c(i) \; mod \; m_l) \; mod \; m_l - m_l \; mod \; m_l + DEC \; mod \; m_l = 0 - 0 + DEC \; mod \; m_l = DEC.$$

Thus, extracted value, *RES'*, and embedded value, *DEC*, are the same, and their binary representation is also the same, since *EC'*=*EC*, and the correctness is proved for the **Case I.2.1.2**.

4. Proof for the **Case I.2.2.** $D \le \frac{m_l}{2}$ and **Case I.2.2.1** $RES > DEC$.

From (5), **Case I.2.2** condition, and **Case I.2.2.1** condition,

$$D = |RES - DEC| = RES - DEC \le \frac{m_l}{2}. \quad \text{(B15)}$$

From (B.15),

$$AV = RES - DEC. \quad \text{(B16)}$$

From (11) and (B16),

$$P_s(i) = P_c(i) - AV = P_c(i) - RES + DEC. \quad \text{(B17)}$$

Using (4) and Case **I.2.2.1** condition in (B17),

$$P_s(i) = P_c(i) - RES + DEC < P_c(i). \quad \text{(B18)}$$

Then, from **Case I** condition $P_c(i) < T$ and (B18),

$$P_s(i) < P_c(i) < T. \quad \text{(B19)}$$

Hence, both $P_s(i)$ and $P_c(i)$ are less than T.

In extraction algorithm, using (B1), (B17), (2) and (22),

$$RES' = P_s(i) mod\ m_l = (Pc(i) - P_c(i) mod\ m_l) mod\ m_l + DEC mod\ m_l$$
$$= 0 + DEC mod\ m_l = DEC.$$

Hence, extracted value, RES', and embedded value, DEC, are the same, and their binary representation is also the same, since $EC'=EC$, and the correctness is proved for the **Case I.2.2.1**

5. Proof of the correctness for the **Case I.2.2.** $D \le \frac{m_l}{2}$ and **Case I.2.2.2** $RES \le DEC$.

From (5) and **Case I.2.2** condition,

$$D = |RES - DEC| \le \frac{m_l}{2}. \quad \text{(B20)}$$

From (10) and **Case I.2.2.2** condition,

$$AV = D = DEC - RES. \quad \text{(B21)}$$

From (12) and (B21),

$$P_s(i) = P_c(i) + AV = P_c(i) + D. \quad \text{(B22)}$$

From (B20) and (B22),

$$P_s(i) = P_c(i) + D \le P_c(i) + \frac{m_l}{2}. \quad \text{(B23)}$$

Then from (B23) and **Case I.2** condition $\frac{m_l}{2} \le P_c(i) < T - \frac{m_l}{2}$,

$$P_s(i) \le P_c(i) + \frac{m_l}{2} < T. \quad \text{(B24)}$$

Hence, both $P_s(i)$ and $P_c(i)$ are less than T.

In extraction algorithm, using (B1), (B21), (B22), (4), and (22),

$$RES' = (P_c(i) + DEC - RES) mod\ m_l = (Pc(i) - P_c(i) mod\ m_l) mod\ m_l + DEC mod\ m_l$$
$$= 0 + DEC mod\ m_l = DEC.$$

Hence, extracted value, RES', and embedded value, DEC, are the same, and their binary representation is also the same, since $EC'=EC$, and the correctness is proved for the **Case I.2.2.2**.

Note that in the items 1-5 above of the proof of RT-M correctness, conditions (24)-(26) were not used.

6. Proof of the correctness for the **Case I** $P_c(i) < T$ and **Case I.3** $T - \frac{m_l}{2} \le P_c(i) < T$.

From (4), **Case I.3** condition, and (13),

$$P_s(i) = P_c(i) - RES + DEC = P_c(i) - P_c(i) mod\ m_l + DEC, \quad \text{(B25)}$$
$$P_c(i) - P_c(i) mod\ m_l = k\, m_l, \quad \text{(B26)}$$

where $k \ge 0$ is some integer.

