Compression and Cryptography Algorithms for 3D Remote Sensing Point Cloud Data Based on 3D-TCICM and ECC

Pengfei Yan, Shixian Nan, Xiufang Feng, Yongfei Wu, Jie Yang, and Hao Zhang, *Member, IEEE*

***Abstract*—This paper proposes an asymmetric encryption and compression method based on three-dimensional Tent-Cubic-ICMIC map (3D-TCICM) and elliptic curve encryption (ECC) for 3D remote sensing point cloud. Due to the large size, rich data and high security of remote sensing point cloud, an encryption and compression method with high computing efficiency and high security is required. The point cloud is first divided into blocks, then compressed by point cloud library (PCL) using octree, and then the resulting data stream is encrypted. During encryption, a chaotic system 3D-TCICM with excellent chaotic behavior is proposed. Its initial values are generated by point cloud data and ECC algorithm. The sender and receiver only hold the public key or private key respectively, which further improves the security of the algorithm. The encryption factors are also generated for subsequent encryption operations, which improves the plaintext correlation. The encryption algorithm includes innovative scrambling algorithm between multiple blocks (MBS), helix diffusion in GF(2^8) field and multiple S-boxes substitution, in which different S-boxes are provided for each data block. Simulation results indicate that the proposed scheme surpasses existing algorithms in terms of security, randomness, and plaintext correlation, all while maintaining a lower computational complexity. The algorithm exhibits robustness against various attacks such as differential, statistical, chosen-plaintext, known-plaintext, chosen-ciphertext, noise, and cropping attacks.**

***Index Terms*—3D point cloud, asymmetric encryption, compression, hyperchaotic system, remote sensing.**

I. Introduction

The remote sensing point clouds record the location, color and other information of the ground objects in detail, which is more intuitive than the plane image, as shown in Fig. 1.

Manuscript received 12 June 2023; revised 28 August 2023; accepted 10 September 2023. Date of publication 13 September 2023; date of current version 3 October 2023. This work was supported in part by the Shanxi Provincial Key Research and Development Project under Grant 202102020101007, in part by Shanxi Provincial Postgraduate Education Innovation Project under Grant 2022Y287, in part by Shanxi Provincial Postgraduate Research Innovation Project under Grant 2023KY292, in part by the National Natural Science Foundation of China under Grant 61702356, and in part by the Natural Science Foundation of ShanXi Province under Grants 20210302124050 and 202203021221074. *(Corresponding author: Hao Zhang.)*

Pengfei Yan is with the College of Software, Taiyuan University of Technology, Jinzhong 030600, China, and also with the College of Computer, Data Science, Taiyuan University of Technology, Jinzhong 030600, China.

Shixian Nan, Xiufang Feng, and Jie Yang are with the College of Software, Taiyuan University of Technology, Jinzhong 030600, China.

Yongfei Wu and Hao Zhang are with the College of Computer, Data Science, Taiyuan University of Technology, Jinzhong 030600, China (e-mail: zhangh545@126.com).

Digital Object Identifier 10.1109/JPHOT.2023.3314840

Therefore, they are often used in agricultural and forestry census, topographic mapping, address disaster assessment, automatic driving, etc. The application scenarios of them are mostly confidential, and they need more secure algorithms to encrypt them. Compared with plane images, remote sensing point clouds have complex data structures, richer data information, and larger data volumes, which make storage and transmission more difficult. They need to be compressed as much as possible when important information is accurately stored to improve storage and transmission efficiency.

Chaotic system is suitable for encryption because of its uncertainty, non repeatability, unpredictability and initial value sensitivity. Lu et al. [1] utilized an improved Chebyshev chaotic system (ICCS) to encrypt the 3D medical model, thereby significantly enhancing data security. Patro et al. [2] introduced a two-stage pixel diffusion-based image encryption technique using piecewise linear chaotic map (PWLCM). Dash et al. [3] employed PWLCM for enhanced bit-level privacy protection in medical digital images. Wang et al. [4] designed a 4D hyperchaos system, combined with DNA technology to design an encryption algorithm, and verified its reliability. Wang et al. [5] proposed a Logistic-Chebyshev dynamic coupled map lattices (LCDCML), and took the proposed one-dimensional Logistic-Chebyshev map (1DLCM) as its dynamic coupling coefficient. Then propose an image encryption algorithm at pixel level and bit level. Lai et al. [6] proposed a no equilibrium chaotic system with double period, which added an additional variable and constant term on the basis of the original chaotic system, thus obtaining better chaotic characteristics. It can be seen from recent research that high-dimensional chaotic systems are widely used in encryption.

Elliptic curve cryptography (ECC) is a public key encryption technology, which uses the properties of elliptic curve equations to generate keys, and the generated keys are faster and more secure. Li et al. [7] converted multiple images into fast response codes, and use them together with a strength key as the input of the ECC system. Then, they propose an encryption system that can encrypt 16 images at the same time. Jasra et al. [8] implemented a computationally efficient encryption and authentication algorithm using an ECC substitution system, and design a color image encryption algorithm combining dynamic DNA and chaotic systems. Sahasrabuddhe et al. [9] used chaotic system and ECC to generate cryptographic images and share the secret key, and generate three-dimensional images from

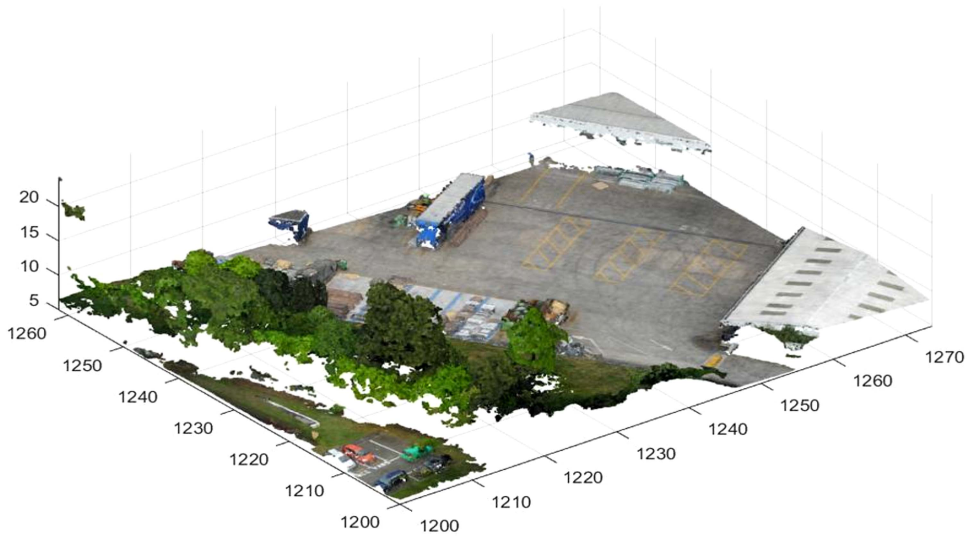


Fig. 1. Remote sensing point cloud.

multiple images for encryption. The research shows that ECC is an effective method to generate encryption algorithm keys.

Octree is often used to compress point cloud data because of its spatial structure. Zhang et al. [10] introduced the quadtree and binary tree into the geometric division step based on the octree division scheme, and propose an adaptive geometric division and coding scheme to achieve point cloud compression. Que et al. [11] use voxel context to compress data with octree structure based on the deep learning framework, and implement dynamic and static point cloud compression schemes. Nguyen et al. [12] adaptively divided a point cloud into multiple octree voxel blocks. Using neural networks to estimate the probability distribution of voxel occupancy, a lossless point cloud geometric compression method is proposed.

At present, there is no encryption-compression algorithm for remote sensing point clouds, because they need not only high security, but also high computing efficiency. According to the characteristics of remote sensing point clouds, this paper proposes an asymmetric encryption and compression algorithm based on chaotic system, ECC and octree compression, in which octree compression is realized by PCL. The innovation points of this paper are:

1) A hyperchaotic system 3D Tent-Cubic-ICMIC map (3D-TCICM) is proposed, which has good ergodicity, randomness, unpredictability and other chaotic characteristics;
2) Octree compression is used and implemented by PCL to convert 3D data into 2D data and block it, reducing the amount of calculation and time complexity, facilitates subsequent encryption, and is easy to store and transmit;
3) The point cloud and ECC are used to generate the initial values of the chaotic system, encryption factors, public key and private key of the algorithm, so as to improve the security of the algorithm and deepen the association with the plaintext;
4) The proposed scrambling between multiple blocks, helix diffusion in GF(2^8) field, and multiple S-boxes substitution all involve encryption factors, which improve the plaintext sensitivity. In addition, different S-boxes are provided for each data block to improve the security.

The overall structure of this paper is as follows: In Section II, relevant technologies are introduced. Section III introduces the proposed encryption and compression scheme. Section IV is the experiment and analysis. Section V is the conclusion.

II. Preliminaries

This section introduces chaotic system, ECC and PCL compression, which are all applicable to remote sensing point clouds. A new hyperchaotic system 3D-TCICM is proposed, and its chaotic trajectory, Lyapunov exponent, permutation entropy and randomness are tested and compared with existing systems.

A. 3D Tent-Cubic-ICMIC Map

The Tent map is a piecewise linear map with uniform distribution, which is often used for image encryption. Its mapping formula is shown in (1).

$$x_{i+1} = \begin{cases} 2px_i, & 0 \le x_i < 0.5 \\ 2p(1-x_i), & 0.5 \le x_i < 1 \end{cases} \tag{1}$$

where the control parameter $p \in [0, 1]$.

The Cubic map is a traditional one-dimensional chaotic map, which is widely used. Its expression is:

$$v_{i+1} = pv_i(1 - v_i^2) \tag{2}$$

where the control parameter $p \in (2.3, 3]$.

The one-dimensional iterative map with infinite collapses (ICMIC) [13] have inverse bifurcations and special main periodic windows, and have good chaotic characteristics, which are well known by scholars. It is expressed as:

$$x_{i+1} = \sin\left(\frac{a}{x_i}\right) \tag{3}$$

where the control parameter $a \neq 0$.

However, one-dimensional chaotic maps have simple forms, small key space, and chaotic behavior can be easily predicted, which can not meet the encryption requirements. Therefore, in this paper, multiple one-dimensional chaotic maps are modulated and coupled to form a three-dimensional chaotic system with complex chaotic behavior, large key space, and difficult to predict. This paper designs a 3D-TCICM based on the Tent map, the Cubic map and the ICMIC. It ensures high security for remote sensing point clouds with large amount of information under low computational load. Its expression is:

$$\begin{cases} x_{i+1} = \begin{cases} \pi^t \left(2px_i + \sin\left(\frac{a}{y_i}\right)\right) \bmod 1, \\ \qquad 0 \le x_i < 0.5 \\ \pi^t \left(2pz_i(1-z_i^2) + \sin\left(\frac{a}{y_i}\right)\right) \bmod 1, \\ \qquad 0.5 \le x_i < 1 \end{cases} \\ y_{i+1} = \begin{cases} \pi^t \left(2py_i + \sin\left(\frac{a}{x_i}\right)\right) \bmod 1, \\ \qquad 0 \le y_i < 0.5 \\ \pi^t \left(2pz_i(1-z_i^2) + \sin\left(\frac{a}{x_i}\right)\right) \bmod 1, \\ \qquad 0.5 \le y_i < 1 \end{cases} \\ z_{i+1} = \begin{cases} \pi^t \left(2pz_i + \sin\left(\frac{a}{x_i}\right)\right) \bmod 1, \\ \qquad 0 \le z_i < 0.5 \\ \pi^t \left(2px_i(1-x_i^2) + \sin\left(\frac{a}{y_i}\right)\right) \bmod 1, \\ \qquad 0.5 \le z_i < 1 \end{cases} \end{cases} \tag{4}$$

where the control parameter $a \neq 0$, $p \in [0, 1]$ and $t \in (0, 19)$.

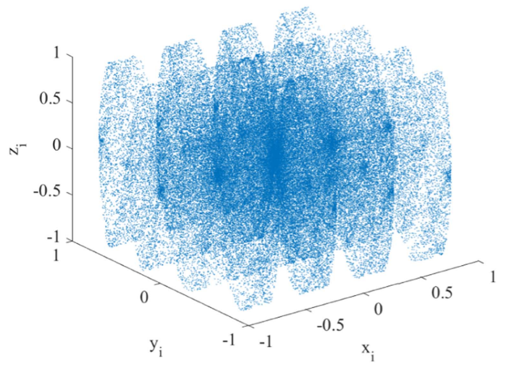
B. Chaotic Behaviors

In this part, the chaotic trajectories, Lyapunov exponents and permutation entropies of 3D-TCICM are compared with the existing chaotic systems (3D-SIMM) [14], (3D-LICC) [15], (3D-Logistic) [16], (3D-ICM) [17] and (3D-Henon) [18], and the randomness of 3D-TCICM chaotic sequence is tested.

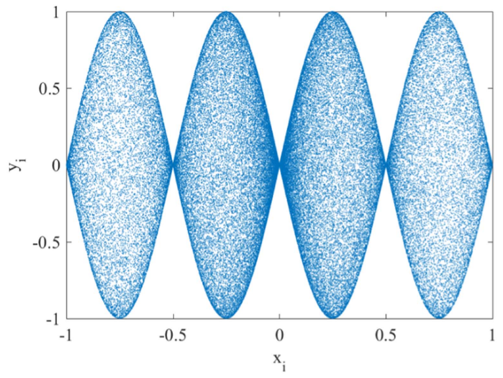
1) Chaotic Trajectory: Fig. 2 shows the motion trajectories of 3D-SIMM in x-y-z plane, x-y plane, x-z plane and y-z plane respectively. The initial values are $x = 0.3, y = 0.4$ and $z = 0.5$. For 3D-SIMM, $a = 1$, $b = 2\pi$ and $c = 11.5$. Obviously, 3D-SIMM has many blank windows, and its coordinate space cannot be completely traversed.

Fig. 3 shows the motion trajectories of 3D-LICC in x-y-z plane, x-y plane, x-z plane and y-z plane respectively. The initial values are $x = 0.3, y = 0.4$ and $z = 0.5$. For 3D-LICC, $a = 3.9$, $b = \pi$ and $c = \pi$. Obviously, 3D-LICC traverses more widely, but clustering occurs in the part where $x = 0$, $y = 0$ and $z = 0$.

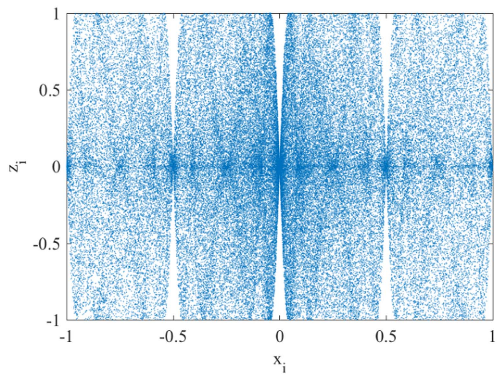
Fig. 4 shows the motion trajectories of 3D-TCICM in x-y-z plane, x-y plane, x-z plane and y-z plane respectively. The initial values are $x = 0.3, y = 0.4$ and $z = 0.5$. For 3D-TCICM, $a = 10, p = 0.5$ and $t = 10$. Compared with 3D-SIMM and 3D-LICC, 3D-TCICM has a wider range of motion, a more uniform trajectory, no blank windows and no aggregation phenomenon, so it has better ergodicity, larger chaotic range and better chaotic performance.



