Article

Vector Map Encryption Method Based on Secret Sharing

Fanshuo Liu , Baiyan Wu * , Xi Liu * , Zixuan Bu , Haodong Zhang

Posted Date: 20 January 2025

doi: 10.20944/preprints202501.1387.v1

Keywords: vector map; data encryption; secret sharing; map encryption; geographic information system

Article

Vector Map Encryption Method Based on Secret Sharing

Fanshuo Liu [1,2], Baiyan Wu [1,2,*], Xi Liu [3,*], Zixuan Bu [1,2] and Haodong Zhang [1,2]

[1] National-Local Joint Engineering Laboratory of Geo-Spatial Information Technology, Hunan University of Science and Technology, Xiangtan 411201, China
[2] School of Earth Sciences and Spatial Information Engineering, Hunan University of Science and Technology, Xiangtan 411201, China
[3] Hunan Engineering Research Center of Geographic Information Security and Application, Changsha 410000, China
* Correspondence: B. Wu: wby@hnust.edu.cn; Tel.: +86-18973260701; X. Liu: lx0050@qq.com

Abstract: Vector map data is of great value and widely used in different fields. The issues for its data security have become increasingly urgent in the modern information age. Encryption technology converts the plaintext data into ciphertexts, making the data unreadable to unauthorized users, thus, plays a vital role in safeguarding sensitive information in various scenarios. Although several encryption algorithms for vector maps have been developed, most of the existing methods lack some very important security related properties, such as the disaster tolerance property, probabilistic property, diffusion property and the robustness to data RST (Rotation, Scaling and Translation) transformations, which greatly affects the security of the encryption algorithms. In this paper, a novel vector map encryption algorithm based on (k, n)-threshold secret sharing is proposed, which encrypts one map into n map shares and reconstructs the plaintext map by collecting at least k shares, thus improving the algorithm's security and achieving the disaster tolerance property. Moreover, random numbers and cipher-feedback mode are cooperated into the encryption process in the proposed method to achieve probabilistic and diffusion properties. In addition, quantized polar coordinate is defined and original map coordinates are transformed into quantized polar coordinates before encryption and decryption process to achieve robustness to data RST transformations. Experiments on map data of different types (including points, polylines, and polygons) demonstrate the effectiveness and superiority of the proposed method.

Keywords: vector map; data encryption; secret sharing; map encryption; geographic information system

1. Introduction

In modern information age, Geographic Information Systems (GIS) have been widely applied in various fields such as urban planning, environmental monitoring, traffic management, and the military industry [1–5]. Vector maps, as one of the main GIS data, contain rich geographic coordinate information [6,7]. However, with the popularity of vector maps, how to protect the security and privacy of this sensitive data has become an increasingly prominent challenge [8–11]. Currently, there are two main issues for the security protection of vector maps: confidentiality protection and copyright protection [12]. Copyright protection mainly includes technologies such as blockchain [13] and digital watermarking [14,15] which can embed copyright information into vector maps while ensuring data integrity. Confidentiality protection is mainly based on encryption technology, which converts the plaintext data into unreadable ciphertexts to prevent unauthorized users to access data. At present, a few vector map encryption schemes have been proposed and applied to the field of geographic information protection [16]. Existing encryption schemes can be roughly divided into three categories: Classical cryptography-based encryption schemes [17], spatial domain-based encryption schemes [23,24], and transform domain-based encryption schemes [18,19].

Classical cryptography-based encryption schemes treat the vector map as a binary file and encrypt and decrypt the vector map as a whole. These encryption algorithms can be either symmetric encryption techniques or asymmetric encryption techniques. Symmetric encryption schemes mainly employ Advanced Encryption Standard (AES) [20] and Data Encryption Standard (DES) [21], etc. while asymmetric encryption schemes mainly employ the Rivest Shamir Adleman algorithm (RSA) [22] and the ellipse curve cryptography algorithm (ECC) [23], etc. The algorithms of these encryption types are simple to implement and consume few resources. However, these algorithms have limited randomness and do not consider the structural characteristics of vector data, resulting in poor usability.

Spatial domain-based encryption schemes directly shuffle the spatial distribution of map vertices through various chaotic systems, scrambles, and obfuscations. In terms of chaotic systems, the local encryption method proposed in [24], uses the SM3 algorithm and a four-dimensional hyperchaotic system to generate pseudo-random sequences, and utilizes these sequences to perform confusion, scrambling, and encryption processing on vertex coordinates. Another example is the encryption method based on the double random position permutation strategy [25], which uses the double random position permutation strategy to encrypt all the coordinates of the vector map. This method can prevent a one-to-one mapping relationship between the plaintext and the ciphertext vector maps. But the encryption effect of this algorithm on points is relatively weak. The encryption algorithm based on image scrambling regards the vector map as a special kind of image and encrypts it using image scrambling techniques. Common scrambling methods include the Arnold transformation [26] and nonlinear scrambling [27], etc. These encryption algorithms have a fast encryption speed and wide applicability. However, they have poor probabilistic property, not semantically secure.

Transform domain-based encryption schemes, converting data from the spatial domain to the transform domain for encryption, belong to another classic type of encryption method. Commonly used transforms in these encryption methods include the Discrete Cosine Transform (DCT), the Discrete Wavelet Transform (DWT), and the Discrete Fourier Transform (DFT) [28]. Pham et al. [29] employed the Advanced Encryption Standard and keys to encrypt the feature vertices of the backbone objects and randomized all the vertices of the backbone objects through the random Gaussian distribution algorithm. Van et al. [30] proposed to classify the objects in each map layer, and all the coordinates of important objects are encrypted by using Discrete Wavelet Transform (DWT) and Discrete Fourier Transform (DFT).

In summary, although some representative models and algorithms for vector map data encryption have been proposed, there still exist the following deficiencies for these methods: (1) Plaintext map data is encrypted into only one ciphertext map, and only one ciphertext map is required for data decryption. The security of such encryption models is not very sufficient for the sensitive vector map data. (2) When the only encrypted map data is lost or damaged, it is difficult to restore the encrypted map to its original version, which will lead to permanent data loss and leakage or damage to the integrity of the data. (3) In most classical cryptography algorithms, when the same plaintext data is encrypted twice, the results of each encryption are the same. This enables attackers to discover the patterns of the encryption algorithm by analyzing a large number of plaintext and ciphertext pairs collected, and then crack the key or the plaintext. Moreover, some asymmetric encryption algorithms have poor diffusion properties, which reduces the security of the algorithms. (4) Many existing algorithms lack robustness to data RST (Rotation, Scaling and Translation) transformations which are the very common operations for vector map data. Thus, decryption for the encrypted data often fails after RST operations. (5) Some methods lack versatility. The encryption algorithms cannot support all 2D vector data types (points, polylines, polygons), or the encryption effect varies depending on the data elements, which will affect the overall security of the methods.

To address the issues above, this paper proposes a novel vector map encryption method based on (k, n)-threshold secret sharing, which encrypts one map into n map shares and reconstructs the plaintext map by collecting at least k shares, thus improving the algorithm's security and achieving the disaster tolerance property. Moreover, random numbers and cipher-feedback mode are cooperated into the encryption process in the proposed method to achieve probabilistic and diffusion properties. In addition, quantized polar coordinate is defined and original map coordinates are transformed into quantized polar coordinates before encryption and decryption process to achieve robustness to data RST transformations.

The remainder of this paper is organized as follows. Section 2 introduces the Shamir's secret sharing method. Section 3 describes in detail the methodology of the proposed method. Experimental results and performance of the proposed method are discussed and evaluated in Section 4. Finally, a conclusion is drawn in Section 5.

2. Shamir's Secret Sharing Method

In 1979, Shamir and Blakley proposed a (k, n)-threshold secret sharing scheme [31]. The method is to divide the secret information s into n shares and distribute them to n respective participants. Any set that contains at least k participants is an authorized subset and can reconstruct s correctly, while a set that contains $k - 1$ or fewer participants is an unauthorized subset and cannot get any information of s. Shamir's secret sharing scheme is widely used. The implementation process is as follows:

Select a finite field F_p, where $p \geq n$. Select $k - 1$ random numbers a_i $(i = 1,2,\cdots,k-1)$ from F_p and construct the polynomial

$$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \bmod p \tag{1}$$

Select n non-zero elements $x_1, x_2, \cdots, x_n$ from F_p which are different from each other and publicly known. Then, the n shares for s are obtained as $f(x_q)$, $q = 1,2,\cdots,n$.

When at least k shares $f(x_{t_j})$ $(j = 1,2,\cdots,k \text{ and } t_j \in \{1,2,\cdots,n\})$ are collected, the original data s can be reconstructed. The reconstruction of the original data is achieved through Lagrange interpolation, represented as:

$$f(x) = \sum_{j=1}^{k} \left(f\left(x_{t_j}\right) \prod_{\substack{1 \le l \le k \\ l \ne j}} \frac{x - x_{t_l}}{x_{t_j} - x_{t_l}} \right) mod\ p \tag{2}$$

Substituting $x = 0$ into the above formula, the original data $s = f(0)$ can be obtained.

3. Methodology

Vector maps use the vector data model to represent geographical entities and describe geospatial phenomena through geometric objects such as points, polylines, and polygons as well as their attributes. The geometric objects consist of ordered 2D coordinates sequences. The encryption operation is performed on each coordinate to obtain encrypted map data. In this paper, each coordinate is encrypted into n cipher coordinates by using (k, n) threshold secret sharing. After all coordinates are encrypted, n cipher map shares are obtained. Only by collecting at least k cipher shares can the plaintext map be reconstructed. As the secret sharing model processes data in the finite field, the map coordinates represented by real data are converted into integers on finite field before secret sharing operations. The conversion from real coordinates to integer coordinates is completed by using a data quantization process based on predefined quantization steps. In addition, in order to make the map decryption robust to the RST (Rotation, Scaling and Translation) operations which are the common operations for map data, the data quantization process and the secret sharing based encryption are all performed under a constructed polar coordinate system.

Specifically, the encryption process for a vector map data includes four sub-processes: (1) The original map coordinates are transformed into polar coordinates under a constructed polar coordinate system. (2) Quantization processes are performed to the polar coordinates using predefined quantization steps to obtain quantized polar coordinates. Each quantized polar coordinate is then divided into quantization index part and quantization fraction part. (3) Secret sharing operation is performed to each quantization index part and n ciphertexts for each index part are obtained accordingly. Each ciphertext for index part is then combined with the fraction part to form n ciphertexts for each quantized polar coordinate. (4) The n ciphertexts of each quantized polar coordinate are then processed using inverse quantization and inverse polar coordinate transformation in sequence and n encrypted map shares for the original map are finally obtained. The flowchart of the encryption process is shown in Figure 1.

