Multimedia Tools and Applications (2019) 78: 10227–10263
https://doi.org/10.1007/s11042-018-6586-0

A new RGB image encryption using generalized Vigenére-type table over symmetric group associated with virtual planet domain

Manish Kumar[1] · R. N. Mohapatra[2] · Sajal Agarwal[1] · G. Sathish[1] · S. N. Raw[3]

Received: 12 October 2017 / Revised: 10 July 2018 / Accepted: 22 August 2018 /
Published online: 5 September 2018

Abstract

The primary aim of this paper is to provide an efficient encryption algorithm for RGB images. A *new*, *fast*, and *secure* RGB image encryption algorithm using generalized Vigenére table over the symmetric group S_n associated with Virtual Planet Domain (VPD) is proposed. Also a new method to generate random key space using generalized Vigenére cipher is introduced. Randomness of the proposed key space has been verified by using NIST statistical test suite. A formula for the key space has also been obtained and it is shown that this key space resists brute force attack. The VPD has been designed to encode RGB image into VPD, which provides a fine interlacing among binary bit for each pixel. The proposed algorithm has been tested on standard RGB images. Robustness of the proposed algorithm has been successfully verified by using commonly known attacks (such as, differential, cropped, and noise, entropy attacks). Finally, the proposed technique has been compared with other existing algorithms and the data (shown in tables) confirm that the proposed algorithm is competitive and can resist exhaustive attacks efficiently.

Keywords Virtual planet domain · Virtual planet rules · Virtual planet diffusion · Generalized Vigenére-type table · Image encryption

1 Introduction

The main objective of this work is to provide high security for transmission of color images in an open network. Due to proliferation of electronic gadgets (smartphones, tablets, laptop, computers, hard-drives, and e-drives) for transferring important information in the form of images through networks and the interest of an adversary to corrupt the transmitted images, it is certain that images are prone to attacks. Images are frequently used in diverse areas such as military, engineering services, scientific experiments, medical imaging, advertising, online education and training etc.. Networks and communication technologies provide seve-

✉ Manish Kumar
manish.math.bhu@gmail.com

Extended author information available on the last page of the article.

ral different modes for transferring images. But, protecting images from an adversary has become a challenge. Therefore, the security of image transmission has become an important agenda in the digital world.

Over the last few years, deoxyribonucleic acid (DNA) coding based cryptography algorithms for image encryption have been proposed by various authors such as (see [3, 5–8, 13, 25, 27, 30, 32]). Recently, RGB image encryption algorithms are presented in [6]. This utilizes a diffusion process involving a chaotic map. DNA and elliptic curve Diffie–Hellman cryptography have also been used to encrypt RGB images in [7]. The complimentary rule of DNA is used to encrypt images (see [5, 7, 32]). Image encryption using DNA and one-time pad is also proposed in [3]. This method requires biological experiments and is computationally demanding. DNA sequences are created using four basic nucleic acids viz. Adenine (A), Cytosine (C), Guanine (G), Thymine (T). A and T; C and G are compliments of each other. The total number of possible combinations is 4! = 24 out of which only 8 follow the complimentary rule [3].

The weakness of the DNA encoding is discussed in [19]. Cryptography with DNA strands cannot be implemented for frequent use is shown in [3] and it is concerned with cryptographic one time pads, substitution and XOR operations in DNA encoded domain. Attacks on image encryption in DNA encoded domains are discussed in [26]. In [13], the authors have mentioned that the proposed scheme has security flaws (see [9]). Likewise, [29] has studied the security issues and confirmed that the method proposed in [28] can be broken easily. In DNA encoding, the 8-bit pixel values are converted into four 2-bit DNA sequences. However, the DNA based cryptography allows only eight DNA complementary base pairings which diminishes the number of operations in general theory of 2-bit binary operations (i.e., addition, subtraction, etc.).

The main object of this paper is to propose an efficient image encryption algorithm using generalized Vigenére-type table over symmetric group S_n of a finite set of n symbols associated with virtual planet domain, which provides a huge key space, fast encryption technique and high level of security. A new VPD is constructed with a view to offering fine interlacing of pixels in each (red, green, and blue) layer of RGB image and become immune to common attacks. Motivated by the DNA cryptography and to overcome the above drawbacks of DNA and its eight complementary rules —we constructed VPD with $8! = 40320$ possible rules to assign eight virtual planets and enhance the size of key space and its security. Further, VPD encoding is designed in such a way that, it is sensitive to the location of each planets and creates a new key space. Randomness of the proposed key space has been ascertained by using NIST statistical test suite. The results are demonstrated in the Appendix. Thus, the proposed algorithm works well to encrypt images including sparse and completely black too.

The algorithm proposed in [4], involves DNA diffusion and scrambling for image in DNA encoded domain. In order to provide high security, a new algorithm is proposed where diffusion and scrambling over VPD domain is carried out. Further proposed algorithm is designed in such a way that it interlace R, G and, B components provides high security. However, the algorithm proposed in [4] and [17] mixes R, G and B components only in DNA add operation step and DNA addition respectively. Furthermore, in [1], the same diffusion and scrambling steps are introduced in R, G and B components individually to get the encrypted image. The proposed algorithms demonstrates clearly the importance of intermixing between R, G and B components along with effective diffusion process. Thus, the proposed algorithm offers good scrambling and diffusion in order to achieve high security.

1.1 The rest of this paper is organized as follows

In Section 2 the key generation procedure using generalized Vigenére-type table over symmetric group S_n of a finite set of n symbols is described in brief. In Section 3, the method used in this paper for information encoding using planetary domain scheme is introduced. Encryption algorithm is explained in Section 4. Then computer simulations are carried out to demonstrate and verify the procedure on some standard color images and performance of the proposed algorithm, security analysis are addressed in Section 5. Efficiency of our proposed encoding scheme is illustrated in Section 6. In Section 7, comparison of the proposed procedure with the methods given in [2, 4, 6, 7, 9, 15, 20, 23, 24, 31] has been done. In Section 8 conclusion drawn from the present work is mentioned.

2 Generalized Vigenére-type table over Symmetric group S_n and random key generation procedure

In this section, a new method to generate random key space using generalized Vigenére-type table over symmetric group S_n (which prevents an adversary from using a brute force attack to find a key) is proposed. The generalized Vigenére-type table has been obtained by using elements of the symmetric group S_n. Since there are $n!$ elements in S_n, so each subsequent column of the generalized Vigenére-type table is filled by an element of S_n selected at random without replacement manner. The generalized Vigenére-type table is designed large enough to make such a search computationally infeasible. A random sequence using generalized Vigenére-type table is generated and encryption/decryption keys are obtained. The generalized Vigenére-type table over symmetric group S_n is of size $n \times q$, where $1 \leq q \leq n!$. Thus the total number of options to fill the generalized Vigenére-type table without repeating of elements is $(n!) \times (n! - 1) \times (n! - 2) \times \cdots \times (n! - q + 1)$ ways.

For instance, if one takes $n = 3$, and $q = 3$ then one gets the symmetric group S_3 (i.e. symmetric group of equilateral triangle), and the Table 1 is created using the elements of S_3.

Similarly, if one takes $n = 26$, $q = 26$ and use the first 26 elements from S_{26}, (say s^1_{26}, s^2_{26}, $s^3_{26} \dots s^{26}_{26}$) then the table columns from 1 to q can be filled with s^1_{26}, s^2_{26}, s^3_{26} and s^{26}_{26} receptively. This gives a Vigenére table which is well known table in the literature as shown in the Table 2, where symbols are used as alphabets.

Thus in general, the generalized Vigenére-type table size $n \times q$ on S_n group is depicted in Table 3. This is used to generate a random sequence.

2.1 Generating key sequence using generalized Vigenére-type table

In this paper, three sequences (Fig. 1) of random numbers, say $P = \{P_i\}$, $Q = \{Q_i\}$ and $R = \{R_i\}$ each of length $N \times M \times 8$ (where the size of original image to be encrypted is $N \times M \times 3$) are generated. The sequences will be obtained using the following procedure:

Table 1 Generalized Vigenére-type table of size 3×3 on S_3 group

row/column	1	2	3
1	1	2	2
2	3	3	1
3	2	1	3

Table 2 Generalized Vigenére-type table of size 26 × 26 on S_{26} group

row/column	1	2	3	4	5	...	24	25	26
1	1	2	3	4	5	...	24	25	26
2	2	3	4	5	6	...	25	26	1
3	3	4	5	6	7	...	26	1	2
4	4	5	6	7	8	...	1	2	3
5	.	.	.	.	.	...	.	.	.
.	.	.	.	.	.	...	.	.	.
.	.	.	.	.	.	...	.	.	.
26	26	1	2	3	4	...	23	24	25

1. Employ the generalized Vigenére-type table on S_n group (as described in Table 3) of size $n \times q$.
2. Take three seeds $seed_1$, $seed_2$, and $seed_3$ such that the value of each seeds is less than the minimum of n and q

$$\left.\begin{array}{l} seed_1 = HASH(R_{sum}) \pmod{min(n,q)}, \\ seed_2 = HASH(G_{sum}) \pmod{min(n,q)}, \\ seed_3 = HASH(B_{sum}) \pmod{min(n,q)} \end{array}\right\} \tag{1}$$

for the purpose of generating random key sequence, here R_{sum}, G_{sum} and B_{sum} are sum of all pixels of R, G and B layers respectively and $HASH$ represents hash function.
3. Let $P_1 = seed_1$, $Q_1 = seed_2$ and $R_1 = seed_3$. The complete sequence $P = \{P_i\}$, $Q = \{Q_i\}$ and $R = \{R_i\}$ for $i = 2, 3, \ldots, (N \times M \times 8)$ can be generated by the following procedure:

$$\left.\begin{array}{l} P_i = GVTSG(P_{i-1}, Q_{i-1}), \\ Q_i = GVTSG(Q_{i-1}, R_{i-1}), \\ R_i = GVTSG(P_{i-1}, R_{i-1}) \end{array}\right\} \tag{2}$$

The function $GVTSG(x, y)$ takes the entry from x^{th} row and y^{th} column in the generalized Vigenére-type Table 3.
4. **Generation of sequence $L = \{L_i\}$:** For $i = 1, 2, 3, \ldots, (N \times M \times 8)$,

$$L_i = P_{i-1} \oplus Q_{i-1} \oplus R_{i-1}, \tag{3}$$

Table 3 Generalized Vigenére-type table of size $n \times q$ on S_n group

row/column	1	2	3	4	5	...	q-2	q-1	q
1	111	211	3	100	5	...	10	151	22
2	7	3	114	23	43	...	111	n	143
3	3	34	15	134	211	...	34	101	17
4	200	5	6	67	n	...	100	2	163
5	.	.	.	.	.	...	.	.	.
.	.	.	.	.	.	...	.	.	.
.	.	.	.	.	.	...	.	.	.
n	25	1	212	199	8	...	50	190	71

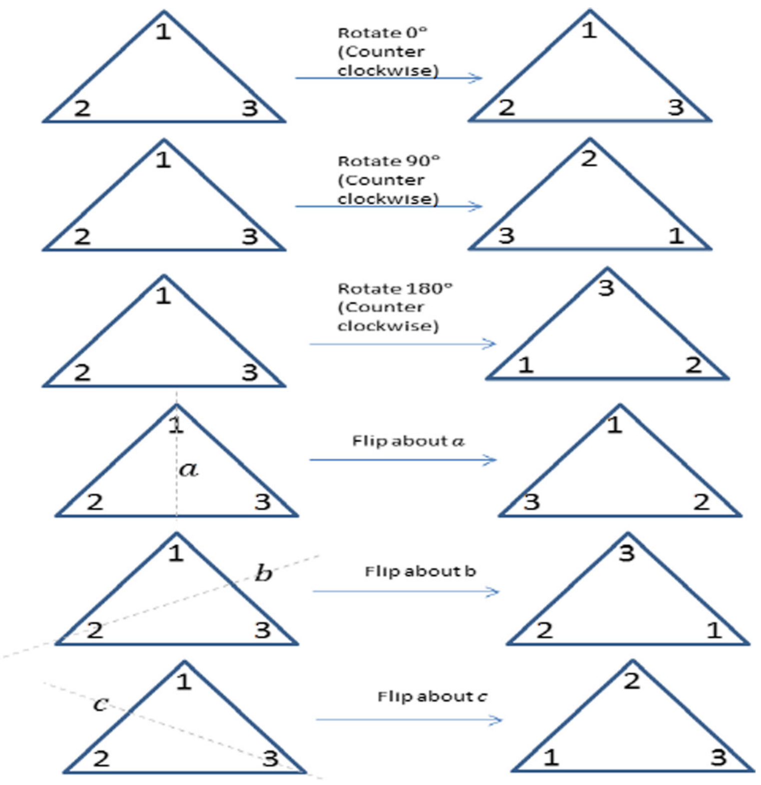

Rotate 0° (Counter clockwise)
Rotate 90° (Counter clockwise)
Rotate 180° (Counter clockwise)
Flip about a
Flip about b
Flip about c
1
2
3
a
b
c

Fig. 1 Symmetric group of equilateral triangle

here $\oplus$ represents bitwise XOR-operation.

5. **Generation of sequence RK and CK:** Consider the sequence $\{L_1, L_2, L_3, \ldots, L_n\}$ and sort this sequence, then mark the previous positions (index) after sorting, as our new key is RK. Now take the set $\{L_{n+1}, L_{n+2}, L_{n+3}, \ldots, L_{n+m*8}\}$ and sort. Mark the previous positions (index) after sorting, as our new key is CK.
6. **Generation of sequence $T = \{T_i\}$ and $U = \{U_i\}$:** For $i = 1, 2, 3, \ldots, (N \times M \times 8)$:

$$\left.\begin{aligned} T_i &= L_i \pmod 8, \\ U_i &= P_i \pmod 8) \end{aligned}\right\} \tag{4}$$

Reshape T and U into the size of $N \times 8M$.

7. **Generation of sequence $V = \{V_i\}$:** For $i = 1, 2, 3, \ldots, (N \times M \times 8)$,

$$V_i = Q_{i-1} \oplus R_{i-1} \tag{5}$$

Express $V = V_i$ as a matrix of size $N \times 8M$.

Figure 2a, b, c, d, e and f are plotted for different orders of seeds and generalized Vigenére-type Tables. Figure 2f depicts that all 3 sequences ($P(red)$, $Q(green)$ and $R(blue)$) coincides only, if all seeds and generalized Vigenére-type table are same. Hence, Fig. 2 reveals that the generated sequences P, Q and R are very sensitive with respect to seeds and generalized Vigenére-type table. Randomness of the sequences P, Q, and R has been proved by using NIST statistical test suite and results are presented in the Appendix.

