Security Analysis of a Color Image Encryption Scheme Based on Dynamic Substitution and Diffusion Operations

George Teşeleanu[1,2]

[1] Advanced Technologies Institute
10 Dinu Vintilă, Bucharest, Romania
tgeorge@dcti.ro
[2] Simion Stoilow Institute of Mathematics of the Romanian Academy
21 Calea Grivitei, Bucharest, Romania

Abstract. In 2019, Essaid *et al.* proposed an encryption scheme for color images based on chaotic maps. Their solution uses two enhanced chaotic maps to dynamically generate the secret substitution boxes and the key bytes used by the cryptosystem. Note that both types of parameters are dependent on the size of the original image. The authors claim that their proposal provides enough security for transmitting color images over unsecured channels. Unfortunately, this is not the case. In this paper, we introduce two cryptanalytic attacks for Essaid *et al.*'s encryption scheme. The first one is a chosen plaintext attack, which for a given size, requires 256 chosen plaintexts to allow an attacker to decrypt any image of this size. The second attack is a a chosen ciphertext attack, which compared to the first one, requires 512 chosen ciphertexts to break the scheme for a given size. These attacks are possible because the generated substitution boxes and key bits remain unchanged for different plaintext images.

1 Introduction

Because the use of social media is growing exponentially, the protection of digital images has become a sensitive topic. Therefore, the protection of images against theft and illegal distribution has attracted much attention. As a result, many researchers have proposed a variety of image encryption schemes. One of the most popular types of image encryption schemes are those based on chaotic maps, due to their high sensitivity to the previous states, initial conditions or both. This property makes them highly desirable because their sensitivity makes it difficult to predict their behaviour or outputs. Hence, novel chaos based cryptographic algorithms have been proposed over the years. We refer the reader to [7,22,24,40] for some surveys of such proposals. Unfortunately, insufficient security analysis and a lack of design guidelines have led to the discovery of serious security flaws in a substantial number of chaos based image encryption schemes. To illustrate our point we further present a list of broken schemes in Table 1. Note that the list is not comprehensive.

Scheme	[20]	[32]	[10]	[11]	[29]	[3]	[8]	[23]	[9]
Broken by	[31]	[2]	[35]	[1]	[34]	[8]	[14]	[13]	[38]
Scheme	[26]	[17]	[27]	[28]	[36]	[37]	[12]	[25]	[21]
Broken by	[30]	[19]	[33]	[39]	[4]	[18]	[6]	[15]	[16]

Table 1. Broken chaos based image encryption algorithms.

In [5] a chaos based encryption scheme is proposed. The authors use the Enhanced Logistic Map (ELM) and Enhanced Sine Map (ESM) as pseudorandom number generators (PRNGs). Using these two PRNGs, Essaid *et al.* randomly generate two substitution boxes (s-boxes), which are then used to compute the rest of the s-boxes required by the cryptosystem. Then, the PRNGs are combined to create the necessary key bytes. Since ELM and ESM are simply used as PRNGs and the scheme's weakness is independent of the employed generators, we omit their description and simply consider the two s-boxes and the key bytes as being randomly generated.

In this paper we conducted a security analysis on the Essaid *et al.* scheme. More precisely, we propose a chosen plaintext attack and a chosen ciphertext attack that allow an attacker to decrypt all images of a given size. In order to achieve this, we need the corresponding ciphertexts of 256 chosen plaintexts or the corresponding plaintexts of 512 chosen ciphertexts. For completeness, we also analysed the Essaid *et al.* scheme when all the s-boxes are randomly generated. This should be the most secure version of their scheme, since there are no relationships between the s-boxes that would allow an attacker to filter out the correct key. Unfortunately, even when the s-boxes are random, our proposed attacks succeed in recovering the encrypted images except two or four pixels, depending on the type of attack.

Structure of the paper. We provide the necessary preliminaries in Section 2. In Section 3 we show how an attacker can recover the private key and secret s-boxes in a chosen plaintext scenario. We also provide a key and s-boxes recovery attack in a chosen ciphertext attack in Section 4. We conclude in Section 5.

2 Preliminaries

Notations. In this paper, the subset $\{1, \ldots, s-1\} \in \mathbb{N}$ is denoted by $[1, s)$. The action of selecting a random element x from a sample space X is represented by $x \overset{\$}{\leftarrow} X$, while $x \leftarrow y$ indicates the assignment of value y to variable x. In the case of matrices, the $\leftarrow$ operator assigns the values position by position and the $=$ operator tests the equality between all positions of the two matrices. We use the C++ language operator & as reference to a variable. By H and W we denote an image's height and width. Also, we denote by $L = H + W - 1$. Hexadecimal numbers will always contain the prefix `0x`.

Algorithm 1: Encryption algorithm.

```
Input: A plaintext $p$, an s-box list $s$, a secret seed $seed$ and a secret key $k$
Output: A ciphertext $c$
for $i \in [0, H)$ and $j \in [0, W)$ do
    if $i = 0$ and $j = 0$ then $c_{0,0} \leftarrow (seed \oplus k_{0,0} + s_0[p_{0,0}]) \bmod 256$
    else if $j = 0$ then $c_{i,0} \leftarrow (c_{i-1,W-1} \oplus k_{i,0} + s_i[p_{i,0}]) \bmod 256$
    else $c_{i,j} \leftarrow (c_{i,j-1} \oplus k_{i,j} + s_{i+j}[p_{i,j}]) \bmod 256$
return $c$
```

Algorithm 2: Decryption algorithm.

```
Input: A ciphertext $c$, an inverse s-box list $s^{-1}$, a secret seed $seed$ and a
       secret key $k$
Output: A plaintext $p$
for $i \in [0, H)$ and $j \in [0, W)$ do
    if $i = 0$ and $j = 0$ then $p_{0,0} \leftarrow s_0^{-1}[(c_{0,0} - seed \oplus k_{0,0}) \bmod 256]$
    else if $j = 0$ then $p_{i,0} \leftarrow s_i^{-1}[(c_{i,0} - c_{i-1,W-1} \oplus k_{i,0}) \bmod 256]$
    else $p_{i,j} \leftarrow s_{i+j}^{-1}[(c_{i,j} - c_{i,j-1} \oplus k_{i,j}) \bmod 256]$
return $p$
```

2.1 Essaid *et al.* Image Encryption Scheme

In this section we present Essaid *et al.*'s encryption (Algorithm 1) and decryption (Algorithm 2) algorithms as described in [5]. We further consider two cases

- s-boxes s_0 and s_1 are randomly generated and the remaining ones are generated using Algorithm 3;
- all the s-boxes in list s are randomly generated.