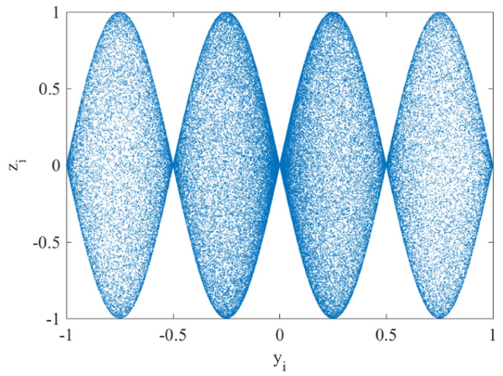
(a) x-y-z plane



(b) x-y plane



(c) x-z plane



(d) y-z plane

Fig. 2. Trajectories diagrams of 3D-SIMM in four different planes.

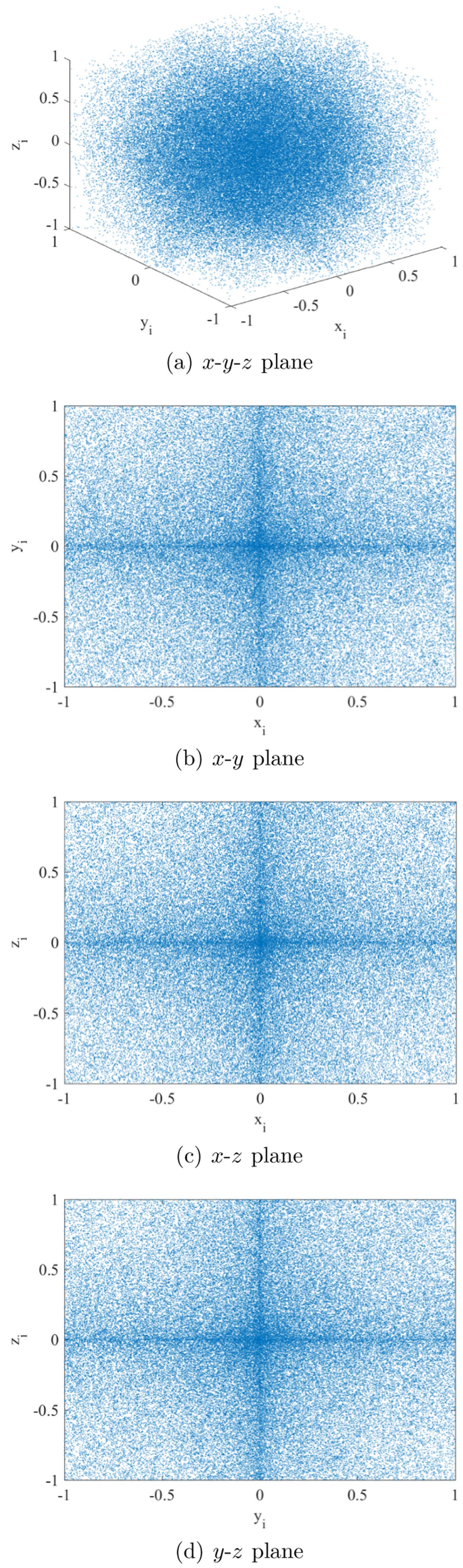


Fig. 3. Trajectories diagrams of 3D-LICC in four different planes.

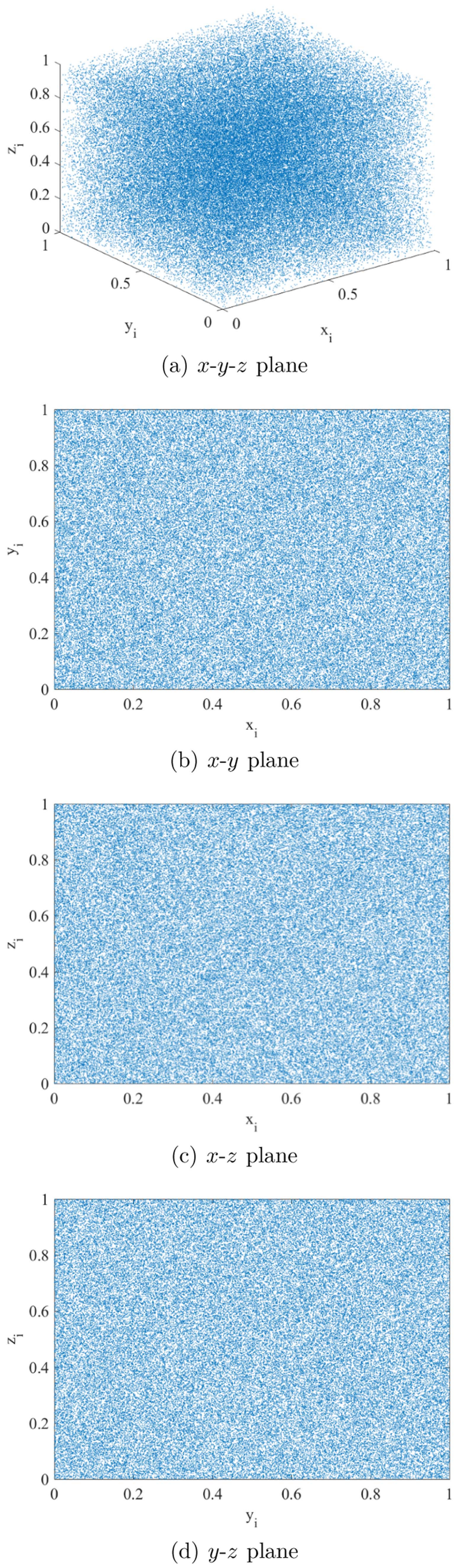


Fig. 4. Trajectories diagrams of 3D-TCICM in four different planes.

2) Lyapunov Exponent: Lyapunov exponent (LE) is one of the indexes to measure the performance of chaotic systems. The formula is:

$$\lambda_{F(x)} = \lim_{k\to+\infty} \frac{1}{k} \sum_{i=0}^{k-1} \ln |F'(x_i)| \tag{5}$$

where k is the number of iterations. When LE is greater than 0, it indicates that the system is chaotic. When two or more LEs are greater than 0, it indicates that the system is hyperchaotic.

It can be seen from Fig. 5 that in the interval shown in the figure, the LEs of 3D-TCICM are greater than 0, the smallest LE is greater than 2, and the largest LE is greater than 11.

Fig. 6 shows the LEs diagrams of 3D-SMM and 3D-LICC when $a \in (0, 1)$. The smallest LE is about 2 and the largest LE is stable around 5.

However, the smallest LE of 3D-TCICM is stable around 5 when $a \in (0, 1)$. It can be concluded that compared with the other two systems, 3D-TCICM has a larger LE and better chaotic performance.

3) Permutation Entropy: Permutation entropy (PE) is used to test the complexity and randomness of time series. In the calculation process, the idea of permutation is introduced, which is expressed as follows:

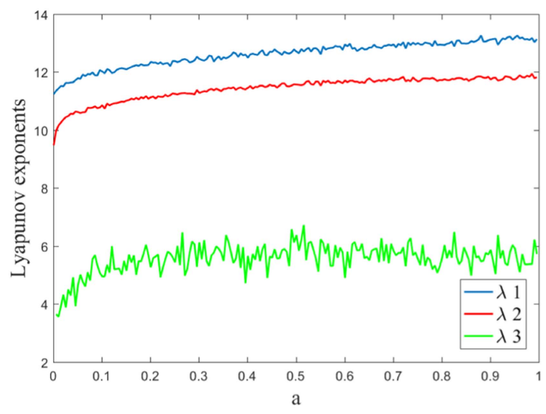
$$\mathrm{PE}(m) = -\frac{\sum_{i=1}^{K} P_i \ln(P_i)}{\ln(m!)} \tag{6}$$

where the result is between 0 and 1. The closer to 1, the more complex and random the sequence is.

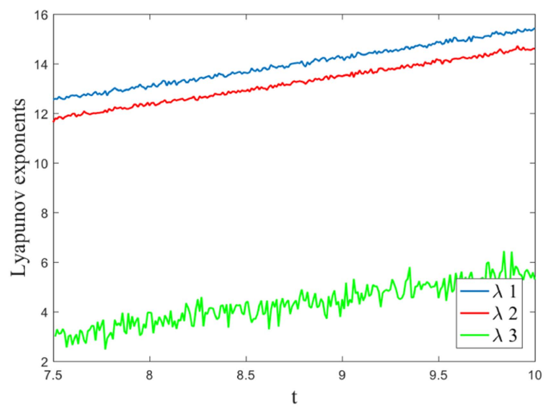
Fig. 7 shows the PEs of 3D-SIMM, 3D-ICM, 3D-Henon and 3D-TCICM. The PE of 3D-Henon is far smaller than the other three systems, indicating that the sequence has the worst randomness. When $a \in (0, 0.4)$, the PE of 3D-SIMM is less than 3D-TCICM and 3D-ICM, indicating that its chaotic performance is poor within a certain range. When $a \in (0, 0.3)$, the PE of 3D-ICM is slightly lower than that of 3D-TCICM, indicating that its performance is slightly inferior to that of 3D-TCICM. In conclusion, 3D-TCICM has better chaotic performance, and its pseudo-random sequence is more random and complex.

4) NIST SP800–22 Test Of Chaotic Sequences: NIST SP800-22 test is a set of methods used to test the randomness of pseudo-random sequences, which includes 15 methods. The length of the tested sequence is usually 10^3 10^7. Here, a sequence with a length of 10^6 is used for testing. When P-value is greater than 0.01, the sequence passes the test, indicating that it has randomness. Table I shows the test results of 3D-TCICM. It can be seen that the P-value of each item is greater than 0.01, indicating that the system can generate pseudo-random sequences with high randomness, which are difficult to predict.

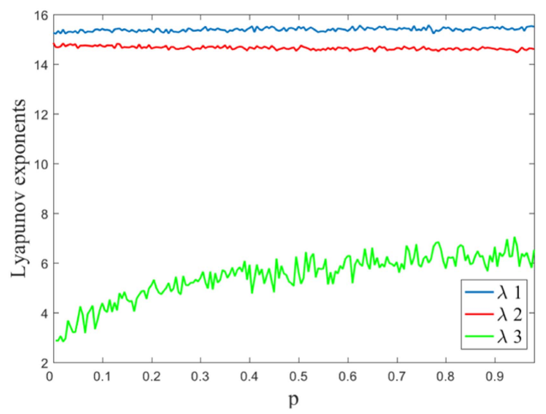
Because the point cloud of remote sensing image has the characteristics of large size, large amount of data, rich information and high security, the chaotic system required for encryption should have high security and low computational complexity. While 3D-TCICM is a three-dimensional hyperchaotic system coupled by multiple one-dimensional chaotic maps, which is simple to generate and ensures low complexity under the premise of high security.



(a) LEs with $t = 10$, $p = 0.5$



(b) LEs with $a = 10$, $p = 0.5$



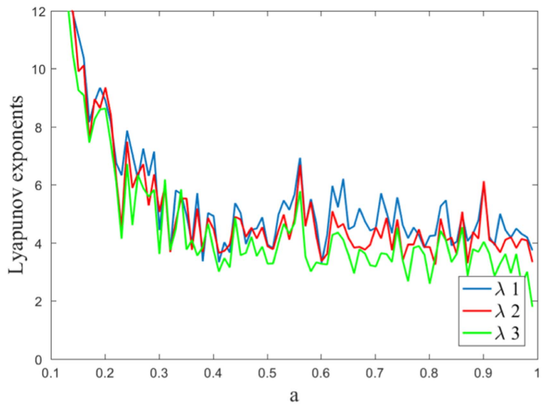
(c) LEs with $a = 10$, $t = 10$

Fig. 5. LEs of 3D-TCICM.

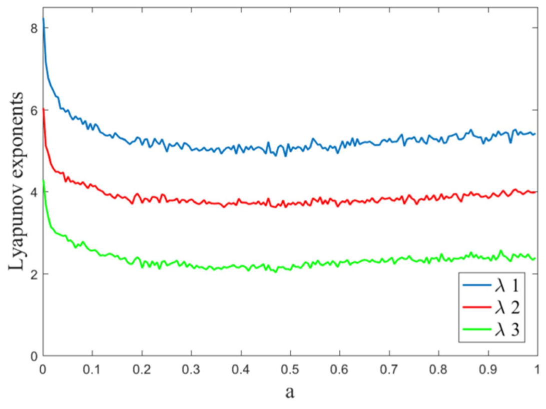
C. Elliptic Curves Cryptography

ECC is a public key asymmetric encryption algorithm based on discrete logarithm problem. Compared with RSA (Rivest Shamir Adleman), it can achieve the same level of security with smaller keys, and at the same time, it makes memory consumption lower and computing efficiency higher. The elliptic curve E in finite field F_p is expressed as:

$$y^2 = (x^3 + ax + b) \bmod p \tag{7}$$



(a) LEs of 3D-SMM



(b) LEs of 3D-LICC

Fig. 6. LEs of 3D-SMM and 3D-LICC when $a \in (0, 1)$.

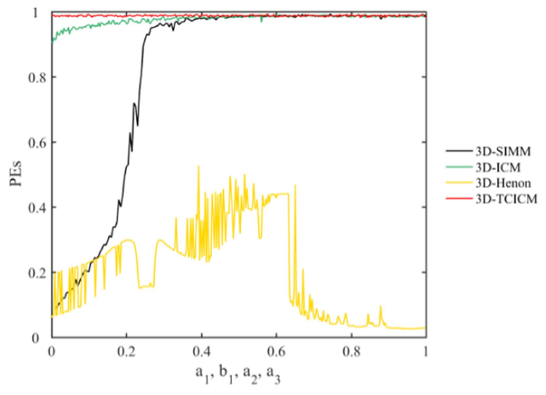


Fig. 7. PEs of different chaotic maps.

where $a, b, x, y \in F_p$, the prime $p > 3$ and $4a^2 + 27b^3 \neq 0 \pmod p$.

E is a set composed of all solutions (x, y) satisfying (7) and the point at infinity O. G is the base point of the curve with upper degree n ($nG = O$). There is an integer k smaller than n. According to the addition rule, it is easy to calculate $K = kG$, K is a point on the curve, but on the contrary, given K and G, it is difficult to calculate n. This is the elliptic curve discrete logarithm problem (ECDLP), which is the basis for elliptic curve encryption. During encryption, k is the private key and K is the public key.