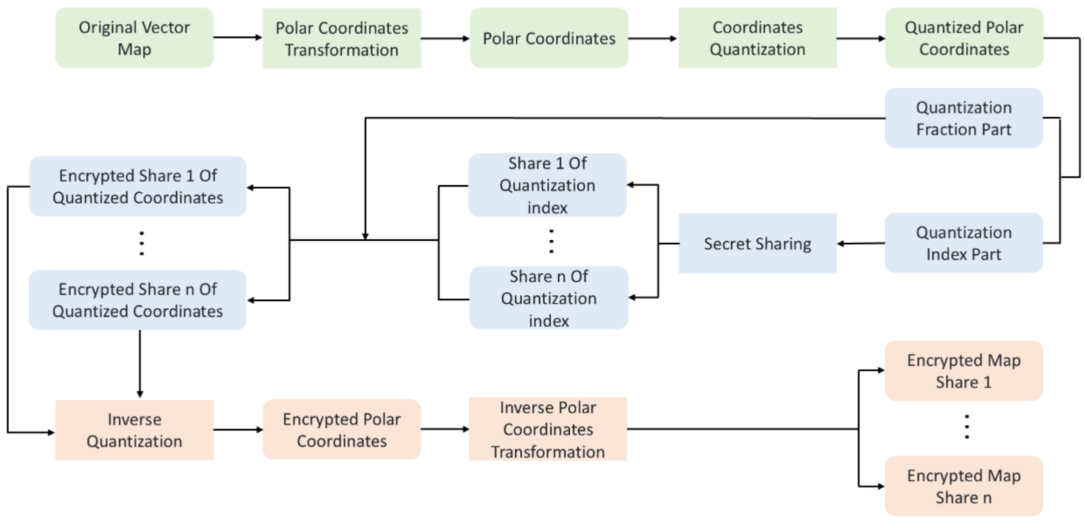

Original Vector Map
Polar Coordinates Transformation
Polar Coordinates
Coordinates Quantization
Quantized Polar Coordinates
Quantization Fraction Part
Encrypted Share 1 Of Quantized Coordinates
Share 1 Of Quantization index
⋮
Secret Sharing
Quantization Index Part
Encrypted Share n Of Quantized Coordinates
Share n Of Quantization index
Encrypted Map Share 1
Inverse Quantization
Encrypted Polar Coordinates
Inverse Polar Coordinates Transformation
Encrypted Map Share n

Figure 1. Encryption process of vector maps.

The decryption process consists of four sub-processes as follows. (1) For each encrypted map share, polar coordinate transformation is performed to all the encrypted coordinates, producing encrypted polar coordinates. (2) Quantization processes are performed to all the encrypted polar coordinates using the corresponding quantization steps to produce quantized cipher polar coordinates. And each quantized cipher polar coordinate is divided into quantization index part and fraction part. (3) For each index part, collect at least k corresponding shares to reconstruct the original index part. The original index part is then combined with the corresponding fraction part to form the original quantized polar coordinate. (4) The original quantized polar coordinates are performed inverse quantization operations and inverse polar coordinate transformations sequentially to obtain the original vector map data. The flowchart of the decryption process is shown in Figure 2.

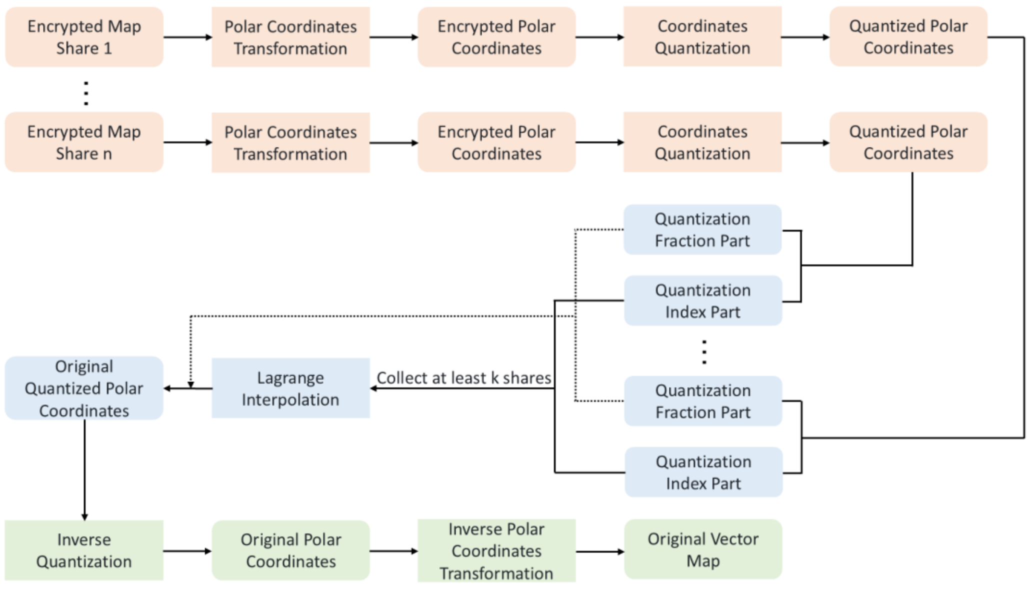

Encrypted Map Share 1
Polar Coordinates Transformation
Encrypted Polar Coordinates
Coordinates Quantization
Quantized Polar Coordinates
Encrypted Map Share n
Polar Coordinates Transformation
Encrypted Polar Coordinates
Coordinates Quantization
Quantized Polar Coordinates
Quantization Fraction Part
Quantization Index Part
Quantization Fraction Part
Quantization Index Part
Original Quantized Polar Coordinates
Lagrange Interpolation
Collect at least k shares
Inverse Quantization
Original Polar Coordinates
Inverse Polar Coordinates Transformation
Original Vector Map

Figure 2. Encryption process of vector maps.

3.1. *Construction of Polar Coordinate System*

RST transformations do not affect data usage and are common operations for vector map data. To ensure that the decryption of cipher map is robust to the RST operations, map encryption and decryption are performed in a constructed polar coordinate system. The construction of the polar coordinate system and the transformation from Cartesian coordinates to polar coordinates are described as follows.

For a given 2D vector map, its vertex sequence$\{V_i(x_i, y_i) | i = 1,2,\cdots,N\}$ is first obtained, where N is the number of vertices. Choose two vertices V_{p_1} and V_{p_2} in the sequence according to a predefined key key_1 with V_{p_1} as the pole and the ray $V_{p_1}V_{p_2}$ as the initial ray to construct a polar coordinate system. Then, the vertex coordinates $V_i(x_i, y_i)(i = 1,2,\cdots,N)$ are transformed into coordinates under the constructed polar system to obtain the polar coordinates $V_i(r_i, \theta_i)$. The transformation from Cartesian coordinates to polar coordinates can be represented by

$$r_i = \left\| V_{p_1} V_i \right\| \tag{3}$$

$$\begin{cases} \theta_i = sign(\overrightarrow{V_{p_1}V_{p_2}} \times \overrightarrow{V_{p_1}V_i} \bullet \overrightarrow{e_z}) \times \arccos\left(\dfrac{\|V_{p_1}V_i\|^2 + \|V_{p_1}V_{p_2}\|^2 - \|V_{p_2}V_i\|^2}{2\|V_{p_1}V_i\| \times \|V_{p_1}V_{p_2}\|}\right) & if\ \left\|\overrightarrow{V_{p_1}V_{p_2}} \times \overrightarrow{V_{p_1}V_i}\right\| \neq 0 \\ \theta_i = \arccos\left(\dfrac{\left\|V_{p_1}V_i\right\|^2 + \left\|V_{p_1}V_{p_2}\right\|^2 - \left\|V_{p_2}V_i\right\|^2}{2\left\|V_{p_1}V_i\right\| \times \left\|V_{p_1}V_{p_2}\right\|}\right) & if\ \left\|\overrightarrow{V_{p_1}V_{p_2}} \times \overrightarrow{V_{p_1}V_i}\right\| = 0 \end{cases} \tag{4}$$

where $\overrightarrow{e_z}$ is the unit vector of Z-axis. And the inverse polar coordinate transformation can be represented by

$$x_i = r_i\, cos(\theta_i) + x_{p_1} \tag{5}$$

$$y_i = r_i\, sin(\theta_i) + y_{p_1} \tag{6}$$

3.2. *Quantization Process to Polar Coordinates*

There are two necessities for performing quantization process to all the polar coordinates. (1) Although the polar radii keep invariant under the map rotation and translation operations, their values will be changed under map scaling operation. Consequently, if encryption process is directly performed to the polar coordinates, the decryption of cipher map will fail after map scaling transformation. Thus, to ensure that the decryption of cipher map is robust to map scaling operation, before encryption and decryption processes, each polar radius is quantized by using a quantization step which is defined as a fixed proportion of the length between the two chosen vertices V_{p_1} and V_{p_2}. The quantized polar radii possess RST-invariant capability. (2) The polar coordinates are real numbers, while the secret sharing model is defined on integers in finite field. Thus, the polar coordinates are transformed into integers on a finite field before secret sharing operations. The polar coordinates' transformation from real numbers to integer numbers is achieved by quantization process.

For a given vertex $V_i(r_i, \theta_i)$, the quantization formula is as follows:

$$\begin{cases} r_i^q = {}^{r_i}/_{q_r} \\ \theta_i^q = {}^{\theta_i}/_{q_\theta} \end{cases} \tag{7}$$

where q_r is the quantization step for polar radii and q_θ is the quantization step for polar angles. q_r is defined as a fixed proportion of the length between vertices V_{p_1} and V_{p_2} and computed as formula (8).

$$q_r = \rho\sqrt{\left(x_{p_1} - x_{p_2}\right)^2 + \left(y_{p_1} - y_{p_2}\right)^2} \tag{8}$$

where ρ is an adaptation factor for the quantization step, and the value of q_r can be tuned by tuning the value of ρ.

The inverse quantization process is represented by

$$\begin{cases} r_i = q_r \times r_i^q \\ \theta_i = q_\theta \times \theta_i^q \end{cases} \tag{9}$$

$V_i^q\left(r_i^q, \theta_i^q\right)$ represents the quantized polar coordinates. A quantization index number is defined as the integer part of the quantized polar coordinate and a quantization fraction number is defined as the decimal part of the quantized polar coordinate. For polar radius, the quantization index number is denoted as $r_i^{q\_int}$, and the quantization fraction number is denoted as $r_i^{q\_dec}$. For polar angle, the quantization index number is denoted as $\theta_i^{q\_int}$, and the quantization fraction number is denoted as $\theta_i^{q\_dec}$. It is obvious that $r_i^q = r_i^{q\_int} + r_i^{q\_dec}$ and $\theta_i^q = \theta_i^{q\_int} + \theta_i^{q\_dec}$. The quantization index number set $\left\{\left(r_i^{q\_int}, \theta_i^{q\_int}\right) | i \neq p_1 \ and \ i \neq p_2\right\}$ constitutes the encryption domain. Secret sharing operations will be performed on the encryption domain.