The first version is according to the original paper [5], while the second one is introduced to show that the scheme remains broken even if the s-boxes are chosen at random. Note that the *seed* and the key bytes $k_{i,j}$ are randomly generated.

Algorithm 3: S-box table generator.

```
Input: Two s-boxs $s_0$ and $s_1$
Output: An s-box list $s$
for $i \in [2, L)$ and $j \in [0, 256)$ do $s_i[j] \leftarrow s_{i-2}[s_{i-1}[j]]$
return $s$
```

3 Chosen Plaintext Attack

In a chosen plaintext attack (CPA), the attacker A has temporary access to the encryption machine $\mathcal{O}_{enc}$ and can interrogate it on different inputs. Therefore,

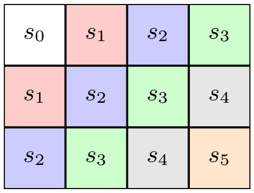
A constructs some plaintexts that are useful for his attack and then using $\mathcal{O}_{enc}$ obtains the corresponding ciphertexts.

We further show that Essaid *et al.*'s image encryption scheme is insecure in the chosen plaintext scenario, regardless of whether the generation method of the s-boxes is random or Algorithm 3 is used. The only difference between the two cases is the run time of the attack.

3.1 Randomly generated s-boxes

Before formally stating our attack, we first provide an example in order to provide the intuition behind our CPA attack.

Example 1. We further assume that we encrypt images of height 3 and width 4. We present in Figure 1 how Essaid *et al.*'s encryption algorithm (see Algorithm 1) uses the generated s-boxes. We can see that the algorithm uses the same s-box for each cell on a given minor diagonal.



s_0	s_1	s_2	s_3
s_1	s_2	s_3	s_4
s_2	s_3	s_4	s_5

Fig. 1. Used s-boxes in an image with $H = 3$ and $W = 4$.

Lets assume that the image I_{pv} we want to encrypt has the same pixel value pv everywhere. We further write $c_{i,j}[pv]$ for the ciphertext byte $c_{i,j}$ computed for I_{pv}. Then, if we write the ciphertext equations for the third minor diagonal we obtain

$$c_{0,2}[pv] \leftarrow (c_{0,1}[pv] \oplus k_{0,2} + s_2[pv]) \bmod 256 \tag{1}$$

$$c_{1,1}[pv] \leftarrow (c_{1,0}[pv] \oplus k_{1,1} + s_2[pv]) \bmod 256 \tag{2}$$

$$c_{2,0}[pv] \leftarrow (c_{1,3}[pv] \oplus k_{2,0} + s_2[pv]) \bmod 256. \tag{3}$$

From Equations (1) and (2) we derive

$$c_{0,2}[pv] - c_{1,1}[pv] \equiv (c_{0,1}[pv] \oplus k_{0,2} - c_{1,0}[pv] \oplus k_{1,1}) \bmod 256. \tag{4}$$

If we request $\mathcal{O}_{enc}$ the ciphertexts for all 256 images $I_0, \ldots, I_{255}$ we obtain 256 equations of type Equation (4). By checking all the values x and y that satisfy

$$c_{0,2}[pv] - c_{1,1}[pv] \equiv (c_{0,1}[pv] \oplus x - c_{1,0}[pv] \oplus y) \bmod 256,$$

for all pv values, we find the correct key pair $(k_{0,2}, k_{1,1})$ and an equivalent key pair $(k_{0,2} \oplus \texttt{0x80}, k_{1,1} \oplus \texttt{0x80})$. The second solution is derived from the following relations

$$\begin{aligned} c_{0,2}[pv] &\equiv (c_{0,1}[pv] \oplus k_{0,2} + 128 + s_2[pv] + 128) \bmod 256 \\ &\equiv (c_{0,1}[pv] \oplus k_{0,2} \oplus \texttt{0x80} + s_2[pv] \oplus \texttt{0x80}) \bmod 256 \\ c_{1,1}[pv] &\equiv (c_{1,0}[pv] \oplus k_{1,1} + 128 + s_2[pv] + 128) \bmod 256 \\ &\equiv (c_{1,0}[pv] \oplus k_{1,1} \oplus \texttt{0x80} + s_2[pv] \oplus \texttt{0x80}) \bmod 256 \end{aligned}$$

which lead to

$$c_{0,2}[pv] - c_{1,1}[pv] \equiv (c_{0,1}[pv] \oplus x \oplus \texttt{0x80} - c_{1,0}[pv] \oplus y \oplus \texttt{0x80}) \bmod 256.$$

After computing $k_{0,2}$, using Equation (1) we recover the correct s-box entries

$$s_2[pv] \leftarrow (c_{0,2}[pv] - c_{0,1}[pv] \oplus k_{0,2}) \bmod 256$$

and from Equation (3) we obtain

$$k_{2,0} = (c_{2,0}[0] - s_2[0] \bmod 256) \oplus c_{1,3}[0].$$

We also obtain an equivalent s-box $\tilde{s}_2$ from

$$\tilde{s}_2[pv] \leftarrow (c_{0,2}[pv] - c_{0,1}[pv] \oplus k_{0,2} \oplus \texttt{0x80}) \bmod 256$$

and from Equation (3) we obtain

$$k_{2,0} \oplus \texttt{0x80} = (c_{2,0}[0] - \tilde{s}_2[0] \bmod 256) \oplus c_{1,3}[0].$$

We can easily see that both the correct key bytes $k_{0,2}, k_{1,1}, k_{2,0}$ and s-box s_2, and the equivalent key bytes $k_{0,2} \oplus \texttt{0x80}, k_{1,1} \oplus \texttt{0x80}, k_{2,0} \oplus \texttt{0x80}$ and s-box $\tilde{s}_2$ can be used for decryption. Since we cannot determine the order of the correct and equivalent key bytes when we brute force x and y, we assume that the first solution we obtain is the correct one.

We repeat the same procedure for the second, forth and fifth minor diagonals. The only key bytes and s-boxes that we cannot recover using the above technique are $k_{0,0}$, $k_{2,3}$, s_0 and s_5.

Example 2. To illustrate the method presented in Example 1 we encrypted all I_{pv} using $seed = \texttt{0x08}$ and the parameters presented in Table 2. For simplicity we omitted the `0x` prefix.