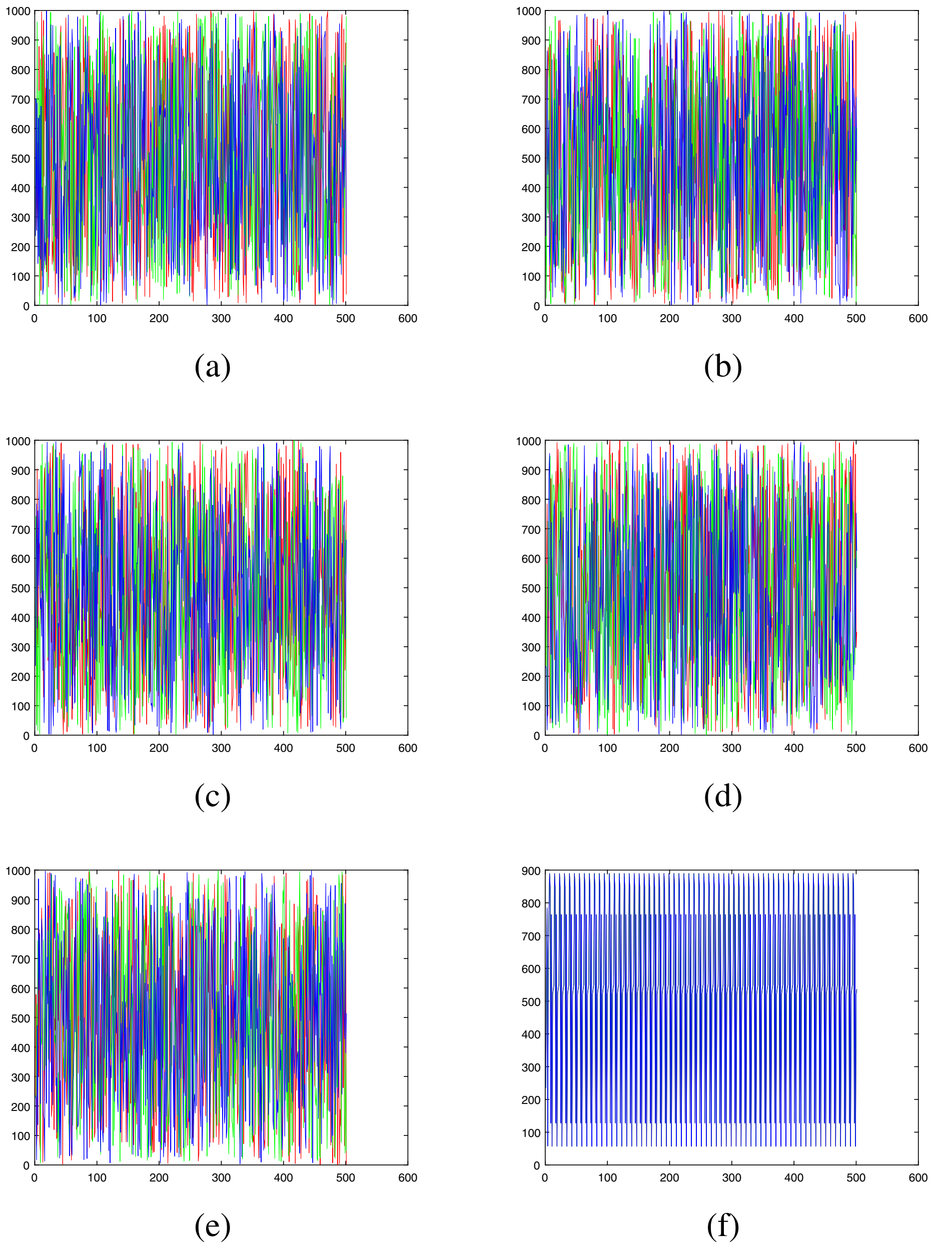


Fig. 2 Sequence plots for $P(red)$, $Q(green)$ and $R(blue)$: **a** $(seed_1, seed_2, seed_3) = (235, 235, 236)$, **b** $(seed_1, seed_2, seed_3) = (235, 236, 236)$, **c** $(seed_1, seed_2, seed_3) = (235, 235, 236)$ with table changed, **d** $(seed_1, seed_2, seed_3) = (235, 236, 236)$ with table changed, **e** $(seed_1, seed_2, seed_3) = (235, 235, 235)$ with table changed, **f** $(seed_1, seed_2, seed_3) = (235, 235, 235)$

3 Rules of virtual planet encoding scheme

This section provides a brief explanation (divided into six Sub-Sections) about steps involved in encryption algorithm.

3.1 Virtual planetary encoding scheme

In view of the limitations of DNA coding (such as, eight complementary rules, biological experiments and extreme computation, etc.), a new idea has been introduced to construct *VPD* encoding scheme by using eight virtual planets from the virtual solar system: namely, Mercury (Me), Venus (Ve), Earth (Ea), Mars (Ma), Jupiter (Ju), Saturn (Sa), Uranus (Ur) and Neptune (Ne). In this encoding scheme, each planet will be represented by a particular 3-bit binary number (since, with three bits, 8 different 3-bit binary numbers are produced, therefore every 3-bit binary number can be uniquely represented by a certain planet) as shown in Table 4.

3.1.1 40320-rules for virtual planets

Inspired by the work in [10], one can assign 40320-rules to eight virtual planets, and simulate the position vectors of all planets for 100 years. In order to simulate the position vectors, we have used Newton's law of gravitation, where each planet is represented as a mass point as shown in (6). This makes it possible to calculate the force of each planet. "The force on each planet is simply the sum of the gravitational forces from the sun and all of the other planets in the solar system". In vector notation, we have

$$\vec{F_{ij}} = m_i \frac{d^2 r_i}{dt^2} = G m_i \sum_{j \neq i} m_j \frac{r_j - r_i}{|r_i - r_j|^3}, \tag{6}$$

where G, m, and r are universal constant of gravitation, mass of each planets, and their positions in space respectively.

Newton's second law states that, $\vec{F}$ applied to object mass m induce the object to move with acceleration, $\vec{a}$; $\vec{F} = m\vec{a}$. Hence, to solve for the motion of each object (i.e., find the positions $r_i = (x_i, y_i, z_i)$ and $r_j = (x_j, y_j, z_j)$ with respect to time t), we have

$$Fx_{ij} = m_i \frac{d^2 x_i}{dt^2}, \quad Fy_{ij} = m_i \frac{d^2 y_i}{dt^2}, \quad Fz_{ij} = m_i \frac{d^2 z_i}{dt^2}. \tag{7}$$

Equation (7) has been solved for each dimensions and the position vectors for all planets are computed. An array $Y := [Sun\ Me_{r_i^t}\ Ve_{r_i^t}\ Ea_{r_i^t}\ Ma_{r_i^t}\ Ju_{r_i^t}\ Sa_{r_i^t}\ Ur_{r_i^t}\ Ne_{r_i^t}]$ of size 1×9 has been defined where $x_{r_i^t}$ is position vector of planet x at time t and calculated the following eight distances in meter (m) at time t as shown in Fig. 3, and represented by a set $d_t(x_{r_i^t}, y_{r_i^t}) := \{d_t(Sun, Me_{r_i^t}), d_t(Me_{r_i^t}, Ve_{r_i^t}), d_t(Ve_{r_i^t}, Ea_{r_i^t}),$ $d_t(Ea_{r_i^t}, Ma_{r_i^t}), d_t(Ma_{r_i^t}, Ju_{r_i^t}), d_t(Ju_{r_i^t}, Sa_{r_i^t}), d_t(Sa_{r_i^t}, Ur_{r_i^t}), d_t(Ur_{r_i^t}, Ne_{r_i^t})\}$. Then modulo k over of each distance is represented as $d_t(x_{r_i^t}, y_{r_i^t})$ and sort them in an ascending order. Thereafter, allocate each planet (in an increasing order) to the 3-bit binary number (in an ascending order) as shown in Table 5. For instance, at time $t = 351311h\ 04m$ these distances are taken and rule-21 has been achieved (shown in Table 4 and plotted in Fig. 8).

Table 4 12 rules out of 40320 rules for virtual planets and their respective 3-bit binary number which will be used for converting image from binary to VPD

3-bit binary number	Rule-21	Rule-20075	Rule-18936	Rule-37241	Rule-26765	Rule-10559	Rule-37604	Rule-17256	Rule-19188	Rule-34438	Rule-20462	Rule-4265
000	Ne	Ur	Ne	Ur	Ma	Ea	Sa	Sa	Ne	Ne	Ve	Me
001	Ea	Ma	Ea	Ma	Ea	Ve	Ea	Ma	Ju	Sa	Sa	Ur
010	Ma	Ju	Ur	Ve	Ve	Me	Ma	Ur	Ur	Ur	Ne	Ne
011	Ju	Me	Me	Ne	Ju	Ma	Ve	Me	Me	Ju	Ea	Ma
100	Ve	Ne	Sa	Ju	Ur	Ne	Ne	Ne	Ea	Ea	Me	Ju
101	Sa	Sa	Ma	Ea	Me	Ju	Ur	Ve	Sa	Ve	Ma	Sa
110	Me	Ea	Ve	Sa	Ne	Sa	Ju	Ju	Ve	Me	Ju	Ve
111	Ur	Ve	Ju	Me	Sa	Ur	Me	Ea	Ma	Ma	Ur	Ea

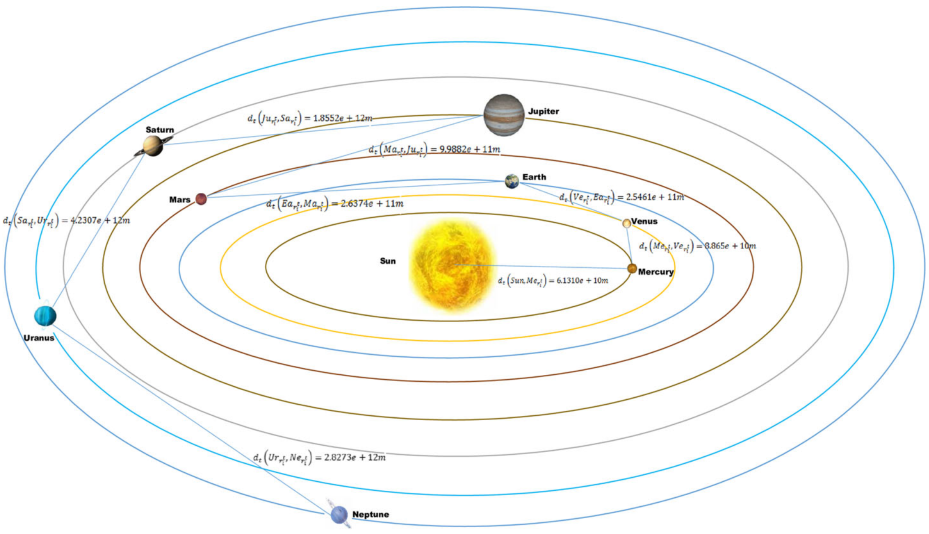

Jupiter
Saturn
$d_t(Ju_{r_i^t}, Sa_{r_i^t}) = 1.8552e + 12m$
$d_t(Ma_{r_i^t}, Ju_{r_i^t}) = 9.9882e + 11m$
Earth
Mars
$d_t(Ea_{r_i^t}, Ma_{r_i^t}) = 2.6374e + 11m$
$d_t(Ve_{r_i^t}, Ea_{r_i^t}) = 2.5461e + 11m$
Venus
$d_t(Sa_{r_i^t}, Ur_{r_i^t}) = 4.2307e + 12m$
$d_t(Me_{r_i^t}, Ve_{r_i^t}) = 8.865e + 10m$
Sun
Mercury
$d_t(Sun, Me_{r_i^t}) = 6.1310e + 10m$
Uranus
$d_t(Ur_{r_i^t}, Ne_{r_i^t}) = 2.8273e + 12m$
Neptune

Fig. 3 Original solar system with distance between planets in meter (m)

One can choose any modulo value for k which lies between 200 and 10^{10} (as the minimum distance between any planets is at least in terms of 10^{10}). At different times t all 40320 rules (for 100 years) in VPD can be achieved by the above process. To show randomness on distance modulo k, the distance of each planet over a hundred year period with 1.752 hours gap have been simulated and 5,00,000 points have been generated. Finally, correlation between all planets are presented in Table 6.

4 Proposed algorithm

The proposed encryption and decryption procedure are given below:

Table 5 Assignment of the rule-21

Distance of virtual planets (mod$k$)	Distance after sorting	Rule-21
$d_t(Sun_{r_i^t}, Me_{r_i^t})(\text{mod}256) = 232.1170$	$d_t(Ur_{r_i^t}, \mathbf{Ne_{r_i^t}})(\text{mod}256) = 17.2876$	Ne = 000
$d_t(Me_{r_i^t}, Ve_{r_i^t})(\text{mod}256) = 219.8610$	$d_t(Ve_{r_i^t}, \mathbf{Ea_{r_i^t}})(\text{mod}256) = 29.3695$	Ea = 001
$d_t(Ve_{r_i^t}, Ea_{r_i^t})(\text{mod}256) = 29.3695$	$d_t(Ea_{r_i^t}, \mathbf{Ma_{r_i^t}})(\text{mod}256) = 136.8814$	Ma = 010
$d_t(Ea_{r_i^t}, Ma_{r_i^t})(\text{mod}256) = 136.8814$	$d_t(Ma_{r_i^t}, \mathbf{Ju_{r_i^t}})(\text{mod}256) = 174.3091$	Ju =011
$d_t(Ma_{r_i^t}, Ju_{r_i^t})(\text{mod}256) = 174.3091$	$d_t(Me_{r_i^t}, \mathbf{Ve_{r_i^t}})(\text{mod}256) = 219.8610$	Ve =100
$d_t(Ju_{r_i^t}, Sa_{r_i^t})(\text{mod}256) = 221.1003$	$d_t(Ju_{r_i^t}, \mathbf{Sa_{r_i^t}})(\text{mod}256) = 221.1003$	Sa = 101
$d_t(Sa_{r_i^t}, Ur_{r_i^t})(\text{mod}256) = 234.9780$	$d_t(Sun_{r_i^t}, \mathbf{Me_{r_i^t}})(\text{mod}256) = 232.1170$	Me=110
$d_t(Ur_{r_i^t}, Ne_{r_i^t})(\text{mod}256) = 17.2876$	$d_t(Sa_{r_i^t}, \mathbf{Ur_{r_i^t}})(\text{mod}256) = 234.9780$	Ur = 111

