International Journal of Geo-Information

MDPI

Article

Commutative Encryption and Reversible Watermarking Algorithm for Vector Maps Based on Virtual Coordinates

Qianyi Dai [1,2], Baiyan Wu [1,2,*], Fanshuo Liu [1,2], 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; driveghost155@gmail.com (Q.D.); 23012001005@mail.hnust.edu.cn (F.L.); 24012001014@mail.hnust.edu.cn (Z.B.); zhd@mail.hnust.edu.cn (H.Z.)

[2] School of Earth Sciences and Spatial Information Engineering, Hunan University of Science and Technology, Xiangtan 411201, China

* Correspondence: wby@hnust.edu.cn; Tel.: +86-18973260701

Abstract: The combination of encryption and digital watermarking technologies is an increasingly popular approach to achieve full lifecycle data protection. Recently, reversible data hiding in the encrypted domain (RDHED) has greatly aroused the interest of many scholars. However, the fixed order of first encryption and then watermarking makes these algorithms unsuitable for many applications. Commutative encryption and watermarking (CEW) technology realizes the flexible combination of encryption and watermarking, and suits more applications. However, most existing CEW schemes for vector maps are not reversible and are unsuitable for high-precision maps. To solve this problem, here, we propose a commutative encryption and reversible watermarking (CERW) algorithm for vector maps based on virtual coordinates that are uniformly distributed on the number axis. The CERW algorithm consists of a virtual interval step-based encryption scheme and a coordinate difference-based reversible watermarking scheme. In the encryption scheme, the map coordinates are moved randomly by multiples of virtual interval steps defined as the distance between two adjacent virtual coordinates. In the reversible watermarking scheme, the difference expansion (DE) technique is used to embed the watermark bit into the coordinate difference, computed based on the relative position of a map coordinate in a virtual interval. As the relative position of a map coordinate in a virtual interval remains unchanged during the coordinate scrambling encryption process, the watermarking and encryption operations do not interfere with each other, and commutativity between encryption and watermarking is achieved. The results show that the proposed method has high security, high capacity, and good invisibility. In addition, the algorithm applies not only to polyline and polygon vector data, but also to sparsely distributed point data, which traditional DE watermarking algorithms often fail to watermark.

Keywords: vector map; commutative encryption and reversible watermarking; virtual coordinates; difference expansion

Citation: Dai, Q.; Wu, B.; Liu, F.; Bu, Z.; Zhang, H. Commutative Encryption and Reversible Watermarking Algorithm for Vector Maps Based on Virtual Coordinates. *ISPRS Int. J. Geo-Inf.* **2024**, *13*, 338. https://doi.org/10.3390/ijgi13090338

Academic Editors: Wolfgang Kainz, Jun Feng, Changqing Luo and Mamoun Alazab

Received: 21 June 2024
Revised: 14 September 2024
Accepted: 20 September 2024
Published: 22 September 2024

1. Introduction

Vector maps are frequently used in many fields, such as navigation, land management, energy development, urban planning, and public safety, which leads to an important and unique status for vector maps [1]. Protecting the privacy and integrity of vector map data is of great significance in promoting the healthy development of the geographic information industry [2]. At present, encryption and digital watermarking technologies are the mainstream technical means to effectively protect the security of geographic information [3]. Encryption technology primarily addresses the issue of confidentiality in the process of information storage and transmission. Plaintext (the original readable information) is converted into ciphertext (the encrypted information) through specific algorithms to prevent unauthorized users from accessing and understanding the information. However, the

security of encryption technology relies heavily on the management of the key; if the key is lost or stolen, then the encrypted information will be completely exposed [4]. Different from encryption technology, digital watermarking technology mainly solves the problems of copyright protection and data traceability. Normally, a specific kind of information (e.g., copyright information, identification) is embedded into digital media (e.g., images, audio, video) in an imperceptible way [5]. In particular, although digital watermarking is designed to be invisible to human eyes or inaudible to human ears, in reality, the watermark embedding alters the original data, and this impact needs to be kept within boundaries [6]. For vector maps, combining encryption and digital watermarking technologies is imperative to achieve full lifecycle data protection [7].