Using (B26) in (B25), one gets

$$P_s(i) = k\, m_l + DEC. \quad \text{(B27)}$$

From **Case I** condition, (B26), and (24),

$$k < k_1. \quad \text{(B28)}$$

Hence, from (B1), (B27), (B28), and (24),

$$P_s(i) = km_l + DEC < km_l + m_l = (k+1)\, m_l \le T = k_1 * m_l. \quad \text{(B29)}$$

Thus, both $P_s(i)$ and $P_c(i)$ are less than T. In extraction algorithm, using (B1), and (B29) in (22),

$$RES' = P_s(i) mod\ m_l = (km_l + DEC) mod\ m_l = km_l mod\ m_l + DEC\ mod\ m_l = DEC.$$

Thus, extracted value, RES', and embedded value, DEC, are the same, and their binary representation is also the same, since, by (3), $EC'=EC$, and the correctness for the Case I.3 is also proved but now, in that item 6, we need the use of the condition (24). Note that this condition was violated in the Counterexample 1 showing incorrect work of RT.

7. Proof of the correctness for the **Case II** $P_c(i) \ge T$ and **Case II.1** $P_c(i) > 255 - \frac{m_u}{2} + 1$,

From (1), (5),

$$0 \le DEC < 2^{EC} \le m_u. \quad \text{(B30)}$$

From (2),

$$0 \le RES = P_c(i)\ mod\ m_u < m_u. \quad \text{(B31)}$$

From **Case II.1** condition and (14),

$$P_s(i) = 256 - m_u + DEC \quad \text{(B32)}$$

Using **Case II.1** condition and (25)-(27), assuming that

$$256 = k * m_u, \text{(B33)}$$
$$P_c(i) > 256 - \frac{m_u}{2} = k * m_u - \frac{m_u}{2} = \left(k - \frac{1}{2}\right) * m_u, \text{(B34)}$$
$$P_c(i) \ge T = k_3 * m_u. \quad \text{(B35)}$$

Since according to (27), $T < 256$, from (B33), (B35),

$$k_3 < k. \quad \text{(B36)}$$

According to (B31), (B33), (B35), and (B36),

$$P_s(i) = 256 - m_u + DEC = (k-1) * m_u + DEC \ge k_3 * m_u + DEC$$
$$= T + DEC \ge T \quad \text{(B37)}$$

From (B37), we see that both, $P_s(i)$ and $P_c(i)$ are not less than T.

In extraction algorithm, from (23), (B30), (B32), (B33),

$$RES' = P_s(i) mod\ m_u = (256 - m_u + DEC) mod\ m_u = \big((k-1)m_u + DEC\big) mod\ m_u = DEC\ mod\ m_u = DEC. \quad \text{(B38)}$$

From (B38), the extracted value, RES' and embedded value, DEC, and their binary representation is also the same, since $EC'=EC$, are the same, and the correctness is proved for the **Case II.1**. Note that in this **Case II.1**, considered in item 7, conditions (25)-(27) are used, and in the Counterexample 3, condition (25) is violated, and in the Counterexample 4, condition (26) is violated.

8. Prove now correctness for the **Case II.2.1** $D > \frac{m_u}{2}$ and **Case II.2.1.1** $RES > DEC$.

From (5), **Case II.2.1** condition, and **Case II.2.1.1** condition,,

$$D = |RES - DEC| = RES - DEC \frac{m_u}{2}. \text{(B39)}$$

From (15) and (B39),

$$AV = m_u - RES + DEC. \quad \text{(B40)}$$

From (16) and (B40),

$$P_s(i) = P_c(i) + m_u - RES + DEC. \text{(B41)}$$

From (2), **Case II.2.** condition $T + \frac{m_u}{2} < P_c(i) \le 255 - \frac{m_u}{2}$ +1, and (B41),

$$P_s(i) = P_c(i) + m_u - P_c(i) mod\ m_u + DEC > P_c(i) + DEC \ge P_c(i) > T. \quad \text{(B42)}$$

Then, from (B42),

$$P_s(i) > T. \quad \text{(B43)}$$

Thus, both $P_s(i)$and $P_c(i)$ are greater than T.