Significance of boldsymbols for planets are used to allocated planet rules

Table 6 Correlation coefficient over distance of the eight virtual planets for 5,00,000 points over 100 years

Distance of virtual planets	$d_t(Sun_{r_i^t}, Me_{r_i^t})$	$d_t(Me_{r_i^t}, Ve_{r_i^t})$	$d_t(Ve_{r_i^t}, Ea_{r_i^t})$	$d_t(Ea_{r_i^t}, Ma_{r_i^t})$	$d_t(Ma_{r_i^t}, Ju_{r_i^t})$	$d_t(Ju_{r_i^t}, Sa_{r_i^t})$	$d_t(Sa_{r_i^t}, Ur_{r_i^t})$	$d_t(Ur_{r_i^t}, Ne_{r_i^t})$
$d_t(Sun_{r_i^t}, Me_{r_i^t})$	1.0000	0.0023	– 0.0007	0.0004	0.0016	– 0.0080	– 0.0021	0.0007
$d_t(Me_{r_i^t}, Ve_{r_i^t})$		1.0000	0.0003	– 0.0012	– 0.0037	0.0000	– 0.0014	0.0000
$d_t(Ve_{r_i^t}, Ea_{r_i^t})$			1.0000	– 0.0005	0.0005	– 0.0023	0.0017	– 0.0005
$d_t(Ea_{r_i^t}, Ma_{r_i^t})$				1.0000	0.0014	0.0008	0.0015	0.0004
$d_t(Ma_{r_i^t}, Ju_{r_i^t})$					1.0000	0.0029	0.0011	– 0.0018
$d_t(Ju_{r_i^t}, Sa_{r_i^t})$						1.0000	0.0002	0.0010
$d_t(Sa_{r_i^t}, Ur_{r_i^t})$							1.0000	– 0.0038
$d_t(Ur_{r_i^t}, Ne_{r_i^t})$								1.0000

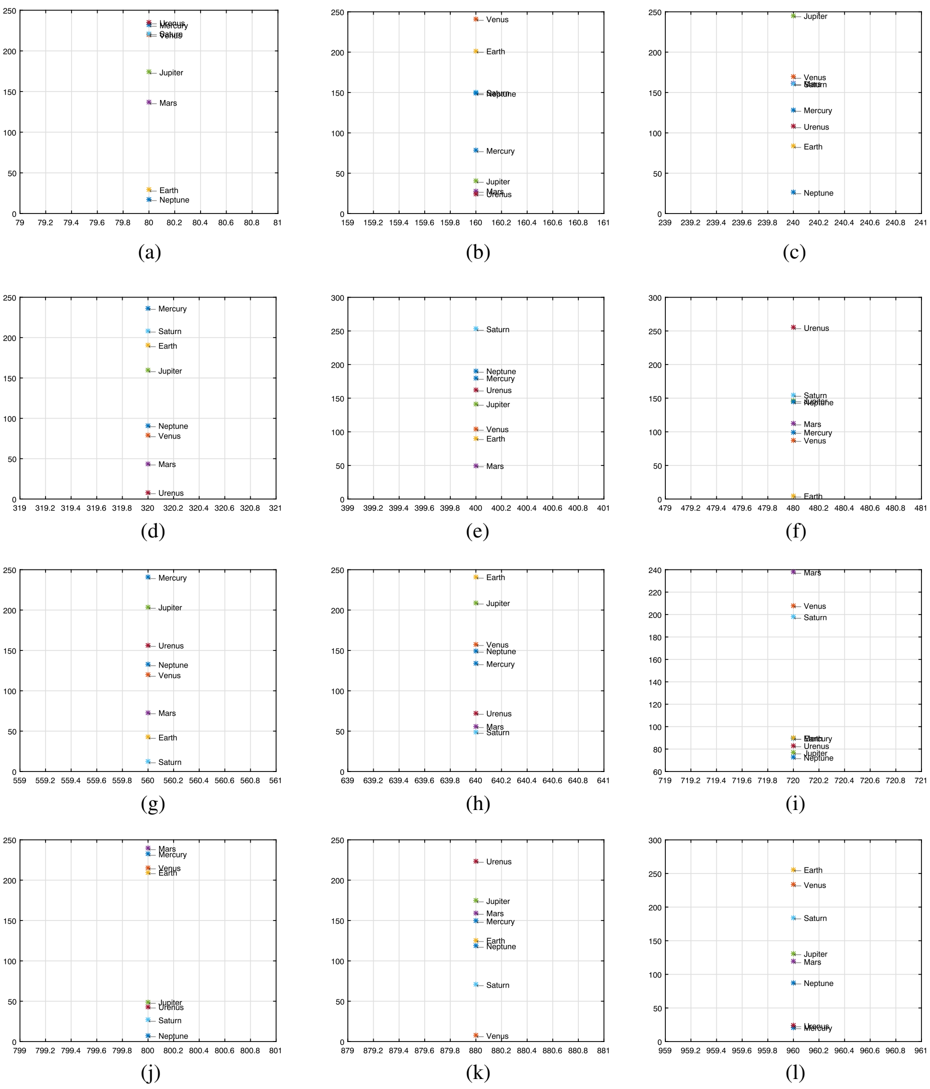


Fig. 4 Distance plots for the eight virtual planets of 12 rules given in the Table 4 over modulo 256 for 100 years: **a** Rule-21 at time $t = 351311h\ 04m$, **b** Rule-20075 at time $t = 139292h\ 76m$, **c** Rule-18936 at time $t = 209372h\ 76m$, **d** Rule-37241 at time $t = 6526h\ 20m$, **e** Rule-26765 at time $t = 245h\ 28m$, **f** Rule-10559 at time $t = 125776h\ 08m$, **g** Rule-37604 at time $t = 489692h\ 76m$, **h** Rule-17256 at time $t = 16740h\ 36m$, **i** Rule-19188 at time $t = 584151h\ 84m$, **j** Rule-34438 at time $t = 339958h\ 08m$, **k** Rule-20462 at time $t = 519906h\ 00m$, **l** Rule-4625 at time $t = 464577h\ 84m$

4.1 Encryption procedure

The original RGB image encrypted successfully by using planetary encoding, virtual planet diffusion, virtual planet transformation, permutation shuffling, original planet transformation and planet decoding. Each step is described in detail as follows:

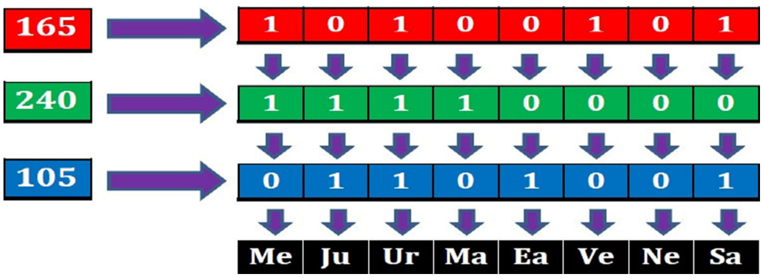


Fig. 5 RGB pixel encoding in VPD using the Rule-21 as given in the Table 4

4.1.1 Encoding original image

Every pixel of 8-bit original image I can be represented as a 8-bit binary number. In RGB image there are 3 layers of matrices (i.e., red layer, green layer and blue layer). If we choose (Fig. 4) the first pixel of every matrix (same as, the pixel at location (1,1)) then we will get 3 pixel values. Converting each of these three 3 pixels to their 8-bit binary equivalent gives a total of 24-bits. Since each planet corresponds to a 3-bit binary number, some unique pattern of eight planets will be able to represent these 24 bits (as shown in Fig. 5: for example, let the first pixel value at red, green and blue matrices be 165, 240, 105, respectively. Convert each of these numbers to their 8-bit binary equivalent numbers. Now we can choose the first bit from every row. In this example (see Fig.5), it is 110. Now one can refer to Table 4 (Rule-21) and look for a planet corresponding to 110. The required planet is Mercury (Me). Likewise, the second planet will be Jupiter (Ju) which is corresponding to 011. Continuing in the same manner, one can get a total of 8 planets, which uniquely represents these 3 set of pixel values.

Proceeding further in this manner, one observes that every set of three pixels corresponding to 3 matrices, that can be encoded as a pattern of eight virtual planets from the virtual solar system. After encoding the entire image, a 3-dimensional matrix (I) of size $N \times M \times 3$ will be converted to a 2-dimensional matrix (E^1) of size $r \times c$ where $r = N$ and $c = M \times 8$) as shown in Fig. 6.

4.1.2 Virtual planet diffusion

Virtual planet diffusion is performed by using XOR operation (like, usual bitwise-XOR between the binary bits represented by each virtual planet) among different planets (for instance, to diffuse Earth (Ea) and Jupiter (Ju), where Earth corresponds to 001 and Jupiter corresponds to 011, then XOR-ing of these two yield 010 which corresponds to the planet Mars (Ma), so diffusing Earth and Jupiter gives Mars). List of all possible combinations

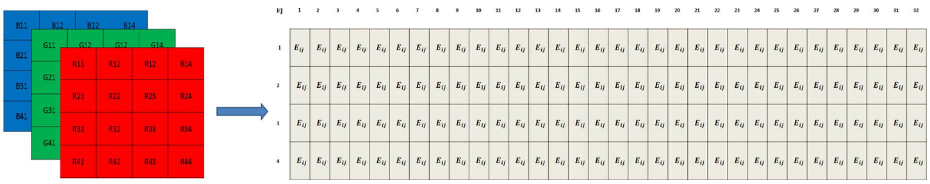


Fig. 6 RGB image of size 4 × 4 × 3 encoded into a matrix of size 4 × 32 in VPD

Table 7 A comprehensive table depicting XOR operation between each virtual planet using the Rule-21 as shown in Table 4

Virtual planets	Ne	Ea	Ma	Ju	Ve	Sa	Me	Ur
Ne	Ne	Ea	Ma	Ju	Ve	Sa	Me	Ur
Ea	Ea	Ne	Ju	Ma	Sa	Ve	Ur	Me
Ma	Ma	Ju	Ne	Ea	Me	Ur	Ve	Sa
Ju	Ju	Ma	Ea	Ne	Ur	Me	Sa	Ve
Ve	Ve	Sa	Me	Ur	Ne	Ea	Ma	Ju
Sa	Sa	Ve	Ur	Me	Ea	Ne	Ju	Ma
Me	Me	Ur	Ve	Sa	Ma	Ju	Ne	Ea
Ur	Ur	Me	Sa	Ve	Ju	Ma	Ea	Ne

of diffusion between virtual planets are shown in Table 7 by using Rule-21. The proposed virtual planet diffusion provides more confusion to the adversary, because of 40320 possibilities to do XOR operation.

Virtual planet diffusion is introduced to diffuse the image in the encoded domain. To do this, one can select the sequence U (as generated in step 6 of subsection 2.1) and E^1_{ij} are the virtual planets in encoded image E^1 obtained in step 1 where $i = 1, 2, 3, \dots r$ and $j = 1, 2, 3, \dots c$. The transformed E^2_{ij} can be obtained by using the following diffusion operation defined by (8) given below:

$$E^2_{ij} = E^1_{ij} \oplus U_{ij}, \tag{8}$$

where U_{ij} is the $(i, j)^{th}$ element of U in VPD domain (i.e.,if $U_{ij} = 7$, then VPD form of U_{ij} is equal to $Uranus(Ur)$ as per rule-21 in Table 4); The symbol $\oplus$ denotes the diffusion between two planets. The diffusion operation is shown in Fig. 7 produces the diffused image E^2.

4.1.3 Virtual planet transform

For a given number T_{ij} (where $T_{ij} = 0, 1, 2, \dots, 7$) and for a given virtual planet, one can transform a planet to some other planet by taking it T_{ij} positions ahead of this planet as shown in Fig. 8. i.e. Observe from the Fig. 9, from Mercury (Me) take $T_{ij} = 3$ positions ahead and replace by Earth (Ea). Now all the eight virtual planets in E^2 will be transformed

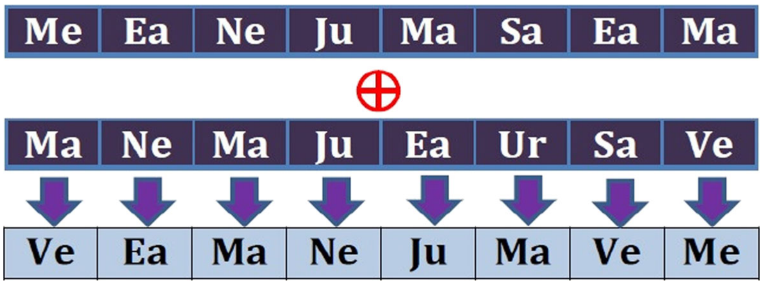


Fig. 7 Representation of virtual planet diffusion process between two *Planets*

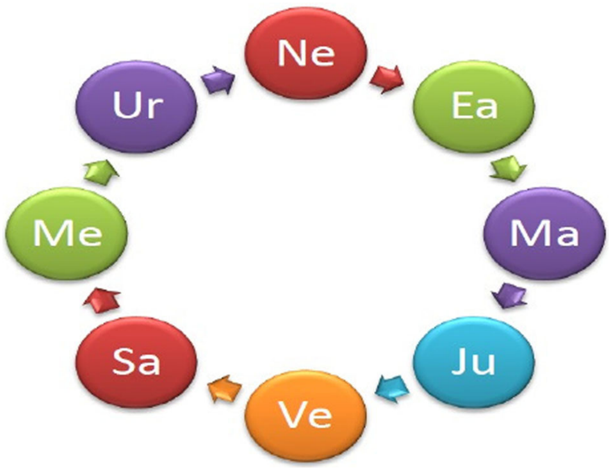


Fig. 8 Clockwise rotation of virtual planets which will be used during virtual planet transform

by the above method. Repeat it for all the $(r \times c)$ planets in E^2 provides E^3. This process yields robust support to the proposed algorithm against attacks (such as, plain-text; chosen cipher-text, etc.).