Over the past decade, a number of algorithms for reversible data hiding in the encrypted domain (RDHED) have been proposed [8–12], in which the plaintext data are first encrypted. Then, additional information is embedded in the encrypted data. RDHED is suitable for scenarios where the data embedder is an unauthorized user and has no access to the original data. In RDHED, the original data can be recovered after additional data extraction, making RDHED very suitable for vector maps, which have high requirements for data accuracy. However, existing RDHED algorithms mainly focus on raster images, and few RDHED algorithms for vector maps have been proposed. Peng et al. [13] proposed an RDHED algorithm for 2D vector graphics based on a real-number reversible mapping model, which strikes a good balance between watermark invisibility, capacity, and security. However, watermark extraction can only be carried out in the ciphertext domain for this algorithm while, in practical applications, watermark extraction in both ciphertext and plaintext domains is often required. Subsequently, Peng et al. [14] proposed another RDHED algorithm for 2D vector graphics, which realizes watermark extraction in both ciphertext and plaintext domains and suits more application scenarios. Although not specially designed for vector maps, these algorithms have reference significance for encrypting and watermarking vector maps. Jang et al. [15] proposed a crypto-marking technique for vector maps, which watermarks the map data first and then encrypts the data using a progressive perceptual encryption method. The watermarking and the progressive perceptual encryption are independent. However, the fixed order of first marking and then encrypting in this method limits its application scenarios.

In the algorithms mentioned above, the order of encryption and watermarking is fixed, which is not flexible enough for practical applications. In order to realize flexible interchangeable operations between watermarking and encryption, some scholars have further proposed commutative encryption and watermarking (CEW) schemes. These schemes can be categorized into three types, according to the mechanism employed to achieve commutativity: separate domain-based CEW, homomorphic encryption-based CEW, and feature invariant-based CEW.

Separate domain-based CEW schemes achieve commutativity by performing encryption and watermarking operations in two different domains. Jiang et al. [16] proposed a CEW scheme based on orthogonal decomposition. The component coefficients of the orthogonal decomposition are independent, while the composite vector is sensitive to any changes in the component coefficients. The 2D image can be orthogonally decomposed into two domains based on this property for encryption and watermarking. This algorithm has no special requirements for encryption and watermarking and, so, it is highly universal. However, some plaintext information is exposed in these schemes, leading to security problems.

Homomorphic encryption-based CEW schemes achieve commutativity using homomorphisms. Lian [17] proposed a quasi-commutative watermarking scheme based on a homomorphic encryption method with an additive mechanism to encrypt the video as a whole, which can realize simple homomorphic watermarking operations but lacks robustness. For vector maps, Wu et al. [18] encrypted quantized integer coordinates based on Paillier homomorphic encryption and embedded the watermark using homomorphic opera-

tions in the ciphertext domain. Although homomorphic encryption algorithms can provide excellent data security, they suffer from computational complexity and low efficiency.

Feature invariant-based CEW schemes fulfill commutativity by embedding watermark information into a certain feature space, which remains unchanged during the encryption or decryption process. Compared with multimedia data such as images, vector maps have various spatial features, which make it more convenient to discover and construct some special invariants to realize commutativity between encryption and watermarking. For example, Ren et al. [19] constructed two feature invariants based on the sum of inner angles and the storage direction of two adjacent objects according to the inherent characteristics of the vector map and designed a CEW scheme based on these two feature invariants. However, the watermark capacity of this algorithm is only half of the number of objects in a vector map. Li et al. [20] proposed a new CEW algorithm for vector maps based on coordinate values, which serve as feature invariants. It is important to note that the above two algorithms only apply to polyline and polygon objects, and are unsuitable for point data. To achieve a higher watermarking capacity, Ren et al. [21] further proposed a vector map CEW method based on the congruence relationship and geometric feature, in which the angles and the distance ratios are selected as the geometric features, and the watermark is embedded into the residuals of the congruence operations performed on the geometric features. The watermark capacity of this algorithm achieves 2 bits per vertex. For improved robustness, Ren [22] proposed a CEW method for vector data based on singular value decomposition (SVD) in which the singular values are selected as the feature invariants.

In the literature, a few CEW schemes for vector maps have been proposed. However, most existing CEW algorithms focus on watermark robustness, while few focus on reversibility and suit the application scenarios that have high requirements for data accuracy. In this regard, Guo et al. [23] proposed a lossless CEW algorithm for vector data in which no data distortion is introduced after watermarking. However, the watermark capacity is small. Tan et al. [24] proposed a CEW algorithm based on zero watermarking and permutation encryption that can be applied to high-precision vector maps. However, the algorithm only applies to vector maps represented by polylines and polygons and cannot be applied to points.

Different from the schemes mentioned above, this study proposes a novel commutative encryption and reversible watermarking (CERW) method for vector maps in which commutativity between encryption and watermarking, as well as reversibility of watermarking, are achieved. This means that the original ciphertext map without a watermark can be recovered from the ciphertext map with a watermark, and the original plaintext map without a watermark can be recovered from the plaintext map with a watermark. The proposed CERW scheme is constructed based on virtual coordinates, which are uniformly distributed on the number axis. The commutativity between encryption and watermarking is achieved using feature invariance, which is defined as the relative position of a map coordinate in a virtual interval formed by two adjacent virtual coordinates. In the encryption part, the map coordinates are moved by random multiples of the virtual interval step. In the reversible watermarking part, the watermark bit is embedded into the difference computed based on the map coordinate's relative position in a virtual interval using the difference expansion (DE) technique. Through introducing virtual coordinates, the proposed CERW scheme can achieve a higher watermark capacity and better watermark invisibility than traditional DE watermarking algorithms. Furthermore, the proposed CERW scheme suits all kinds of vector maps, including point maps, such as POI maps.