In Table 3's first half we provide the key $\tilde{k}_0$ and the s-boxes $\tilde{s}_{0,1}, \ldots, \tilde{s}_{0,5}$ obtained from the first solutions of the equations of type Equation (4). For completeness, in the second half we present the key $\tilde{k}_1$ and the s-boxes $\tilde{s}_{1,1}, \ldots, \tilde{s}_{1,5}$ obtained from the second solutions.

The formal description of our chosen plaintext attack is provided in Algorithm 4 and is a generalization of the method presented in Example 1. For

k	21 c1 75 88 38 0c ad ad ef 62 84 d4
s_0	cd f0 a2 e2 ed ... ce 14 6a 3e fc
s_1	25 de 5a 72 bb ... 41 23 e5 10 a4
s_2	26 5c 73 1e 34 ... 49 c2 7b ca 46
s_3	bd dd 6d a0 4a ... e2 1c cc 75 06
s_4	62 fa ee 0f 1d ... f5 69 31 54 a1
s_5	b9 26 ec 68 65 ... c2 c3 5f 8f 6e

Table 2. Used secrets.

$\tilde{k}_0$	?? 41 75 08 b8 0c 2d 2d ef e2 04 ??
$\tilde{s}_{0,1}$	a5 5e da f2 3b ... c1 a3 65 90 24
$\tilde{s}_{0,2}$	26 5c 73 1e 34 ... 49 c2 7b ca 46
$\tilde{s}_{0,3}$	3d 5d ed 20 ca ... 62 9c 4c f5 86
$\tilde{s}_{0,4}$	e2 7a 6e 8f 9d ... 75 e9 b1 d4 21

$\tilde{k}_1$	?? c1 f5 88 38 8c ad ad 6f 62 84 ??
$\tilde{s}_{1,1}$	25 de 5a 72 bb ... 41 23 e5 10 a4
$\tilde{s}_{1,2}$	a6 dc f3 9e b4 ... c9 42 fb 4a c6
$\tilde{s}_{1,3}$	bd dd 6d a0 4a ... e2 1c cc 75 06
$\tilde{s}_{1,4}$	62 fa ee 0f 1d ... f5 69 31 54 a1

Table 3. Computed secrets.

completeness, in Algorithm 4 we compute two possible key, s-boxes pairs just as in Example 2. Note that in order to be able to use our attack we must have $H, W \geq 2$.

The complexity of Algorithm 4 is $\mathcal{O}(2^{24}L + HW)$ and we need 256 oracle queries. For example, if we encrypt 2 megapixels[3] images we obtain that the complexity of Algorithm 4 is $\mathcal{O}(2^{24} \cdot 2^{11.45} + 2^{20.87}) = \mathcal{O}(2^{35.45})$. In the case of 12 megapixels[4], we obtain $\mathcal{O}(2^{24} \cdot 2^{12.77} + 2^{23.5}) = \mathcal{O}(2^{36.77})$.

3.2 Essaid *et al.*'s generation method for s-boxes

As in the previous subsection, we begin with an example.

Example 3. Compared to Examples 1 and 2, in the case of Essaid *et al.*'s generation method it is enough to compute $\tilde{s}_{0,1}$, $\tilde{s}_{1,1}$, $\tilde{s}_{0,2}$, $\tilde{s}_{1,2}$, $\tilde{s}_{0,3}$ and $\tilde{s}_{1,3}$ and the remaining s-boxes can be easily deduced using Algorithm 3. Note that in this case we can also compute s_0 and s_5.

The first thing that we do after deducing $\hat{s}_{0,1}$, $\hat{s}_{1,1}$, $\hat{s}_{0,2}$, $\hat{s}_{1,2}$ from the 256 encrypted images is to compute $\hat{s}_{0,1} \circ \hat{s}_{0,2}$, $\hat{s}_{0,1} \circ \hat{s}_{1,2}$, $\hat{s}_{1,1} \circ \hat{s}_{0,2}$, $\hat{s}_{1,1} \circ \hat{s}_{1,2}$ and check which one coincides with $\hat{s}_{0,3}$ or $\hat{s}_{1,3}$. This leads to two possible combinations, one for $\hat{s}_{0,3}$ and one for $\hat{s}_{1,3}$. We denote the first combination by $\tilde{s}_{0,1}, \tilde{s}_{0,2}$ and $\tilde{s}_{0,3}$ and the second one by $\tilde{s}_{1,1}, \tilde{s}_{1,2}$ and $\tilde{s}_{1,3}$.

[3] $W \times H = 1600 \times 1200$

[4] $W \times H = 4000 \times 3000$

Algorithm 4: CPA attack (randomly generated).

```
for $t \in [0, 256)$ do
    for $i \in [0, H)$ and $j \in [0, W)$ do $p_{i,j} \leftarrow t$
    Send the plaintext $p$ to the encryption oracle $\mathcal{O}_{enc}$.
    Receive the ciphertext $\bar{c}$ from the encryption oracle $\mathcal{O}_{enc}$.
    $c_t \leftarrow \bar{c}$
for $j \in [1, L-1)$ do
    $\alpha \leftarrow \max(0, j-(W-1))$; $\beta \leftarrow \min(j, H-1)$; $pos \leftarrow 0$
    for $x \in [0, 256)$ and $y \in [0, 256)$ do
        $ctr \leftarrow 0$
        for $t \in [0, 256)$ do
            $f \leftarrow c_{t,\alpha,j-\alpha} - c_{t,\alpha+1,j-(\alpha+1)} \bmod 256$
            if $j \neq \alpha+1$ then $g \leftarrow (c_{t,\alpha,j-\alpha-1} \oplus x - c_{t,\alpha+1,j-\alpha-2} \oplus y) \bmod 256$
            else $g \leftarrow (c_{t,\alpha,j-\alpha-1} \oplus x - c_{t,\alpha,W-1} \oplus y) \bmod 256$
            if $f = g$ then $ctr \leftarrow ctr + 1$
        if $ctr = 256$ then
            $\tilde{k}_{pos,\alpha,j-\alpha} \leftarrow x$; $\tilde{k}_{pos,\alpha+1,j-(\alpha+1)} \leftarrow y$
            for $t \in [0, 256)$ do $\tilde{s}_{pos,j}[t] \leftarrow (c_{t,\alpha,j-\alpha} - c_{t,\alpha,j-\alpha-1} \oplus x) \bmod 256$
            for $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,j}[\tilde{s}_{pos,j}[t]] \leftarrow t$
            for $t \in [\alpha+2, \beta+1)$ do
                $\tilde{k}_{pos,t,j-t} \leftarrow ((c_{0,t,j-t} - \tilde{s}_{pos,j}[0]) \bmod 256) \oplus c_{0,t,j-t-1}$
            $pos \leftarrow pos + 1$
return $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$
```