4.1.4 Permutation shuffling

A new permutation shuffling has been introduced over E^3 (obtained in step 3) using permutation keys (RK and CK generated in step 5 of Section 2). Permutation shuffling offers sensitivity to permutation keys and can be viewed in Fig. 12f and g. The permutation shuffled image E^4 will be obtained by the following process: Let the output image after permutation shuffling be E^4, then we have

$$E^4(RK(i), CK(j)) := E^3(i, j). \tag{9}$$

Similarly, decryption process is reverse way of (9) and can be seen in (10):

$$E^3(i, j) := E^4(RK(i), CK(j). \tag{10}$$

4.1.5 Original planetary transform

Consider the same order of planets in Original Planetary Domain (OPD) = $\{Mercury(Me)$, $Venus(Ve)$, $Earth(Ea)$, $Mars(Ma)$, $Jupiter(Ju)$, $Saturn(Sa)$, $Uranus(Ur)$,

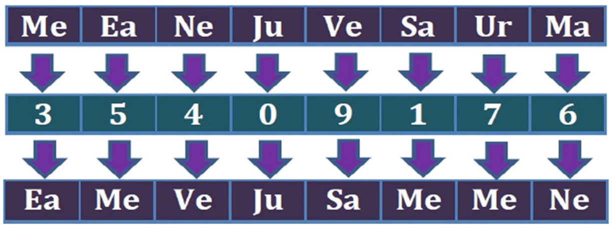


Fig. 9 Substation of virtual planet transformation process on a single Cell

$Neptune(Ne)\}$ as shown in Fig. 3. The transformation from VPD to OPD can be obtained in the following way,

$$K = (J + KEY) \pmod 8 + 1, \qquad KEY = V, \tag{11}$$

where J and K represent position of planets in OPD. Here the J^{th} planet in VPD will be replaced by K^{th} planet in OPD over matrix E^4 (obtained after step 4) produces the output matrix E^5. Here VPD order follows the order of Rule-21 in Table 4 which is VPD $= \{Neptune(Ne), Earth(Ea), Mars(Ma), Jupiter(Ju), Venus(Ve), Saturn(Sa), Mercury(Me), Uranus(Ur)\}$.

The inverse can be found as follows:

$$J = (K - 1 - KEY) \pmod 8, \qquad KEY = V. \tag{12}$$

The main reason for introducing the original planetary transform is to connect VPD to Original Solar System (OSS) and provide randomness to the encoding scheme. This process transforms the planets by using original planet order from the solar system bringing connection with OPD.

4.1.6 Virtual planet decoding

Decode E^5 results in encrypted image of size $N \times M \times 3$. Here the number 3 is used to take into account that the three planes. Flowchart of encryption algorithm is given in Fig. 10.

4.2 Decryption procedure

Similarly, the decryption can be performed by taking inverse operation of the encryption process.

4.3 Mathematical model

A mathematical model has been incorporated to justify the proposed algorithm in terms of a new planet diffusion scheme. Further, planet encoding and permutation shuffling offers better security as explained below:

(i) Encoding: As discussed in key space analysis there are $8! = 40320$ rules. Change in the rule will make, the pixel values to be changed. Let us take, rule-21 used in encryption. Let us consider adversary have chosen (guessed) the rule in which only first two planets (Ne-000, Ea-001) are interchanged as(Ne-001, Ea-000) remaining all same. Let us say encrypted image is encoded during decryption results in pixel values changed wherever the virtual planets $Ne - 001$ and $Ea - 000$ are occurring. Our encoding scheme itself offers security as correct rule should be chosen for decryption (Clearly depicted in key change analysis).

(ii) Virtual planet diffusion: This step is executed with the help of Table 7, with key U generated in 2.1 and rule chosen. Before analysis, a look at Table 7, every column and row of table is permutation of virtual planets chosen. Out of 40320 permutations only eight are selected and filled in the Table 7. So the correct rule should be chosen to make this table. And key U (generated in the step VI of Section 2.1) plays important role to decrypt image.

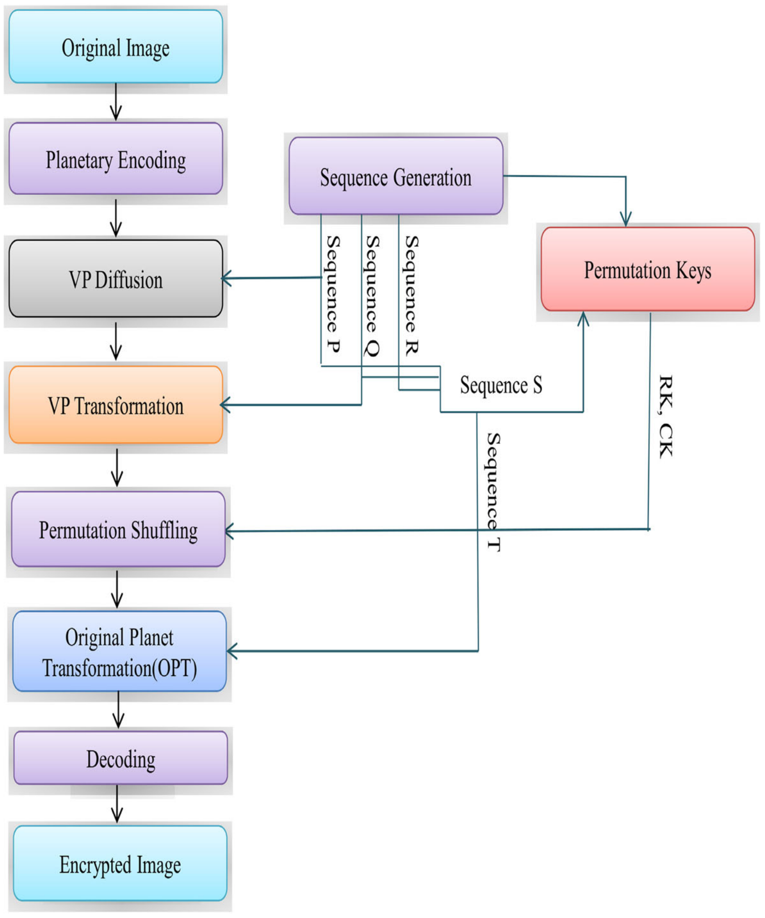


Fig. 10 Flowchart of the encryption process

Analysis: An adversary would require a correct rule out of 40320 rules and key U to break this transformation. And Table 7 is resembling as substitution box along with key U.

(iii) Virtual planet transform: Let us take virtual planet x is $Ne-001$ and take $t \in T$ is 3. After virtual planet transform x becomes y which is $Ju-011$.

Ne-000	Ea-001	Ma-010	Ju-011	Ve-100	Sa-101	Me-110	Ur-111

Suppose an adversary try to break this step by guessing the key $t = 4$. From y $(Ju-011)$ going back will results in $Ur-111$. So $Ne-000$ will be replaced by $Ur-111$ and it results in corresponding pixel change during decryption.

And rule-21 plays a major role during this transition. Change of rule will destroy total structure during this transformation.

(iv) Permutation shuffling: We demonstrate this step by considering a matrix A given by

$$A = \begin{bmatrix} Sa & Ju & Me \\ Ve & Ea & Ne \\ Ma & Ur & Ve \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}, RK = \{2, 1, 3\}, \quad CK = \{3, 1, 2\}.$$

Now, one can form a permutation matrix B with the help of keys $RK(i)$ and $CK(j), \quad \forall i = 1, 2, 3, \quad$ and $\forall j = 1, 2, 3,$

$$B = \begin{bmatrix} (2,3) & (2,1) & (2,2) \\ (1,3) & (1,1) & (1,2) \\ (3,3) & (3,1) & (3,2) \end{bmatrix}.$$

Shuffle the position of each entry of matrix A by using permutation matrix B to get $A^{'}$

$$A' = \begin{bmatrix} x_{23} & x_{21} & x_{22} \\ x_{13} & x_{11} & x_{12} \\ x_{33} & x_{31} & x_{32} \end{bmatrix} = \begin{bmatrix} Ne & Ve & Ea \\ Me & Sa & Ju \\ Ve & Ma & Ur \end{bmatrix}.$$

This resembles random shuffling over virtual planets and shuffles as well as change each pixel value in encoded domain.

(v) The above analysis in step 3 can be performed by taking virtual planets in the original planet order.

Me-110	Ve-100	Ea-001	Ma-010	Ju-011	Sa-101	Ne-000	Ur-111

(vi) Virtual planet decoding: The similar analysis suits is used here as explained in step 1.

5 Simulated results, performance and security analysis

Here standard images of size 512×512×3 are considered (as shown in Fig. 11). The original images (Baboon 11a, Lena 11d, Fruits 11g, and Peppers 11j) are encrypted and shown in the Fig. 11b, e, h and k respectively. Correctly decrypted images using the same set of keys are shown in Fig. 11c, f, i and l respectively.

An algorithm is efficient, if it resists all kinds of attacks (such as, cryptanalytic, statistical, cipher text, differential, brute force and entropy attacks). We have performed key space sensitivity, statistical analysis and differential attack to prove robustness of the proposed algorithm.

5.1 Formula for key space

1. There are 40320 planet rules, so choice of incorrect planet rule during decryption will give the wrong image.
2. During encryption a generalized Vigenére-type table is employed to generate keys and the total possible options to fill generalized Vigenére-type table over s_n of size $n \times q$ is $(n!) \times (n! - 1) \times (n! - 2) \times \cdots \times (n! - q + 1)$.
3. Finally 3 seeds for encryption is used. The maximum value of each seed will be up to $\min(n, q)$, for 3 seeds the total number of possibility is $(\min(n, q))^3$.

So the size of the key space is $40320 \times (n!) \times (n! - 1) \times (n! - 2) \times \cdots \times (n! - q + 1) \times (\min(n, q))^3$ which is good enough to resist brute force attack. Thus, the proposed algorithm can be apt for security purpose on handling of big data.

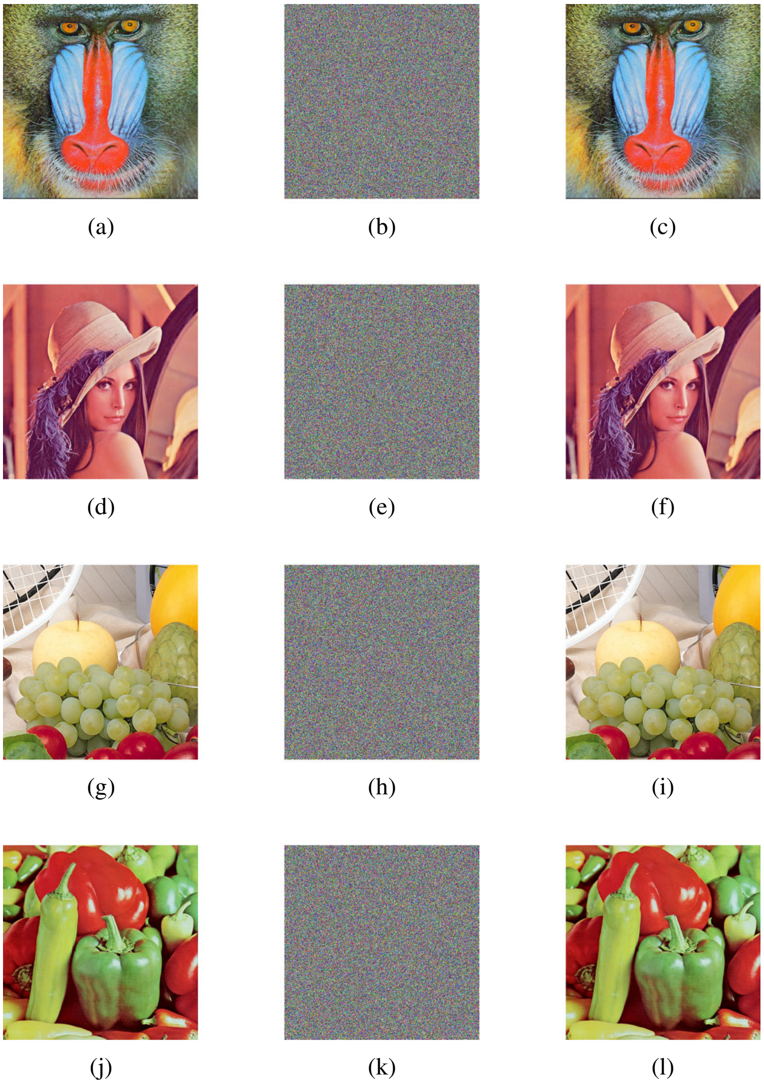


Fig. 11 Encryption and decryption Results using Rule-21 (given in Table 4, Vigenére-type table and three seeds $P_1 = Q_1 = R_1 = 235$) : **a** original Baboon image, **b** encrypted Baboon image, **c** correctly decrypted Baboon image, **d** original Lena image, **e** encrypted Lena image, **f** correctly decrypted Lena image, **g** original Peppers image, **h** encrypted Peppers image, **i** correctly decrypted Peppers image, **j** original fruits image, **k** encrypted fruits image, and **l** correctly decrypted fruits image

5.2 Key sensitivity analysis

High key sensitivity is required for secure image crypto-system, so that the original image can not be decrypted correctly, even though there is a marginal difference in the correct key. Using these keys, Fig. 11b is decrypted, but original image can never be retrieved as shown

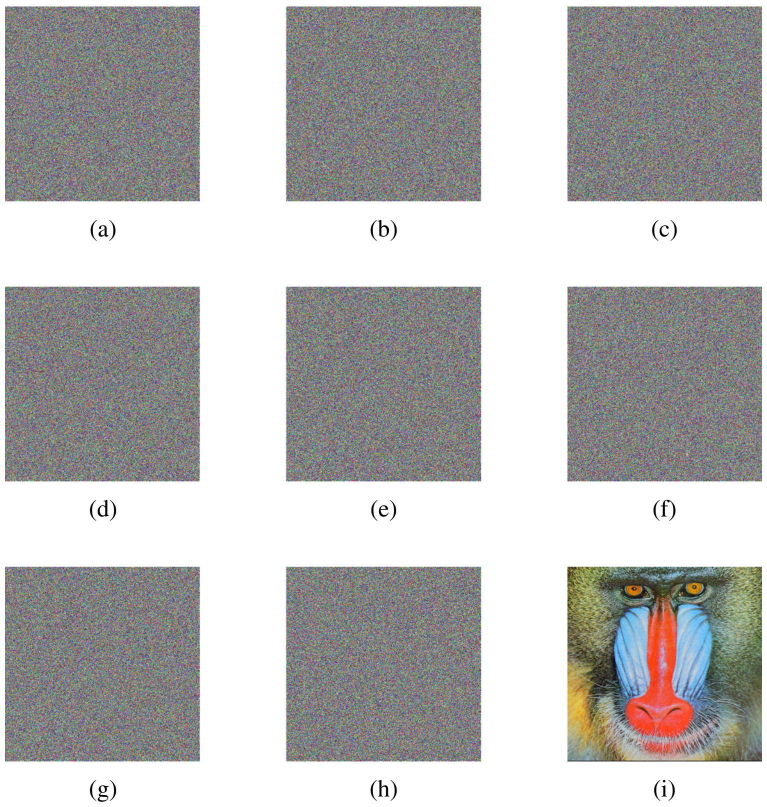


Fig. 12 Key sensitivity results: **a** incorrectly decrypted image using wrong seeds used for key generation in step 3 of Section 2.1 ($P_1 = 235$, $Q_1 = 235$, $R_1 + 1 = 236$), **b** incorrectly decrypted image using wrong seeds used for key generation in step 3 of Section 2.1 ($P_1 + 1 = 236$, $Q_1 = 235$, $R_1 = 235$), **c** incorrectly decrypted image using wrong seeds used for key generation in step 3 of Section 2.1 ($P_1 = 235$, $Q_1 - 1 = 234$, $R_1 = 235$), **d** incorrectly decrypted image using extended vigenére table by columns 1 and 235 interchanged, **e** incorrectly decrypted image using extended vigenére table by columns 1 and 236 interchanged, **f** incorrectly decrypted image using extended vigenére table by columns 1 and 234 interchanged, **g** incorrectly decrypted image extended vigenére table by columns 235 and 236 interchanged, **h** incorrectly decrypted image with shifting the position of planets order (Ne-001 and Ea-000) while original order is (Ne-000 and Ea-001) as Rule-21. **i** correctly decrypted image with all correct keys

in Fig. 12a, b, c, d, e, f, g and h. Even for small changes in encoding scheme the decrypted image changes drastically. Hence, from the above key space analysis, one can infer that the proposed algorithm is highly key sensitive.