3.3. Map Encryption Based on Secret Sharing

In the original Shamir's secret sharing scheme [31], the constant item s can be replaced by elements in the encryption domain to produce the shares of the quantization index numbers, and the coefficients $a_1, \cdots, a_{k-1}$are all randomly selected integers. The use of random numbers in the sharing process can promote the security level. However, the scheme cannot achieve diffusion properties. One bit change in secret s can only change the shares for itself and has no impact to the shares of other elements in the encryption domain, which may lead to the failure of resisting differential cryptanalysis, chosen-plaintext attack, and so on. Therefore, the cipher-feedback mode of AES is cooperated into the original Shamir's secret sharing scheme by using a random share of the previous sharing result as the last coefficient of the polynomial in the current sharing operation [32].

Then, the proposed (k, n)-threshold secret sharing scheme can be constructed as follows. (1) Generate n different integers $\{x_1, x_2, \cdots, x_n\}$ from a secret key denoted as key_2; (2) for the i-th element in the encryption domain, two $k-1$ degree polynomials are constructed by

$$\begin{cases} f_i^r(x) = (r_i^{q\_int} + \sum_{c=1}^{k-2} a_c x^c + f_{i-1}^r\left(x_{t_r}\right) x^{k-1}) \bmod p_r \\ f_i^\theta(x) = (\theta_i^{q\_int} + \sum_{c=1}^{k-2} b_c x^c + f_{i-1}^\theta\left(x_{t_\theta}\right) x^{k-1}) \bmod p_\theta \end{cases} \tag{10}$$

where $a_1, \cdots, a_{k-2}$ and $b_1, \cdots, b_{k-2}$ are all randomly selected integers. $f_{i-1}^r\left(x_{t_r}\right)$ and $f_{i-1}^\theta\left(x_{t_\theta}\right)$ are randomly selected previous sharing results, in which t_r and t_θ are random in each sharing and randomly selected from $\{1, 2, \cdots, n\}$. For the first element $i = 1$, $f_0^r\left(x_{t_r}\right)$ and $f_0^\theta\left(x_{t_\theta}\right)$ can be any random integers satisfying $f_0^r\left(x_{t_r}\right) < p_r$ and $f_0^\theta\left(x_{t_\theta}\right) < p_\theta$. When $x \in \{x_1, x_2, \cdots, x_n\}$, n shares for polar radius and polar angle can be generated as $\{f_i^r(x_1), f_i^r(x_2), \cdots f_i^r(x_n)\}$ and $\{f_i^\theta(x_1), f_i^\theta(x_2), \cdots f_i^\theta(x_n)\}$, respectively. Then, $V_i^{q\_int\_j}\left(r_i^{q\_int\_j} = f_i^r(x_j), \theta_i^{q\_int\_j} = f_i^\theta(x_j)\right)$ represents the i-th encrypted quantization index number of share j.

In the above secret sharing scheme, a change of one element only influences the shares of itself and the following elements, and has no impact to the shares of preceding elements. In order to make the change of one element diffuse across the shares of all elements in the encryption domain, the above secret sharing operations can be performed in two rounds, and the sharing result of the second round is used as the final encryption result. In the second round, it is noted that, for the first element, the last coefficient of the secret sharing polynomial is randomly selected from the n shares of the last element produced in the first round.

After secret sharing operations on encryption domain, the encrypted quantization index numbers represented as $V_i^{q\_int\_j}\left(r_i^{q\_int\_j}, \theta_i^{q\_int\_j}\right)(i = 1,2,\cdots,N,\ i \neq p_1, i \neq p_2$ and $j = 1,2,\cdots,n)$ are combined with the quantization fraction numbers to form n shares of the quantized polar coordinates. For the i-th vertex in share j, its encrypted quantized polar coordinates denoted as $V_i^{q\_j}(r_i^{q\_j}, \theta_i^{q\_j})$ are computed as formula (11).

$$\begin{cases} r_i^{\mathrm{q\_j}} = r_i^{\mathrm{q\_int\_j}} + r_i^{\mathrm{q\_dec}} \\ \theta_i^{\mathrm{q\_j}} = \theta_i^{\mathrm{q\_int\_j}} + \theta_i^{\mathrm{q\_dec}} \end{cases} \tag{11}$$

Finally, sequential operations of inverse quantization and inverse polar coordinate transformation are performed to all encrypted quantized polar coordinates $V_i^{q\_j}(r_i^{q\_j}, \theta_i^{q\_j})(i = 1,2,\cdots,N, i \neq p_1, i \neq p_2$ and $j = 1,2,\cdots,n)$ to obtain the final n shares denoted as $\left\{\left(j, share_j\right) | j \in \{1,2,\cdots,n\}\right\}$ of encryption results for the original vector map. $KEY = (key_1, key_2)$ is saved for decryption process.

3.4. Decryption Process

The flowchart for decryption process is shown as Figure 2. When any k shares with their identities $\left(t_s, share_{t_s}\right), s \in \{1,2,\cdots,k\}, t_s \in \{1,2,\cdots,n\}$ have been collected, polar coordinate transformation and quantization process are first performed to all the coordinates of each share sequentially, obtaining the encrypted quantized polar coordinates $V_i^{q\_t_s}(r_i^{q\_t_s}, \theta_i^{q\_t_s})(i = 1,2,\cdots,N, i \neq p_1, i \neq p_2, s = 1,2,\cdots,k \text{ and } t_s \in \{1,2,\cdots,n\})$. Then, the encrypted quantization index numbers $r_i^{q\_int\_t_s}(s = 1,2,\cdots,k)$ for polar radius can be obtained as the integer part of $r_i^{q\_t_s}$ and the quantization fraction numbers $r_i^{q\_dec}$ for polar radius can be obtained as the decimal part of $r_i^{q\_t_s}$. Similarly, the encrypted quantization index numbers $\theta_i^{q\_int\_t_s}(s = 1,2,\cdots,k)$ for polar angle and the quantization fraction numbers $\theta_i^{q\_dec}$ for polar angle can be obtained as the integer part and decimal part of $\theta_i^{q\_t_s}$, respectively. It is easy to see that $r_i^{q\_int\_t_s} = f_i^r\left(x_{t_s}\right)$ and $\theta_i^{q\_int\_t_s} = f_i^\theta\left(x_{t_s}\right)$. Finally, one can compute x_{t_s} based on key_2, and reconstruct the polynomials f_i^r and f_i^θ using the Lagrange interpolation as formula (2), recovering the original quantization index numbers $r_i^{q\_int}$ and $\theta_i^{q\_int}$ as $f_i^r(0)$ and $f_i^\theta(0)$, respectively.

The decrypted original quantization index numbers $\left(r_i^{q\_int}, \theta_i^{q\_int}\right)$ are combined with the quantization fraction numbers $\left(r_i^{q\_dec}, \theta_i^{q\_dec}\right)$ to form the original quantized polar coordinates $V_i^q(r_i^q, \theta_i^q)$ by using $r_i^q = r_i^{q\_int} + r_i^{q\_dec}$ and $\theta_i^q = \theta_i^{q\_int} + \theta_i^{q\_dec}$. Subsequently, inverse quantization process and inverse polar coordinate transformation are performed to all the decrypted quantized polar coordinates $V_i^q(r_i^q, \theta_i^q)$ to obtain the final decrypted original vector map.

4. Experiments

4.1. Testing Environment and Datasets

The proposed algorithm is implemented using the Java (JDK8) language and the open-source library for geospatial data processing–GDAL. The experimental programs were run on 64-bit Windows 11 with AMD Ryzen™ 7 6800H CPU @ 3.20 GHz and 16G of memory.

Experiments on point, polyline and polygon feature maps were designed to validate the effectiveness and efficiency of the proposed method for vector map encryption. To better demonstrate the applicability of the proposed method, as shown in Figure 3, vector map datasets from OpenStreetMap (OSM) were used in the experiments, covering different volumes, feature types, and precisions. Specifically, Datasets A to C are the Points of Interest (POI), railways network and administrative division of prefecture-level cities in Zhejiang Province, China, respectively. Datasets D to F are the Points of Interest (POI), waterway network and water bodies of South Korea, respectively. The statistical information of these datasets is listed in Table 1.

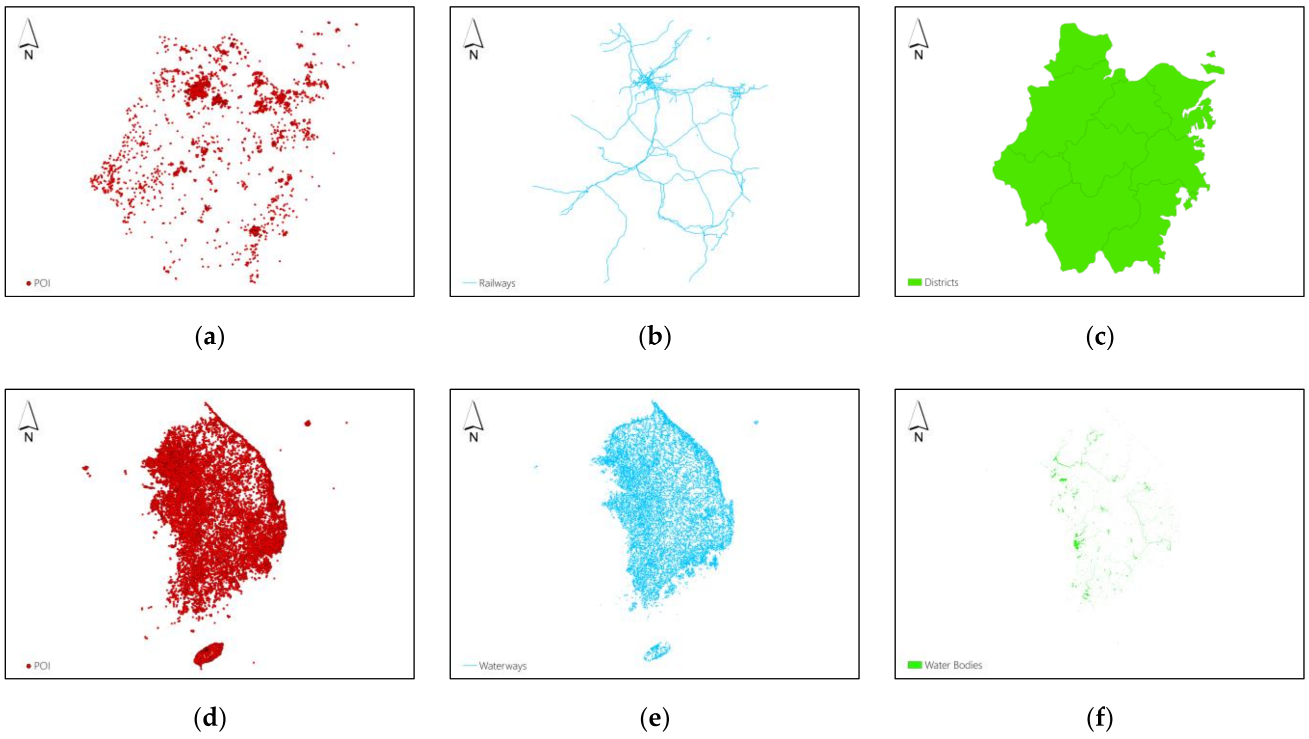


Figure 3. (**a**) Dataset A;(**b**) Dataset B;(**c**) Dataset C; (**d**) Dataset D; (**e**) Dataset E; (**f**) Dataset F.