TABLE I
NIST SP800-22 Test

Test Items	P-value	Result
Monobit frequency test	0.0554	success
Block Frequency test	0.9273	success
Runs test	0.5536	success
Longest-run-of ones in a block	0.7572	success
Binary matrix rank test	0.9735	success
Discrete Fourier transform test	0.6424	success
Non-overlapping template matching test	0.5711	success
Overlapping template matching test	0.3905	success
Maurer's universal statistical test	0.4951	success
Linear complexity test	0.8124	success
Serial test	0.1789	success
Approximate entropy test	0.6069	success
Cumulative sums test	1.0000	success
Random excursions test	0.5931	success
Random excursions variant test	0.5740	success

The addition rule on the elliptic curve is: suppose there are two points $P(x_1, y_1)$, $Q(x_2, y_2)$ on the curve, and $P \neq -Q$. Assume that '$\otimes$' is expressed as point addition, and two point addition is expressed as:

$$P(x_1, y_1) \otimes Q(x_2, y_2) = R(x_3, y_3) \tag{8}$$

where

$$\begin{cases} x_3 = (k^2 - x_1 - x_2) \bmod p \\ y_3 = (k(x_1 - x_3) - y_1) \bmod p \end{cases} \tag{9}$$

and k is the slope, expressed as

$$k = \begin{cases} \frac{3x_1^2 + a}{2y_1} \bmod p, & P = Q \\ \frac{y_2 - y_1}{x_2 - x_1} \bmod p, & P \neq Q \end{cases} \tag{10}$$

D. Point Cloud Library Compression

Point Cloud Library (PCL) is a large cross platform open source C++ programming library built on the basis of point cloud related research, which can be used for point cloud acquisition, filtering, segmentation, registration, feature extraction, compression and other operations.

PCL uses octree for compression, and octree was proposed in the 1980 s. Suppose that there are some points in Cartesian coordinates, the coordinates can be equally divided into eight quadrants, and each quadrant can establish a Cartesian coordinate and divide into eight quadrants, and so on, until each point can be accurately represented, that is, a lossless compressed octree partition. A cube composed of eight equal sized cubes is regarded as a whole. The nodes of the octree can be divided into three categories. When all eight cubes are unoccupied, the node is a white node; all occupied nodes are called black nodes; some occupied nodes are called gray nodes, while white nodes and black nodes are called leaf nodes, as shown in Fig. 8.

TABLE II
ADVANCED PARAMETER REPRESENTATION AND VALUE OF PCL COMPRESSION

Parameter	Explain
pointResolution_arg	Defines the coding accuracy of point coordinates. This parameter should be set to a value less than the sensor accuracy.
octreeResolution_arg	Defines the voxel size of the expanded octree. The larger voxel resolution makes compression faster, but the compression quality is degraded.
doVoxelGridDownDownSampling_arg	If this parameter is activated, only the hierarchical octree data structure will be encoded, and the decoding object will generate a point in the voxel center. With this method, the point cloud will be down sampled during compression, and high compression performance will be achieved
iFrameRate_arg	The point cloud is compressed by differential coding. With this method, the difference between the newly introduced point cloud and the previously encoded point cloud is coded to obtain the maximum compression performance.
doColorEncoding_arg	Enable color texture component encoding compression.
colorBitResolution_arg	This parameter defines the number of bits occupied by each color component after encoding.

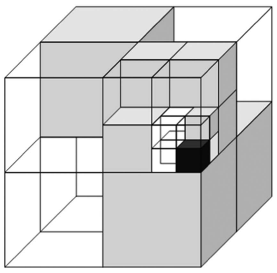


Fig. 8. Nodes and leaf nodes of octree.

The basic idea of PCL octree compression is to record the entire octree structure and then entropy code it. For point cloud compression, PCL not only compresses the content of its points, but also compresses the color information. During PCL compression, can adjust the parameters to obtain compression results with different precision and effects. Advanced parameter representation and value of PCL compression are shown in Table II.

In this paper, set the parameters “pointResolution_arg”, “octreeResolution_arg”, “doVoxelGridDownDownSampling_arg”, “iFrameRate_arg”, “doColorEncoding_arg” and “colorBitResolution_arg” as “0.001”, “0.01”, “true”, “100”, “true” and “8” respectively. It ensures high calculation efficiency with small loss.

III. ENCRYPTION-COMPRESSION SCHEME

Considering the characteristics of remote sensing point clouds, such as large amount of data, large size and high security, the encryption and compression algorithm suitable for them is designed. First, the initial values of the chaotic system and the public and private keys for encryption and decryption are generated from the point cloud’s information and ECC algorithm. The asymmetric encryption algorithm further improves the security, and generates encryption factors in this process. The encryption factors are used in the subsequent encryption steps to improve the security of the algorithm and deepen the connection with plaintext. The point cloud is first divided into blocks, which are compressed by PCL block by block. The 3D data is compressed and mapped to 2D, which reduces the computational complexity and facilitates storage and transmission. PCL near lossless compression retains remote sensing details to the maximum extent, and whether it is completely lossless depends on the data and computer accuracy. After that, scrambling with multiple blocks, helix diffusion in GF(2^8) field, and multiple S-boxes substitution are performed. These algorithms have high security and low computational complexity, and are suitable for large size and high security remote sensing point clouds. Fig. 9 shows the encryption and decryption process of the algorithm.

A. Key Generation Phase

ECC can reach a higher security level with a smaller key, and it requires less memory space and has high computing efficiency. It is suitable for remote sensing point clouds with large size and rich data. The key of this algorithm is generated by the point cloud information, SHA-512 and ECC algorithm, which improves the plaintext sensitivity and further improves the security by using asymmetric encryption. The specific way to generate the key is as follows:

Step 1: Decompose the color components R, G, B of the remote sensing point cloud, and obtain the hash sequences SHA_R, SHA_G, SHA_B by SHA-512 function.

Step 2: An elliptic curve E_p is obtained from (7), and the base point $G(G_x, G_y)$ on E_p is selected by the hash sequence of the point cloud in the following way:

$$\begin{cases} G_x = x\left(\lfloor (SHA_R(i) + 100) \times 10^{10} \rfloor \bmod (\text{length}(x) + 1)\right) \\ G_y = y\left(\lfloor (SHA_G(j) + 100) \times 10^{10} \rfloor \bmod (\text{length}(y) + 1)\right) \end{cases} \tag{11}$$

where x and y are the coordinate sequences of the elliptic curve E_p, and i and j are positive integer values less than the length of the hash sequences.

Step 3: The private key k is generated from the point cloud color sequences and its hash sequences, and the formula is:

$$k = \left\lfloor \begin{array}{c} \sum_{i=1}^{64} (SHA_R(i) + SHA_G(i) + SHA_B(i)) \\ \times \sum_{i=1}^{M} (R(i) + G(i) + B(i)) \end{array} \right\rfloor \bmod n + 1 \tag{12}$$

where n is the order, M is the length of each color sequence.

Step 4: The public key K is obtained from the private key k and the base point G:

$$K = kG \tag{13}$$

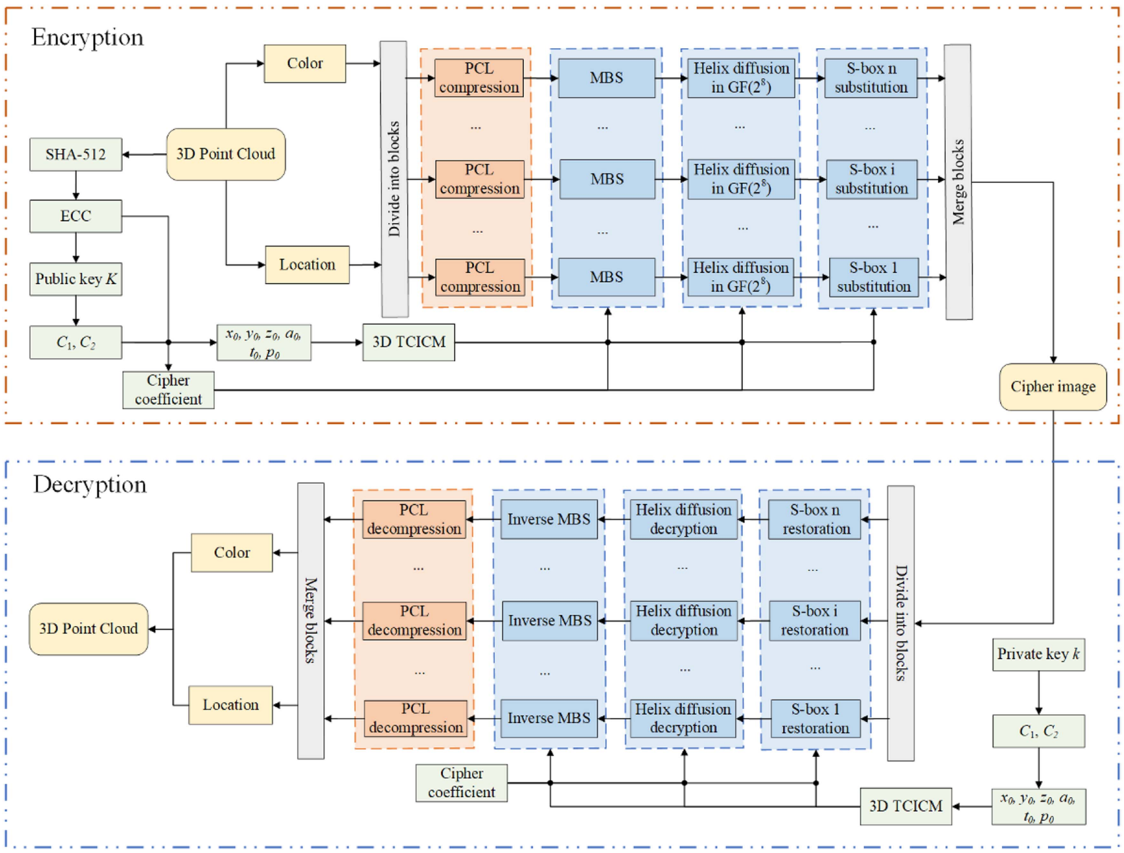


Fig. 9. Block diagram of encryption and decryption algorithm.

Then E_p, K and G are sent to the sender for generating the initial values of the chaotic system.

After receiving the information, the sender generates the initial values of the chaotic system. The specific process is as follows:

Step 1: Map the sum of point cloud color values onto the elliptic curve E_p to obtain point $M(M_x, M_y)$:

$$\begin{cases} M_x = x \\ \times \left(\sum_{i=1}^{M} (R(i) + G(i) + B(i)) \bmod \text{length}(x) + 1 \right) \\ M_y = y \\ \times \left(\sum_{i=1}^{M} (R(i) + G(i) + B(i)) \bmod \text{length}(y) + 1 \right) \end{cases} \quad (14)$$

Step 2: Use SHA-512 to generate a random integer r ($r < n$) related to plaintext:

$$r = \left\lfloor \sum_{i=1}^{64} (SHA_R(i) + SHA_G(i) + SHA_B(i)) \times 10^{10} \right\rfloor \bmod n + 1 \quad (15)$$

Step 3: Calculate $C_1 = M + rK$ and $C_2 = rG$. C_1, C_2 and M are used to generate initial values of chaotic systems. The initial values generation process is as follows:

$$\begin{cases} x_0 = M \times (C_1 + C_2) \times \sum_{i=1}^{64} SHA_R(i) \bmod 1 \\ y_0 = M \times (C_1 + C_2) \times \sum_{i=1}^{64} SHA_G(i) \bmod 1 \\ z_0 = M \times (C_1 + C_2) \times \sum_{i=1}^{64} SHA_B(i) \bmod 1 \\ t_0 = M \times (C_1 - C_2) \times \sum_{i=1}^{64} SHA_R(i) \bmod 19 \\ p_0 = M \times (C_1 - C_2) \times \sum_{i=1}^{64} SHA_G(i) \bmod 1 \\ a_0 = M \times (C_1 - C_2) \times \sum_{i=1}^{64} SHA_B(i) \bmod 19 \end{cases} \quad (16)$$

where “+” represents that the corresponding coordinate values of two points are added, and then the horizontal and vertical coordinate values of the results are added; “–” represents that the corresponding coordinate values of two points are subtracted, and then the horizontal and vertical coordinate values of the results are added.

Finally, C_1 and C_2 can be transmitted to the receiver for decryption. The receiver obtains C_1, C_2 and k to calculate M, and the process is as follows:

$$C_1 - kC_2 = M + rK - k(rG) = M + rkG - krG = M \quad (17)$$

After the receiver obtains C_1, C_2 and M, the initial values of the chaotic system can be obtained by (16), and then the subsequent steps can be decrypted using the initial values. The asymmetric encryption and decryption process is shown in Fig. 10. ECC is used to generate the initial values of the chaotic system, and the sender and the receiver only have the public key

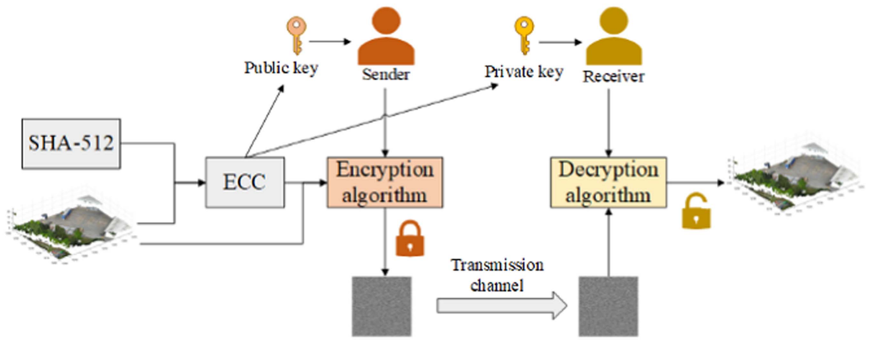


Fig. 10. Asymmetric encryption and decryption process.

or private key, which improves the security of the key. For remote sensing point clouds, the security of the algorithm is improved under the condition of high computing efficiency.

B. Scrambling Between Multiple Blocks

Due to the large volume of remote sensing point clouds, normal scrambling algorithms have high computational complexity and low computational efficiency for them, so this part proposes a scrambling algorithm between multiple blocks (MBS). Before scrambling, the point clouds are divided into blocks, and then the blocks are scrambled according to the pseudo-random sequence generated of the chaotic system 3D-TCICM, reducing the computational complexity and the correlation between blocks. In addition, encryption factor is added in the scrambling process to increase the correlation between the algorithm and plaintext. The scrambling algorithm is as follows:

Step 1: Divide the data into $blockNum$ blocks, the side length is m, that is, the size of a block is $blockSize = m \times m$.