5.3 Statistical analysis

Statistical analysis on the proposed algorithm, demonstrates its superior confusion and diffusion properties that can strongly resist statistical attack. This is done by testing the distribution of pixels of the cipher-images and study of correlation among the adjacent pixels in the cipher-image. The details of statistical analysis are provided in the following Sub-Sections.

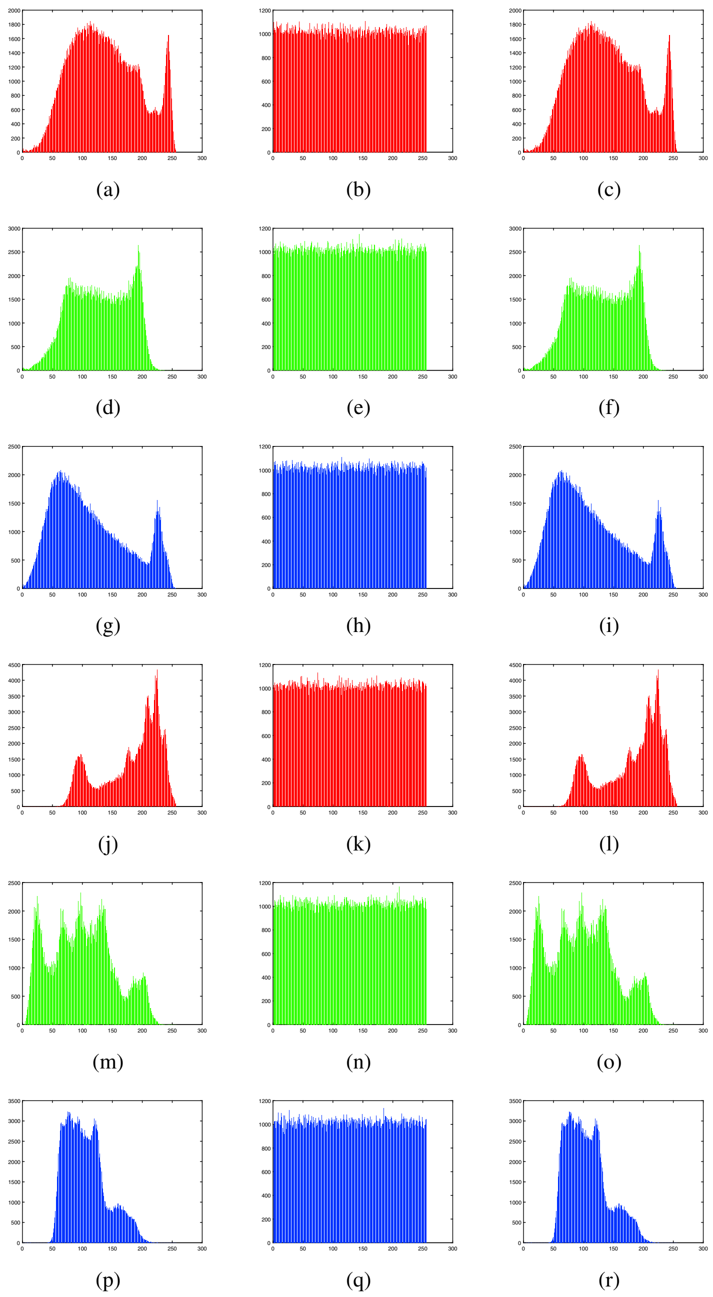


Fig. 13 Histogram analysis: **a** original Baboon red component, **b** encrypted Baboon red component, **c** decrypted Baboon red component, **d** original Baboon green component, **e** encrypted Baboon green component, **f** decrypted Baboon green component, **g** original Baboon blue component, **h** encrypted Baboon blue component, **i** decrypted Baboon blue component, **j** original Lena red component, **k** encrypted Lena red component, **l** decrypted Lena red component, **m** original Lena green component, **n** encrypted Lena green component, **o** decrypted Lena green component, **p** original Lena blue component, **q** encrypted Lena blue component, **r** decrypted Lena blue component

5.3.1 Histograms of the encrypted images

Histogram is a graphical representation of the pixel intensity distribution of an image. Therefore, histogram provides a clear illustration of how the pixels in an image are distributed by plotting the number of pixels at each intensity level. Histograms of the red, green and blue components of original images (Fig. 11a and d) are shown in Fig. 13a, d, g, j, m and p respectively. And histograms of the red, green and blue components of cipher-image (Fig. 11b and e) are shown in Fig. 13b, e, h, k, n and q respectively. Observe that the histograms of cipher-image is almost constant for verity of images. The interesting fact about the proposed algorithm is that, it works well for any kind of image. Therefore, the proposed algorithm will work perfectly for variety of images.

5.3.2 The mean square error and peak signal-to-noise ratio

The Mean Square Error (MSE) between the original and encrypted/decrypted image is calculated by the following formula:

$$MSE = \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} [f(i,j) - f_o(i,j)]^2, \tag{13}$$

where f and f_o are the intensity functions of encrypted/decrypted and original images and (i, j) the position of pixels. The zero values of MSE between original and decrypted image (Table 8), demonstrates that the original image is unchanged by the proposed algorithm.

Peak Signal-to-Noise Ratio (PSNR) between the original and decrypted image is calculated by the following formula:

$$PSNR = 20 \times log \frac{MAX_I^2}{\sqrt{MSE}}, \tag{14}$$

where MAX_I represents the maximum intensity value presented in the image. The PSNR value between original and encrypted images should be sufficiently small to ensure the originality between original and encrypted images. For the proposed algorithm, MSE and PSNR between original and decrypted images are turns out to be always zero and infinity respectively and shown in the Table 8. Hence, the values presented in the Table 8 confirm that the proposed algorithm renders full information without any loss in the original image. Table 9 shows that the MSE and PSNR between original and encrypted images are depicting encrypted image pixels are distributed randomly.

Table 8 MSE-PSNR between original and decrypted image of Baboon, Lena, Fruits and peppers

RGB components of image	MSE	PSNR
R	0	∞
G	0	∞
B	0	∞

Table 9 Mean square error and signal to noise power ratio between original and encrypted images of Baboon, Lena, Fruits and Peppers respectively

	Baboon		Lena		Fruits		Peppers	
RGB Components of image	MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR
R	8.6615e+03	8.7549	1.0657e+04	7.8543	1.1164e+04	7.6518	8.0051e+03	9.0972
G	7.7537e+03	9.2357	9.0921e+03	8.5442	9.9729e+03	8.1426	1.1237e+04	7.6243
B	9.4352e+03	8.3833	7.0973e+03	9.6198	9.1072e+03	8.5370	1.1144e+04	7.6606

5.3.3 Correlation analysis

Correlation between two adjacent pixels can be tested in three ways, by taking two vertically adjacent pixels or by taking two horizontally adjacent pixels or by taking two diagonally adjacent pixels of the encrypted image as shown in Tables 10, 11 and 12. One can randomly select 10,000 pairs of adjacent pixels (for horizontal and vertical) and 1,000 pairs of adjacent pixels (for diagonal) then calculate their correlation coefficient; r_{xy} of each adjacent pair is calculated for the plain and ciphered images using the following formulas:

$$\left.\begin{array}{l} r_{xy} = \frac{cov(x,y)}{\sqrt{D_x}\sqrt{D_y}}, \\ cov(x,y) = E\left[(x-E(x))(y-E(y))\right], \\ E(x) = \frac{1}{L}\sum_{i=1}^{L} x_i, \\ D_x = \frac{1}{L}\sum_{i=1}^{L}(x_i - E(x))^2, \end{array}\right\} \tag{15}$$

where x and y are values of two adjacent pixels in each channel, and L denotes the total number of samples taken. This phase essentially shows that after encryption the correlation among the image pixels is broken whereas decryption will bind the pixels with the original correlation. The correlation plots are presented in Figs. 14 and 15.

5.3.4 Differential attacks

Variations in the original image can be made by adversary, who then make use of the algorithm which is proposed to encrypt the original image before and after changing pixels, and through estimating similarity of two encrypted images, they discover the relationship

Table 10 Horizontal correlation of Baboon, Lena, Peppers and Fruits images over 10,000 points

Image	Baboon		Lena		Fruits		Peppers	
components	Plain image	Cipher image	Plain image	Cipher image	Plain image	Cipher image	Plain image	Cipher image
R	0.9320	1.5237e-04	0.9464	5.8436e-05	0.9699	3.5840e-06	0.9304	1.3688e-04
G	0.8058	1.1662e-05	0.8970	4.2606e-05	0.9622	2.0231e-07	0.9793	9.9400e-05
B	0.8466	1.4087e-04	0.9290	1.6229e-04	0.9394	5.0299e-05	0.9318	4.4381e-05

Table 11 Vertical correlation of Baboon, Lena, Peppers and Fruits images over 10,000 points

Image	Baboon		Lena		Fruits		Peppers	
components	Plain image	Cipher image	Plain image	Cipher image	Plain image	Cipher image	Plain image	Cipher image
R	0.8458	1.3977e− 04	0.9822	2.4262e− 04	0.9492	1.7866e− 05	0.9285	2.8312e− 05
G	0.8444	4.5896e− 04	0.9714	2.0516e− 04	0.9581	5.7035e− 05	0.9683	2.4262e− 06
B	0.8908	1.9685e− 04	0.8926	1.0307e− 04	0.9012	1.0376e− 04	0.9751	6.1088e− 06

between original and encrypted images. However, if any minor change in the original image causes a large change in the encrypted image then the differential attack becomes useless. In this paper, a new method to improve the capacity to withstand differential attacks by calculating the total occurrences of each virtual planet in the encoded image. To analyze the capacity of resisting differential attacks, NPCR and UACI analysis [21] are often used.

5.3.5 NPCR analysis

The Number of Pixels Change Rate (NPCR) refers to the change rate of the number of pixels of ciphered image while one pixel of plain-image is changed. And it is calculated as follows:

$$NPCR = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} D(i, j)}{N \times M} \times 100. \tag{16}$$

If $C^1(i, j) = C^2(i, j)$ then $D(i, j) = 0$. Otherwise $D(i, j) = 1$.

5.3.6 UACI analysis

The Unified Average Change Intensity (UACI) measures the average intensity of differences between two ciphered images. It is mathematically defined as

$$UACI = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} \frac{C^1(i,j)-C^2(i,j)}{255}}{N \times M} \times 100, \tag{17}$$

where C^1 and C^2 are encrypted images before and after one pixel change in original image.

Table 13 shows the values of NPCR ($> 99\%$) and UACI (33%) for each color component between two cipher-image's. Experimental results show the estimated expectations and

Table 12 Diagonal correlation of Baboon, Lena, Peppers and Fruits images over 1,000 points

Image	Baboon		Lena		Fruits		Peppers	
components	Plain image	Cipher image	Plain image	Cipher image	Plain image	Cipher image	Plain image	Cipher image
R	0.7406	10018e− 04	0.9042	2.7345e− 05	0.8930	9.2457e− 05	0.8850	3.9007e− 04
G	0.7609	2.0336e− 04	0.8456	1.9034e− 05	0.9129	2.0034e− 05	0.9192	3.9299e− 04
B	0.7519	7.9664e− 04	0.7387	0.7387e− 05	0.9321	5.8810e− 04	0.8608	1.2745e− 05

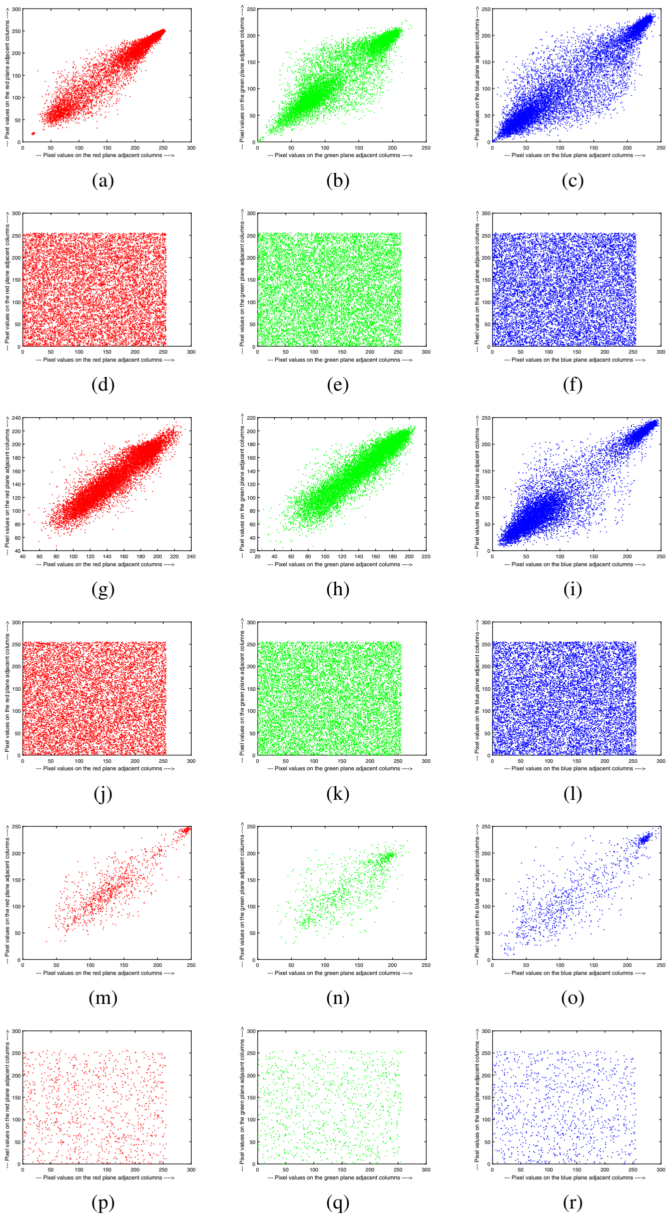


Fig. 14 Correlation analysis for Baboon image: **a** horizontal correlation of original red component, **b** horizontal correlation of original green component, **c** horizontal correlation of original blue component, **d** horizontal correlation of encrypted red component, **e** horizontal correlation of encrypted green component, **f** horizontal correlation of encrypted blue component, (g) vertical correlation of original red component, **h** vertical correlation of original green component, **i** vertical correlation of original blue component, **j** vertical correlation of encrypted red component, **k** vertical correlation of encrypted green component, **l** vertical correlation of encrypted blue component, **m** diagonal correlation of original red component, **n** diagonal correlation of original green component, **o** diagonal correlation of original blue component, **p** diagonal correlation of encrypted red component, **q** diagonal correlation of encrypted green component, **r** diagonal correlation of encrypted blue component