Table 1. Detailed information of the vector map data.

Datasets	Layer	Feature type	Number of features	Number of vertices	Size
Dataset A	POI	point	15298	15298	419KB
Dataset B	Railway	polyline	12656	102165	2.29MB
Dataset C	Prefecture boundaries	polygon	12	1803	29KB
Dataset D	POI	point	246950	246950	6.75MB
Dataset E	Waterways	polyline	31870	777420	13.9MB
Dataset F	Water bodies	polygon	24036	1023278	17MB

4.2. Encryption and Decryption Results

The encryption and decryption operations on the vector map data are performed utilizing the proposed method. The threshold secret sharing schemes of (2,3), (3,5) and (5,9) are implemented in the experiments, and some results are shown in Figures 4-8. Figures 4(a-c) display the three shares of Dataset A obtained by using the proposed (2,3)-threshold secret sharing method, and Figures 4(d-f) display the decryption results obtained by collecting any two shares to reconstruct the plaintext data. Figures 5(a-c) display the three shares of Dataset B obtained by using the proposed (2,3)-threshold secret sharing method, and Figures 5(d-f) display the decryption results obtained by collecting any two shares to reconstruct the plaintext data. Figures 6(a-c) display the three shares of Dataset C obtained by using the proposed (2,3)-threshold secret sharing method, and Figures 6(d-f) display the decryption results obtained by collecting any two shares to reconstruct the plaintext data. Figures 7(a-i) display the nine shares of Dataset F obtained by using the proposed (5,9)-threshold secret sharing method. Figure 8(a) displays the decryption result of Dataset F by collecting

4 of 9 shares to reconstruct the plaintext data, and Figures 8(b-f) display the decryption results of Dataset F by collecting 5, 6, 7, 8 and 9 shares to reconstruct the plaintext data, respectively.

From the results, it can be found that the vertex distribution of the shares produced by the proposed (k,n)-threshold secret sharing scheme is chaotic from the visual effect, and it is difficult to get a bit of valuable information and patterns about the original maps, thus realizing the protection of the map information. The decryption results shown in Figures 4(d-f), 5(d-f) and 6(d-f) validate that collecting any k of n shares produced by the proposed (k,n)-threshold secret sharing can reconstruct the plaintext data correctly. It means that if no more than $n-k$ shares are destroyed, the plaintext data still can be reconstructed correctly by the remaining shares. This verifies the disaster tolerance property and promotes the security level of the proposed encryption scheme.

The decryption results shown in Figures 8(a-f) indicate that collecting less than k shares can not obtain any information of the original vector map, while collecting no less than k shares can reconstruct the plaintext data correctly, and this further proves the high security level of the proposed encryption scheme.

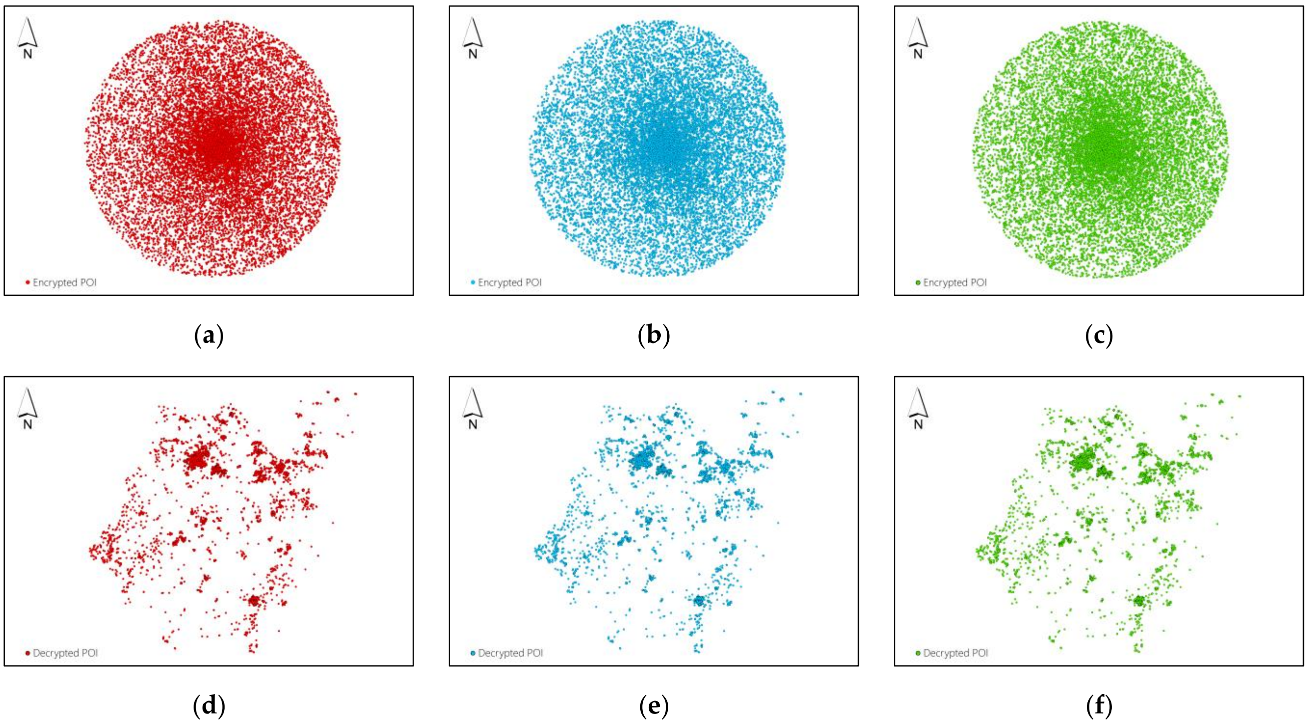
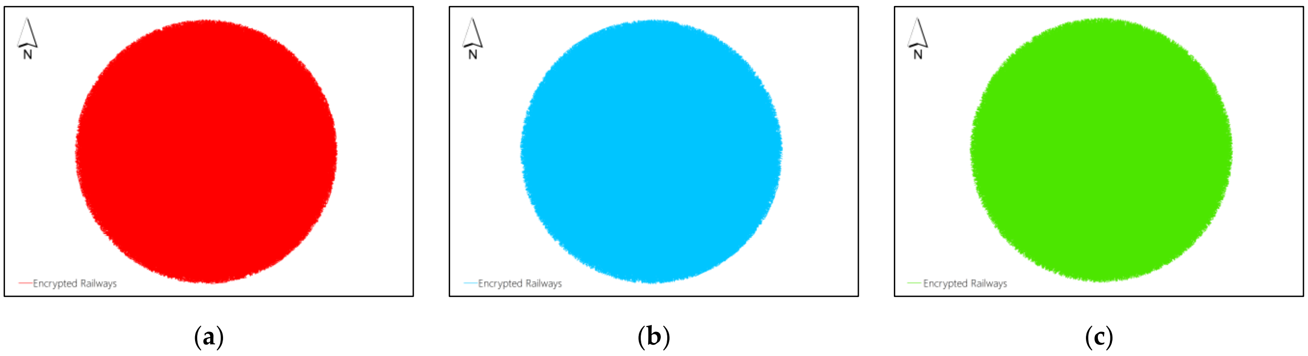
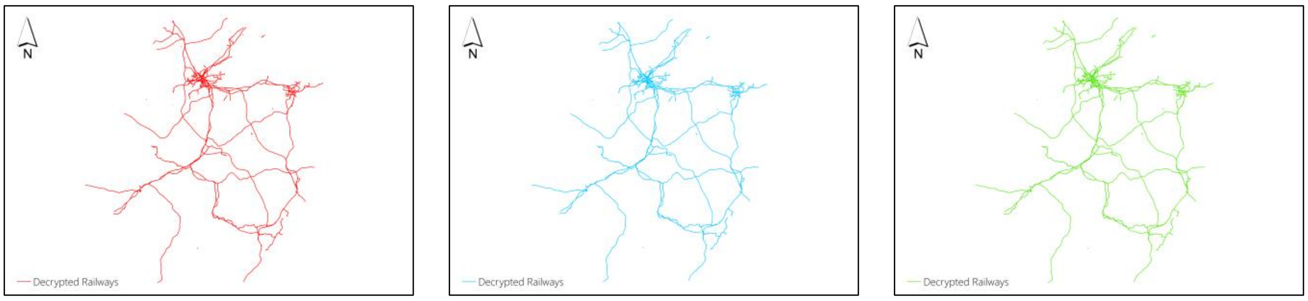