Step 2: Take the initial values x_0, y_0, z_0, a_0, p_0, t_0 into 3D-TCICM, iterate $10 \times M \times N$ ($M \times N$is the size of color data of point cloud) times, and then obtain pseudo-random sequences s_1, s_2, s_3. The sequences S_1, S_2 and S_3 are obtained according to the following formula:

$$\begin{cases} S_1 = \lfloor s_1 \times 2^{16} \rfloor \bmod 256 \\ S_2 = \lfloor s_2 \times 2^{16} \rfloor \bmod 256 \\ S_3 = \lfloor s_3 \times 2^{16} \rfloor \bmod 256 \end{cases} \tag{18}$$

Step 3: Add the horizontal and vertical coordinate values of C_1 to get a number, which is recorded as C_1'. Similarly, C_2' is obtained. Then convert them to odd numbers C_X and C_Y according to (19), and bring them into Algorithm 1 as encryption factors to increase the association between algorithm and plaintext.

$$\begin{cases} C_X = 2 \times \left[\frac{C_1'}{2}\right] + 1 \\ C_Y = 2 \times \left[\frac{C_2'}{2}\right] + 1 \end{cases} \tag{19}$$

Algorithm 1 is the scrambling process of MBS algorithm. Fig. 11 shows the first round scrambling process of MBS.

Algorithm 1: The MBS Algorithm.

```
Input: The data $P$, the sequences $S_1$ and $S_2$, the encryption
    factors $C_X$ and $C_Y$
Output: The encrypted $E$
1: Determine the serial numbers of the scrambling blocks
   by pseudo-random sequence $S_1$, and save them as
   $index_1$:
   $\begin{cases} [\sim, S_{index_1}] = \text{sort}\,(S_1(1 : blockNum)) \\ index_1 = (C_X \oplus S_{index_1}) \bmod blockNum + 1 \end{cases}$
2: Determine the scrambling position. In order to make
   each piece of data have different scrambling positions,
   different parts of sequence $S_2$ are selected for
   scrambling.
   $\begin{cases} [\sim, S_{index_2}(1,:)] = \\ \quad \text{sort}\,(S_2\,((i-1) \times blockSize) + 1 : i \times blockSize) \\ index_2 = (C_Y \oplus S_{index_2}) \bmod blockSize + 1 \end{cases}$
   where $i = 1, 2, \ldots, blockNum$.
3: Divide $P$ into $blockNum$ blocks of size $blockSize$,
   and convert it into a one-dimensional array. The $j$-th
   element in the $i$-th block is exchanged with the
   $index_2(j)$-th element in the $index_1(i)$-th block,
   where $i = 1, 2, \ldots, blockNum$, and
   $j = 1, 2, \ldots, blockSize$.
4: Several scrambled blocks are integrated into a
   complete data $E$.
```

The decryption algorithm of MBS is its inverse process, and the formula of inverse scrambling is:

$$\begin{cases} t = P'(i,j) \\ P'(i,j) = P'\,(index_1(i), index_2(j)) \\ P'\,(index_1(i), index_2(j)) = t \end{cases} \tag{20}$$

where $i = blockNum, blockNum - 1, \ldots, 2, 1,$ and $j = blockSize, blockSize - 1, \ldots, 2, 1,$ and P' is the data to be decrypted after blocking.

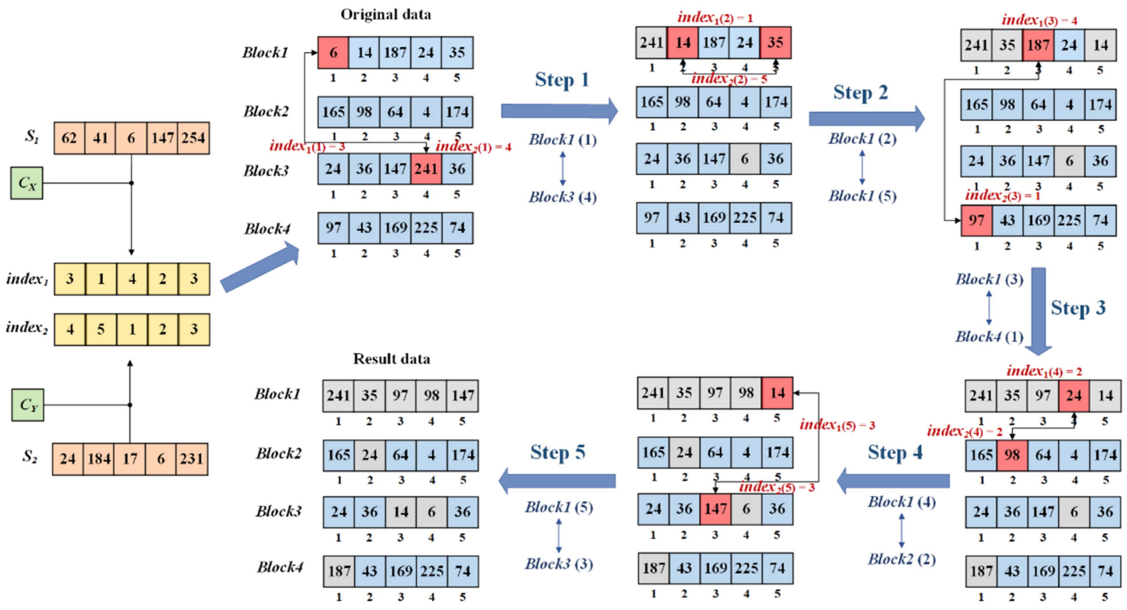


Fig. 11. First round scrambling process of MBS.

C. Helix Diffusion in GF(2^8)

Diffusion algorithm can hide a pixel to other pixels to achieve encryption. The classical diffusion algorithms include XOR diffusion, modulo addition diffusion and the combination of the two. Zhang et al. [19] has analyzed and evaluated many diffusion algorithms, but many of them are less resistant to known & chosen plaintext attacks. It is mentioned in the paper that the helix cipher cryptanalysis can be converted into searching the unknown pairs (k_1, k_2) of the following formula:

$$(k_1 \dot{+} k_2) \oplus (k_1 \dot{+} (k_2 \oplus \beta)) = y \tag{21}$$

where '$\dot{+}$' is modular addition, '$\oplus$' is XOR. Its complexity is 2^{n-2}, and it can be used as a diffusion algorithm with anti attack [20]. Based on the above formula and [21], this part proposes a helix diffusion algorithm based in GF(2^8) field. This algorithm is improved on the basis of [21], and the constant field is converted to the GF(2^8) field, which further improves the security of the algorithm. The diffusion algorithm is as follows:

Step 1: First obtain pseudo-random sequences S_1, S_2 and S_3 according to (22):

$$\begin{cases} S_1 = \lfloor s_1 \times 2^{16} \times (C_X + C_Y) \rfloor \bmod 256 \\ S_2 = \lfloor s_2 \times 2^{16} \times (C_X + C_Y) \rfloor \bmod 256 \\ S_3 = \lfloor s_3 \times 2^{16} \times (C_X + C_Y) \rfloor \bmod 256 \end{cases} \tag{22}$$

Then initialize diffusion parameters:

$$\begin{cases} B = \text{zeros}(M, N) \\ C = \text{zeros}(M, N) \\ D = \text{zeros}(M, N) \\ E = \text{zeros}(M, N) \\ B_0 = 0 \\ D_0 = 0 \end{cases} \tag{23}$$

where M and N are the size of plaintext.

Step 2: First, carry out forward diffusion according to (24).

$$\begin{cases} B(i) = GF_{2^8}\left(B(i-1) + S_1(i) + P(i)\right) \\ C(i) = \left(GF_{2^8}\left(S_1(i) + S_2(i) + S_3(i)\right)\right) \\ \qquad \oplus \left(GF_{2^8}\left(S_1(i) + S_2(i) + S_3(i) \oplus B(i)\right)\right) \end{cases} \tag{24}$$

where pixel $i = 1, 2, \ldots, M \times N$. Especially, $B(0) = B_0$.

Step 3: After that, reverse diffusion. When pixel $i = M \times N$, the diffusion process is:

$$\begin{cases} D(M \times N) = GF_{2^8}\left(D_0 + S_1(M \times N) + C(M \times N)\right) \\ t_1 = GF_{2^8}\left(S_1(M \times N) + S_2(M \times N) + S_3(M \times N)\right) \\ t_2 = GF_{2^8}\begin{pmatrix} S_1(M \times N) + S_2(M \times N) \\ + S_3(M \times N) \oplus D(M \times N) \end{pmatrix} \\ E(M \times N) = t_1 \oplus t_2 \end{cases} \tag{25}$$

Step 4: When pixel $i < M \times N$ (i.e., $i = M \times N - 1, \ldots, 2, 1$), the diffusion process is:

$$\begin{cases} D(i) = GF_{2^8}\left(D(i+1) + S_1(i) + C(i)\right) \\ t_1 = GF_{2^8}\left(S_1(i) + S_2(i) + S_3(i)\right) \\ t_2 = GF_{2^8}\left(S_1(i) + S_2(i) + S_3(i) \oplus D(i)\right) \\ E(i) = t_1 \oplus t_2 \end{cases} \tag{26}$$

where E is the final diffusion result.

Diffusion decryption operations are divided into forward diffusion decryption and reverse diffusion decryption. The forward diffusion decryption is as follows:

Step 1: Obtain pseudo-random sequences S_1, S_2 and S_3, assume that the data to be decrypted is A, and initialize reverse diffusion parameters:

$$\begin{cases} B = \text{zeros}(M, N) \\ C = \text{zeros}(M, N) \\ F = \text{zeros}(M, N) \\ G = \text{zeros}(M, N) \\ D_0 = 0 \\ E_0 = 0 \end{cases} \tag{27}$$

Step 2: When pixel $i = M \times N$, the decryption of reverse diffusion is as follows:

$$\begin{cases} t_3 = GF_{2^8}\begin{pmatrix} S_1(M \times N) + S_2(M \times N) \\ +S_3(M \times N) \end{pmatrix} \oplus A(M \times N) \\ D(M \times N) = GF_{2^8}\left(t_3 - S_1(M \times N) - S_2(M \times N)\right) \\ B(M \times N) = S_3(M \times N) \oplus D(M \times N) \\ C(M \times N) = GF_{2^8}\left(B(M \times N) - D_0 - S_1(M \times N)\right) \end{cases} \tag{28}$$

When pixel $i < M \times N$ (i.e., $i = M \times N - 1, \ldots, 2, 1$), the reverse diffusion process is:

$$\begin{cases} t_3 = GF_{2^8}\left(S_1(i) + S_2(i) + S_3(i)\right) \oplus A(i) \\ D(i) = GF_{2^8}\left(t_3 - S_1(i) - S_2(i)\right) \\ B(i) = S_3(i) \oplus D(i) \\ C(i) = GF_{2^8}\left(B(i+1) - B(i) - S_1(i)\right) \end{cases} \tag{29}$$

Step 3: Similarly, the forward diffusion decryption also involves two situations. When pixel $i = 1$, the decryption process is:

$$\begin{cases} t_4 = GF_{2^8}\left(S_1(1) + S_2(1) + S_3(1)\right) \oplus C(1) \\ E(1) = GF_{2^8}\left(t_4 - S_1(1) - S_2(1)\right) \\ F(1) = S_3(1) \oplus E(1) \\ G(1) = GF_{2^8}\left(F(1) - E_0 - S_1(1)\right) \end{cases} \tag{30}$$

When pixel $i > 1$ (i.e., $i = 2, 3, \ldots, M \times N$), the decryption process is:

$$\begin{cases} t_4 = GF_{2^8}\left(S_1(i) + S_2(i) + S_3(i)\right) \oplus C(i) \\ E(i) = GF_{2^8}\left(t_4 - S_1(i) - S_2(i)\right) \\ F(i) = S_3(i) \oplus E(i) \\ G(i) = GF_{2^8}\left(F(i) - F(i-1) - S_1(i)\right) \end{cases} \tag{31}$$

where G is the final decryption result.

D. Multiple S-Boxes Substitution

Due to the large volume of remote sensing point cloud, it is first divided into blocks and then encrypted. In the multiple S-boxes substitution phase, each block uses a different S-box, which improves the security of the algorithm. In the S-box generation and substitution phase, the encryption factor generated by ECC in Section III-A is added to improve the security and enhance the correlation with plaintext. The S-box is generated as follows:

Step 1: Obtain pseudo-random sequences S_1, S_2 according to (18). Set the size of S-box as $s_m = 16$, $s_n = 16$. Intercept the sequences and sort the results to get i_1, i_2:

$$\begin{cases} [\sim, index_1] = \text{sort}\left(S_1\begin{pmatrix} (i-1) \times s_m \times s_n + 1 \\ : i \times s_m \times s_n \end{pmatrix}\right) \\ [\sim, index_2] = \text{sort}\left(S_2\begin{pmatrix} (i-1) \times s_m \times s_n + 1 \\ : i \times s_m \times s_n \end{pmatrix}\right) \end{cases} \tag{32}$$

where i is the serial number of the block. Different sequences are selected according to different blocks, so that different S-boxes are generated for each block.

Step 2: Get the ith prime number $C_{prime}, i = C_1 \times C_2$, where "$\times$" represents that the horizontal and vertical coordinate values of C_1 and C_2 are added and then the results are multiplied. Use C_{prime} and the sequences to get the new index sequences U, W related to plaintext:

$$\begin{cases} U = (C_{prime} \otimes index_1) \bmod 256 + 1 \\ W = (C_{prime} \otimes index_2) \bmod 256 \end{cases} \tag{33}$$

Step 3: Scramble the values of sequence W to get the prototype of S-box:

$$V(i) = W\left(U(i)\right) \tag{34}$$

where $i = 1, 2, \ldots, 256$. Reshape V as a matrix with the size of $s_m \times s_n$ to get the S-box $sbox$.

The substitution algorithm of S-box is shown in Algorithm 2. The substitution process of the first pixel is shown in Fig. 12.

The reverse substitution method of S-box decryption is shown in Algorithm 3.

E. Encryption and Decryption Process

This part is the process of encryption-compression and decryption-decompression of remote sensing point cloud. The steps of encryption and compression are as follows:

Step 1: Decompose the color component R, G, B of remote sensing point cloud for subsequent steps.

Step 2: The sender holds public key K, and generate the initial values x_0, y_0, z_0, t_0, p_0, a_0 for the chaotic system 3D-TCICM according to ECC algorithm in Section III-A. Then generate pseudo random sequences S_1, S_2, S_3 according to (18).

Step 3: Divide the remote sensing point cloud into multiple sub point clouds. And compress them in turn with PCL octree compresson algorithm in Section II-D, then gather them into one-dimensional data P.

Step 4: Divide the data P into blocks with the size of $blockSize$ and the number of blocks of $blockNum$. The size of the last block whose size is less than $blockSize$ is $blockSize2$.

Step 5: The encryption factors C_X and C_Y are obtained according to (19). Bring the data P, sequences S_1, S_2, S_3 and encryption factors C_X and C_Y into the Algorithm 1. The MBS algorithm is used to scramble blocks of the same size to reduce the correlation between data. The scrambled sequence is P_s.