Let $pos \in [0, 2)$. After computing $\tilde{s}_{pos,1}, \tilde{s}_{pos,2}$ and $\tilde{s}_{pos,3}$, the remaining s-boxes can be calculated as follows: $\tilde{s}_{pos,4} = \tilde{s}_{pos,3} \circ \tilde{s}_{pos,2}$, $\tilde{s}_{pos,5} = \tilde{s}_{pos,4} \circ \tilde{s}_{pos,5}$ and $\tilde{s}_{pos,0} = \tilde{s}_{pos,2} \circ \tilde{s}^{-1}_{pos,1}$. Once all the s-boxes are known, we can easily compute the key $\tilde{k}_{pos}$.

We remark that only one of the two solutions is the correct one. We can see that from the following relation

$$\tilde{s}_3[i] = s_3[i] \oplus \texttt{0x80} = s_2[s_1[i]] \oplus \texttt{0x80} = \tilde{s}_2[s_1[i]].$$

Since $\tilde{s}_3$ is not generated by $\tilde{s}_2[\tilde{s}_1[i]]$, we do not obtain the equivalent key, s-boxes pair, and thus one of the solutions will not decrypt images correctly.

The formal description of our chosen plaintext attack is provided in Algorithm 5 and is a generalization of the method presented in Example 3. The complexity of Algorithm 5 is $\mathcal{O}(2^{24} + 2^8 L + HW)$ and we need 256 oracle queries. For example, if we encrypt 2 megapixels images we obtain that the complexity of Algorithm 5 is $\mathcal{O}(2^{24} + 2^8 \cdot 2^{11.45} + 2^{20.87}) = \mathcal{O}(2^{24.21})$. In the case of 12 megapixels, we obtain $\mathcal{O}(2^{24} + 2^8 \cdot 2^{12.77} + 2^{23.5}) = \mathcal{O}(2^{24.85})$.

Note that according to [5] the security of their scheme is $\mathcal{O}(2^{128})$. Using 256 encrypted images and Algorithm 5, we lower the security strength of Essaid *et al.*'s scheme from 128 bits to approximately 24 bits.

Algorithm 5: CPA attack (Essaid *et al.*'s generation method).

```
for $j \in [1, 4)$ do
    $\alpha \leftarrow \max(0, j - (W - 1))$; $pos \leftarrow 0$
    for $x \in [0, 256)$ and $y \in [0, 256)$ do
        $ctr \leftarrow 0$
        for $t \in [0, 256)$ do
            $f \leftarrow c_{t,\alpha,j-\alpha} - c_{t,\alpha+1,j-(\alpha+1)} \bmod 256$
            if $j \neq \alpha + 1$ then $g \leftarrow (c_{t,\alpha,j-\alpha-1} \oplus x - c_{t,\alpha+1,j-\alpha-2} \oplus y) \bmod 256$
            else $g \leftarrow (c_{t,\alpha,j-\alpha-1} \oplus x - c_{t,\alpha,W-1} \oplus y) \bmod 256$
            if $f = g$ then $ctr \leftarrow ctr + 1$
        if $ctr = 256$ then
            $\tilde{x}_{pos,j-1} \leftarrow x$; $\tilde{y}_{pos,j-1} \leftarrow y$
            for $t \in [0, 256)$ do
                $\hat{s}_{pos,j-1}[t] \leftarrow (c_{t,\alpha,j-\alpha} - c_{t,\alpha,j-\alpha-1} \oplus x) \bmod 256$
            $pos \leftarrow pos + 1$
for $i \in [0, 2)$ and $j \in [0, 2]$ do
    for $t \in [0, 256)$ do $f = \hat{s}_{i,0}[\hat{s}_{j,1}[t]]$
    if $f = \hat{s}_{0,2}$ then $\tilde{s}_{0,1} \leftarrow \hat{s}_{i,0}$; $\tilde{s}_{0,2} \leftarrow \hat{s}_{j,1}$; $\tilde{s}_{0,3} \leftarrow \hat{s}_{0,2}$
    if $f = \hat{s}_{1,2}$ then $\tilde{s}_{1,1} \leftarrow \hat{s}_{i,0}$; $\tilde{s}_{1,2} \leftarrow \hat{s}_{j,1}$; $\tilde{s}_{1,3} \leftarrow \hat{s}_{1,2}$
for $pos \in [0, 2)$ do
    for $j \in [4, L)$ and $t \in [0, 256)$ do $\tilde{s}_{pos,j}[t] \leftarrow \tilde{s}_{pos,j-2}[\tilde{s}_{pos,j-1}[t]]$
    for $j \in [1, L)$ and $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,j}[\tilde{s}_{pos,j}[t]] \leftarrow t$
    for $t \in [0, 256)$ do $\tilde{s}_{pos,0}[t] \leftarrow \tilde{s}_{pos,2}[\tilde{s}^{-1}_{pos,1}[t]]$
    for $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,0}[\tilde{s}_{pos,0}[t]] \leftarrow t$
    for $i \in [0, H)$ and $j \in [0, W)$ do
        if $i = 0$ and $j = 0$ then $\tilde{k}_{pos,0,0} = (c_{0,0,0} - \tilde{s}_{pos,0}[0]) \bmod 256$
        else if $j = 0$ then $\tilde{k}_{pos,i,0} = (c_{0,i,0} - \tilde{s}_{pos,i}[0]) \bmod 256$
        else $\tilde{k}_{pos,i,j} = (c_{0,i,j} - \tilde{s}_{pos,i+j}[0]) \bmod 256$
return $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$
```

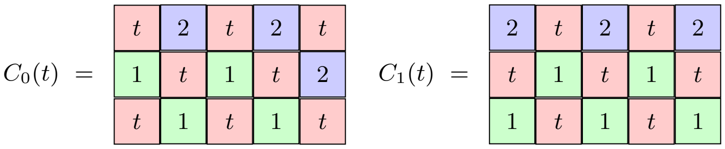
4 Chosen Ciphertext Attack

Compared to the chosen plaintext attack, in a chosen ciphertext attack (CCA), A has temporary access to the decryption machine $\mathcal{O}_{dec}$. Therefore, A constructs some ciphertexts that are useful for his attack and then using $\mathcal{O}_{dec}$ obtains the corresponding plaintexts.