In extraction algorithm, from (2), (23), (B30), and (B41),

$$RES' = P_s(i) mod\ m_u = (P_c(i) - P_c(i) mod\ m_u) mod\ m_u + m_u mod\ m_u + DEC\ mod\ m_u = 0+0+ DEC\ mod\ m_u = DEC.$$

Thus, the extracted value, *RES'*, is the same as the embedded value, *DEC*, and their binary representation is also the same, since *EC'=EC*, and the correctness is proved for the **Case II.2.1.1** Note that in this item 8, as in items 1-5, conditions of the RT-A, (24)-(26), are not used.

9. Prove the correctness for the **Case II.2.1.** $D > \frac{m_u}{2}$ and **Case II.2.1.2** $RES \leq DEC$.

From (5) and **Case II.2.1** condition $D > \frac{m_u}{2}$,

$$D = |RES - DEC| > \frac{m_u}{2}. \quad (B44)$$

From **Case II.2.1.2** condition and (B44),

$$DEC > RES + \frac{m_u}{2}. \quad (B45)$$

From (2) and (B45),

$$DEC > P_c(i)\ mod\ m_u + \frac{m_u}{2}. \quad (B46)$$

From (15) and (B44),

$$AV = m_u - DEC + RES. \quad (B47)$$

From (17) and (B47)

$$P_s(i) = P_c(i) - m_u - RES + DEC. \quad (B48)$$

From (2), (B46), and (B48),

$$P_s(i) > P_c(i) - m_u - P_c(i) mod\ m_u + P_c(i)\ mod\ m_u + \frac{m_u}{2} = P_c(i) - \frac{m_u}{2}. \quad (B49)$$

Then, from (B49), and **Case II.2** condition $T + \frac{m_u}{2} < P_c(i) \leq 255 - \frac{m_u}{2} +1$,

$$P_s(i) > P_c(i) - \frac{m_u}{2} > T. \quad (B50)$$

From (B50), both $P_s(i)$ and $P_c(i)$ are greater than *T*.

In extraction algorithm, from (2), (23), (B30), and (B48),

$$RES' = P_s(i) mod\ m_u = (P_c(i) - P_c(i) mod\ m_u) mod\ m_u - m_u mod\ m_u + DEC\ mod\ m_u = 0 - 0 + DEC\ mod\ m_u = DEC.$$

Thus, the extracted value, *RES'*, is the same as the embedded value, *DEC*, and their binary representation is also the same, since *EC'=EC*, and the correctness is proved for this item 9 without use of the RT-M conditions (24)-(26).

10. Prove now correctness for the **Case II.2.2** $D \leq \frac{m_u}{2}$ and **Case II.2.2.1** *RES > DEC.*

From (5), and **Case II.2.2** condition**,**

$$D = |RES - DEC| \leq \frac{m_u}{2}. \quad (B51)$$

From **Case II.2.2.1** condition, and (B51),

$$DEC \geq RES - \frac{m_u}{2}. \quad (B52)$$

From (2), and (B52),

$$DEC \geq P_c(i) mod\ m_u - \frac{m_u}{2}. \quad (B53)$$

From (18), and **Case II.2.2.1** condition,

$$AV = RES - DEC. \quad (B54)$$

From (B54), and (19),

$$P_s(i) = P_c(i) - AV = P_c(i) - RES + DEC. \quad (B55)$$

Using (2), (B53), and (B55),

$$P_s(i) = P_c(i) - RES + DEC \geq P_c(i) - P_c(i) mod\ m_u + P_c(i) mod\ m_u - \frac{m_u}{2} = P_c(i) - \frac{m_u}{2}. \quad (B56)$$

Then, from (B56), and **Case II.2** condition $T + \frac{m_u}{2} < P_c(i) \leq 255 - \frac{m_u}{2} +1$,

$$P_s(i) \geq P_c(i) - \frac{m_u}{2} > T. \quad (B57)$$

Thus, we see that $P_s(i)$ and $P_c(i)$ are both greater than *T*.