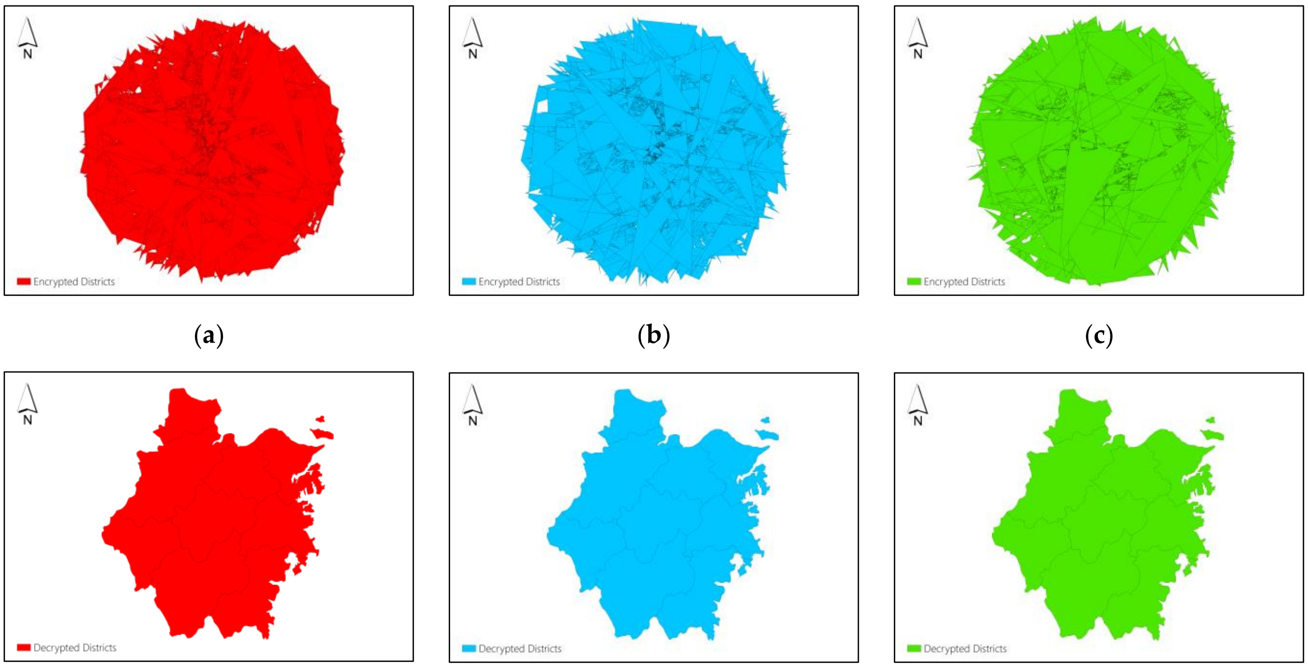
Figure 4. (**a**)-(**c**) The encrypted shares of Dataset A; (**d**) The decryption result with shares (**a**) and (**b**); (**e**) The decryption result with shares (**b**) and (**c**); (**f**) The decryption result with shares (**a**) and (**c**).





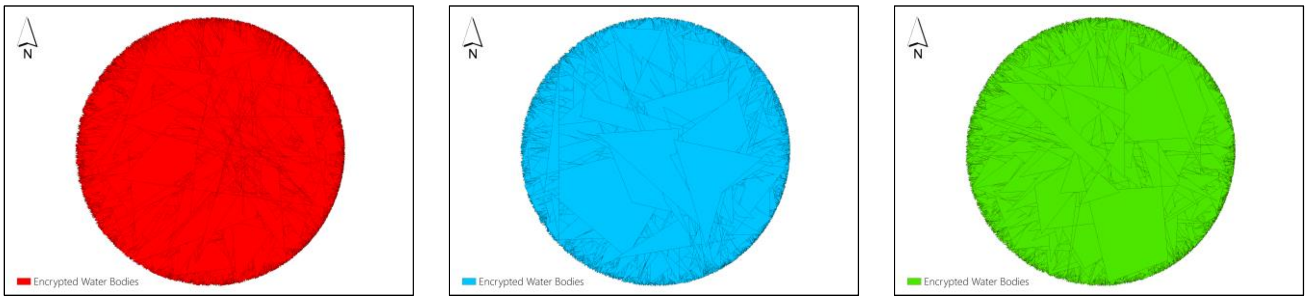
(**d**) (**e**) (**f**)

Figure 5. (**a**)-(**c**) The encrypted shares of Dataset B; (**d**) The decryption result with shares (**a**) and (**b**); (**e**) The decryption result with shares (**b**) and (**c**); (**f**) The decryption result with shares (**a**) and (**c**).



(**d**) (**e**) (**f**)

Figure 6. (**a**)-(**c**) The encrypted shares of Dataset C; (**d**) The decryption result with shares (**a**) and (**b**); (**e**) The decryption result with shares (**b**) and (**c**); (**f**) The decryption result with shares (**a**) and (**c**).



(**a**) (**b**) (**c**)

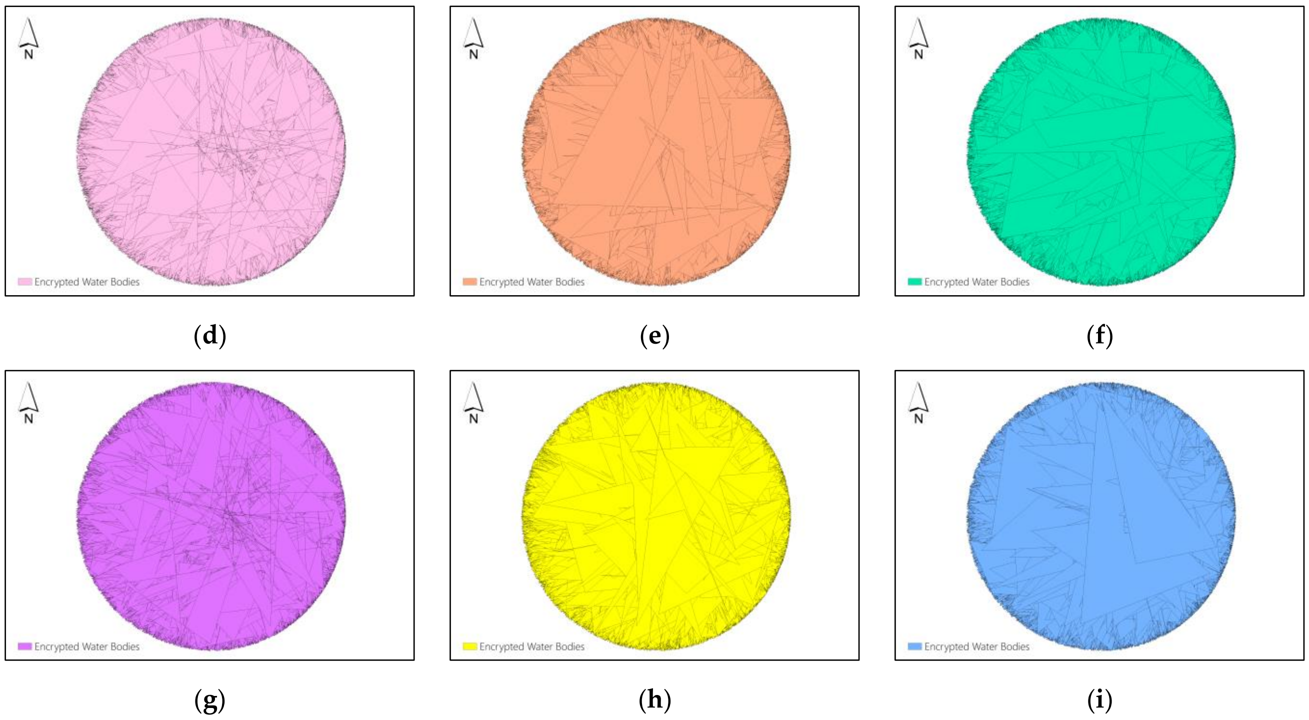


Figure 7. (**a**)-(**i**) The encrypted shares of Dataset F obtained by using the proposed (5,9)-threshold secret sharing method.

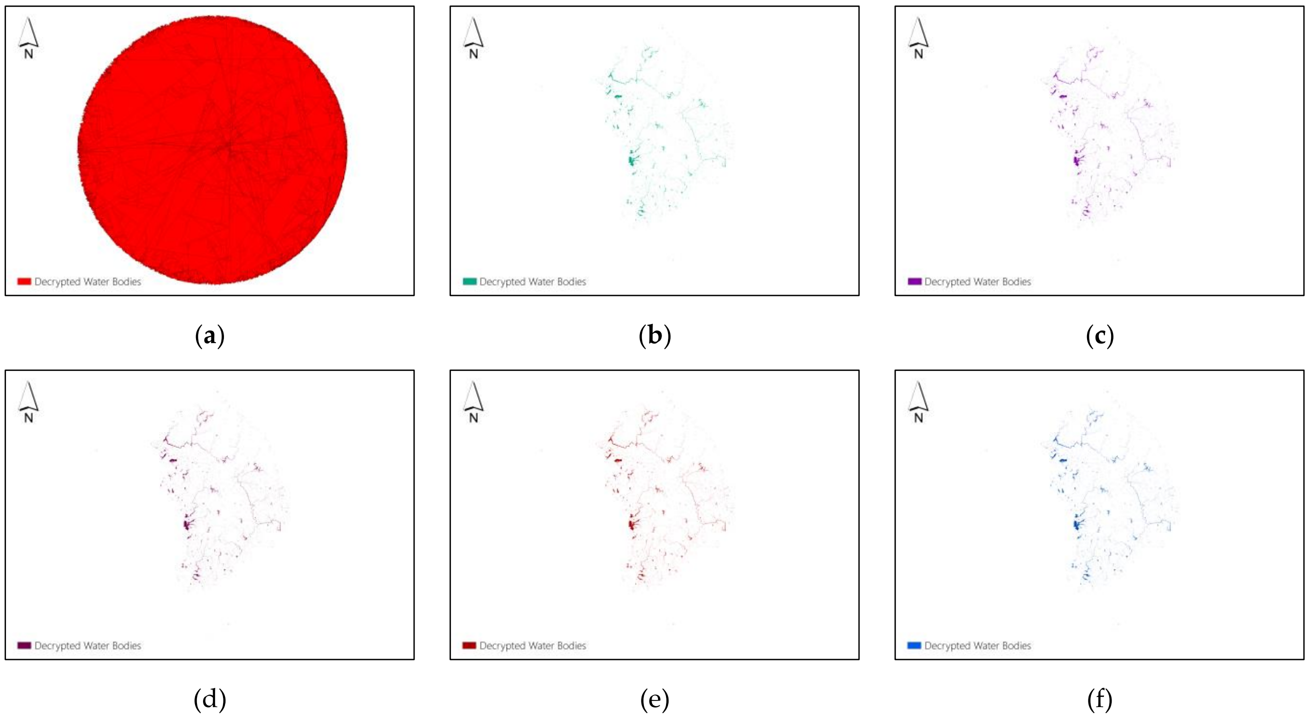


Figure 8. (**a**)-(**f**) The decryption results of Dataset F by collecting 4, 5, 6, 7, 8 and 9 shares, respectively.

4.3. Encryption Effects

In vector maps, the adjacent coordinates generally exhibit strong correlations. An effective encryption process should disrupt such correlations between coordinates and scramble all the vertices in the vector map in a random way. Thus, the RMSE (Root Mean Square Error) between the original map and the encrypted map will be high. In this section, the correlation of adjacent coordinates and the RMSE between the original and encrypted maps are adopted to measure the efficacy of the proposed encryption method.

4.3.1. Correlation of Adjacent Coordinates (CAC)

The correlation between adjacent coordinates in the vector map is calculated as:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{12}$$

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{13}$$

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y}) \tag{14}$$

$$r_{xy} = \frac{cov(x,y)}{\sigma_x \sigma_y} \tag{15}$$

Table 2 shows the results of the adjacency coordinate correlation in dataset B and C. It can be seen that the CAC of the original map is quite high, while that of the encrypted map shares are close to zero. The CAC of the decrypted map is the same as that of the original map, indicating that the decrypted map data is identical to the original map data. This demonstrates that the proposed encryption algorithm in this paper can effectively break the correlation within the original map data.