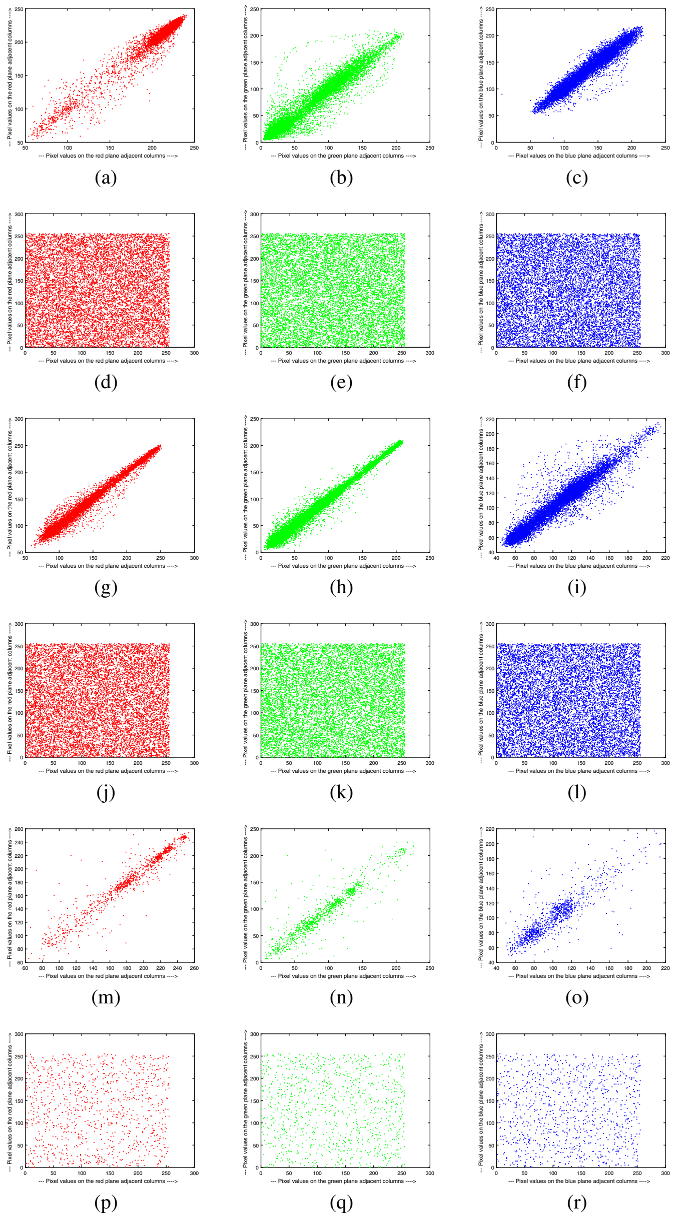


Fig. 15 Correlation analysis for Lena image: **a** horizontal correlation of original red component, **b** horizontal correlation of original green component, **c** horizontal correlation of original blue component, **d** horizontal correlation of encrypted red component, **e** horizontal correlation of encrypted green component, **f** horizontal correlation of encrypted blue component, **g** vertical correlation of original red component, **h** vertical correlation of original green component, **i** vertical correlation of original blue component, **j** vertical correlation of encrypted red component, **k** vertical correlation of encrypted green component, **l** vertical correlation of encrypted blue component, **m** diagonal correlation of original red component, **n** diagonal correlation of original green component, **o** diagonal correlation of original blue component, **p** diagonal correlation of encrypted red component, **q** diagonal correlation of encrypted green component, **r** diagonal correlation of encrypted blue component

Table 13 NPCR and UACI values of the proposed algorithm

Testing cipher-image	NPCR(%)			UACI(%)		
	R	G	B	R	G	B
Baboon	99.6056	99.6105	99.5956	33.4571	33.4763	33.4852
Lena	99.6155	99.6110	99.6271	33.4373	33.4744	33.3921
Peppers	99.6075	99.6098	99.6166	33.4645	33.5029	33.4013
Average of each component	99.6095	99.6102	99.6131	33.4530	33.4845	33.4262
Average for all images	99.6109			33.4546		

variance of NPCR and UACI are very close to the theoretical values, which justify the validity of theoretical values [14]. Hence, the proposed scheme is immune against differential attacks.

5.3.7 Statistical test for NPCR

From [21], one can demonstrate the proposed algorithm is good by using statistical test for NPCR. Suppose we have two cipher-images C^1 and C^2 of size $512 \times 512 \times 3$ each, then hypotheses (H_0 and H_1) with significance level α for $N(C^1, C^2)$ are

$$H_0 : N(C^1, C^2) = \mu_N, \tag{18}$$

$$H_1 : N(C^1, C^2) < \mu_N. \tag{19}$$

Reject H_0, if $N(C^1, C^2) < N^*_\alpha$, otherwise accept H_0, where

$$N^*_\alpha = \mu_N - \phi^{-1}(\alpha)\sigma_N = \frac{\left(F - \phi^{-1}(\alpha)\sqrt{\frac{F}{MN}}\right)}{F+1}, \tag{20}$$

$$\mu_N = \frac{F}{F+1}, \tag{21}$$

$$\sigma_N^2 = \frac{F}{(F+1)^2 MN}, \tag{22}$$

where F largest pixel value in the original image.

Table 14 Statistical test for NPCR

Testing cipher-image	F = 255				
	μ_N	σ_N	$N^*_{0.05}$	$N^*_{0.01}$	$N^*_{0.001}$
Numerical values	99.6094	0.0122	99.5893	99.5810	99.5717
Baboon (99.6038)			PASS	PASS	PASS
Lena (99.6179)			PASS	PASS	PASS
Peppers (99.6101)			PASS	PASS	PASS

Observe from the Table 14, $N(C^1, C^2)$ values for Baboon, Lena, and Peppers exceeds N^*_α values for $\alpha = 0.05, 0.01, 0.001$. So we can accept the null hypothesis (H_0). Hence, the NPCR values confirm that the proposed algorithm is good.

5.3.8 Statistical test for UACI

Likewise, again from [21], one can demonstrate the proposed algorithm is good by using statistical test for UACI. Assuming that, we have two cipher-images C^1 and C^2 of size $512 \times 512 \times 3$ each, then hypotheses (H_0 and H_1) with significance level α for $U(C^1, C^2)$ are

$$H_0 : U(C^1, C^2) = \mu_U, \tag{23}$$

$$H_1 : N(C^1, C^2) < \mu_U, \tag{24}$$

Reject H_0, if $U(C^1, C^2) \notin (U^{*+}_\alpha, U^{*-}_\alpha)$, otherwise accept H_0, where

$$U^{*+}_\alpha = \mu_U + \phi^{-1}(\alpha/2)\sigma_U, \tag{25}$$

$$U^{*-}_\alpha = \mu_U - \phi^{-1}(\alpha/2)\sigma_U, \tag{26}$$

$$\mu_U = \frac{F+2}{3F+3}, \tag{27}$$

$$\sigma_U^2 = \frac{(F+2)(F^2+2F+3)}{18(F+1)^2 MNF}, \tag{28}$$

Notice from Table 15, $U(C^1, C^2)$ values for Baboon, Lena, and Peppers belongs to the interval $(U^{*+}_\alpha, U^{*-}_\alpha)$ for $\alpha = 0.05, 0.01, 0.001$. So we can accept the null hypothesis (H_0). Hence, the UACI values confirm that the proposed algorithm is good.

5.3.9 Cropped attack analysis

The cropped attack analysis has been performed to check the robustness against data loss for real-time applications of the proposed algorithm. Two different types of cropped images have been decrypted using the correct set of keys. One image contains cropping on different position of layers (R, G, and B) in the image, whereas the second image contains cropping on same position. The main objective of the analysis is to make sure that the decrypted image can be recognized visually. From Fig. 16, one can see that the proposed algorithm is robust against cropping attack.

Table 15 Statistical test for UACI

Testing cipher-image	F = 255				
	μ_U	σ_U	$U^{*+}_{0.05}/U^{*-}_{0.05}$	$U^{*+}_{0.01}/U^{*-}_{0.01}$	$U^{*+}_{0.001}/U^{*-}_{0.001}$
Numerical values	33.4635	0.0462	33.3730	33.3445	33.3115
			33.5541	33.5826	33.6156
Baboon (33.4729)			PASS	PASS	PASS
Lena (33.4346)			PASS	PASS	PASS
Peppers (33.4562)			PASS	PASS	PASS

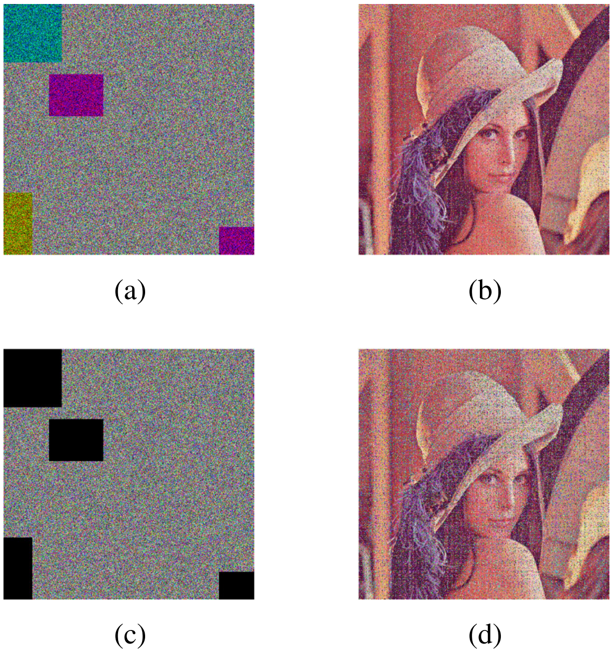


Fig. 16 Cropped attack experiment: **a** encrypted image with different locations from each layer (R, G, and B) is cropped **b** decrypted result; **c** encrypted image with same locations from each layer (R, G, and B) is cropped; **d** decrypted result

5.3.10 Transmission of noise attack analysis

Transmission loss analysis has been performed to analyze the robustness of the algorithm to the noise that might be picked up during real-time transmission of images. The analysis is performed by introducing salt and pepper noise in the encrypted image synthetically. The image is then decrypted using the correct set of keys and the decrypted image is checked for visual recognition. Figure 17, shows that the proposed algorithm is invulnerable against noise attack.

5.3.11 Information entropy analysis

The information entropy is an important model for scaling the uncertainty [11, 18] and distribution of the gray values present in the image [22, 31]. As gray values scattered more uniformly, the information entropy is nearest to its ideal value:

$$H(s) = \sum_{i=0}^{2^N-1} p(s_i) \log_2 \left(\frac{1}{p(s_i)} \right), \tag{29}$$

where $p(s_i)$ is the probability of mark s_i presence. The information entropy for a solely random information source s with 2^N states is $H(s) = N$ bits. As the information source have 8 bit gray image, the information entropy is $H(s) = 8$ bits. For our proposed encryption algorithm information entropies are given in Table 16 evidence that the proposed algorithm is robust against the entropy attack.

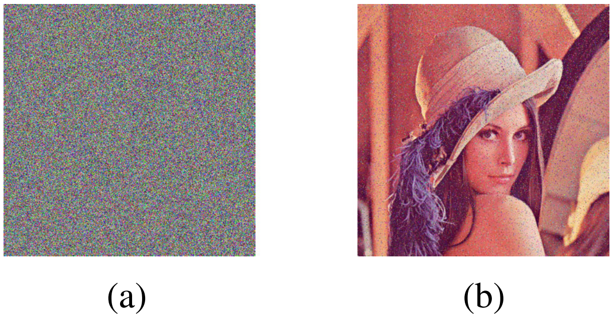


Fig. 17 Transmission noise attack analysis: **a** noise attack image; **b** decrypted result

5.4 Run time of the algorithm

The proposed algorithm is designed in such a way that the encryption and decryption process take 0.5853 seconds with 16 core processor, Windows 10 Pro and MATLAB R2016a. Thus, one can conclude that the run time of the proposed algorithm is less. The algorithm deals only with XOR, addition and substitution steps for which the run time is of $O(1)$ for every pixel. As we are dealing with RGB image of size $N \times M \times 3$, then the run time of proposed algorithm is of $O(N \times M)$.

6 Efficiency of the encoding scheme

The performance of our proposed algorithm is shown on VPD.

i Observe from the Fig. 18a and b decrypting with wrong decryption permutation key (CK) results in wrong decrypted image. If one observes carefully the results from Fig. 18a, results in completely different randomized picture from correct image (Fig. 18c) in encoded domain.

  (a) Virtual planet encoding,
  (b) Permutation shuffling,
  (c) Virtual planet decoding.