In extraction algorithm, using (B30), (B55), (2), and (23), $RES' = P_s(i)\ mod\ m_u = (P_c(i) - P_c(i) mod\ m_u) mod\ m_u + DEC\ mod\ m_u = 0 + DEC\ mod\ m_u = DEC$.

Thus, the extracted value, *RES'*, is the same as the embedded value, *DEC*, and their binary representation is also the same, since *EC'=EC*, and the correctness is proved for this item 10 without use of the RT-M conditions (24)-(26).

11. Now prove correctness in the **Case II.2.2** $D \leq \frac{m_l}{2}$ and **Case II.2.2.2** *RES ≤ DEC.*

From (18), and **Case II.2.2.2** condition *RES ≤ DEC,*

$$AV = D = DEC - RES >= 0. \quad (B58)$$

Using (20) and (B58),

$$P_s(i) = P_c(i) + D \geq P_c(i) > T. \quad (B59)$$

Thus, from (B59), both $P_s(i)$, $P_c(i)$ are greater than *T*.

In the extraction algorithm, from (2), (23), (B30), (B58), and (B59),

$$RES' = (P_c(i) - P_c(i) mod\ m_u)\ mod\ m_u + DEC\ mod\ m_u = 0 + DEC\ mod\ m_u = DEC.$$

Thus, the extracted value, *RES'*, is the same as the embedded value, *DEC*, and their binary representation is also the same, since *EC'=EC*, and the correctness is proved for this item 11 without use of the RT-M conditions (24)-(26).

12. Prove now correctness for the **Case II.3** $T \leq P_c(i) \leq T + \frac{m_u}{2}$.

From (2) and (21),

$$P_s(i) = P_c(i) - RES + DEC = P_c(i) - P_c(i) mod\ m_u DEC. \quad (B60)$$

$$P_c(i) - P_c(i) mod\ m_u = k * m_u. \quad (B61)$$

$$P_s(i) = k * m_u + DEC. \quad (B62)$$

From (2), (26), (B61), and **Case II.3** condition, $T + \frac{m_u}{2} = k_3 * m_u + \frac{m_u}{2} \geq P_c(\mathrm{i}) = k * m_u + RES \geq \mathrm{T}$

$$= k_3 * m_u. \quad (B63)$$

From (B63),

$$k_3 * m_u \leq k * m_u + RES \leq (k_3 + 0.5) m_u < (k_3 + 1) m_u. \quad (B64)$$

From (B64),

$$k_3 \leq k < k_3 + 1. \quad (B65)$$

From (B65),

$$k = k_3. \quad (B66)$$

From (B30), (B62), (B66), and (26),

$$P_s(i) = k m_u + DEC = k_3 m_u + DEC = T + DEC \geq T. \quad (B67)$$

Thus, from (B67) and **Case II.3** condition, we have that both stego-pixel value, $P_s(i)$, and original cover pixel value, $P_c(i)$, are not less than *T*.

In the extraction algorithm, from (23), (25), (B30), and (B67),

$$RES' = P_s(i) mod\ m_u = (T + DEC) mod\ m_u = (k_3 * m_u + DEC) mod\ m_u = 0 + DEC\ mod\ m_u = DEC.$$

Thus, the result of extraction is the same as the secret embedded, and their binary representation is also the same, since *EC'=EC*, and the correctness for this item 12 is proved. Proof of item 12 needs usage of the condition (26) introduced in the RT-M. Note that in the Counterexample 2, condition (26) is violated.