Table 2. The CACs of the original maps, the encrypted maps, and the decrypted maps.

Datasets		X-Coordinates	Y-Coordinates
Dataset B (railway)	original map	0.963269	0.985731
	share 1	0.006523	0.009533
	share 2	0.001108	0.002775
	share 3	0.001677	-0.004052
	decrypted map	0.963269	0.985731
Dataset C (prefecture boundaries)	original map	0.984837	0.988346
	share 1	-0.005180	0.007334
	share 2	0.001434	-0.002722
	share 3	0.004277	-0.001738
	decrypted map	0.984837	0.988346

4.3.2. RMSE Between the Original and Encrypted Maps

RMSE is used to describe how big the offset distance between the two corresponding vertices in the original map and encrypted map is. To eliminate the influence of dimensionality, normalized RMSE ($N\text{-}RMSE$) is proposed in this paper by normalizing the original RMSE using the maximum distance between two vertices in the original map. The calculation of $N\text{-}RMSE$ is represented by formula (16), where d_i is the offset distance between two corresponding vertices in the original and encrypted maps, N is the number of vertices, and $max\text{-}dis$ is the maximum distance between two vertices in the original map. Obviously, the larger the $N\text{-}RMSE$, the greater the difference between the original and encrypted maps. Table 3 shows the normalized minimum offset distance (N-Min offset distance), normalized maximum offset distance (N-Max offset distance) and $N\text{-}RMSE$ between original and encrypted maps for Dataset A, B and C. The results shown in Table 3 suggest that the encrypted maps are totally different from the original maps and the proposed encryption method has very good encryption effects.

$$N\text{-}RMSE = \frac{1}{max\text{-}dis}\sqrt{\frac{\sum_{i=1}^{i=N} d_i^2}{N}} \quad (16)$$

Table 3. Normalized offset distances between original and encrypted maps.

Datasets		N-Min offset Distance	N-Max offset distance	N-RMSE
	share 1	0.683244	0.968387	0.828142
Dataset A	share 2	0.679105	0.972651	0.838721
	share 3	0.681582	0.973746	0.827944
	share 1	0.756818	0.890549	0.822524
Dataset B	share 2	0.757976	0.895330	0.826558
	share 3	0.758600	0.899626	0.839536
	share 1	0.689633	0.962224	0.820770
Dataset C	share 2	0.690028	0.960687	0.821879
	share 3	0.690179	0.963291	0.823647

4.4. Encryption and Decryption Security

An encryption scheme which possesses probabilistic property encrypts the same plaintext data into two different ciphertexts in two encryption processes, thus, achieving semantic security. Furthermore, if an encryption scheme possesses diffusion property, a bit change in plaintext data will diffuse across the whole dataset in ciphertext version after encryption process, resulting in totally different encrypted data, which enable the encryption scheme to resist differential cryptanalysis, chosen-plaintext attack, and so on. Probabilistic property and diffusion property are very important properties for promoting the security of the encryption algorithms. Many existing vector map encryption schemes lack these properties, which greatly affects the security of the algorithms. Thus, the probabilistic and diffusion properties of the proposed encryption method are evaluated in this section. In addition, the robustness of the proposed decryption process to RST transformations is also evaluated in this section, since the RST transformations are very common data operations for vector maps.

4.4.1. Probabilistic Property

Probabilistic property ensures that the encryption process results in different ciphertexts when a plaintext is encrypted twice. In this section, plaintext maps are encrypted twice using the proposed encryption method. The normalized minimum offset distance (N-Min offset distance), normalized maximum offset distance (N-Max offset distance) and N-RMSE between two corresponding shares produced by encryption twice are computed and the results are listed in Table 4. Table 4 demonstrates that the offset distances between corresponding shares obtained by encrypting the same dataset twice are big, and the corresponding shares are completely different, which proves that the proposed vector map encryption method possesses probabilistic property.

Table 4. Normalized offset distances between corresponding shares obtained by encryption twice.

Datasets		N-Min offset distance	N-Max offset distance	N-RMSE
	share 1	0.117439	0.796279	0.633435
Dataset A	share 2	0.133791	0.784559	0.641268
	share 3	0.165460	0.795543	0.632560

Dataset B	share 1	0.156770	0.772710	0.623962
	share 2	0.123437	0.763173	0.636590
	share 3	0.107197	0.770605	0.628593
Dataset C	share 1	0.177662	0.786009	0.658562
	share 2	0.124753	0.796304	0.644872
	share 3	0.154923	0.806452	0.652654

4.4.2. Diffusion Property

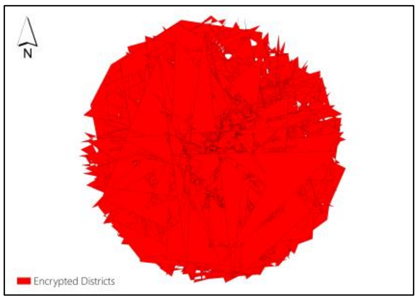
Diffusion property ensures that one bit change in a vertex coordinate will diffuse across all the vertex coordinates after encryption process, which promotes the encryption algorithm's ability of resisting differential cryptanalysis, chosen-plaintext attack, and so on. In this section, one bit of a vertex coordinate in a dataset is changed first. Then, the proposed encryption algorithm is performed to the dataset before and after change, respectively. Finally, the N-Min offset distance, N-Max offset distance and N-RMSE between two corresponding shares produced by encrypting the dataset before and after change are computed, and the results are listed in Table 5. Table 5 shows that the offset distances between corresponding shares are big, and that the corresponding shares obtained by encryption before and after one bit change are totally different, which verifies that the proposed vector map encryption method possesses diffusion property.

Table 5. Normalized offset distances between corresponding shares obtained by encryption before and after one bit change.

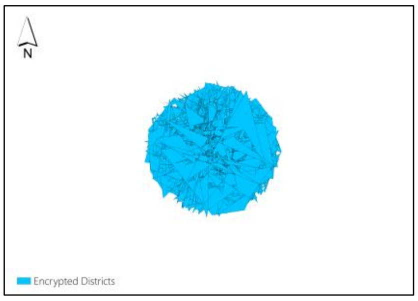
Datasets		N-Min offset distance	N-Max offset distance	N-RMSE
Dataset A	share 1	0.172385	0.842934	0.704336
	share 2	0.103453	0.855600	0.719485
	share 3	0.205342	0.869501	0.704567
Dataset B	share 1	0.153835	0.840998	0.719462
	share 2	0.159384	0.875234	0.712842
	share 3	0.129503	0.853555	0.723853
Dataset C	share 1	0.110485	0.893650	0.744845
	share 2	0.190289	0.856047	0.748534
	share 3	0.239604	0.867975	0.740185

4.4.3. Robustness of Decryption Process

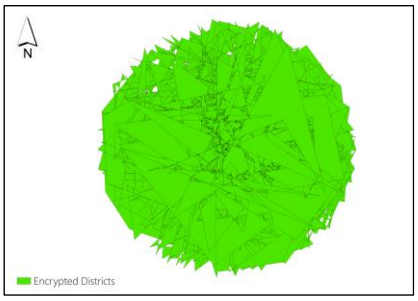
RST transformations are very common data operations for vector maps. The corresponding decryption algorithm for an encryption scheme should be robust to RST transformations so that the encrypted map can be correctly decrypted even after the RST transformations are performed to it. In this section, Dataset C is encrypted using the proposed (2,3)-threshold secret sharing scheme shown as Figures 6(a-c), and one of three shares is selected to perform RST transformations. Share 1 is selected for example. The transformed share 1 after rotation, scaling and translation operations is shown in Figures 9(a), 9(b) and 9(c), respectively. Then, the transformed share 1 and share 2 are selected to reconstruct the plaintext map. As the coordinate system of the transformed share 1 is different from the original coordinate system used by share 2, the reconstructed plaintext map can use either the transformed coordinate system of share 1 or the original coordinate system of share 2. The decrypted maps using the transformed coordinate system and the original coordinate system and the overlay of the two decrypted maps are shown in the sub figures of Figure 10, respectively. Figure 10 demonstrates that even the shares involved in decryption are transformed by RST operations, the plaintext map can still be correctly decrypted, and the decrypted map can use any coordinate system used by the shares involved in the decryption.



(**a**) share 1 after rotation

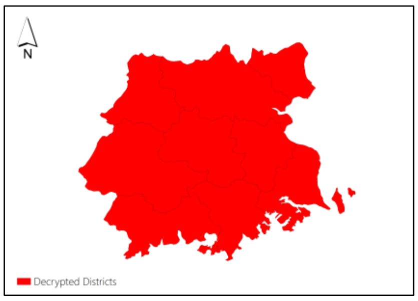


(**b**) share 1 after scaling

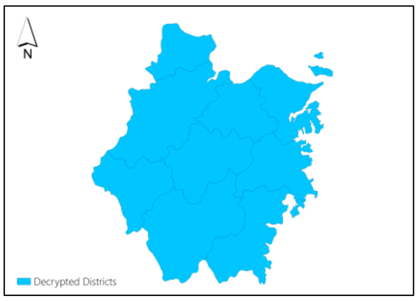


(**c**) share 1 after translation

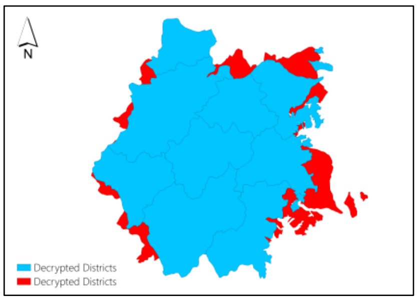
Figure 9. Transformed share 1 after RST operations.



