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

George Teşeleanu[1,2][a]
[1]*Advanced Technologies Institute, 10 Dinu Vintilă, Bucharest, Romania*
[2]*Simion Stoilow Institute of Mathematics of the Romanian Academy, 21 Calea Grivitei, Bucharest, Romania*

Keywords: Image Encryption Scheme, Chaos Based Encryption, Cryptanalysis.

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 (Zolfaghari and Koshiba, 2022; Muthu and Murali, 2021; Hosny, 2020)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.

In (Essaid et al., 2019) 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, de-

[a] https://orcid.org/0000-0003-3953-2744

Teşeleanu, G.
Security Analysis of a Color Image Encryption Scheme Based on Dynamic Substitution and Diffusion Operations.
DOI: 10.5220/0011646300003405
In *Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023)*, pages 410-417
ISBN: 978-989-758-624-8; ISSN: 2184-4356

Table 1: Broken chaos based image encryption algorithms.

Scheme	(Matoba and Javidi, 2004)	(Wang et al., 2012)	(Huang et al., 2014)	(Khan, 2015)	(Song and Qiao, 2015)	(Chen et al., 2015)
Broken by	(Wang et al., 2019)	(Arroyo et al., 2013)	(Wen et al., 2021)	(Alanazi et al., 2021)	(Wen et al., 2019)	(Hu et al., 2017)
Scheme	(Hu et al., 2017)	(Niyat et al., 2017)	(Hua and Zhou, 2017)	(Pak and Huang, 2017)	(Liu et al., 2018)	(Shafique and Shahid, 2018)
Broken by	(Li et al., 2019a)	(Li et al., 2018)	(Yu et al., 2021)	(Wang et al., 2018)	(Ma et al., 2020)	(Wen and Yu, 2019)
Scheme	(Sheela et al., 2018)	(Wu et al., 2018)	(Yosefnezhad Irani et al., 2019)	(Khan and Masood, 2019)	(Pak et al., 2019)	(Mondal et al., 2021)
Broken by	(Zhou et al., 2019)	(Chen et al., 2020)	(Liu et al., 2020)	(Fan et al., 2021)	(Li et al., 2019b)	(Li et al., 2021)

pending 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.

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

2 PRELIMINARIES

Notations. In this paper, the subset $\{1,\dots,s-1\} \in \mathbb{N}$ is denoted by $[1,s)$. The action of selecting a random element x from a sample space X is represented by $x \xleftarrow{\$} X$, 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`.

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 (Essaid et al., 2019). 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 (Essaid et al., 2019), 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 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

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 O_{enc} and can interrogate it on different inputs. Therefore, A constructs some plaintexts that are useful for his attack and then using 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 3.1. *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.*

Figure 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

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

From Equations (1) *and* (2) *we derive*

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

If we request 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

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

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

which lead to

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

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.

The formal description of our chosen plaintext attack is provided in Algorithm 4 and is a generalization of the method presented in Example 3.1. For completeness, in Algorithm 4 we compute two possible key, s-boxes pairs.Note that in order to be able to use our attack we must have $H, W \geq 2$.

The complexity of Algorithm 4 is $O(2^{24}L + HW)$ and we need 256 oracle queries. For example, if we encrypt 2 megapixels[1] images we obtain that the complexity of Algorithm 4 is $O(2^{24} \cdot 2^{11.45} + 2^{20.87}) = O(2^{35.45})$. In the case of 12 megapixels[2], we obtain $O(2^{24} \cdot 2^{12.77} + 2^{23.5}) = O(2^{36.77})$.

3.2 Essaid *et Al.*'s Generation Method for S-Boxes

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

Example 3.2. *Compared to Example 3.1, 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 $\tilde{s}_{1,3}$.*

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

[1] $W \times H = 1600 \times 1200$
[2] $W \times H = 4000 \times 3000$

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 O_{enc}.
  Receive the ciphertext $\bar{c}$ from the encryption oracle 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 $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$

$\tilde{s}_{pos,0} = \tilde{s}_{pos,2} \circ \tilde{s}^{-1}_{pos,1}$. *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 3.2. The complexity of Algorithm 5 is $O(2^{24} + 2^8 L + HW)$ and we need 256 oracle queries. For example, if we encrypt 2 megapixels images we obtain that the complexity of Algorithm 5 is $O(2^{24} + 2^8 \cdot 2^{11.45} + 2^{20.87}) = O(2^{24.21})$. In the case of 12 megapixels, we obtain $O(2^{24} + 2^8 \cdot 2^{12.77} + 2^{23.5}) = O(2^{24.85})$.

Note that according to (Essaid et al., 2019) the security of their scheme is $O(2^{128})$. 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 O_{dec}. Therefore, A constructs some ciphertexts that are useful for his attack and then using 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.