Thus, in the RT-M Modification Correctness Theorem proof represented by items 1-12 above, only items 6, 7, 12 use conditions (24)-(26) introduced in the RT-M.

Thus, if any particular secret embedded into some cover pixel is extracted correctly that is proved in Items 1-12, then, if the sequence of the pixels used for extraction dictated by a PRNG is exactly the same as used for embedding then overall secret message extracted is the same as the embedded one. Thus, the theorem is fully proved.

References

[1] Wang SJ (2005) Steganography of capacity required using modulo operator for embedding secret image. Applied Mathematics and Computation 164:99–116

[2] Chang CC, Chan CS, Fan YH (2006) Image hiding scheme with modulus function and dynamic programming strategy on partitioned pixels. Pattern Recognition 39(6):1155-1167

[3] Chen WY (2007) Color image steganography scheme using set partitioning in hierarchical trees coding, digital Fourier transform and adaptive phase modulation. Applied Mathematics and Computation 185(1):432-448

[4] Lin SJ, Lin JC (2007) VCPSS: A two-in-one two-decoding-options image sharing method combining visual cryptography (VC) and polynomial-style sharing (PSS) approaches. Pattern Recognition 40(12):3652-3666

[5] Wang CM, Wu NI, Tsai CS, Hwang MS (2008) A high quality steganographic method with pixel-value differencing and modulus. Journal of Systems and Software 81(1):150-158

[6] Yang CH (2008) Inverted pattern approach to improve image quality of information hiding by LSB substitution. Pattern Recognition 41(8):2674-2683

[7] Chang YJ, Wang RZ, Lin JC (2008) Hiding images using modified search-order coding and modulus function. International Journal of Pattern Recognition and Artificial Intelligence 22(6):1215-1240

[8] Chang CC, Hsiehb YP, Lina CH (2008) Sharing secrets in stego images with authentication. Pattern Recognition 41(10):3130-3137

[9] Hsieha YP, Changa CC, Liua LJ (2008) A two-codebook combination and three-phase block matching based image-hiding scheme with high embedding capacity. Pattern Recognition 41(10):3104-3113

[10] Lin IC, Lin YB, Wang CM (2009) Hiding data in spatial domain images with distortion tolerance. Computer Standards & Interfaces 31(2):458-464

[11] Eslami Z, Razzaghi SH, Ahmadabadi JZ (2010) Secret image sharing based on cellular automata and steganography. Pattern Recognition 43(1):397-404

[12] Chang CC, Hsieh YP (2010) A chaotic map-based adaptive variable-size LSB method with pixel-value differencing and modulus. Image Science Journal 58(1):49-60

[13] Chen LST, Lin SJ, Lin JC (2010) Reversible JPEG-based hiding method with high hiding-ratio. International Journal of Pattern recognition and Artificial Intelligence 24(3):433-456

[14] Lou DC, Wu NI, Wang CM et al (2010) A novel adaptive steganography based on local complexity and human vision sensitivity. Journal of Systems and Software 83(7):1236-1248

[15] Yang CH, Weng CY, Wang SJ et al (2010) Varied PVD+LSB evading detection programs to spatial domain in data embedding systems. Journal of Systems and Software 83(10):1635-1643

[16] Wu NI, Fu KC, Wang CM (2010) A novel data hiding method for gray scale images based on pixel-value differencing and modulus function. Journal of Internet technology 11(7):1071-1081

[17] Liao X, Wen QY, Zhang J (2011) A steganographic method for digital images with four-pixel differencing and modified LSB substitution. Journal of Visual Communication and Image Representation 22(1):1-8

[18] Lin PY, Lee JS, Chang CC (2011) Protecting the content integrity of digital imagery with fidelity preservation. ACM Transaction on Multimedia Computing Communications and Applications 7(3):1-20

[19] Lin CC (2011) An information hiding scheme with minimal image distortion. Computer Standards & Interfaces 33(5):477-484