Step 6: The last data block with size $blockSize2$ is randomly inserted into P_s according to the sequence index. The sequence

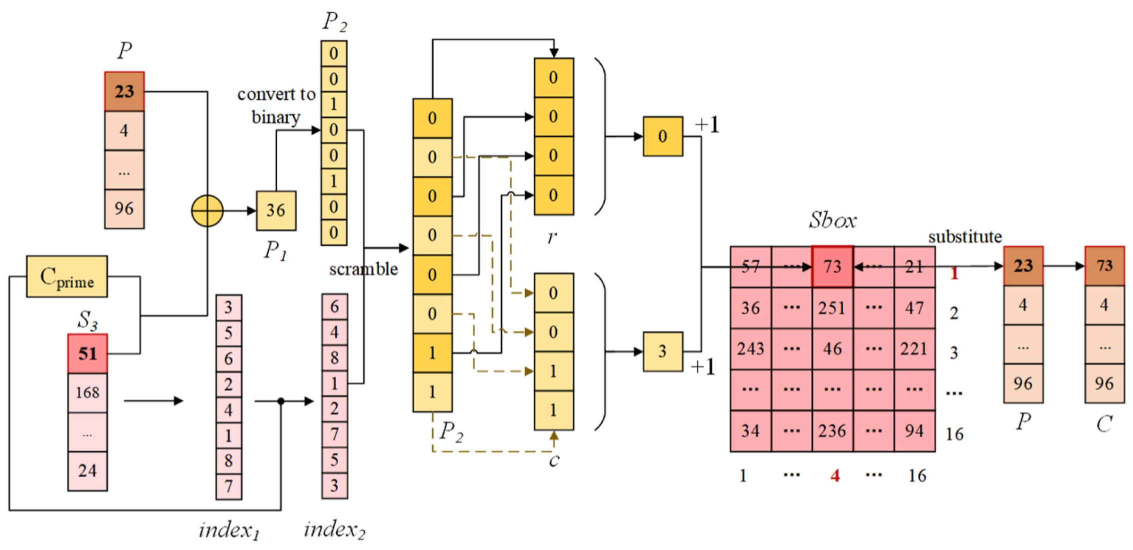


Fig. 12. Substitution process of the first pixel in P.

Algorithm 2: The S-Box Substitution Algorithm.

```
Input: The pixel point $P$, the S-box $sbox$, the pseudo-random sequences $S_3$
Output: The substituted value $C$
1: Use encryption factor and pseudo-random sequence to preliminarily encrypt the pixel point:
       $P_1 = P \oplus (C_{prime} \times S_3(i)) \mod 256$
   where $i$ is the $i$-th pixel point. Then $P_1$ is converted to binary sequence $P_2$.
2: Get the index sequence $index_2$ for $P_2$ scrambling, $i$ is the $i$-th pixel point:
       $\begin{cases} [\sim, index_1] = \text{sort}(S_3(i : i+7)) \\ index_2 = (C_{prime} \times index_1) \mod 8 + 1 \end{cases}$
3: Scramble the 8-bit binary number of $P_2$:
       $\begin{cases} t = P_2(k) \\ P_2(k) = P_2(index_2(k)) \\ P_2(index_2(k)) = t \end{cases}$
   where $k = 1, 2, \ldots, 8$.
4: Obtain the binary values of $P_2$ odd bit, which is recorded as $r$ and even bit as $c$, and convert them to decimal number, so as to obtain the substitution variable $row$ and $col$ are obtained:
       $\begin{cases} row = \text{bin2dec}(r) + 1 \\ col = \text{bin2dec}(c) + 1 \end{cases}$
5: Substitute the original pixel point with the value in $sbox(row, col)$:
       $C = sbox(row, col)$
```

Algorithm 3: The S-Box Reverse Substitution Algorithm.

```
Input: The substituted value $C$, the S-box $sbox$, the pseudo-random sequences $S_3$.
Output: The restored value $R$
1: Find the position of the same value as the value $C$ in the S-box, and record its location as $row$ and $col$.
       $[row, col] = \text{find}(sbox == C)$
2: Convert $row$ and $col$ into binary numbers $r$ and $c$:
       $\begin{cases} r = \text{dec2bin}(row - 1, 4) \\ c = \text{dec2bin}(col - 1, 4) \end{cases}$
   Then put the value $r$ in the odd number position, and put the value $c$ in the even number position to get the 8-bit binary number $P_2$:
       $P_2 = [r(1), c(1), r(2), c(2), r(3), c(3), r(4), c(4)]$
3: Obtain the same scrambling sequence $index_2$ as Algorithm 2.
4: Binary sequence $P_2$ is scrambled inversely:
       $\begin{cases} t = P_2(k) \\ P_2(k) = P_2(index_2(k)) \\ P_2(index_2(k)) = t \end{cases}$
   where $k = 1, 2, \ldots, 8$. Then $P_2$ is converted to decimal number.
5: Decrypt $P_2$ to get the final restored $R$:
       $R = P_2 \oplus (C_{prime} \times S_3(i)) \mod 256$
```

index is:

$$[\sim, index] = \text{sort}(S_1(1 : blockSize2)) \tag{35}$$

where index is the value that is inserted into the position of index.

Step 7: Divide P_s into blocks and each block is diffused according to the helix diffusion algorithm in GF(2^8) field in Section III-C to obtain P_d.

Step 8: According to the S-box generation algorithm in Section III-D, generate different S-boxes for each block.

Step 9: Take the S-box and sequence S_3 into Algorithm 2, substitute the value in P_d to get P_{sub}.

Step 10: Finally, the final encryption result E can be obtained by combining the blocks into a whole.

Decryption is the anti operation of the above process. The specific steps are as follows:

Step 1: The receiver obtains C_1, C_2 and the private key k from the ECC algorithm.

Step 2: Take C_1 and C_2 into (17) to obtain M, and then obtain the initial values x_0, y_0, z_0, t_0, p_0, a_0 of the chaotic sequence 3D-TICICM according to (16), the pseudo random sequences S_1, S_2, S_3 according to (18).

Step 3: Encrypted data E is divided into blocks with size $blockSize$ and number of blocks $blockSize$. The size of the last block whose size is less than $blockSize$ is $blockSize2$.

Step 4: Generate different S-boxes for each block according to the S-box generation algorithm in Section III-D. Take the S-box and sequence S_3 into Algorithm 3 for reverse substitution, and get P'_{sub}.

Step 5: P'_{sub} is decrypted according to the decryption algorithm of helix diffusion in GF(2^8) field in Section III-C, the decrypted P'_d is obtained.

Step 6: According to the sequence index in (33), the value of the corresponding index position in P'_d is extracted and collected as the last block.

Step 7: Blocks are scrambled reversely according to the algorithm and (20) in Section III-B, then gather the data into a one-dimensional array P'_s.

Step 8: P'_s is divided into blocks, and each block is decompressed by the octree decompression algorithm.

Step 9: Finally, each compressed block is combined into a whole to obtain the decrypted and decompressed remote sensing point cloud.

IV. Simulation Results and Security Analysis

The algorithm is tested on a computer with Intel Core™ i7-6700HQ CPU @ 2.60 GHz 2.59 GHz, 1 TB hard disk, 8.00 GB RAM and 64-bit Windows10 operating system. The programming language is matlab and C++, and the programming tools are Matlab 2016b and Visual Studio 2019. The initial values of chaotic system is generated by point cloud and ECC algorithm.

A. Simulation Results

The data set is $SensatUrban$, an urban point cloud data set proposed by Hu Q. et al. [22] of Oxford University, which includes nearly 3 billion points with detailed semantic annotation in 7.6 square kilometers of three British cities (Birmingham, Cambridge and York). Fig. 13 shows the point cloud data of 'birmingham_block_11.ply', 'birmingham_block_13.ply' and 'cambridge_block_32.ply' in the dataset and the results of encryption and decryption using the algorithm in this paper. The three point cloud data will be abbreviated as 'BH_11', 'BH_13' and 'CB_32'. The original point clouds after encryption and compression are data streams, which are visualized here.

B. Key Space Analysis

When the key space is greater than 2^{100}, the algorithm can resist violent attacks [23]. The key of this algorithm includes the hash values generated by SHA-512 algorithm, the initial values x_0, y_0, z_0, t_0, p_0, a_0 of the chaotic system and the encryption factors C_X, C_Y, C_{prime}. Assuming that the accuracy of the computer is 10^{-15}, the key space of this algorithm is $2^{256} \times 10^{135} \approx 2^{704} > 2^{100}$. It shows that the algorithm in this paper has enough ability to resist violent attacks. Table III shows the comparison of key spaces of different algorithms. It can be seen that the algorithm in this paper has a larger key space than other algorithms, and is more capable of resisting statistical analysis attacks.

C. Histogram Analysis

The histogram of plaintext image has certain rules, and attackers can infer plaintext information from the histogram. Fig. 14(a)–(c) are histograms of point clouds color information, which can be seen to have regularity. Image data is encrypted with qualified algorithms, which can make the histogram of encrypted image no longer have specific rules and can resist statistical attacks. Fig. 14(d)–(f) are the histograms of the encrypted point cloud. It can be seen that they have no specific rules after encrypting and all obey uniform distribution, indicating that the algorithm can resist statistical attacks.

1) Histogram Variance Analysis: The histogram variance measures the distribution of pixel values in the histogram. The Lower the variance, higher is the grayscale uniformity [28]. The histogram variance is expressed as follow formula:

$$var(Z) = \frac{1}{p^2}\sum_{i=1}^{p}\sum_{j=1}^{p}\frac{1}{2}(z_i - z_j)^2, \tag{36}$$

where p represents the grayscale value, z_i, z_j represent the occurrences of corresponding grayscale values in the histogram.

Table IV presents the histogram variance between original data and cipher data. Notably, the original data exhibits a significant histogram variance. As data volume grows, the histogram variance in encrypted data rises correspondingly.

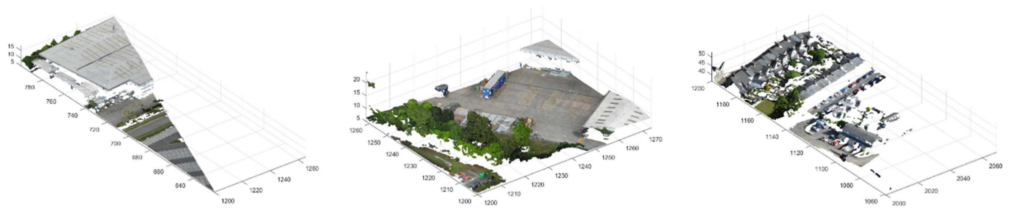
Table IV also compares the results of "Lena" to other algorithms, revealing that, after implementing the proposed scheme, "Lena" demonstrates a reduced histogram variance to 881.6563, suggesting a more even pixel distribution in the cipher image.

2) Chi-square Test Analysis: The chi-square test is commonly used to examine whether an image approximates a uniform distribution. The expression for chi-square test is shown in (37):

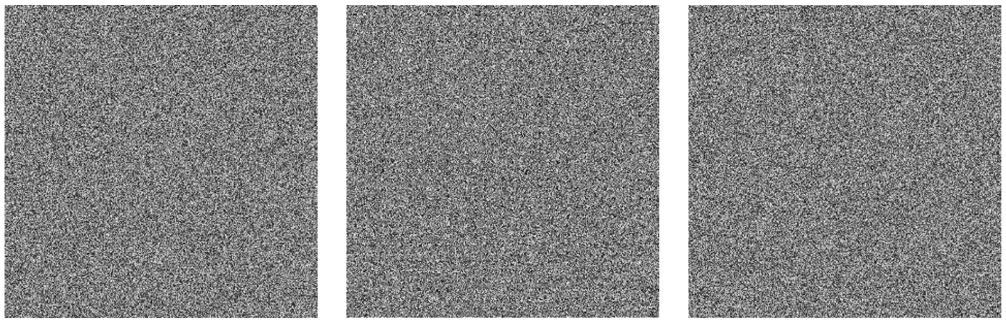
$$\chi^2_{test} = \sum_{i=0}^{255}\frac{(f_i - g)^2}{g}, \tag{37}$$

where $g = (M \times N)/256$, f_i represents the frequency of the i^th pixel value in the image. Table V presents the results of the chi-square test for both original data and cipher data.

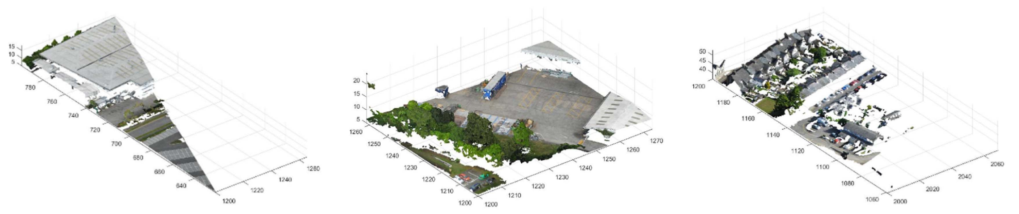
Table V shows that the chi-square test values for original point clouds data are notably high, indicating an uneven pixel



(a) Point clouds BH _11 (b) Point clouds BH_13 (c) Point clouds CB_32



(d) Encrypted and compressed BH_11 (e) Encrypted and compressed BH_13 (f) Encrypted and compressed CB_32



(g) Decrypted and decompressed BH_11 (h) Decrypted and decompressed BH_13 (i) Decrypted and decompressed CB_32

Fig. 13. Dataset used for simulation.

TABLE III
KEY SPACE OF DIFFERENT ALGORITHMS

Algorithm	Key space
Our	2^{704}
Ref. [24]	10^{56}
Ref. [25]	10^{56}
Ref. [26]	10^{79}
Ref. [27]	10^{70}

distribution. In contrast, the test values of cipher data are below the significance levels of 1% ($\chi^2_{255,0.01} = 310.457$) and 5% ($\chi^2_{255,0.05} = 293.2478$). Thus, it can be considered that the grayscale value is uniformly distributed. Compared to other algorithms, the encrypted "Lena" using the proposed method demonstrates a significantly lower chi-square value to 220.4141, indicating superior encryption performance.