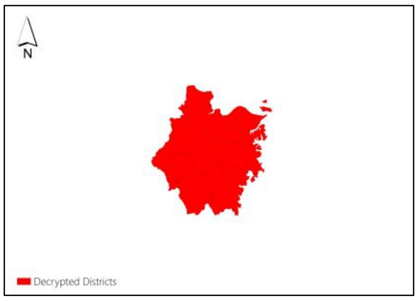
(**a**)



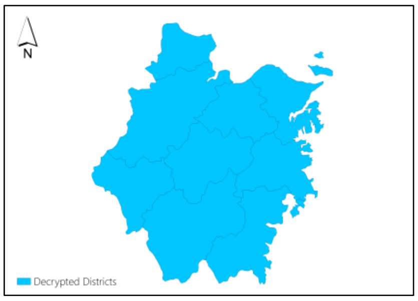
(**b**)



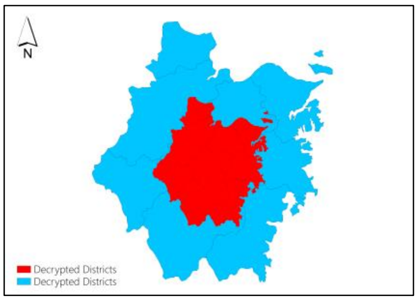
(**c**)



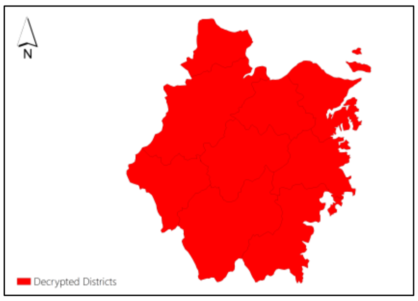
(**d**)



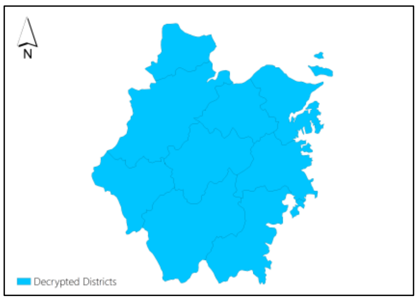
(**e**)



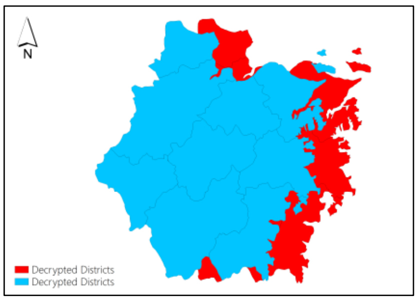
(**f**)



(**g**)



(**h**)



(**i**)

Figure 10. (**a**) decrypted map (with rotated share 1 and share 2) using coordinate system of rotated share 1; (**b**) decrypted map (with rotated share 1 and share 2) using coordinate system of share 2; (**c**) the overlay of (**a**) and (**b**); (**d**) decrypted map (with scaled share 1 and share 2) using coordinate system of scaled share 1; (**e**) decrypted map (with scaled share 1 and share 2) using coordinate system of share 2; (**f**) the overlay of (**d**) and (**e**); (**g**) decrypted map (with translated share 1 and share 2) using coordinate system of translated share 1; (**h**) decrypted map (with translated share 1 and share 2) using coordinate system of share 2; (**i**) the overlay of (**g**) and (**h**).

4.5. *Efficiency Analysis*

Computational efficiency is an important metric for evaluating encryption algorithms. Experiments were conducted using vector maps of three types (point, polyline, polygon) with different data volumes to demonstrate the efficiency of the proposed method. Through theoretical analysis of the algorithm, it can be seen that the parameters n, k used in the (k, n)-threshold secret sharing process and the data volumes are key factors that affect the efficiency of the proposed algorithm. The following experiments were conducted to demonstrate how these factors affect the encryption and decryption efficiency.

4.5.1. Encryption Efficiency

In this section, Datasets A, D, E and F of different data types with different data volumes are selected for experiments. Five encryption experiments are set to test the encryption time of the proposed method. The number n of the shares produced by encryption process is set to 1, 3, 5, 7 and 9 for each respective experiment. Encryption operations are performed to all the selected datasets in each experiment, and each encryption operation is performed 30 times, taking the average value as the result. The experimental results for the five experiments are displayed in Figure 11, which shows the increase in encryption time with the increase of data volume and the number of shares. It can be found in Figure 11 that compared to data volume, the number of shares has a greater impact on encryption time. When n increases, the encryption time increases significantly. However, when the n value is not too much big, it is found that the encryption time of the proposed method is comparable to that of the Local Encryption algorithm [24] and DRPP algorithm [25] which is not demonstrated here due to the limitation in paper length.

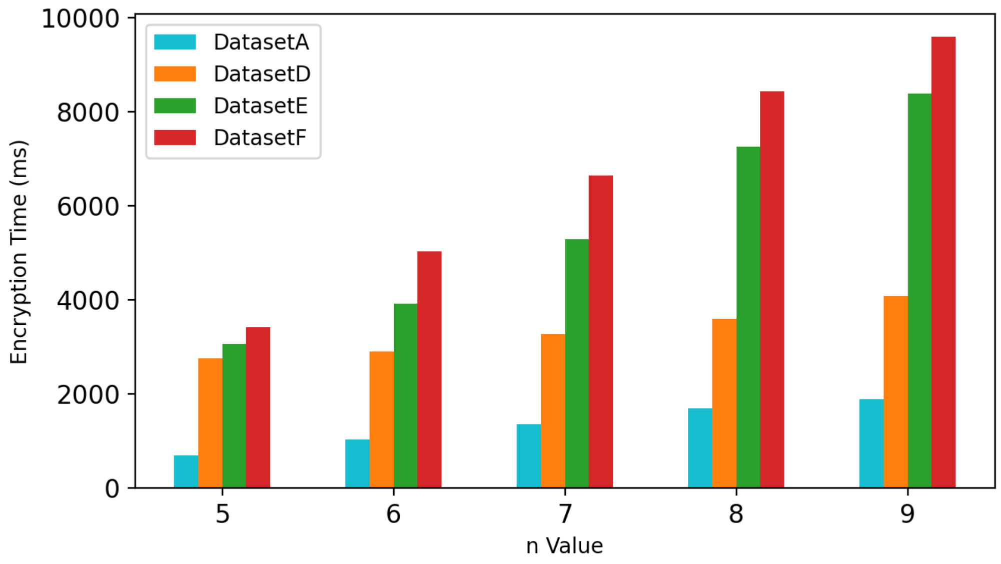


Figure 11. Running time of encryption for different datasets under different n values.

4.5.2. Decryption Efficiency

In this section, the impact of the k value used in the (k, n)-threshold secret sharing process and the data volume on the decryption time is evaluated. In the experiments, n is set to 9, and k is set to 5, 6, 7, 8 and 9, respectively. The selected datasets in section 4.5.1 are used in the decryption experiments. Each decryption operation is performed 30 times, taking the average value as the result. The decryption time for each selected dataset under different k values is displayed in Figure 12, which shows the increase in the decryption time with the increase of data volume and k value. It is noted in Figure 12 that compared to k value, data volume has a much greater impact on decryption time. This is because Lagrange interpolation for decrypting each vertex coordinate is time-consuming and as the number of vertices increases, time consumption becomes more significant. For this reason, the time efficiency of decryption operations is much lower than that of encryption operations for the proposed method.

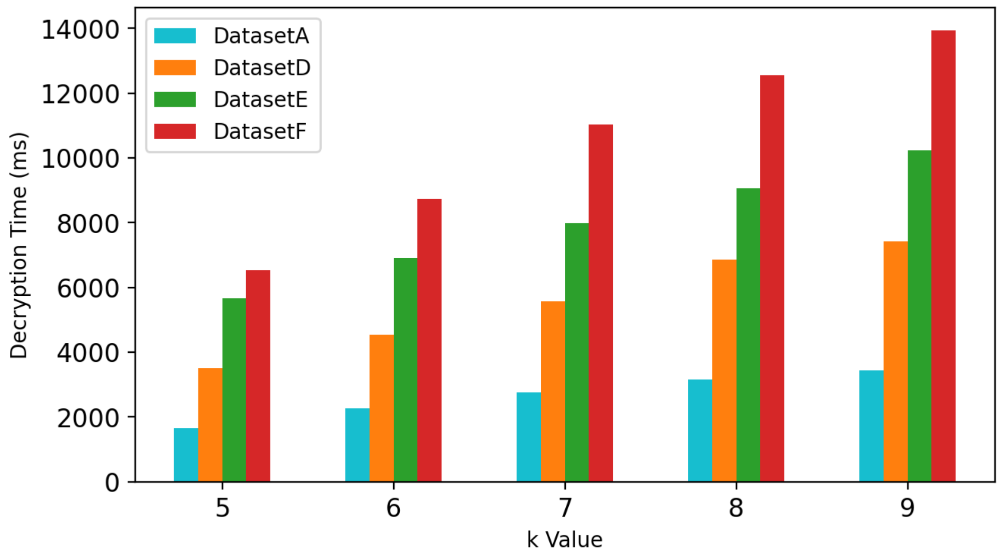


Figure 12. Running time of decryption for different datasets under different k values.

4.6. Comparison with Existing Methods

To evaluate the applicability and effectiveness of the proposed method, a comparison is made with other methods from five aspects, namely, whether the encryption method has disaster tolerance property, probabilistic property, diffusion property, the robustness to data RST transformations and whether it is applicable to points, polylines, and polygons. The comparison results are listed in Table 6. As shown in Table 6, (1) Compared with references [9,24,25,29,30], the proposed method has disaster tolerance property, which means that even some map shares are unavailable, the plaintext map can still be correctly decrypted; (2) Compared with references [9,24,25], the proposed method has probabilistic property. That is, different ciphertexts will be generated after encrypting the same plaintext twice, thus achieving semantic security; (3) Compared with references [29,30], the proposed method has diffusion property, which enhances the algorithm's ability of resisting differential cryptanalysis, chosen-plaintext attack, and so on; (4) Compared with references [24,29,30], the proposed method is robust to data RST transformations, which means the plaintext map can still be decrypted after RST transformations are performed to the ciphertext map; (5) Compared with references [25,29], the proposed method is applicable to different data types such as points , polylines and polygons. Overall, the proposed method is superior compared with the existing methods.

Table 6. Comparison with existing methods.

Methods	**Proposed method**	**Ref [9]**	**Ref [24]**	**Ref [25]**	**Ref [29]**	**Ref [30]**
Disaster tolerance	√	×	×	×	×	×
Probabilistic property	√	×	×	×	√	√
Diffusion property	√	√	√	√	×	×
The robustness to data RST transformations	√	√	×	√	×	×
Applicable to points, polylines and polygons	√	√	√	●	●	√

Note: "√" indicates that the method has this property, "×" indicates that the method does not have this property, and "●" indicates that the method does not fully support this property.

5. Conclusions

This research proposes a novel vector map encryption method based on (k, n)-threshold secret sharing, which encrypts one vector map into n map shares, and only by collecting at least k shares can the plaintext map be reconstructed. The encryption process consists of polar coordinate transformation and quantization, secret sharing process to produce n shares, and inverse quantization and inverse polar coordinate transformation. The corresponding decryption process consists of polar coordinate transformation and quantization, plaintext reconstruction by collecting at least k shares, and inverse quantization and inverse polar coordinate transformation. To validate the effectiveness of the proposed method, encryption results of different datasets using different secret sharing schemes are given, and encryption effects, security and efficiency were analyzed on vector map data of different types with different data volumes. Compared with available representative methods, the proposed encryption method has many very good properties, such as disaster tolerance property, probabilistic property, diffusion property and the robustness to data RST transformations, which greatly enhance the security and practicality of the method.