[20] Lou DC, Hu CH (2012) LSB steganographic method based on reversible histogram transformation function for resisting statistical steganalysis. Information Sciences 188:346-358

[21] Liu S, Chen Y, Jiang F et al (2012) A New Steganographic method based on equivalence class partition. Journal of Computational and Theoretical Nanoscience 9(10):1757-1765

[22] Sajasi S, Moghadam AME (2015) An adaptive image steganographic scheme based on noise visibility function and an optimal chaotic based encryption method. Applied Soft Computing 30:375-389

[23] Jana B (2016) High payload reversible data hiding scheme using weighted matrix. Optik 127(6):3347-3358

[24] Xu WL, Chang CC, Chen TS et al (2016) An improved least-significant-bit substitution method using the modulo three strategy. Displays 42:36-42

[25] Liu L, Chang CC, Wang A (2017) Data hiding based on extended turtle shell matrix construction method. Multimedia Tools and Applications 76(10):12233-12250

[26] Wu NI, Hwang MS (2017) A novel LSB data hiding scheme with the lowest distortion. The Imaging Science Journal 65(6):371-378

[27] Liu L, Chang CC, Anhong W (2017) Data hiding based on extended turtle shell matrix construction method. Multimedia Tools and Applications 76(10):12233-12250

[28] Jana B (2018) Reversible data hiding scheme using sub-sampled image exploiting Lagrange's interpolating polynomial. Multimedia Tools Applications 77:8805–8821

[29] Datta B, Roy S, S. Roy, Bandyopadhyay SM (2019) Multi-bit robust image steganography based on modular arithmetic. Multimedia Tools Applications 78(2):1511-1546

[30] Nashat D, Mamdouh L (2019) An efficient steganographic technique for hiding data. Journal of the Egyptian Mathematical Society 27(1):1-4

[31] Zenati A, Ouarda W, Alimi AM (2020) SSDIS-BEM: A New Signature Steganography Document Image System based on Beta Elliptic Modeling. Engineering Science and Technology, an International Journal 23(3):470-82

[32] Zenati A, Ouarda W, Alimi AM (2021) A new digital steganography system based on hiding online signature within document image data in YUV color space. Multimedia Tools and Applications 80(12):18653-7

[33] Ping P, Yang X, Zhang X, Mao Y, Khalid H (2022) Generating visually secure encrypted images by partial block pairing-substitution and semi-tensor product compressed sensing. Digital Signal Processing 120. https://doi.org/10.1016/j.dsp.2021.103263

[34] Ko HJ, Huang CT, Tseng H W. Wang SJ (2022) Efficient Cost-Reduced With High-Quality Image of Imperceptible Steganography Using Modulo and Magic Cube. IEEE Access 10:67686-67693. doi: 10.1109/ACCESS.2022.3185120

[35] Hu Y, Li X, Ma J (2022) A Novel LSB Matching Algorithm Based on Information Pre-Processing. Mathematics 2022, 10, 8. https://doi.org/10.3390/math10010008.

[36] Chan C-K, Cheng LM (2004) Hiding Data in Images by Simple LSB Substitution. Pattern Recognition 37, 469-474, doi:10.1016/j.patcog.2003.08.007

Alexander G. Chefranov received his Engineer in Applied Mathematics, PhD and Doctor of Engineering Sciences from Taganrog State Radio-Engineering University, Taganrog, Russia. Currently, he is working as Professor in the Department of Computer Engineering of Eastern Mediterranean University. His research interests include information security, parallel processing, distributed systems, real-time systems, scientific computing.

Gürcü Öz received her B.S, M.S. degrees from the Electrical and Electronic Engineering Department and Ph.D. degree from the Computer Engineering Department of Eastern Mediterranean University, in Famagusta, North Cyprus. Currently, she is working as an Associate Professor in the Department of Computer Engineering of Eastern Mediterranean University. Her research interests include computer networks, wireless ad hoc networks, distributed systems, cloud computing, system simulation, information security.