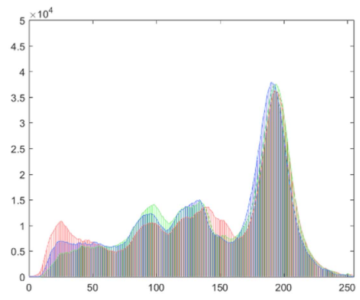
D. Information Entropy Analysis

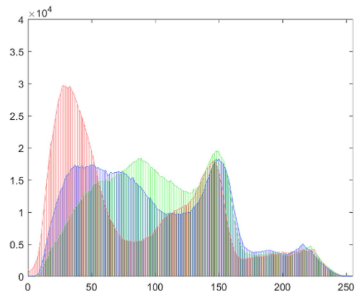
Information entropy can measure the amount of information of an image, and can indicate whether the image has randomness and uncertainty. It was proposed by Shannon in 1948:

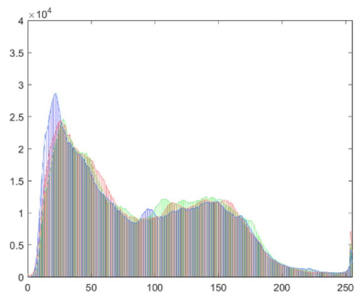
$$H = -\sum_{i=0}^{L-1} p(i) \log_2 p(i) \tag{38}$$

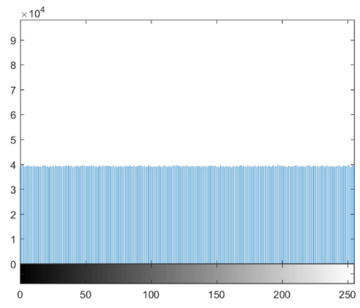
when a pixel is represented by 8 bits, $L = 256$, and the theoretical value of information entropy H is 8. The closer H is to 8, the more information the image has, the lower the degree of visual discrimination, and the more random it is. For the encrypted image, the confidentiality of the image is stronger. Table VI shows the information entropies before and after point clouds encryption. The information entropies of color information are measured before encryption.

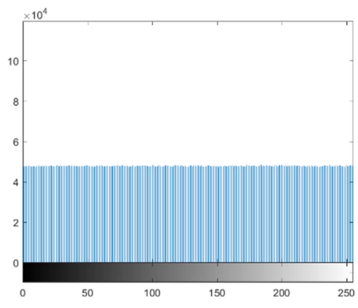
It can be seen from Table VI that the information entropies before encryption and compression are greater than 7, and the information entropies after encryption and compression are equal to 8, indicating that the point cloud data becomes more random and uncertain after encryption and compression, indicating that the algorithm in this paper has high randomness and security. Table VII shows the information entropies of Lena using different algorithms. It can be seen that the information entropy encrypted by the algorithm in this paper is closer to


(a) Point clouds BH_11


(b) Point clouds BH_13


(c) Point clouds CB_32


(d) Encrypted and compressed BH_11


(e) Encrypted and compressed BH_13

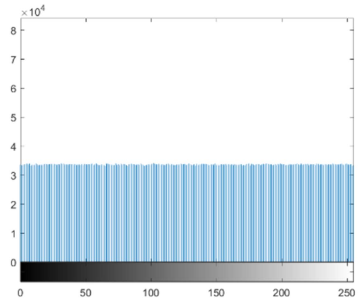

(f) Encrypted and compressed CB_32

Fig. 14. Results for histogram analysis.

TABLE IV
COMPARISON OF HISTOGRAM VARIANCE OF THE PROPOSED SCHEME WITH OTHER ALGORITHMS

Algorithms	Data	Histogram variance	
		Original data	Cipher data
Ours	BH_11	605664451.7	37894.767
	BH_13	315971920.4	35695.0002
	CB_32	384681410.1	28388.6225
	Lena (256×256)	38951	261.6797
	Lena (512×512)	633340	881.6563
Ref. [29]	Lena (512×512)	633400	982.5703
Ref. [30]	Lena (512×512)	633400	974.8

TABLE V
COMPARISON OF CHI-SQUARE VALUES OF THE PROPOSED SCHEME WITH OTHER ALGORITHMS

Algorithms	Data	χ^2_{test}		
		Original data	Cipher data	Results
Ours	BH_11	5287019.222	247.4341	Pass
	BH_13	2937645.388	253.9627	Pass
	CB_32	3563016.691	215. 9773	Pass
	Lena (512×512)	158349.3555	220.4141	Pass
Ref. [29]	Lena (512×512)	-	245.6426	Pass
Ref. [30]	Lena (512×512)	-	233.1353	Pass

8 than other algorithms, indicating that this algorithm is more random, secure and invisible.

TABLE VI
INFORMATION ENTROPIES OF DIFFERENT DATA

Point cloud	Information entropy	
	Origianl data	Cipher data
BH_11	7.539	8.0000
BH_13	7.6362	8.0000
CB_32	7.5824	8.0000
Lena	7.3740	7.9971

TABLE VII
INFORMATION ENTROPIES ON LENA OF DIFFERENT ALGORITHMS

Algorithm	Entropy
Our	7.9971
Ref. [25]	7.9915
Ref. [26]	7.9949
Ref. [27]	7.9969
Ref. [31]	7.996513

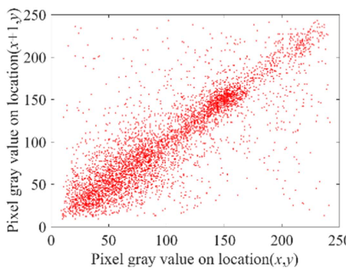
E. Pseudo-Randomness of Cipher Images

In this paper, the pseudo-randomness of cipher images is tested with NIST suite. The cipher images are decomposed into bit streams as the input of NIST test.The results are listed in Table VIII.

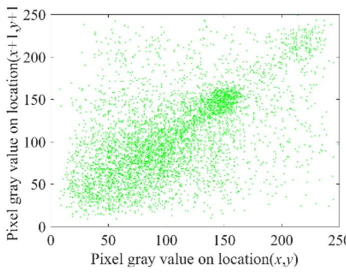
As can be seen from the Table VIII, the P-values for all of the three ciphertext images exceeded 0.01. Therefore, the cipher images can be regarded as uniformly random.

TABLE VIII
PSEUDO-RANDOMNESS TEST RESULTS OF THE CIPHER IMAGES

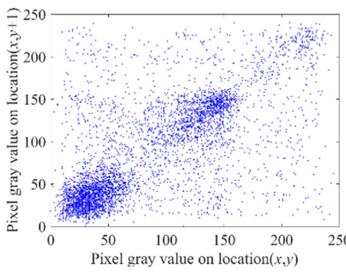
Test Items	P-value			
	BH_11	BH_13	CB_32	Result
Monobit frequency test	0.6484	0.2416	0.1313	Pass
Block Frequency test	0.568	0.5592	0.0642	Pass
Runs test	0.0155	0.0841	0.4531	Pass
Longest-run-of ones in a block	0.0669	0.6383	0.6812	Pass
Binary matrix rank test	0.0265	0.1031	0.0774	Pass
Discrete Fourier transform test	0.6424	0.2701	0.0675	Pass
Non-overlapping template matching test	0.582	0.3817	0.7109	Pass
Overlapping template matching test	0.4572	0.1092	0.1584	Pass
Maurer's universal statistical test	0.99	0.7212	0.0651	Pass
Linear complexity test	0.7129	0.1118	0.6708	Pass
Serial test	0.1299	0.1358	0.2849	Pass
Approximate entropy test	0.5234	0.9266	0.8848	Pass
Cumulative sums test	0.9875	0.9523	0.9996	Pass
Random excursions test	0.4721	0.6008	0.6196	Pass
Random excursions variant test	0.7191	0.1885	0.4078	Pass



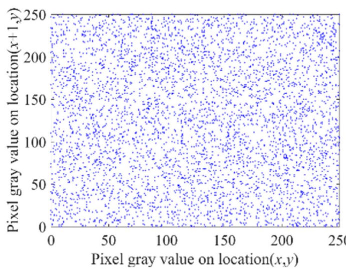
(a) Pixel distribution in horizontal of red component



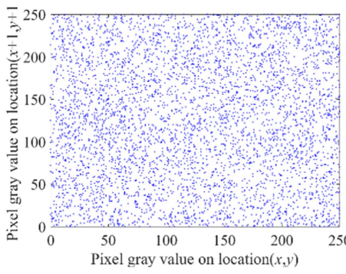
(b) Pixel distribution in diagonal of green component



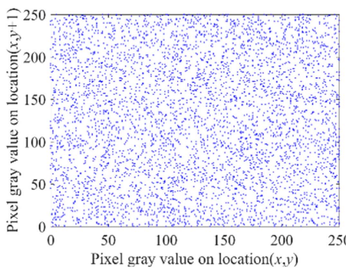
(c) Pixel distribution in vertical of blue component



(d) Encrypted and compressed pixel distribution in horizontal



(e) Encrypted and compressed pixel distribution in diagonal



(f) Encrypted and compressed pixel distribution in vertical

Fig. 15. Pixel distribution of BH_13.

F. Correlation Analysis

The adjacent pixel pairs of plaintext usually have a high correlation. Fig. 15(a)–(c) shows the distribution relationship of adjacent pixel pairs of 'BH_13' color components. It can be seen that the distribution is linear and has a high correlation. Fig. 15(d)–(e) shows the relationship between adjacent pixel pairs after encryption and compression of 'BH_13'. It can be seen that after encryption and compression, adjacent pixel pairs are evenly distributed on the coordinate plane, indicating that they have low correlation.

In addition, the correlation coefficient can be used to analyze the correlation between adjacent pixel pairs. The formula is expressed as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{39}$$

where x and y represent adjacent pixel pairs. The value of r_{xy} is between 0 and 1. The closer to 0, the lower the correlation between x and y; on the contrary, the higher the correlation between them. Table IX shows that 5000 pairs of adjacent

TABLE IX
CORRELATION COEFFICIENTS OF DIFFERENT DATA

Data	Original data			Cipher data		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
BH_11	0.9992	0.9783	0.9781	0.0005	0.0028	0.0001
BH_13	0.9963	0.8994	0.8995	-0.0003	-0.0009	0.0007
CB_32	0.9953	0.9578	0.9574	-0.0001	-0.0011	0.0007
Lena	0.9707	0.9428	0.9205	-0.0014	0.001	0.0003

TABLE X
CORRELATION COEFFICIENTS OF LENA ENCRYPTED BY DIFFERENT ALGORITHMS

Image	Algorithm	Direction		
		Horizontal	Vertical	Diagonal
Lena	Our	-0.0014	0.001	0.0003
	Ref. [17]	-0.0009	-0.0012	0.001
	Ref. [26]	-0.0009	-0.0005	0.0029
	Ref. [27]	-0.0138	-0.0027	-0.0026
	Ref. [31]	0.000938	0.003295	0.000563

TABLE XI
CR OF DIFFERENT POINT CLOUDS

Point Cloud	CR
BH_11	3.817606
BH_13	3.756283
CB_32	3.811355

pixel of multiple point cloud color data are randomly selected respectively, and the correlation coefficients before and after encryption and compression are calculated.

The correlation coefficients of the color data of the original point clouds are close to 1, with strong correlation. After encryption and compression by this algorithm, they are close to 0, with low correlation. It shows that this algorithm has greater randomness and higher security. Table X shows the correlation coefficients of Lena encrypted by different algorithms. It can be seen that the correlation coefficients of this algorithm is closer to 0 than that of other algorithms, indicating that the randomness and security of this algorithm are higher.

G. Comprssion Performance Analysis

Compression ratio (CR) is one of the indexes to measure compression performance. It is expressed as:

$$CR = \frac{M \times N}{m \times n} \tag{40}$$

where $M \times N$ and $m \times n$ are the dimensions of the data before and after compression. The data here is not necessarily two-dimensional data, but a general expression.

In this paper, near lossless PCL compression is used. After compression, a series of encryption operations are performed. Table XI shows the final compression rate of point clouds data.

TABLE XII
RMSE OF THE POINT CLOUDS

Point Cloud	$RMSE_1$	$RMSE_2$
BH_11	0.15%	0.15%
BH_13	0.21%	0.21%
CB_32	0.19%	0.19%

Root mean square error (RMSE) can be used to measure the difference between compressed and reconstructed point clouds. The root mean square error is expressed as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n} (y_i - y_i')^2} \tag{41}$$

where y_i and y_i' are the original value and the value to be measured, respectively, and n is the number to be measured.

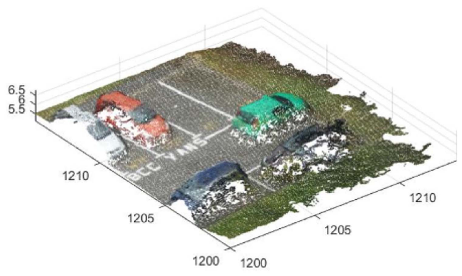
The compression algorithm is near lossless, and whether it is completely lossless depends on the number of bits in the data set and the accuracy of the computer. Because of the high precision of this data set, the highest number of integer and decimal digits reaches 8. In order to ensure the clearness and readability of the reconstructed data and the efficiency of calculation, use the compression parameters in Section II-D for compression. Calculate the RMSE values of the original data and the reconstructed data, and the results are shown in Table XII.

As seen in Table IX, $RMSE_1$ is the RMSE of the reconstructed point cloud and the origin cloud. Then encryption algorithm is added to the compression algorithm, and the result is decrypted and reconstructed, and then $RMSE_2$ is calculated. It can be compared that the values of $RMSE_1$ and $RMSE_2$ are the same, indicating that the encryption algorithm does not cause data loss, and data loss is determined by precision and compression parameters. Under this parameter setting, the compression loss is very small, about equal to 0, which is near lossless compression.

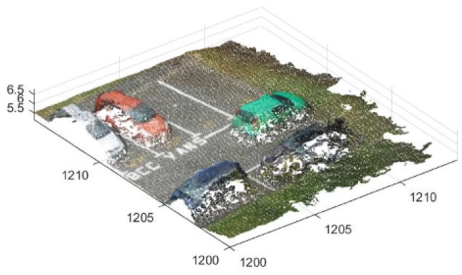
Fig. 16 shows the local magnified comparison of 'BH_13' original data and reconstructed data. In this parameter setting, the loss of original data and reconstructed data can be ignored.

H. Plaintext Sensitivity Analysis

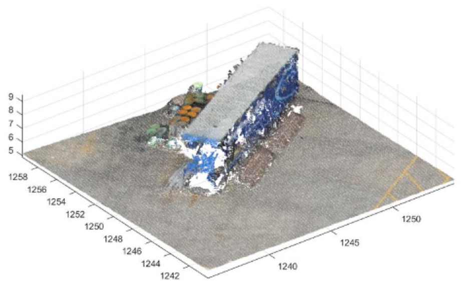
Plaintext sensitivity means that when the same plaintext image is slightly changed, the same encryption algorithm is used to encrypt the image before and after the change, and the encryption results are very different. The difference of results



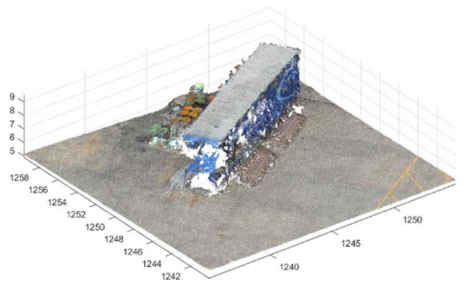
(a) Local image 1 of origin point clouds



(b) Reconstructed result for local image 1



(c) Local image 2 of origin point clouds



(d) Reconstructed result for local image 2

Fig. 16. Local comparison of point cloud 'BH_13'.

can be measured by the number of pixel change rate (NPCR) and the uniform average change intensity (UACI), and the formulas are as follows:

$$\begin{cases} NPCR = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left|\text{Sign}(P(i,j)-P'(i,j))\right|}{M\times N} \times 100\% \\ \text{Sign}(x) = \begin{cases} 1, & x>0 \\ 0, & x=0 \\ -1, & x<0 \end{cases} \end{cases} \quad (42)$$

$$UACI = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N}\left|P(i,j)-P'(i,j)\right|}{255\times M\times N} \times 100\% \quad (43)$$

TABLE XIII
NPCRs and UACIs of Different Data

Image	NPCR(%)	UACI(%)
BH_11	99.6099	33.5004
BH_13	99.6105	33.4681
CB_32	99.6107	33.4785
Lena	99.6092	33.4629

TABLE XIV
NPCR and UACI Values of Lena for Different Algorithms

	Our	Ref. [15]	Ref. [26]	Ref. [31]	Ref. [32]
NPCR(%)	99.6092	99.61	99.6101	99.5819	99.6147
UACI(%)	33.4629	33.4	33.4654	33.5138	33.4723

where P and P' are images before and after the change. The theoretical values of NPCR and UACI are 99.6094% and 33.4635% respectively. Randomly change a pixel value (plus 1) of the point cloud color information, encrypt and compress the images before and after the change, and calculate the NPCR and UACI values of the encrypted image. Repeat for 50 times, calculate the average values of NPCR and UACI, and record them in Table XIII .