The proposed vector map encryption method encrypts a vector map into n shares, and the n shares can be managed by n respective data administrators. Decryption of data can only be carried out with the consent of at least k data administrators, making it particularly suitable for protecting the security of sensitive vector map data. To our knowledge, our work is the first to achieve vector map encryption using (k, n)-threshold secret sharing.

However, the proposed method also has some limitations. The encryption method encrypts one vector map into n map shares, expanding data volume by n times and bringing challenges to data storage. In addition, for the proposed method, as n increases, the encryption efficiency rapidly decreases. Moreover, due to that the Lagrange interpolation for decrypting each vertex coordinate is time-consuming, the decryption efficiency is not very high, too. Avenues for future investigation include exploring more efficient ways to achieve (k, n)-threshold secret sharing to improve the efficiency of the method.

Author Contributions: Conceptualization, B. Wu; methodology, B. Wu and F. Liu; software, F. Liu; validation, X. Liu, Z. Bu and H. Zhang; formal analysis, B. Wu and F. Liu; investigation, B. Wu and F. Liu; resources, X. Liu and B. Wu; data curation, F. Liu; writing—original draft preparation, F. Liu; writing—review and editing, B. Wu; visualization, F. Liu; supervision, B. Wu; project administration, B. Wu and X. Liu; funding acquisition, B. Wu and X. Liu. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Open Project of Hunan Engineering Research Center of Geographic information security and application, grant number HNGISA2024004 and Research Foundation of the Department of Natural Resources of Hunan Province, grant number 20230126XX.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors gratefully acknowledge the comments from the editor and the reviewers.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Ngoc-Giao, P.; Lee, S.-H.; Kwon, K.-R. Perceptual Encryption Based on Features of Interpolating Curve for Vector Map. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2017**, E100A, 1156–1164. https://doi.org/10.1587/transfun.E100.A.1156
2. Jang, B.-J.; Lee, S.-H.; Kwon, K.-R. Perceptual Encryption with Compression for Secure Vector Map Data Processing. *Digit. Signal Prog.* **2014**, 25, 224–243. https://doi.org/10.1016/j.dsp.2013.09.013
3. Duarte, L.; Teodoro, A.C.; Goncalves, J.A.; Ribeiro, J.; Flores, D.; Lopez-Gil, A.; Dominguez-Lopez, A.; Angulo-Vinuesa, X.; Martin-Lopez, S.; Gonzalez-Herraez, M. Distributed Temperature Measurement in a Self-Burning Coal Waste Pile through a GIS Open Source Desktop Application. *ISPRS Int. J. Geo-Inf.* **2017**, 6, 87. https://doi.org/10.3390/ijgi6030087
4. Lee, S.-H.; Huo, X.-J.; Kwon, K.-R. Vector Watermarking Method for Digital Map Protection Using Arc Length Distribution. *IEICE Trans. Inf. Syst.* **2014**, E97D, 34–42. https://doi.org/10.1587/transinf.E97.D.34
5. Peng, Y.; Lan, H.; Yue, M.; Xue, Y. Multipurpose Watermarking for Vector Map Protection and Authentication. *Multimed. Tools Appl.* **2018**, *77*, 7239–7259. https://doi.org/10.1007/s11042-017-4631-z
6. Qiu, Y.; Duan, H. A Novel Multi-Stage Watermarking Scheme of Vector Maps. *Multimed. Tools Appl.* **2021**, *80*, 877–897. https://doi.org/10.1007/s11042-020-09776-8
7. Jiang, L.; Xu, Z.; Xu, Y. Commutative Encryption and Watermarking Based on Orthogonal Decomposition. *Multimed. Tools Appl.* **2014**, 70, 1617–1635. https://doi.org/10.1007/s11042-012-1181-2

8. Zhou, Q.; Xiong, Y.; Zhu, C.; Ren, N. Selective Encryption Algorithm for Vectoral Geographic Data under Feature Point Grouping Strategy. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* **2024**, X-4–2024, 507–514. https://doi.org/10.5194/isprs-annals-X-4-2024-507-2024
9. Jang, B.-J.; Lee, S.-H.; Lim, S.; Kwon, K.-R. Progressive Vector Compression for High-Accuracy Vector Map Data. *Int. J. Geogr. Inf. Sci.* **2014**, 28, 763–779. https://doi.org/10.1080/13658816.2013.872249
10. Abubahia, A.; Cocea, M. Advancements in GIS Map Copyright Protection Schemes - a Critical Review. *Multimed. Tools Appl.* **2017**, 76, 12205–12231. https://doi.org/10.1007/s11042-016-3441-z
11. Wang, Y.; Yang, C.; Ren, N.; Zhu, C.; Rui, T.; Wang, D. An Adaptive Watermark Detection Algorithm for Vector Geographic Data. *KSII Trans. Internet Inf. Syst.* **2020**, 14, 323–343. https://doi.org/10.3837/tiis.2020.01.018
12. Tang, Z.; Zhang, Y.; Huang, J.; He, H.; Ding, Y. A Novel Infringement Detection Method for GIS Vector Data. *ISPRS Int. Geo-Inf.* **2020**, 9, 12. https://doi.org/10.3390/ijgi9010012
13. Ren, N.; Zhao, Y.; Zhu, C.; Zhou, Q.; Xu, D. Copyright Protection Based on Zero Watermarking and Blockchain for Vector Maps. *ISPRS Int. J. Geo-Inf.* **2021**, 10, 294. https://doi.org/10.3390/ijgi10050294
14. Wang, Y.; Yang, C.; Ding, K. Multiple Watermarking Algorithms for Vector Geographic Data Based on Multiple Quantization Index Modulation. *Appl. Sci.-Basel.* **2023**, 13, 12390. https://doi.org/10.3390/app132212390
15. Li, Y.; Zhang, L.; Wang, X.; Zhang, X.; Zhang, Q. A Novel Invariant Based Commutative Encryption and Watermarking Algorithm for Vector Maps. *ISPRS Int. J. Geo-Inf.* **2021**, 10, 718. https://doi.org/10.3390/ijgi10110718
16. Ren, N.; Tong, D.; Cui, H.; Zhu, C.; Zhou, Q. Congruence and Geometric Feature-Based Commutative Encryption-Watermarking Method for Vector Maps. *Comput. Geosci.* **2022**, 159. https://doi.org/10.1016/j.cageo.2021.105009
17. Dakroury, Y.; El-Ghafar, I.A.; Tammam, A. Protecting GIS data using cryptography and digital watermarking. *Int. J. Comput. Sci. Netw. Secur.* **2010**, 10 (1), 75–84. https://doi.org/10.1109/ICCCE.2012.6271256
18. Ren, N.; Zhao, M.; Zhu, C.; Sun, X.; Zhao, Y. Commutative Encryption and Watermarking Based on SVD for Secure GIS Vector Data. *Earth Sci. Inform.* **2021**, 14, 2249–2263. https://doi.org/10.1007/s12145-021-00684-5
19. Zhu, C.; Ren, N.; Xu, D. Geo-Information Security Technology: Progress and Prospects. *Acta Geodaetica et Cartographica Sinica.* **2022**, 51 (6), 1017-1028. https://doi.org/10.11947/j.AGCS.2022.20220172
20. Wright, M.A. The Advanced Encryption Standard. *Network Security.* **2001**, 2001, 11–13. https://doi.org/10.1016/S1353-4858(01)01018-2
21. Biryukov, A.; De Cannière, C. Data Encryption Standard (DES). In *Encyclopedia of Cryptography and Security*; van Tilborg, H.C.A., Ed.; Springer US: Boston, MA, 2005; pp. 129–135. https://doi.org/10.1007/978-1-4419-5906-5_568
22. Rivest, R.L.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM.* **1978**, *21*, 120–126. https://doi.org/10.1145/359340.359342
23. Koblitz, N. Elliptic Curve Cryptosystems. *Math Comput.* **1987**, *48*, 203–209. https://doi.org/10.2307/2007884
24. Ding, C.; Tang, J.; Deng, M.; Liu, H.; Mei, X. A Local Encryption Method for Large-Scale Vector Maps Based on Spatial Hierarchical Index and 4D Hyperchaotic System. *Int. J. Geogr. Inf. Sci.* **2024**, 38, 2272–2300. https://doi.org/10.1080/13658816.2024.2381225
25. Wang, X.; Yan, H.; Zhang, L. Vector Map Encryption Algorithm Based on Double Random Position Permutation Strategy. *ISPRS Int. J. Geo-Inf.* **2021**, 10, 311. https://doi.org/10.3390/ijgi10050311
26. Zhou, N.R.; Hua, T.X.; Gong, L.H.; Pei, D.J.; Liao, Q.H. Quantum Image Encryption Based on Generalized Arnold Transform and Double Random-Phase Encoding. *Quantum Inf. Process.* **2015**, 14, 1193–1213. https://doi.org/10.1007/s11128-015-0926-z
27. Cao, L.; Men, C.; Ji, R. Nonlinear Scrambling-Based Reversible Watermarking for 2D-Vector Maps. *Vis. Comput.* **2013**, *29*, 231–237. https://doi.org/10.1007/s00371-012-0732-x
28. Qu, C.; Du, J.; Xi, X.; Tian, H.; Zhang, J. A Hybrid Domain-Based Watermarking for Vector Maps Utilizing a Complementary Advantage of Discrete Fourier Transform and Singular Value Decomposition. *Comput. Geosci.* **2024**, 183, 105515. https://doi.org/10.1016/j.cageo.2023.105515
29. Pham, G.N.; Ngo, S.T.; Bui, A.N.; Tran, D.; Lee, S.-H.; Kwon, K.-R. Vector Map Random Encryption Algorithm Based on Multi-Scale Simplification and Gaussian Distribution. *Appl. Sci.-Basel.* **2019**, *9*, 4889. https://doi.org/10.3390/app9224889
30. Van, B.N.; Lee, S.-H.; Kwon, K.-R. Selective Encryption Algorithm Using Hybrid Transform for GIS Vector Map. *J. Inf. Process. Syst.* **2017**, *13*, 68–82. https://doi.org/10.3745/JIPS.03.0059
31. Shamir, A. How to Share a Secret. *Commun. ACM.* **1979**, *22*, 612–613. https://doi.org/10.1145/359168.359176
32. Hua, Z.; Wang, Y.; Yi, S.; Zhou, Y.; Jia, X. Reversible Data Hiding in Encrypted Images Using Cipher-Feedback Secret Sharing. *IEEE Trans. Circuits Syst. Video Technol.* **2022**, 32, 4968–4982. https://doi.org/10.1109/TCSVT.2022.3140974

doi:10.20944/preprints202501.1387.v1