In this scenario, we provide two types of attacks against Essaid *et al.*'s cryptosystem, one for images with odd width and one for images with even width. Again the s-box generation method is irrelevant to the success of the attacks, the only thing that is affected is their run time.

4.1 Randomly generated s-boxes

We first provide an example for images with odd width and then one for images with even width. After, we present the formal description of our CCA attack.



$$C_0(t) = \begin{array}{|c|c|c|c|c|} \hline t & 2 & t & 2 & t \\ \hline 1 & t & 1 & t & 2 \\ \hline t & 1 & t & 1 & t \\ \hline \end{array} \qquad C_1(t) = \begin{array}{|c|c|c|c|c|} \hline 2 & t & 2 & t & 2 \\ \hline t & 1 & t & 1 & t \\ \hline 1 & t & 1 & t & 1 \\ \hline \end{array}$$

Fig. 2. Ciphertext patterns for $H = 3$ and $W = 5$.

Example 4. We further assume that we decrypt images of height 3 and width 5. In the first part of the attack we use ciphertexts of type $C_0(t)$ (see Figure 2). If we explicit the relations for the forth minor diagonal we obtain

$$s_3[p_{0,3}[t]] \leftarrow (2 - t \oplus k_{0,3}) \bmod 256 \tag{5}$$

$$s_3[p_{1,2}[t]] \leftarrow (1 - t \oplus k_{1,2}) \bmod 256 \tag{6}$$

$$s_3[p_{2,1}[t]] \leftarrow (1 - t \oplus k_{2,1}) \bmod 256. \tag{7}$$

Since we consider all t values, in Equations (5) and (6) permutation s_3 iterates through all its values. Therefore, all we need is to find the t values for which $p_{0,3} = p_{1,2}$. Therefore, we define

$$tab_0[p_{0,3}[t]] \leftarrow t \quad \text{and} \quad tab_1[p_{1,2}[t]] \leftarrow t,$$

and using Equations (5) and (6) we obtain

$$(2 - tab_0[i] \oplus k_{0,3}) \equiv (1 - tab_1[i] \oplus k_{1,2}) \bmod 256,$$

for all i values. By checking all the values x and y that satisfy

$$(2 - tab_0[i] \oplus x) \equiv (1 - tab_1[i] \oplus y) \bmod 256,$$

for all i values, we find the correct key pair $(k_{0,3}, k_{1,2})$ and the equivalent key pair $(k_{0,3} \oplus \texttt{0x80}, k_{1,2} \oplus \texttt{0x80})$. As in Example 1, we consider that the first solution is the correct one. Once the first two key bytes are known, we can easily compute the third s-box using Equation (5) and the third key byte using Equation (7).

We repeat the process for the second and sixth minor diagonals. In the second part of our attack, we use ciphertexts of type $C_1(t)$ (see Figure 2) and then we use the same procedure as before for the third and fifth minor diagonal. The only key bytes and s-boxes that we cannot recover using the above technique are $k_{0,0}$, $k_{2,4}$, s_0 and s_6.

Example 5. For the even width case, we consider images of height 3 and width 6. In this case we use the same procedure as in Example 4, but instead of using $C_0(t)$ and $C_1(t)$ ciphertext patterns, we use $C_2(t)$ and $C_3(t)$ (see Figure 3). The only difference between the even and odd width cases is that in the even case we cannot recover $k_{0,1}$, $k_{1,0}$ and s_1. This is because in the even case we could not put t on the last cell in the first line of $C_2(t)$ without interfering with the recovery of the other bytes and s-boxes.

$$C_2(t) = \begin{array}{|c|c|c|c|c|c|}\hline t & 2 & t & 2 & t & 2 \\ \hline 1 & t & 1 & t & 1 & t \\ \hline t & 1 & t & 1 & t & 2 \\ \hline \end{array} \qquad C_3(t) = \begin{array}{|c|c|c|c|c|c|}\hline 2 & t & 2 & t & 2 & t \\ \hline t & 1 & t & 1 & t & 2 \\ \hline 1 & t & 1 & t & 1 & t \\ \hline \end{array}$$

Fig. 3. Ciphertext patterns for $H = 3$ and $W = 6$.

Note that we can recover the two key bytes and one s-box if we create one additional ciphertext pattern that satisfies the previously stated condition (see Figure 4). However, we have to ask 256 more oracle queries. Therefore, we decided that is more practical to lose the ability to decrypt two extra bytes, than to ask the additional queries, since images can still be recognized without them. For example, an emoji has a resolution of 32×32 and removing the four pixels it does not affect the informational content.

$$C_4(t) = \begin{array}{|c|c|c|c|c|c|}\hline t & 2 & 3 & 3 & 3 & t \\ \hline 1 & 3 & 3 & 3 & 3 & 3 \\ \hline 3 & 3 & 3 & 3 & 3 & 3 \\ \hline \end{array}$$

Fig. 4. Additional ciphertext pattern for $H = 3$ and $W = 6$.

The formal description of our chosen ciphertext attack is provided in Algorithm 7 and is a generalization of the methods presented in Examples 4 and 5. For completeness, in Algorithm 7 we compute two possible key, s-boxes pairs just as in Algorithm 4. Note that in order to be able to use our attack we must have $H \geq 2$ and $W \geq 3$.

The complexity of Algorithm 7 is the same as the complexity of Algorithm 4, namely $\mathcal{O}(2^{24}L + HW)$. The only difference between the two attacks is that in the case of Algorithm 7 we need 512 decryption oracle queries.

4.2 Essaid *et al.*'s generation method for s-boxes

In the case of Essaid *et al.*'s generation method is enough to compute three s-boxes using the ideas from Examples 4 and 5. Then using similar techniques as the ones from Example 3 we can recover all the s-boxes, and implicitly all the key bytes.

The formal description of our chosen ciphertext attack is provided in Algorithm 9. The complexity of Algorithm 9 is the same as the complexity of Algorithm 5, namely $\mathcal{O}(2^{24} + 2^8 L + HW)$. The only difference is that in the case of Algorithm 9 we need 512 oracle queries.