It can be seen from Table XIII that the experimental results are close to the theoretical values, indicating that the algorithm is sensitive to plaintext. "Lena" was slightly changed, and different algorithms were used for encryption. The values of NPCR and UACI in the encrypted image were calculated, and the results were recorded in Table XIV. It can be seen that the results of this algorithm are closer to the theoretical values, indicating that this algorithm is more sensitive to plaintext and has higher security.

I. *Key Sensitivity Analysis*

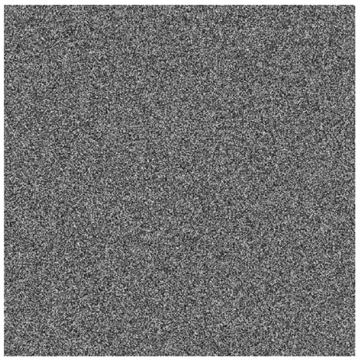
When other factors remain unchanged, the key is slightly changed (such as 10^{-15}), and the resulting encrypted images are very different, indicating that the algorithm is key sensitive. Add 10^{-15} to the initial values of x_0, y_0, z_0, t_0, p_0, a_0 respectively, and then conduct six encryption and compression experiments in turn. The results are shown in Fig. 17.

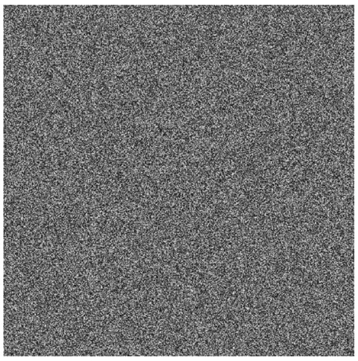
Make the difference images of the encrypted image before and after the initial values change, as shown in Fig. 18. It can be seen that even if the key is slightly changed, the encryption results obtained by this algorithm are still very different, indicating that this algorithm has strong key sensitivity.

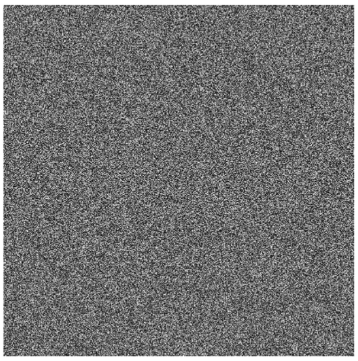
J. *Cryptanalysis*

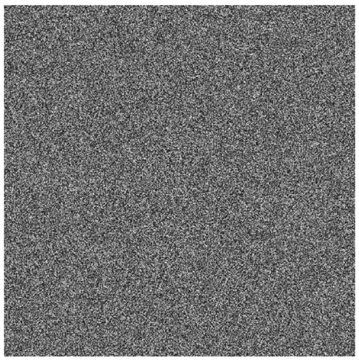
Cryptanalysis often involves known-plaintext, chosen-plaintext, and chosen-ciphertext attacks, all of which have the potential to undermine encryption schemes [33].

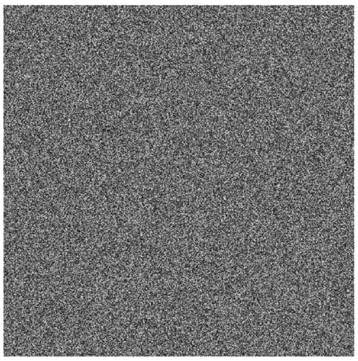
1) Known – Plaintext and Chosen – Plaintext Attack Analysis: In the proposed scheme, the key is composed of the initial values and system parameters of the chaotic system, which are determined by the Hash512 value of the plaintext image. This implies that even minor changes in the image will lead to different initial parameters. Owing to the sensitivity of initial


(a) $x' = x_0 + 10^{-15}$


(b) $y' = y_0 + 10^{-15}$


(c) $z' = z_0 + 10^{-15}$


(d) $t' = t_0 + 10^{-15}$


(e) $p' = p_0 + 10^{-15}$

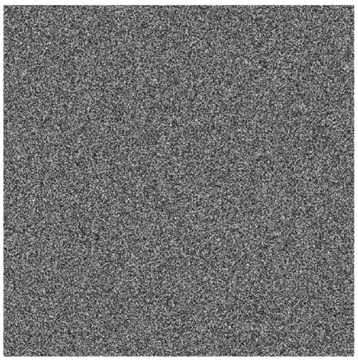
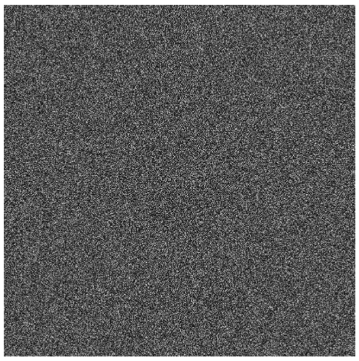
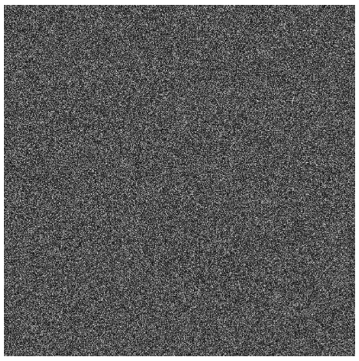
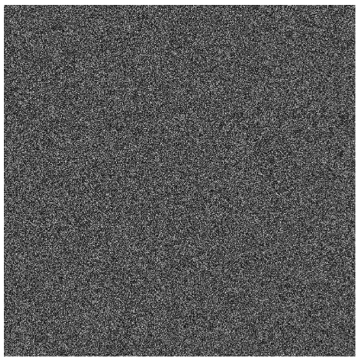
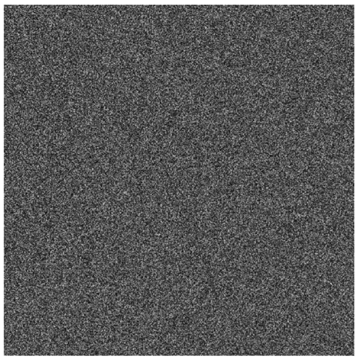

(f) $a' = a_0 + 10^{-15}$

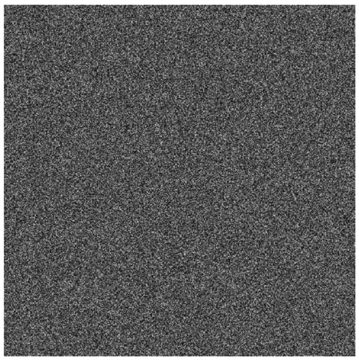
Fig. 17. Encrypted images with modified keys.


(a) —Fig. 13(e) - Fig. 17(a)—


(b) —Fig. 13(e) - Fig. 17(b)—


(c) —Fig. 13(e) - Fig. 17(c)—


(d) —Fig. 13(e) - Fig. 17(d)—


(e) —Fig. 13(e) - Fig. 17(e)—

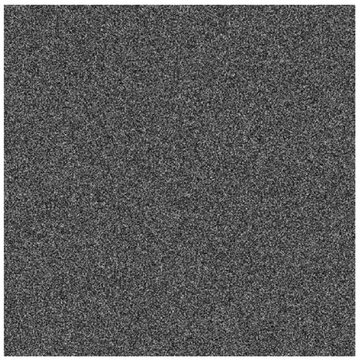
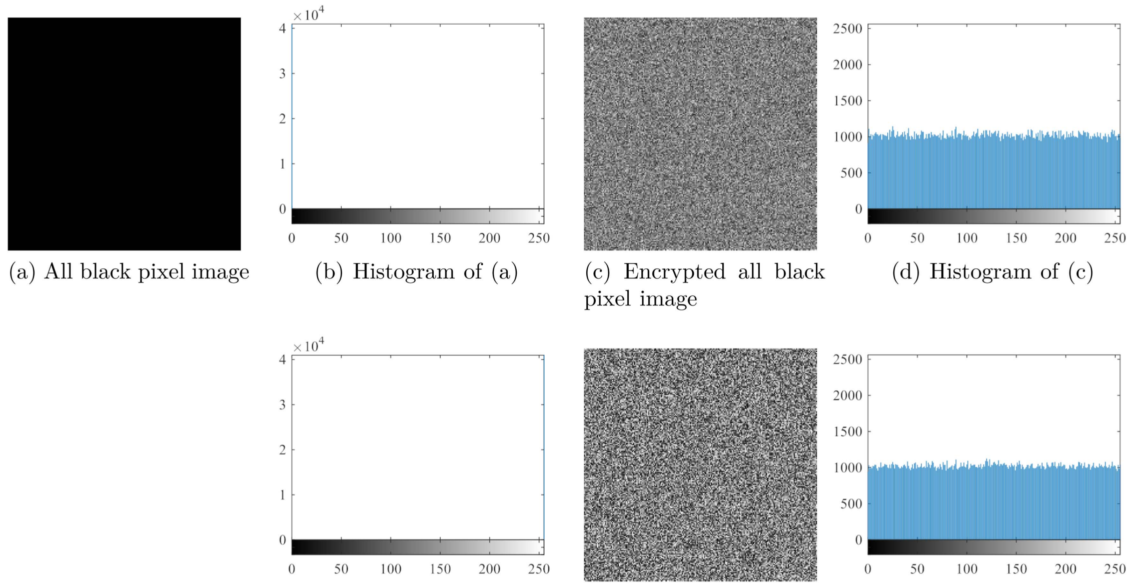

(f) —Fig. 13(e) - Fig. 17(f)—

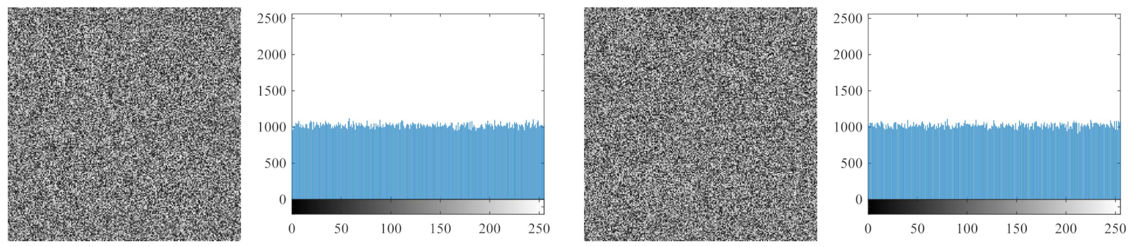
Fig. 18. Difference images.



(a) All black pixel image (b) Histogram of (a) (c) Encrypted all black pixel image (d) Histogram of (c)

(e) All white pixel image (f) Histogram of (e) (g) Encrypted all white pixel image (h) Histogram of (g)

Fig. 19. Simulation results of all-zero and all-one plaintext images.



(a) Result of all-zero cipher image (b) Histogram of (a) (c) Result of all-one cipher image (d) Histogram of (c)

Fig. 20. Simulation results of all-zero and all-one cipher images.

conditions in chaotic system, different parameters produce entirely distinct chaotic sequences, resulting in completely different cipher images. Consequently, even if an attacker possesses a known plaintext, they cannot obtain the correct cipher image.

We conducted a chosen-plaintext attack test utilizing images with all black pixels and all white pixels. The simulation results are shown in Fig. 19.

As observed from Fig. 19, the pronounced contrast in visual appearance between the encrypted images and the plaintext images. Furthermore, the histogram distribution displays uniformity. This characteristic makes it difficult for potential attackers to extract any meaningful information from the original images. This confirms the effectiveness of the proposed scheme against chosen-plaintext attacks.

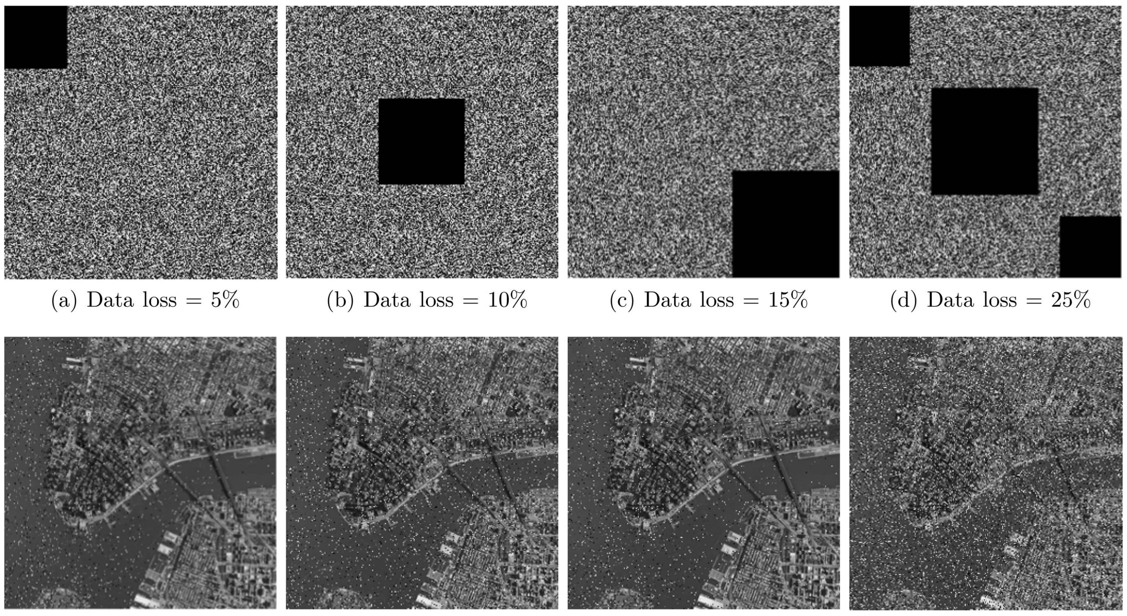
2) Chosen-Ciphertext Attack Analysis: For the chosen-ciphertext attack, we employ an all-zero or an all-one image as the ciphertext image C_0 and its corresponding decoded image P_0 to perform XOR operations to obtain part of the key stream K, and then perform XOR operations on the key stream K and Lena's cipher image C to observe the results. The Fig. 20 illustrates that the chosen-ciphertext attack does not yield meaningful information from the result or its associated histogram, indicating the attack's failure.