$C_0(t) =$

t	2	t	2	t
1	t	1	t	2
t	1	t	1	t

$C_1(t) =$

2	t	2	t	2
t	1	t	1	t
1	t	1	t	1

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

Example 4.1. *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*

$$s_3[p_{0,3}[t]] \leftarrow (2 - t \oplus k_{0,3}) \bmod 256 \quad (5)$$
$$s_3[p_{1,2}[t]] \leftarrow (1 - t \oplus k_{1,2}) \bmod 256 \quad (6)$$
$$s_3[p_{2,1}[t]] \leftarrow (1 - t \oplus k_{2,1}) \bmod 256. \quad (7)$$

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*

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

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 3.1, 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 4.2. *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 4.1, 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.*

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.*

$C_2(t) =$

t	2	t	2	t	2
1	t	1	t	1	t
t	1	t	1	t	2

$C_3(t) =$

2	t	2	t	2	t
t	1	t	1	t	2
1	t	1	t	1	t

Figure 3: Ciphertext patterns for $H = 3$ and $W = 6$.

$C_4(t) =$

t	2	3	3	3	t
1	3	3	3	3	3
3	3	3	3	3	3

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

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 O_{dec}.
    Receive the plaintext $\bar{p}$ from the decryption oracle 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**
          $\tilde{k}_{pos,t,j-t} \leftarrow ((c_{t,j-t} - \tilde{s}_{pos,j}[p_{255,t,j-t}]) \bmod 256) \oplus c_{t,j-t-1}$
        $pos \leftarrow pos + 1$

The formal description of our chosen ciphertext attack is provided in Algorithm 7 and is a generalization of the methods presented in Examples 4.1 and

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 $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$

4.2. 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 $O(2^{24}L + HW)$. The only difference between the two attacks is that in the case of Algorithm 7 we need 512 decryption oracle queries.

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 $\tilde{k}_{pos,i,j} = (c_{0,i,j} - \tilde{s}_{pos,i+j}[p_{255,i,j}]) \bmod 256$

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 4.1 and 4.2. Then using similar techniques as the ones from Example 3.2 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 $O(2^{24} + 2^8 L + HW)$. The only difference is that in the case of Algorithm 9 we need 512 oracle queries.

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 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 $\tilde{k}, \tilde{s}, \tilde{s}^{-1}$

5 CONCLUSIONS

In (Essaid et al., 2019), 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 $O(2^{24})$. 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.

REFERENCES

Alanazi, A. S., Munir, N., Khan, M., Asif, M., and Hussain, I. (2021). Cryptanalysis of Novel Image Encryption Scheme Based on Multiple Chaotic Substitution Boxes. *IEEE Access*, 9:93795–93802.

Arroyo, D., Diaz, J., and Rodriguez, F. (2013). Cryptanalysis of a One Round Chaos-Based Substitution Permutation Network. *Signal Processing*, 93(5):1358–1364.

Chen, J., Chen, L., and Zhou, Y. (2020). Cryptanalysis of a DNA-Based Image Encryption Scheme. *Information Sciences*, 520:130–141.

Chen, J.-x., Zhu, Z.-l., Fu, C., Zhang, L.-b., and Zhang, Y. (2015). An Efficient Image Encryption Scheme Using Lookup Table-Based Confusion and Diffusion. *Nonlinear Dynamics*, 81(3):1151–1166.

Essaid, M., Akharraz, I., Saaidi, A., and Mouhib, A. (2019). A New Approach of Image Encryption Based on Dynamic Substitution and Diffusion Operations. In *SysCoBIoTS 2019*, pages 1–6. IEEE.

Fan, H., Zhang, C., Lu, H., Li, M., and Liu, Y. (2021). Cryptanalysis of a New Chaotic Image Encryption Technique Based on Multiple Discrete Dynamical Maps. *Entropy*, 23(12):1581.

Hosny, K. M. (2020). *Multimedia Security Using Chaotic Maps: Principles and Methodologies*, volume 884. Springer.

Hu, G., Xiao, D., Wang, Y., and Li, X. (2017). Cryptanalysis of a Chaotic Image Cipher using Latin Square-Based Confusion and Diffusion. *Nonlinear Dynamics*, 88(2):1305–1316.

Hua, Z. and Zhou, Y. (2017). Design of Image Cipher Using Block-Based Scrambling and Image Filtering. *Information sciences*, 396:97–113.

Huang, X., Sun, T., Li, Y., and Liang, J. (2014). A Color Image Encryption Algorithm Based on a Fractional-Order Hyperchaotic System. *Entropy*, 17(1):28–38.

Khan, M. (2015). A Novel Image Encryption Scheme Based on Multiple Chaotic S-Boxes. *Nonlinear Dynamics*, 82(1):527–533.

Khan, M. and Masood, F. (2019). A Novel Chaotic Image Encryption Technique Based on Multiple Discrete Dynamical Maps. *Multimedia Tools and Applications*, 78(18):26203–26222.

Li, M., Lu, D., Wen, W., Ren, H., and Zhang, Y. (2018). Cryptanalyzing a Color Image Encryption Scheme Based on Hybrid Hyper-Chaotic System and Cellular Automata. *IEEE access*, 6:47102–47111.