Algorithm 6: Helper functions.

```
Function $choose\_plaintext(parity)$
    for $t \in [0, 256)$ do
        for $i \in [0, H)$ and $j \in [0, W)$ do
            $\alpha \leftarrow \max(0, i + j - (W - 1))$
            if $i + j \equiv parity \bmod 2$ then $c_{i,j} \leftarrow t$
            else if $i = \alpha$ then $c_{i,j} \leftarrow 2$
            else $c_{i,j} \leftarrow 1$
        Send the ciphertext $c$ to the decryption oracle $\mathcal{O}_{dec}$.
        Receive the plaintext $\bar{p}$ from the decryption oracle $\mathcal{O}_{dec}$.
        $p_t \leftarrow \bar{p}$
    return $p_t$
Function $partial\_attack(low, upp, p_t, \&\tilde{k}, \&\tilde{s}, \&\tilde{s}^{-1})$
    for $j \in [low, upp)$ and at each step increment $j$ with $2$ do
        $\alpha \leftarrow \max(0, j - (W - 1))$; $\beta \leftarrow \min(j, H - 1)$; $pos \leftarrow 0$
        for $t \in [0, 256)$ do $tab_0[p_{t,\alpha,j-\alpha}] = t$; $tab_1[p_{t,\alpha+1,j-(\alpha+1)}] = t$
        for $x \in [0, 256)$ and $y \in [0, 256)$ do
            $ctr \leftarrow 0$
            for $t \in [0, 256)$ do
                $f \leftarrow (2 - tab_0[t] \oplus x) \bmod 256$
                $g \leftarrow (1 - tab_1[t] \oplus y) \bmod 256$
                if $f = g$ then $ctr \leftarrow ctr + 1$
            if $ctr = 256$ then
                $\tilde{k}_{pos,\alpha,j-\alpha} \leftarrow x$; $\tilde{k}_{pos,\alpha+1,j-(\alpha+1)} \leftarrow y$
                for $t \in [0, 256)$ do $\tilde{s}_{pos,j}[t] \leftarrow (2 - tab_0[t] \oplus x) \bmod 256$
                for $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,j}[\tilde{s}_{pos,j}[t]] \leftarrow t$
                for $t \in [\alpha + 2, \beta + 1)$ do
                    $\tilde{k}_{pos,t,j-t} \leftarrow ((c_{t,j-t} - \tilde{s}_{pos,j}[p_{255,t,j-t}) \bmod 256) \oplus c_{t,j-t-1}$
                $pos \leftarrow pos + 1$
```

Algorithm 7: CCA attack (randomly generated).

```
Function $main\_odd()$
    $p_t \leftarrow choose\_plaintext(0)$
    $partial\_attack(1, L - 1, p_t, \tilde{k}, \tilde{s}, \tilde{s}^{-1})$
    $p_t \leftarrow choose\_plaintext(1)$
    $partial\_attack(2, L - 1, p_t, \tilde{k}, \tilde{s}, \tilde{s}^{-1})$
    return $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$
Function $main\_even()$
    $p_t \leftarrow choose\_plaintext(0)$
    $partial\_attack(3, L - 1, p_t, \tilde{k}, \tilde{s}, \tilde{s}^{-1})$
    $p_t \leftarrow choose\_plaintext(1)$
    $partial\_attack(2, L - 1, p_t, \tilde{k}, \tilde{s}, \tilde{s}^{-1})$
    return $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$
```

Similar to the case of the CPA attack, using 512 decrypted images and Algorithm 9, we managed to lower the security strength of Essaid *et al.*'s scheme from 128 bits to approximately 24 bits.

Algorithm 8: More helper functions.

```
**Function** $partial\_attack(low, upp, p_t, flag, \&\tilde{k}, \&\tilde{s}, \&\tilde{s}^{-1})$
    **for** $j \in [low, upp)$ *and at each step increment* $j$ *with* $2$ **do**
        $\alpha \leftarrow \max(0, j - (W-1))$; $pos \leftarrow 0$
        **for** $t \in [0, 256)$ **do** $tab_0[p_{t,\alpha,j-\alpha}] = t$; $tab_1[p_{t,\alpha+1,j-(\alpha+1)}] = t$
        **for** $x \in [0, 256)$ *and* $y \in [0, 256)$ **do**
            $ctr \leftarrow 0$
            **for** $t \in [0, 256)$ **do**
                $f \leftarrow (2 - tab_0[t] \oplus x) \bmod 256$
                $g \leftarrow (1 - tab_1[t] \oplus y) \bmod 256$
                **if** $f = g$ **then** $ctr \leftarrow ctr + 1$
            **if** $ctr = 256$ **then**
                $\tilde{x}_{pos,j-1-flag} \leftarrow x$; $\tilde{y}_{pos,j-1-flag} \leftarrow y$
                **for** $t \in [0, 256)$ **do** $\hat{s}_{pos,j-1-flag}[t] \leftarrow (2 - tab_0[t] \oplus x) \bmod 256$
                $pos \leftarrow pos + 1$
**Function** $extract\_key(pos, \&\tilde{k})$
    **for** $i \in [0, H)$ *and* $j \in [0, W)$ **do**
        **if** $i = 0$ *and* $j = 0$ **then** $\tilde{k}_{pos,0,0} = (c_{0,0,0} - \tilde{s}_{pos,0}[p_{255,0,0}]) \bmod 256$
        **else if** $j = 0$ **then** $\tilde{k}_{pos,i,0} = (c_{0,i,0} - \tilde{s}_{pos,i}[p_{255,i,0}]) \bmod 256$
        **else** $\tilde{k}_{pos,i,j} = (c_{0,i,j} - \tilde{s}_{pos,i+j}[p_{255,i,j}]) \bmod 256$
```

5 Conclusions

In [5], the authors described an image encryption scheme that they claimed provided a security strength of 128 bits. Unfortunately, in this paper we showed that the actual security strength of Essaid *et al.*'s scheme is roughly 24 bits. To achieve our security bound, we devised a chosen plaintext attack which needs 256 queries to the encryption oracle. We also describe a chosen ciphertext attack which needs 512 queries to the decryption oracle and has a complexity of $\mathcal{O}(2^{24})$. For completeness, we also show how to attack Essaid *et al.*'s cryptosystem when all its s-boxes are randomly generated. In this case, using our CPA or CCA attacks, we lower the security strength to 36 bits.