  Here, Fig. 18b (steps 1 and 3 are ignored) is less effective to hide original image decrypted using wrong decryption permutation key (CK) in normal RGB domain. This explicitly shows the effect of our VPD.

ii In [26], shuffling process in DNA encoded domain fails when each pixels are same (i.e. either 0, or 85, or 170 or 255) which reveals the secret keys. For an instance, that $R = R_{ij}$, $G = G_{ij}$ and $B = B_{ij}$ layers and let the binary representation of one pixel (say R_{11}, B_{11} and G_{11}) from each layer be $r_1r_2r_3r_4r_5r_6r_7r_8$, $g_1g_2g_3g_4g_5g_6g_7g_8$ and

Table 16 The result of information entropy

RGB Components	R	G	B
Baboon	7.9997	7.9999	7.9994
Lena	7.9999	7.9997	7.9998
Fruits	7.9996	7.9997	7.9995
Peppers	7.9995	7.9998	7.9997

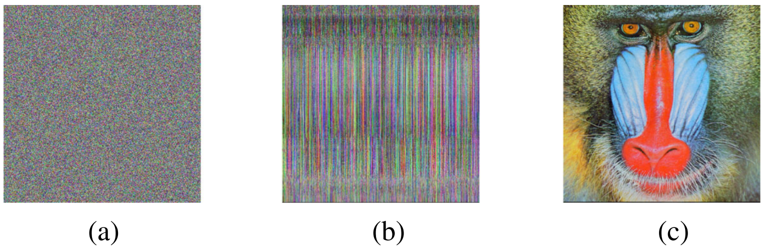


Fig. 18 Performance analysis of encoding scheme: **a** represents incorrectly decrypted image after applying permutation decryption with wrong column key (generated using $seed_1 + 1$) in the encoded domain, **b** represents incorrectly decrypted image after applying permutation decryption with wrong column key (generated using $seed_1 + 1$) in normal domain, **c** represents correctly decrypted image after applying permutation decryption

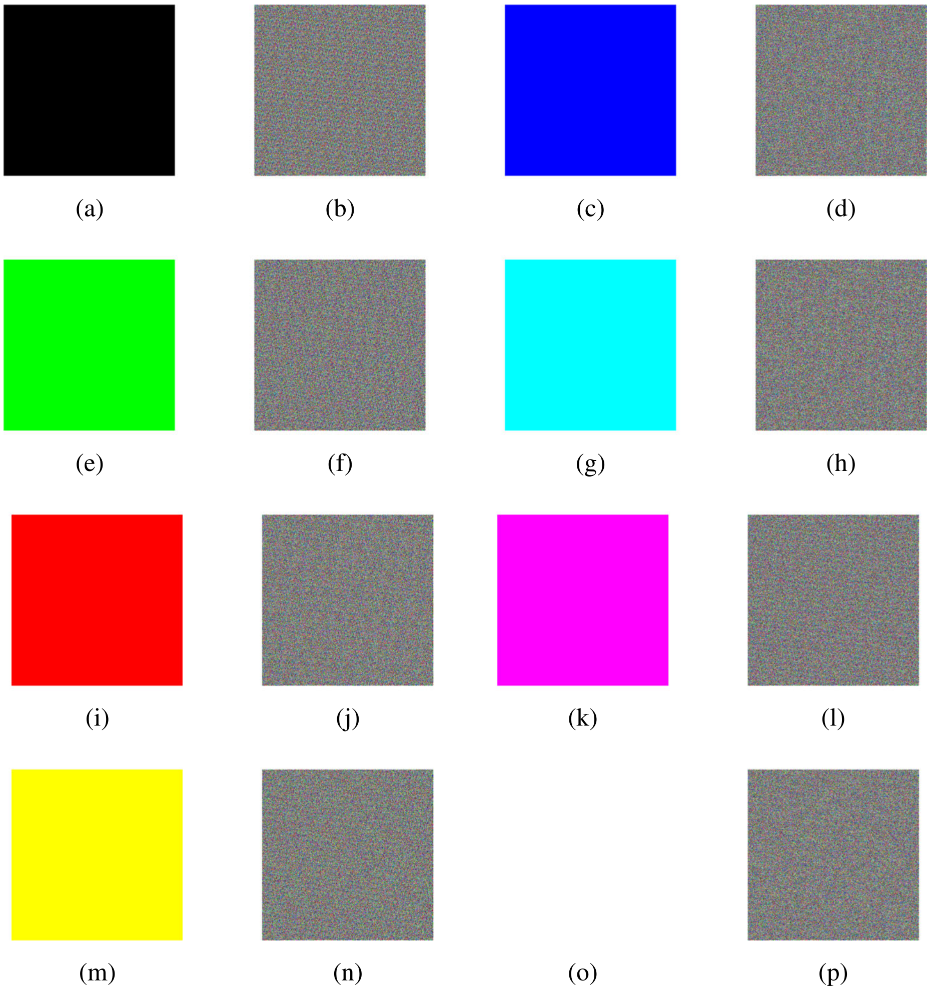


Fig. 19 $\forall i, j$ $[R_{ij}, B_{ij}, B_{ij}]$: **a** [0 0 0], **b** encrypted [0 0 0], **c** [0 0 255], **d** encrypted [0 0 255], **e** [0 255 0], **f** encrypted [0 255 0], **g** [0 255 255], **h** encrypted [0 255 255], **i** [255 0 0], **j** encrypted [255 0 0], **k** [255 0 255], **l** encrypted [255 0 255], **m** [255 255 0], **n** encrypted [255 255 0], **o** [255 255 255], **p** encrypted [255 255 255]

Table 17 Comparison of key space of the proposed algorithm with other existing algorithms

Algorithm	Key space
Ref. [9]	10^{56} (fixed size)
Ref. [20]	10^{70} (fixed size)
Ref. [15]	3.40×10^{38} (fixed size)
Ref. [2]	10^{70} (fixed size)
Ref. [4]	10^{80} (fixed size)
Proposed algorithm	$40320 \times (n!) \times (n! - 1) \times (n! - 2) \times \cdots \times (n! - q + 1) \times (\min(n, q))^3$ (variable size) $> 10^{5040}$ for $n = 255, q = 10$

Table 18 Comparison on NPCR analysis of the proposed method with authors [4, 6, 20, 23, 24, 31]

	NPCR (%)						
RGB Components	Proposed method	Ref. [6]	Ref. [20]	Ref. [31]	Ref. [24]	Ref. [23]	Ref. [4]
Red	99.6095	99.5659	99.58649	–	99.6100	99.6137	99.5926
Green	99.6102	99.5658	99.21722	–	99.6092	99.6053	99.6216
Blue	99.6131	99.5959	98.84796	–	99.6099	99.6079	99.6323
Average	99.6109	99.5759	99.55047	99.6097	99.6089	99.6117	99.6155

Table 19 Comparison on UACI analysis of the proposed method with authors [4, 6, 20, 23, 24, 31]

	UACI (%)						
RGB Components	Proposed method	Ref. [6]	Ref. [20]	Ref. [31]	Ref. [24]	Ref. [23]	Ref. [4]
Red	33.4530	33.2829	33.48347	–	33.4639	33.4655	33.3386
Green	33.4845	33.3459	33.46399	–	33.5042	33.4781	33.3595
Blue	33.4262	33.3270	33.26891	–	33.4776	33.4746	33.1250
Average	33.4545	33.3186	33.40545	33.4500	33.4819	33.4727	33.2743

Table 20 Comparison of horizontal correlation of encrypted image of Lena image with authors [4, 6, 7, 20, 23, 31]

RGB Components	Proposed algorithm	Ref. [6]	Ref. [20]	Ref. [7]	Ref. [31]	Ref. [23]	Ref. [4]
Red	0.000127	0.0181	0.0054	0.003535	–	0.9244	–
Green	0.000042	– 0.0067	0.0059	– 0.009706	–	0.9366	–
Blue	0.000162	0.0154	0.0013	0.018571	–	0.8545	–
Average	0.000110	0.00893	0.0042	0.004133	0.0214	0.9052	0.0265

Table 21 Comparison of run time of the proposed algorithm with other existing algorithms

Algorithm	Running time (Mbits/second)
Ref. [22]	3.36
Ref. [23]	4.87
Proposed algorithm	10.74

$b_1b_2b_3b_4b_5b_6b_7b_8$ receptively. If we design our proposed algorithm based on [26], then attack is possible if all planets in encoded image are equal i.e. $r_1g_1b_1 = r_2g_2b_2 = r_3g_3b_3 = \ldots = r_8g_8b_8$ for all pixels. So there are 8 possible pictures that are $\forall i, j, [R_{ij}, B_{ij}, B_{ij}]$ = [0 0 0], [0 0 255], [0 255 0], [0 255 255], [255 0 0], [255 0 255], [255 255 0], [255 255 255]. However, in our proposed encoding scheme, we have introduced a series of complex diffusion processes that resist against vulnerabilities of chosen plain/cipher text attacks. The encrypted images for above mentioned eight pictures are demonstrated in Fig. 19.

7 Comparison

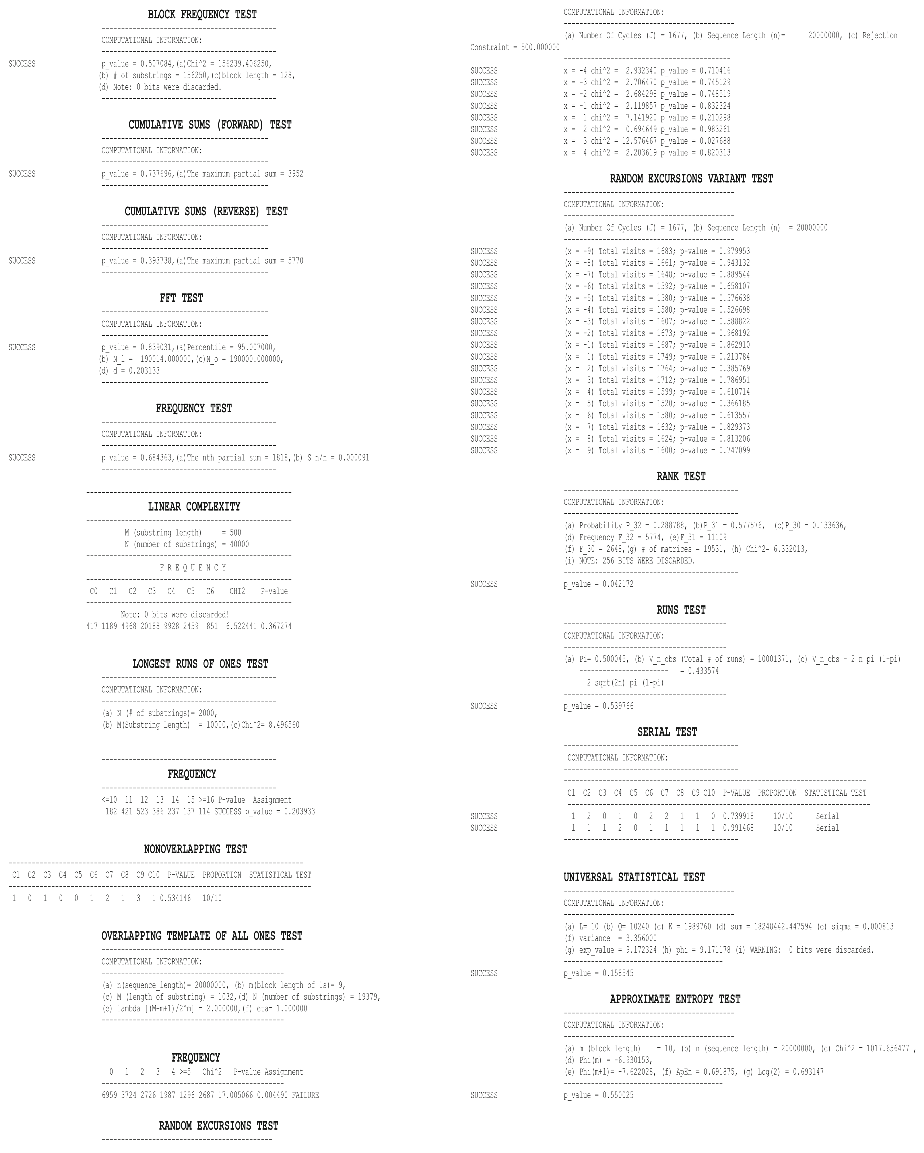
The proposed approach has been compared with the existing techniques and can be seen in Tables 17–21. For a healthy cryptosystem, it is very important to have a huge key space. Table 17 shows that the proposed algorithm has large key space as compared to others. Further, the proposed key space is designed in such a way that, it depends on the size of the image, which provides brute force attack infeasible. Furthermore, data presented in Tables 18 and 19 confirm that the proposed algorithm emerged as the best by ensuring NPCR and UACI up to an optimal level. Correlation coefficients of the proposed and existing algorithms of encrypted image have been shown in Table 20 which shows that the proposed technique has superior correlation between pixels, hence provides robust security against statistical attacks [6, 7, 15]. The data presented in Table 21, shows runtime efficiency of the proposed algorithm. Therefore, from the above analyses, it can be easily seen that the proposed algorithm is efficient, robust and can encrypt RGB images securely as compared to others [2, 4, 6, 7, 9, 15, 20, 23, 24, 31].

8 Conclusion

In this paper, a *new*, *fast* and *secure* RGB image encryption algorithm in VPD is proposed. An advance idea is introduced to construct VPD by using the eight virtual planets (i.e., Me, Ve, Ea, Ma, Ju, Sa, Ur and Ne) from the virtual solar system. This VPD offers to encode RGB image into planet domain, by using any one of the rule out of 40320 rules, which enhance the size of key space and its security. Further the VPD is diffused with original planet domain (OPD) to provide more randomness. The key space resists common attacks (such as, differential, cropped, noise, brute force and entropy attacks, etc.). Figure 12 shows that, the secret keys are highly sensitive and are influenced by the seeds, VPD rules and generalized Vigenére tables. Comparison with existing methods have been done and results are reported in Tables 17–20. The data presented in Tables 17–20 confirm the superiority of the proposed algorithm. Further, key spaces, key sensitivity, histogram analysis, correlation plots, correlation coefficients, MSE, PSNR, NPCR, UACI, and statistical tests for NPCR and UACI have been performed and the results of these analysis suggest that the proposed algorithm can resists towards the exhaustive attacks effectively and can be apt for practical applications.

Acknowledgments We thank to the two anonymous referees for their valuable comments. The first author is thankful to the Science & Engineering Research Board, Government of India for providing financial support through project file no. YSS/2015/000930.