Li, M., Lu, D., Xiang, Y., Zhang, Y., and Ren, H. (2019a). Cryptanalysis and Improvement in a Chaotic Image Cipher Using Two-Round Permutation and Diffusion. *Nonlinear Dynamics*, 96(1):31–47.

Li, M., Wang, P., Liu, Y., and Fan, H. (2019b). Cryptanalysis of a Novel Bit-Level Color Image Encryption Using Improved 1D Chaotic Map. *IEEE Access*, 7:145798–145806.

Li, M., Wang, P., Yue, Y., and Liu, Y. (2021). 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.

Liu, L., Hao, S., Lin, J., Wang, Z., Hu, X., and Miao, S. (2018). Image Block Encryption Algorithm Based on Chaotic Maps. *IET Signal Processing*, 12(1):22–30.

Liu, Y., Qin, Z., Liao, X., and Wu, J. (2020). Cryptanalysis and Enhancement of an Image Encryption Scheme Based on a 1-D Coupled Sine Map. *Nonlinear Dynamics*, 100(3):2917–2931.

Ma, Y., Li, C., and Ou, B. (2020). Cryptanalysis of an Image Block Encryption Algorithm Based on Chaotic Maps. *Journal of Information Security and Applications*, 54:102566.

Matoba, O. and Javidi, B. (2004). Secure Holographic Memory by Double-Random Polarization Encryption. *Applied Optics*, 43(14):2915–2919.

Mondal, B., Behera, P. K., and Gangopadhyay, S. (2021). 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.

Muthu, J. S. and Murali, P. (2021). Review of Chaos Detection Techniques Performed on Chaotic Maps and Systems in Image Encryption. *SN Computer Science*, 2(5):1–24.

Niyat, A. Y., Moattar, M. H., and Torshiz, M. N. (2017). Color Image Encryption Based on Hybrid Hyper-Chaotic System and Cellular Automata. *Optics and Lasers in Engineering*, 90:225–237.

Pak, C., An, K., Jang, P., Kim, J., and Kim, S. (2019). A Novel Bit-Level Color Image Encryption Using Improved 1D Chaotic Map. *Multimedia Tools and Applications*, 78(9):12027–12042.

Pak, C. and Huang, L. (2017). A New Color Image Encryption Using Combination of the 1D Chaotic Map. *Signal Processing*, 138:129–137.

Shafique, A. and Shahid, J. (2018). Novel Image Encryption Cryptosystem Based on Binary Bit Planes Extraction and Multiple Chaotic Maps. *The European Physical Journal Plus*, 133(8):1–16.

Sheela, S., Suresh, K., and Tandur, D. (2018). Image Encryption Based on Modified Henon Map Using Hybrid Chaotic Shift Transform. *Multimedia Tools and Applications*, 77(19):25223–25251.

Song, C. and Qiao, Y. (2015). A Novel Image Encryption Algorithm Based on DNA Encoding and Spatiotemporal Chaos. *Entropy*, 17(10):6954–6968.

Wang, H., Xiao, D., Chen, X., and Huang, H. (2018). Cryptanalysis and Enhancements of Image Encryption Us-

ing Combination of the 1D Chaotic Map. *Signal processing*, 144:444–452.

Wang, L., Wu, Q., and Situ, G. (2019). Chosen-Plaintext Attack on the Double Random Polarization Encryption. *Optics Express*, 27(22):32158–32167.

Wang, X., Teng, L., and Qin, X. (2012). A Novel Colour Image Encryption Algorithm Based on Chaos. *Signal Processing*, 92(4):1101–1108.

Wen, H. and Yu, S. (2019). 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.

Wen, H., Yu, S., and Lü, J. (2019). Breaking an Image Encryption Algorithm Based on DNA Encoding and Spatiotemporal Chaos. *Entropy*, 21(3):246.

Wen, H., Zhang, C., Huang, L., Ke, J., and Xiong, D. (2021). Security Analysis of a Color Image Encryption Algorithm Using a Fractional-Order Chaos. *Entropy*, 23(2):258.

Wu, J., Liao, X., and Yang, B. (2018). Image Encryption Using 2D Hénon-Sine Map and DNA Approach. *Signal processing*, 153:11–23.

Yosefnezhad Irani, B., Ayubi, P., Amani Jabalkandi, F., Yousefi Valandar, M., and Jafari Barani, M. (2019). Digital Image Scrambling Based on a New One-Dimensional Coupled Sine Map. *Nonlinear Dynamics*, 97(4):2693–2721.

Yu, F., Gong, X., Li, H., and Wang, S. (2021). Differential Cryptanalysis of Image Cipher Using Block-Based Scrambling and Image Filtering. *Information Sciences*, 554:145–156.

Zhou, K., Xu, M., Luo, J., Fan, H., and Li, M. (2019). Cryptanalyzing an Image Encryption Based on a Modified Henon Map Using Hybrid Chaotic Shift Transform. *Digital Signal Processing*, 93:115–127.

Zolfaghari, B. and Koshiba, T. (2022). Chaotic Image Encryption: State-of-the-Art, Ecosystem, and Future Roadmap. *Applied System Innovation*, 5(3):57.