Algorithm 9: CCA attack (Essaid *et al.*'s generation method).

```
Function main_odd()
    $p_t \leftarrow choose\_plaintext(0)$
    $partial\_attack(1, 4, p_t, 0, \tilde{k}, \tilde{s}, \tilde{s}^{-1})$
    $p_t \leftarrow choose\_plaintext(1)$
    $partial\_attack(2, 3, p_t, 0, \tilde{k}, \tilde{s}, \tilde{s}^{-1})$
    for $i \in [0, 2)$ and $j \in [0, 2]$ do
        for $t \in [0, 256)$ do $f = \hat{s}_{i,0}[\hat{s}_{j,1}[t]]$
        if $f = \hat{s}_{0,2}$ then $\tilde{s}_{0,1} \leftarrow \hat{s}_{i,0}$; $\tilde{s}_{0,2} \leftarrow \hat{s}_{j,1}$; $\tilde{s}_{0,3} \leftarrow \hat{s}_{0,2}$
        if $f = \hat{s}_{1,2}$ then $\tilde{s}_{1,1} \leftarrow \hat{s}_{i,0}$; $\tilde{s}_{1,2} \leftarrow \hat{s}_{j,1}$; $\tilde{s}_{1,3} \leftarrow \hat{s}_{1,2}$
    for $pos \in [0, 2)$ do
        for $j \in [4, L)$ and $t \in [0, 256)$ do $\tilde{s}_{pos,j}[t] \leftarrow \tilde{s}_{pos,j-2}[\tilde{s}_{pos,j-1}[t]]$
        for $j \in [1, L)$ and $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,j}[\tilde{s}_{pos,j}[t]] \leftarrow t$
        for $t \in [0, 256)$ do $\tilde{s}_{pos,0}[t] \leftarrow \tilde{s}_{pos,2}[\tilde{s}^{-1}_{pos,1}[t]]$
        for $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,0}[\tilde{s}_{pos,0}[t]] \leftarrow t$
        $extract\_key(pos, \tilde{k})$
    return $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$
Function main_even()
    $p_t \leftarrow choose\_plaintext(0)$
    $partial\_attack(3, 4, p_t, 1, \tilde{k}, \tilde{s}, \tilde{s}^{-1})$
    $p_t \leftarrow choose\_plaintext(1)$
    $partial\_attack(2, 5, p_t, 1, \tilde{k}, \tilde{s}, \tilde{s}^{-1})$
    for $i \in [0, 2)$ and $j \in [0, 2]$ do
        for $t \in [0, 256)$ do $f = \hat{s}_{i,0}[\hat{s}_{j,1}[t]]$
        if $f = \hat{s}_{0,2}$ then $\tilde{s}_{0,2} \leftarrow \hat{s}_{i,0}$; $\tilde{s}_{0,3} \leftarrow \hat{s}_{j,1}$; $\tilde{s}_{0,4} \leftarrow \hat{s}_{0,2}$
        if $f = \hat{s}_{1,2}$ then $\tilde{s}_{1,2} \leftarrow \hat{s}_{i,0}$; $\tilde{s}_{1,3} \leftarrow \hat{s}_{j,1}$; $\tilde{s}_{1,4} \leftarrow \hat{s}_{1,2}$
    for $pos \in [0, 2)$ do
        for $j \in [5, L)$ and $t \in [0, 256)$ do $\tilde{s}_{pos,j}[t] \leftarrow \tilde{s}_{pos,j-2}[\tilde{s}_{pos,j-1}[t]]$
        for $j \in [2, L)$ and $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,j}[\tilde{s}_{pos,j}[t]] \leftarrow t$
        for $t \in [0, 256)$ do $\tilde{s}_{pos,1}[t] \leftarrow \tilde{s}_{pos,3}[\tilde{s}^{-1}_{pos,2}[t]]$
        for $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,1}[\tilde{s}_{pos,1}[t]] \leftarrow t$
        for $t \in [0, 256)$ do $\tilde{s}_{pos,0}[t] \leftarrow \tilde{s}_{pos,2}[\tilde{s}^{-1}_{pos,1}[t]]$
        for $t \in [0, 256)$ do $\tilde{s}^{-1}_{pos,0}[\tilde{s}_{pos,0}[t]] \leftarrow t$
        $extract\_key(pos, \tilde{k})$
    return $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$
```

References

1. Alanazi, A.S., Munir, N., Khan, M., Asif, M., Hussain, I.: Cryptanalysis of Novel Image Encryption Scheme Based on Multiple Chaotic Substitution Boxes. IEEE Access **9**, 93795–93802 (2021)
2. Arroyo, D., Diaz, J., Rodriguez, F.: Cryptanalysis of a One Round Chaos-Based Substitution Permutation Network. Signal Processing **93**(5), 1358–1364 (2013)
3. Chen, J.x., Zhu, Z.l., Fu, C., Zhang, L.b., Zhang, Y.: An Efficient Image Encryption Scheme Using Lookup Table-Based Confusion and Diffusion. Nonlinear Dynamics