Appendix



```
                          BLOCK FREQUENCY TEST
                   ---------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   ---------------------------------------------
SUCCESS            p_value = 0.507084,(a)Chi^2 = 156239.406250,
                  (b) # of substrings = 156250,(c)block length = 128,
                  (d) Note: 0 bits were discarded.
                   ---------------------------------------------

                      CUMULATIVE SUMS (FORWARD) TEST
                   -------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   -------------------------------------------
SUCCESS            p_value = 0.737696,(a)The maximum partial sum = 3952
                   -------------------------------------------

                      CUMULATIVE SUMS (REVERSE) TEST
                   -------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   -------------------------------------------
SUCCESS            p_value = 0.393738,(a)The maximum partial sum = 5770
                   -------------------------------------------

                               FFT TEST
                   -------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   -------------------------------------------
SUCCESS            p_value = 0.839031,(a)Percentile = 95.007000,
                  (b) N_l =  190014.000000,(c)N_o = 190000.000000,
                  (d) d = 0.203133
                   -------------------------------------------

                            FREQUENCY TEST
                   ---------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   ---------------------------------------------
SUCCESS            p_value = 0.684363,(a)The nth partial sum = 1818,(b) S_n/n = 0.000091
                   ---------------------------------------------

                ----------------------------------------------------
                            LINEAR COMPLEXITY
                ----------------------------------------------------
                      M (substring length)     = 500
                      N (number of substrings) = 40000
                ----------------------------------------------------
                              F R E Q U E N C Y
                ----------------------------------------------------
                 C0   C1   C2   C3   C4   C5   C6    CHI2    P-value
                ----------------------------------------------------
                      Note: 0 bits were discarded!
                417 1189 4968 20188 9928 2459  851  6.522441 0.367274

                         LONGEST RUNS OF ONES TEST
                   ---------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   ---------------------------------------------
                   (a) N (# of substrings)= 2000,
                   (b) M(Substring Length)  = 10000,(c)Chi^2= 8.496560


                   ---------------------------------------------
                                FREQUENCY
                   ---------------------------------------------
                   <=10  11  12  13  14  15 >=16 P-value  Assignment
                    182 421 523 386 237 137 114 SUCCESS p_value = 0.203933

                          NONOVERLAPPING TEST
-------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
-------------------------------------------------------------------------------
 1   0   1   0   0   1   2   1   3   1 0.534146   10/10

                   OVERLAPPING TEMPLATE OF ALL ONES TEST
                   -----------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   -----------------------------------------------
                   (a) n(sequence_length)= 20000000, (b) m(block length of 1s)= 9,
                   (c) M (length of substring) = 1032,(d) N (number of substrings) = 19379,
                   (e) lambda [(M-m+1)/2^m] = 2.000000,(f) eta= 1.000000
                   -----------------------------------------------


                                FREQUENCY
                     0   1   2   3   4 >=5   Chi^2   P-value Assignment
                   -----------------------------------------------
                   6959 3724 2726 1987 1296 2687 17.005066 0.004490 FAILURE

                          RANDOM EXCURSIONS TEST
                   --------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   --------------------------------------------
                   (a) Number Of Cycles (J) = 1677, (b) Sequence Length (n)=     20000000, (c) Rejection
Constraint = 500.000000
                   --------------------------------------------
SUCCESS            x = -4 chi^2 =  2.932340 p_value = 0.710416
SUCCESS            x = -3 chi^2 =  2.706470 p_value = 0.745129
SUCCESS            x = -2 chi^2 =  2.684298 p_value = 0.748519
SUCCESS            x = -1 chi^2 =  2.119857 p_value = 0.832324
SUCCESS            x =  1 chi^2 =  7.141920 p_value = 0.210298
SUCCESS            x =  2 chi^2 =  0.694649 p_value = 0.983261
SUCCESS            x =  3 chi^2 = 12.576467 p_value = 0.027688
SUCCESS            x =  4 chi^2 =  2.203619 p_value = 0.820313

                        RANDOM EXCURSIONS VARIANT TEST
                   --------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   --------------------------------------------
                   (a) Number Of Cycles (J) = 1677, (b) Sequence Length (n)  = 20000000
                   --------------------------------------------
SUCCESS            (x = -9) Total visits = 1683; p-value = 0.979953
SUCCESS            (x = -8) Total visits = 1661; p-value = 0.943132
SUCCESS            (x = -7) Total visits = 1648; p-value = 0.889544
SUCCESS            (x = -6) Total visits = 1592; p-value = 0.658107
SUCCESS            (x = -5) Total visits = 1580; p-value = 0.576638
SUCCESS            (x = -4) Total visits = 1580; p-value = 0.526698
SUCCESS            (x = -3) Total visits = 1607; p-value = 0.588822
SUCCESS            (x = -2) Total visits = 1673; p-value = 0.968192
SUCCESS            (x = -1) Total visits = 1687; p-value = 0.862910
SUCCESS            (x =  1) Total visits = 1749; p-value = 0.213784
SUCCESS            (x =  2) Total visits = 1764; p-value = 0.385769
SUCCESS            (x =  3) Total visits = 1712; p-value = 0.786951
SUCCESS            (x =  4) Total visits = 1599; p-value = 0.610714
SUCCESS            (x =  5) Total visits = 1520; p-value = 0.366185
SUCCESS            (x =  6) Total visits = 1580; p-value = 0.613557
SUCCESS            (x =  7) Total visits = 1632; p-value = 0.829373
SUCCESS            (x =  8) Total visits = 1624; p-value = 0.813206
SUCCESS            (x =  9) Total visits = 1600; p-value = 0.747099

                                  RANK TEST
                   ---------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   ---------------------------------------------
                   (a) Probability P_32 = 0.288788, (b)P_31 = 0.577576,  (c)P_30 = 0.133636,
                   (d) Frequency F_32 = 5774, (e)F_31 = 11109
                   (f) F_30 = 2648,(g) # of matrices = 19531, (h) Chi^2= 6.332013,
                   (i) NOTE: 256 BITS WERE DISCARDED.
                   ---------------------------------------------
SUCCESS            p_value = 0.042172

                                  RUNS TEST
                   -------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   -------------------------------------------
                   (a) Pi= 0.500045, (b) V_n_obs (Total # of runs) = 10001371, (c) V_n_obs - 2 n pi (1-pi)
                      -----------------------   = 0.433574
                        2 sqrt(2n) pi (1-pi)
                   -------------------------------------------
SUCCESS            p_value = 0.539766

                               SERIAL TEST
                   ---------------------------------------------
                    COMPUTATIONAL INFORMATION:
                   ---------------------------------------------
                   -----------------------------------------------------------------------------
                    C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
                    -----------------------------------------------------------------------------
SUCCESS              1   2   0   1   0   2   2   1   1   0  0.739918    10/10      Serial
SUCCESS              1   1   1   2   0   1   1   1   1   1  0.991468    10/10      Serial
                   ---------------------------------------------


                   UNIVERSAL STATISTICAL TEST
                   --------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   --------------------------------------------
                   (a) L= 10 (b) Q= 10240 (c) K = 1989760 (d) sum = 18248442.447594 (e) sigma = 0.000813
                   (f) variance  = 3.356000
                   (g) exp_value = 9.172324 (h) phi = 9.171178 (i) WARNING:  0 bits were discarded.
                   -----------------------------------------
SUCCESS            p_value = 0.158545

                         APPROXIMATE ENTROPY TEST
                   --------------------------------------------
                   COMPUTATIONAL INFORMATION:
                   --------------------------------------------
                   (a) m (block length)    = 10, (b) n (sequence length) = 20000000, (c) Chi^2 = 1017.656477 ,
                   (d) Phi(m) = -6.930153,
                   (e) Phi(m+1)= -7.622028, (f) ApEn = 0.691875, (g) Log(2) = 0.693147
                   -----------------------------------------
SUCCESS            p_value = 0.550025
```

NIST statistical test suite result

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

References

1. Awdun B, Li G (2016) The color image encryption technology based on DNA encoding & sine chaos. International Conference on Smart City and Systems Engineering, 539–544
2. Gao T, Chen Z (2008) A new image encryption algorithm based on hyper-chaos. Phys Lett A 372:394–400
3. Gehani A, LaBean TH, Reif JH, cryptography DNA-based (2000) DIMACS series in discrete mathematics. Theor Comput Sci 54:23–249
4. Guesmi R, Farah MAB, Kachouri A, Samet M (2015) A novel chaos-based image encryption using DNA sequence operation and secure hash algorithm SHA-2. Nonlinear Dyn 83:1123–1136
5. Head T, Rozenberg G, Bladergroen RS, Breek CKD, Lommerse PHM, Spaink HP (2000) Computing with DNA by operating on plasmids. Biosystems 57:87–93
6. Kumar M, Powduri P, Reddy A (2015) An RGB image encryption using diffusion process associated with chaotic map. J Inf Secur Appl 21:20–30
7. Kumar M, Iqbal A, Kumar P (2016) A new RGB image encryption algorithm based on DNA encoding and elliptic curve Diffie-Hellman cryptography. Signal Process 125:187–202
8. Liu HJ, Wang XY, Kadir A (2012) Image encryption using DNA complementary rule and chaotic maps. Appl Soft Comput 12:1457–1466
9. Liu L, Zhang Q, Wei X (2012) A RGB image encryption algorithm based on DNA encoding and chaos map. Comput Electr Eng 38:1240–1248
10. Malhotra R, Holman M, Ito T (2001) Chaos and stability of the solar system. Proc Natl Acad Sci U S A 98:12342–12343
11. Menezes AJ, Vanstone SA, Oorschot PCV (1997) Handbook of applied cryptography. CRC Press
12. Mokhtar MA, Gobran SN, Badawy ESAME (2014) Colored image encryption algorithm using DNA code and chaos theory. In: 5th international conference on computer & communication engineering, pp 12–15
13. Ozkaynak F, Ozer A, Yavuz S (2013) Security analysis of an image encryption algorithm based on chaos and DNA encoding. Signal Processing and Communications Applications Conference (SIU) 2013 21st: 1–4
14. Pareek NK, Patidar V, Sud KK (2011) A symmetric encryption scheme for color BMP images. International Journal of Computer Application: Special Issue on Network Security and Cryptography, 42–46
15. Pareek NK, Patidar V, Sud KK (2013) Diffusion-substitution based gray image encryption scheme. Digital Signal Process 23:894–901
16. Tong XJ, Wang Z, Zhang M, Liu Y, Xu H, Ma J (2015) An image encryption algorithm based on the perturbed high-dimensional chaotic map. Nonlinear Dyn 80:1493–1508
17. Wang J, Long F, Ou W (2017) CNN-based color image encryption algorithm using DNA sequence operations. International Conference on Security, Pattern Analysis, and Cybernetics, 730–736
18. Wang XY, Chen F, Wang T (2010) A new compound mode of confusion and diffusion for block encryption of image based on chaos. Commun Nonlinear Sci Numer Simul 15:2479–2485
19. Wang Y, Lei P, Yang H, Cao H (2015) Security analysis on a color image encryption based on DNA encoding and chaos map. Comput Electr Eng 46:433–446
20. Wei X, Guo L, Zhang Q, Zhang J, Lian S (2012) A novel color image encryption algorithm based on DNA sequence operation and hyper-chaotic system. J Syst Softw 85:290–299
21. Wu Y, Noonan JP, Agaian S (2011) NPCR and UACI randomness tests for image encryption. Journal of Selected Areas in Telecommunications, 31–38
22. Wu X, Kan H, Kurths J (2015) A new color image encryption scheme based on DNA sequences and multiple improved 1D chaotic maps. Appl Soft Comput 37:24–39
23. Wu X, Wang K, Wang X, Kan H, Kurths J (2018) Color image DNA encryption using NCA map-based CML and one-time keys. Signal Process 148:272–287
24. Wua X, Kana H, Kurthsc J (2015) A new color image encryption scheme based on DNA sequences and multiple improved 1D chaotic maps. Appl Soft Comput 37:24–39
25. Xiao G, Lu M, Qin L, Lai X (2006) New field of cryptography: DNA cryptography. Chin Sci Bull 51:1413–1420
26. Xie T, Liu Y, Tang J (2014) Breaking a novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system. Optik 125:7166–7169
27. Zhang Q, Liu Wei LL (2014) Improved algorithm for image encryption based on DNA encoding and multi-chaotic mapts. AEU-Inter J Electron Commun 68:186–192

28. Zhang Q, Guo L, Wei X (2013) A novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system. Optik 124(18):3596–3600
29. Zhang Y (2015) Cryptanalysis of a novel image fusion encryption algorithm based on DNA sequence operation and hyper-chaotic system. Optik 126:223–229
30. Zhang Y, Fu LHB (2012) Research on DNA Cryptography, Applied Cryptography and Network Security Jaydip Sen (ed). ISBN: 978-953-51-0218-2, InTech
31. Zhen P, Zhao G, Min L, Jin X (2016) Chaos-based image encryption scheme combining DNA coding and entropy. Multimed Tools Appl 75:6303–6319
32. Zheng XD, Xu J, Li Parallel W (2009) DNA arithmetic operation based on n-moduli set. Appl Math Comput 212:174–184

Manish Kumar is presently working as assistant professor in the Department of Mathematics at the Birla Institute of Technology and Science, Pilani, Hyderabad Campus, Hyderabad, India. Dr. Kumar obtained his Master of Science in Mathematics from Banaras Hindu University, Varanasi, Ph. D. from Department of Applied Mathematics at Indian Institute of Technology (Indian School of Mines), Dhanbad, and received various awards. Dr. Kumar research areas are Pseudo-differential operators, distribution theory, wavelet analysis and its applications, digital image processing, cryptography, and information security.

R. N. Mohapatra is an Academic Director & Professor in the Department of Mathemtics at the University of Central Florida, Orlando, Florida, USA. His research interests include classical and modern analysis, approximation theory and splines.

Sajal Agarwal did his Masters in Mathematics and BE in Computer Science from BITS Pilani Hyderabad Campus, India. His current research interests include multimedia security, machine learning and software engineering.

G. Sathish did his M.Tech (IMSC) from IIT Madras and currently pursuing his Ph.D at BITS Pilani Hyderabad Campus, Hyderabad, India. His current research interests include cryptography, image processing and wireless communication.

S. N. Raw is an Assistant Professor in Department of Mathematics, National Institute of Technology Raipur, India since 2013. He has completed his Ph. D. degree in 2012 from IIT (ISM) Dhnabad, India and completed his PG & UG degree from BHU Varanasi, India. His current research interests include nonlinear dynamics and chaos, sensitivity analysis, mathematical biology, real world problems.

Affiliations

Manish Kumar[1] · R. N. Mohapatra[2] · Sajal Agarwal[1] · G. Sathish[1] · S. N. Raw[3]

R. N. Mohapatra
ramm@mail.ucf.edu

Sajal Agarwal
sagarwal175@gmail.com

G. Sathish
sathish90.india@gmail.com

S. N. Raw
shardaraw@gmail.com

[1] Department of Mathematics, Birla institute of Technology and Science-Pilani, Hyderabad Campus, Hyderabad, Telangana, 500078, India

[2] Department of Mathematics, University of Central Florida, Orlando, FL 32816, USA

[3] Department of Mathematics, National Institute of Technology Raipur, Raipur, Chhattisgarh, 492010, India