K. Robust Analysis

During the transmission of cipher images, issues like data loss or noise attacks can arise, necessitating that the algorithm possess a robust resistance to these challenges. In this section, we evaluate the proposed scheme using remote sensing images from "New York" and cropping attacks and noise attack analysis.

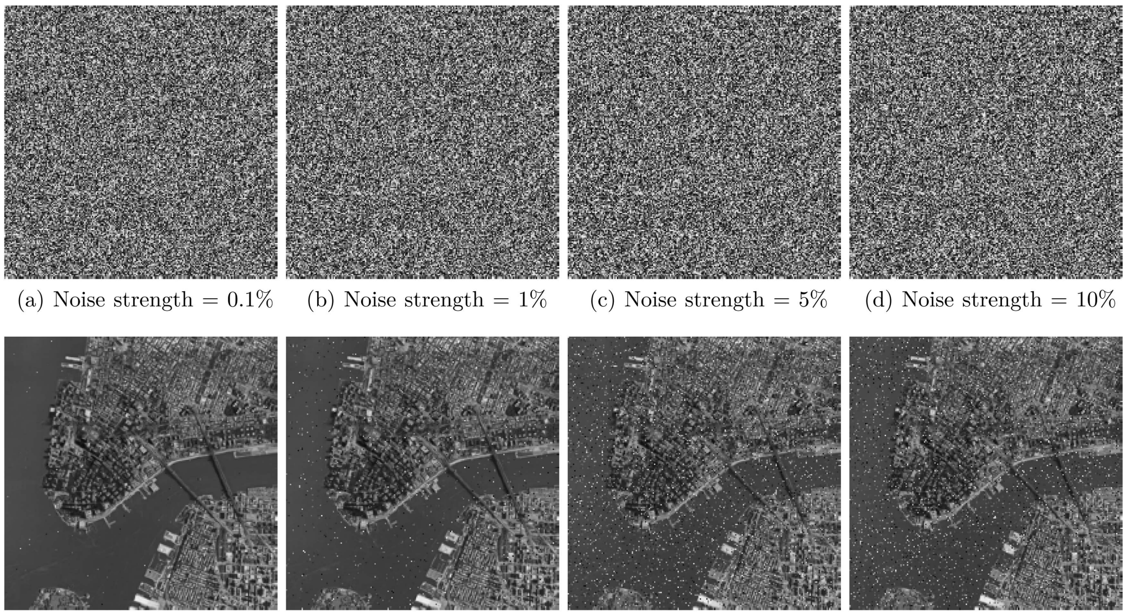
1) Cropping Attacks Analysis: Cropping attacks refers to the process of simulating data loss by setting specific values of the cipher image to 0, followed by decryption to observe if the result image remains clear. We tested various positions and proportions of cropping attacks on cipher images. The Cropping attacks results are illustrated in Fig. 21.

The Fig. 21 demonstrates that data is lost at the beginning, middle, or end of the cipher images. However, the decrypted


(a) Data loss = 5% (b) Data loss = 10% (c) Data loss = 15% (d) Data loss = 25%

(e) Decrypted image of (a) (f) Decrypted image of (b) (g) Decrypted image of (c) (h) Decrypted image of (d)

Fig. 21. Cropping attacks results of "New York" image.


(a) Noise strength = 0.1% (b) Noise strength = 1% (c) Noise strength = 5% (d) Noise strength = 10%

(e) Decrypted image of (a) (f) Decrypted image of (b) (g) Decrypted image of (c) (h) Decrypted image of (d)

Fig. 22. Salt & Pepper noise attack results of "New York" image.

images retain resemblance to the original images, confirming the proposed scheme has the ability to resist the cropping attacks.

2) Noise Attacks Analysis: Fig. 22(a)–(d) displays the effects of adding 0.1%, 1%, 5% and 10% Salt & pepper noise to ciphertext images. Fig. 22(e)–(h) presents the results from the decryption of the corresponding ciphertexts. The figure reveals that even with a 0.1% noise attack, the difference between the decrypted and original images is minimal. Additionally, after a 10% noise attack, the decrypted image remains discernible, suggesting that the proposed scheme effectively withstands noise attacks.

L. Time Complexity Analysis

The proposed compression-encryption algorithm encompasses PCL compression, ECC encryption with private key, iterative generation of chaotic sequences, MBS, Helix diffusion and multiple S-boxes substitution. Assuming an initial three-dimensional point cloud data size of m, the computational complexity for PCL compression is $O(m \log m)$. However, PCL compression has high computing efficiency because PCL uses advanced high-performance computing technologies such as OpenMP, GPU, CUDA, etc. to improve computing efficiency in parallel. The ECC encryption involves two scalar multiplications and one point addition operation, leading to complexities of $O(\log k)$ and $O(1)$ respectively, where k is the binary length of the private key.

Assume that the compressed data size is $M \times N$. The time complexity of generating 3D-TCICM system based sequence is $O(3 \times M \times N)$. In the MBS stage, all pixels have been processed through (20), with a computational complexity of $O(3 \times M \times N)$. In the diffusion stage, all the diffusion operations are performed by using bitwise XOR operations and MOD operation, so the computational complexity to perform Helix diffusion of two numbers having n-bits is $O(2^{n-2} \times M \times N)$. In the process of Multiple S-boxes substitution, for 8-bit pixels, their computational complexity is $O(8 \times M \times N)$. Thus, the total computational complexity of the proposed encryption process is $O(M \times N)$.

In a word, the compression part of the algorithm has a high computational efficiency, and the encryption part has a low computational complexity, which is suitable for large remote sensing point clouds.

V. Conclusion

This paper proposes an asymmetric encryption and compression algorithm for remote sensing point clouds. Firstly, the remote sensing point cloud is divided into blocks and compressed to reduce the computational complexity. The compressed data is encrypted based on the proposed chaotic system 3D-TCICM. It has broad chaotic trajectories, high LEs and PE close to 1, that is, outstanding chaotic behavior. Its initial values are generated by ECC algorithm and point cloud. Asymmetric encryption algorithm improves the security and the plaintext sensitivity of key. And the encryption factors generated by ECC algorithm are applied to subsequent scrambling, diffusion and S-box substitution, which improves the security of the overall algorithm. The MBS reduces the correlation between blocks and the computational complexity. Helix diffusion in GF(2^8) field carries out forward and reverse diffusion respectively, which improves the diffusion effect. Multiple S-boxes subsitiute each piece of data respectively, ensuring the diversity of S-boxes and improving the randomness of the algorithm. The experimental analysis shows that this algorithm has a large key space, chaotic and random encryption results, and has plaintext sensitivity and key sensitivity. It has low computational complexity under the condition of ensuring high security. To sum up, it is a qualified encryption and compression algorithm for remote sensing point cloud. Currently, the proposed scheme has only been tested through simulation and hasn't been implemented on hardware. In the future, we aim to enhance the temporal efficiency of our algorithm by leveraging parallel processing on advanced platforms such as CUDA, GPU, CPU, and OpenCL.

References

[1] Y. Lu, M. Gong, Z. Gan, X. Chai, L. Cao, and B. Wang, "Exploiting one-dimensional improved chebyshev chaotic system and partitioned diffusion based on the divide-and-conquer principle for 3D medical model encryption," *Chaos Solitons Fractals*, vol. 171, Jun. 2023, Art. no. 113449.

[2] K. A. K. Patro, M. P. Kumar, and B. Acharya, "An efficient dual-stage pixel-diffusion based multimedia-image encryption using one-type 1D chaotic maps," *Sadhana-Acad. Proc. Eng. Sci.*, vol. 47, no. 3, 2022, Art. on. 161.

[3] S. Dash, S. Padhy, S. A. Devi, S. Sachi, and K. A. K. Patro, "An efficient intra-inter pixel encryption scheme to secure healthcare images for an IoT environment," *Expert Syst. Appl.*, vol. 231, Nov. 2023, Art. no. 120622.

[4] S. Wang, Q. Peng, and B. Du, "Chaotic color image encryption based on 4D chaotic maps and DNA sequence," *Opt. Laser Technol.*, vol. 148, Apr. 2022, Art. no. 107753.

[5] X. Wang and X. Du, "Pixel-level and bit-level image encryption method based on logistic-Chebyshev dynamic coupled map lattices," *Chaos Solitons Fractals*, vol. 155, Feb. 2022, Art. no. 111629.

[6] Q. Lai, H. Zhang, P. D. K. Kuate, G. Xu, and X.-W. Zhao, "Analysis and implementation of no-equilibrium chaotic system with application in image encryption," *Appl. Intell.*, vol. 52, no. 10, pp. 11448–11471, 2022.

[7] W. Li, X. Chang, A. Yan, and H. Zhang, "Asymmetric multiple image elliptic curve cryptography," *Opt. Lasers Eng.*, vol. 136, Jan. 2021, Art. no. 106319.

[8] J. Bhat and A. H. Moon, "Color image encryption and authentication using dynamic dna encoding and hyper chaotic system," *Expert Syst. Appl.*, vol. 206, Nov. 2022, Art. no. 117861.

[9] A. Sahasrabuddhe and D. S. Laiphrakpam, "Multiple images encryption based on 3D scrambling and hyper-chaotic system," *Inf. Sci.*, vol. 550, pp. 252–267, 2021.

[10] X. Zhang and W. Gao, "Adaptive geometry partition for point cloud compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4561–4574, Dec. 2021.

[11] Z. Que, G. Lu, and D. Xu, "Voxelcontext-Net: An octree based framework for point cloud compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6038–6047.

[12] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Lossless coding of point cloud geometry using a deep generative model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 12, pp. 4617–4629, Dec. 2021.

[13] D. He, C. He, L. Jiang, H. Zhu, and R. Hu, "Chaotic characteristics of a one-dimensional iterative map with infinite collapses," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 48, no. 7, pp. 900–906, Jul. 2001.

[14] W. Liu, K. Sun, Y. He, and M. Yu, "Color image encryption using three-dimensional sine ICMIC modulation map and DNA sequence operations," *Int. J. Bifurcation Chaos*, vol. 27, no. 11, Oct. 2017, Art. no. 1750171.

[15] C. Wei and G. Li, "A selective image encryption scheme using LICC hyperchaotic system," *IET Image Process.*, vol. 16, no. 12, pp. 3342–3358, Oct. 2022.

[16] S. Patel, K. P. Bharath, and R. M. Kumar, "Symmetric keys image encryption and decryption using 3D chaotic maps with DNA encoding technique," *Multimedia Tools Appl.*, vol. 79, no. 43/44, pp. 31739–31757, Nov. 2020.

[17] W. Yan, Z. Jiang, X. Huang, and Q. Ding, "A three-dimensional infinite collapse map with image encryption," *Entropy*, vol. 23, no. 9, Sep. 2021, Art. no. 1221.

[18] Z. Yan, "Q-S (complete or anticipated) synchronization backstepping scheme in a class of discrete-time chaotic (hyperchaotic) systems: A symbolic-numeric computation approach," *Chaos*, vol. 16, no. 1, Mar. 2006, Art. no. 013119.

[19] L. Y. Zhang et al., "On the security of a class of diffusion mechanisms for image encryption," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1163–1175, Apr. 2018.

[20] J. Chen, L. Chen, L. Y. Zhang, and Z. Zhu, "Medical image cipher using hierarchical diffusion and non-sequential encryption," *Nonlinear Dyn.*, vol. 96, no. 1, pp. 301–322, Apr. 2019.

[21] X. Feng, S. Nan, R. Ma, and H. Zhang, "A lossless compression and encryption method for remote sensing image using LWT, Rubik's cube and 2D-CCM," *Int. J. Bifurcation Chaos*, vol. 32, no. 10, Aug. 2022, Art. no. 2250149.

[22] Q. Hu et al., “Towards semantic segmentation of urban-scale 3D point clouds: A dataset, benchmarks and challenges,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4975–4985.

[23] Q. Xu, K. Sun, C. Cao, and C. Zhu, “A fast image encryption algorithm based on compressive sensing and hyperchaotic map,” *Opt. Lasers Eng.*, vol. 121, pp. 203–214, Oct. 2019.

[24] Y. Zhang, Y. He, P. Li, and X. Wang, “A new color image encryption scheme based on 2DNLCML system and genetic operations,” *Opt. Lasers Eng.*, vol. 128, May 2020, Art. no. 106040.

[25] J. Li et al., “Holographic encryption algorithm based on bit-plane decomposition and hyperchaotic Lorenz system,” *Opt. Laser Technol.*, vol. 152, Aug. 2022, Art. no. 108127.

[26] S. Nan, X. Feng, Y. Wu, and H. Zhang, “Remote sensing image compression and encryption based on block compressive sensing and 2D-LCCCM,” *Nonlinear Dyn.*, vol. 108, no. 3, pp. 2705–2729, May 2022.

[27] Z. Gan, X. Chai, J. Bi, and X. Chen, “Content-adaptive image compression and encryption via optimized compressive sensing with double random phase encoding driven by chaos,” *Complex Intell. Syst.*, vol. 8, no. 3, pp. 2291–2309, Jun. 2022.

[28] K. A. K. Patro and B. Acharya, “Secure multi-level permutation operation based multiple colour image encryption,” *J. Inf. Secur. Appl.*, vol. 40, pp. 111–133, Jun. 2018.

[29] K. A. K. Patro, A. Soni, P. K. Netam, and B. Acharya, “Multiple grayscale image encryption using cross -coupled chaotic maps,” *J. Inf. Secur. Appl.*, vol. 52, Jun. 2020, Art. no. 102470.

[30] K. A. K. Patro, B. Acharya, and V. Nath, “Secure, lossless, and noise-resistive image encryption using chaos, hyper-chaos, and DNA sequence operation,” *IETE Tech. Rev.*, vol. 37, no. 3, pp. 223–245, May 2020.

[31] H. Zhao, S. Wang, and X. Wang, “Fast image encryption algorithm based on multi-parameter fractal matrix and MPMCML system,” *Chaos Solitons Fractals*, vol. 164, Nov. 2022, Art. no. 112742.

[32] M. Li, M. Wang, H. Fan, K. An, and G. Liu, “A novel plaintext-related chaotic image encryption scheme with no additional plaintext information,” *Chaos Solitons Fractals*, vol. 158, May 2022, Art. no. 111989.

[33] K. A. K. Patro and B. Acharya, “An efficient colour image encryption scheme based on 1-D chaotic maps,” *J. Inf. Secur. Appl.*, vol. 46, pp. 23–41, Jun. 2019.