81(3), 1151–1166 (2015)
4. Chen, J., Chen, L., Zhou, Y.: Cryptanalysis of a DNA-Based Image Encryption Scheme. Information Sciences **520**, 130–141 (2020)
5. Essaid, M., Akharraz, I., Saaidi, A., Mouhib, A.: A New Approach of Image Encryption Based on Dynamic Substitution and Diffusion Operations. In: SysCoBIoTS 2019. pp. 1–6. IEEE (2019)
6. Fan, H., Zhang, C., Lu, H., Li, M., Liu, Y.: Cryptanalysis of a New Chaotic Image Encryption Technique Based on Multiple Discrete Dynamical Maps. Entropy **23**(12), 1581 (2021)
7. Hosny, K.M.: Multimedia Security Using Chaotic Maps: Principles and Methodologies, vol. 884. Springer (2020)
8. Hu, G., Xiao, D., Wang, Y., Li, X.: Cryptanalysis of a Chaotic Image Cipher using Latin Square-Based Confusion and Diffusion. Nonlinear Dynamics **88**(2), 1305–1316 (2017)
9. Hua, Z., Zhou, Y.: Design of Image Cipher Using Block-Based Scrambling and Image Filtering. Information sciences **396**, 97–113 (2017)
10. Huang, X., Sun, T., Li, Y., Liang, J.: A Color Image Encryption Algorithm Based on a Fractional-Order Hyperchaotic System. Entropy **17**(1), 28–38 (2014)
11. Khan, M.: A Novel Image Encryption Scheme Based on Multiple Chaotic S-Boxes. Nonlinear Dynamics **82**(1), 527–533 (2015)
12. Khan, M., Masood, F.: A Novel Chaotic Image Encryption Technique Based on Multiple Discrete Dynamical Maps. Multimedia Tools and Applications **78**(18), 26203–26222 (2019)
13. Li, M., Lu, D., Wen, W., Ren, H., Zhang, Y.: Cryptanalyzing a Color Image Encryption Scheme Based on Hybrid Hyper-Chaotic System and Cellular Automata. IEEE access **6**, 47102–47111 (2018)
14. Li, M., Lu, D., Xiang, Y., Zhang, Y., Ren, H.: Cryptanalysis and Improvement in a Chaotic Image Cipher Using Two-Round Permutation and Diffusion. Nonlinear Dynamics **96**(1), 31–47 (2019)
15. Li, M., Wang, P., Liu, Y., Fan, H.: Cryptanalysis of a Novel Bit-Level Color Image Encryption Using Improved 1D Chaotic Map. IEEE Access **7**, 145798–145806 (2019)
16. Li, M., Wang, P., Yue, Y., Liu, Y.: Cryptanalysis of a Secure Image Encryption Scheme Based on a Novel 2D Sine–Cosine Cross-Chaotic Map. Journal of Real-Time Image Processing **18**(6), 2135–2149 (2021)
17. Liu, L., Hao, S., Lin, J., Wang, Z., Hu, X., Miao, S.: Image Block Encryption Algorithm Based on Chaotic Maps. IET Signal Processing **12**(1), 22–30 (2018)
18. Liu, Y., Qin, Z., Liao, X., Wu, J.: Cryptanalysis and Enhancement of an Image Encryption Scheme Based on a 1-D Coupled Sine Map. Nonlinear Dynamics **100**(3), 2917–2931 (2020)
19. Ma, Y., Li, C., Ou, B.: Cryptanalysis of an Image Block Encryption Algorithm Based on Chaotic Maps. Journal of Information Security and Applications **54**, 102566 (2020)
20. Matoba, O., Javidi, B.: Secure Holographic Memory by Double-Random Polarization Encryption. Applied Optics **43**(14), 2915–2919 (2004)
21. Mondal, B., Behera, P.K., Gangopadhyay, S.: A Secure Image Encryption Scheme Based on a Novel 2D Sine–Cosine Cross-Chaotic (SC3) Map. Journal of Real-Time Image Processing **18**(1), 1–18 (2021)
22. Muthu, J.S., Murali, P.: Review of Chaos Detection Techniques Performed on Chaotic Maps and Systems in Image Encryption. SN Computer Science **2**(5), 1–24 (2021)

23. Niyat, A.Y., Moattar, M.H., Torshiz, M.N.: Color Image Encryption Based on Hybrid Hyper-Chaotic System and Cellular Automata. Optics and Lasers in Engineering **90**, 225–237 (2017)
24. Özkaynak, F.: Brief Review on Application of Nonlinear Dynamics in Image Encryption. Nonlinear Dynamics **92**(2), 305–313 (2018)
25. Pak, C., An, K., Jang, P., Kim, J., Kim, S.: A Novel Bit-Level Color Image Encryption Using Improved 1D Chaotic Map. Multimedia Tools and Applications **78**(9), 12027–12042 (2019)
26. Pak, C., Huang, L.: A New Color Image Encryption Using Combination of the 1D Chaotic Map. Signal Processing **138**, 129–137 (2017)
27. Shafique, A., Shahid, J.: Novel Image Encryption Cryptosystem Based on Binary Bit Planes Extraction and Multiple Chaotic Maps. The European Physical Journal Plus **133**(8), 1–16 (2018)
28. Sheela, S., Suresh, K., Tandur, D.: Image Encryption Based on Modified Henon Map Using Hybrid Chaotic Shift Transform. Multimedia Tools and Applications **77**(19), 25223–25251 (2018)
29. Song, C., Qiao, Y.: A Novel Image Encryption Algorithm Based on DNA Encoding and Spatiotemporal Chaos. Entropy **17**(10), 6954–6968 (2015)
30. Wang, H., Xiao, D., Chen, X., Huang, H.: Cryptanalysis and Enhancements of Image Encryption Using Combination of the 1D Chaotic Map. Signal processing **144**, 444–452 (2018)
31. Wang, L., Wu, Q., Situ, G.: Chosen-Plaintext Attack on the Double Random Polarization Encryption. Optics Express **27**(22), 32158–32167 (2019)
32. Wang, X., Teng, L., Qin, X.: A Novel Colour Image Encryption Algorithm Based on Chaos. Signal Processing **92**(4), 1101–1108 (2012)
33. Wen, H., Yu, S.: Cryptanalysis of an Image Encryption Cryptosystem Based on Binary Bit Planes Extraction and Multiple Chaotic Maps. The European Physical Journal Plus **134**(7), 1–16 (2019)
34. Wen, H., Yu, S., Lü, J.: Breaking an Image Encryption Algorithm Based on DNA Encoding and Spatiotemporal Chaos. Entropy **21**(3), 246 (2019)
35. Wen, H., Zhang, C., Huang, L., Ke, J., Xiong, D.: Security Analysis of a Color Image Encryption Algorithm Using a Fractional-Order Chaos. Entropy **23**(2), 258 (2021)
36. Wu, J., Liao, X., Yang, B.: Image Encryption Using 2D Hénon-Sine Map and DNA Approach. Signal processing **153**, 11–23 (2018)
37. Yosefnezhad Irani, B., Ayubi, P., Amani Jabalkandi, F., Yousefi Valandar, M., Jafari Barani, M.: Digital Image Scrambling Based on a New One-Dimensional Coupled Sine Map. Nonlinear Dynamics **97**(4), 2693–2721 (2019)
38. Yu, F., Gong, X., Li, H., Wang, S.: Differential Cryptanalysis of Image Cipher Using Block-Based Scrambling and Image Filtering. Information Sciences **554**, 145–156 (2021)
39. Zhou, K., Xu, M., Luo, J., Fan, H., Li, M.: Cryptanalyzing an Image Encryption Based on a Modified Henon Map Using Hybrid Chaotic Shift Transform. Digital Signal Processing **93**, 115–127 (2019)
40. Zolfaghari, B., Koshiba, T.: Chaotic Image Encryption: State-of-the-Art, Ecosystem, and Future Roadmap. Applied System Innovation **5**(3), 57 (2022)