The remainder of this paper is organized as follows. Section 2 introduces some preliminaries involved in this paper, including the pseudo-random number-generation method, difference expansion (DE) technique, and principle of the cosine-transform-based chaotic system (CTBCS). Section 3 describes the proposed algorithm in detail. Section 4 discusses the key parameters involved in the algorithm. Section 5 verifies the effectiveness and analyzes the performance of the proposed algorithm by implementing a series of experiments. Finally, Section 6 summarizes and overviews the whole work.

2. Preliminary

2.1. Pseudo-Random Number Generation Based on Hash Function and Streaming Cryptography Algorithm

In cryptography, pseudo-random numbers are usually utilized to increase the unpredictability and security of the encryption process [25]. The hash function is a method that converts an input (information) of arbitrary length to an output of fixed length. In watermarking algorithms, the hash function is usually chosen for data integrity checking, password storage, and random number generation [26]. The common functions are MD5 (Message Digest Algorithm 5), SHA-1 (Secure Hash Algorithm), and SHA-2. SHA-2 is subdivided into the SHA-256, SHA-384, and SHA-512 algorithms. Among them, SHA-512 runs and processes data faster and is more secure than SHA-256 and SHA-384.

The stream cipher is a cryptographic method that generates a stream of ciphertext by performing an exclusive-or operation between a plaintext stream and a key stream, and can be used to provide high-quality, high-entropy pseudo-random number sequences with good statistical properties and randomness. The Salsa 20 algorithm can realize fast encryption, which is a stream cipher algorithm with high security [27].

We chose SHA-512 and Salsa20 and designed an SHA-512-Salsa20 generator to generate secure and reliable pseudo-random numbers. The flowchart for the pseudo-random number generation is displayed in Figure 1 and the specific steps are as follows:

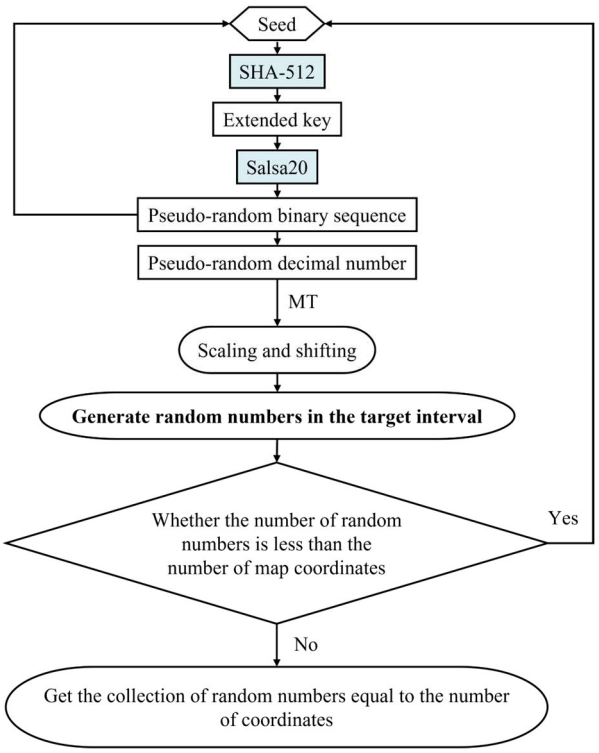


Figure 1. Flowchart of the pseudo-random number-generation process.

Step 1: A seed is selected as the initial value input, and then the seed is input into SHA-512 to obtain the output of SHA-512 as the extended key. This extended key is used as the key for the stream cipher.

Step 2: The Salsa20 algorithm is initialized using the extended key.

Step 3: The Salsa20 algorithm is used to generate a pseudo-random binary sequence and then convert the sequence to a decimal number.

Step 4: The pseudo-random binary sequence generated in the previous step is used as a new seed into SHA-512, and the output of SHA-512 is obtained as a new extended key. This step is used to increase randomness and enhance security.

Step 5: In conjunction with the Mersenne Twister (MT) algorithm, the pseudo-random number generated in Step 3 is scaled and shifted using Equation (1) in order to generate a random number within the target interval.

$$\text{r\_number} = \text{floor}[(\text{pr\_number} - \text{a})/(\text{b} - \text{a}) \times \text{MT(c, d)}] \quad (1)$$

A pseudo-random number will be generated from each run of the stream cipher algorithm. a denotes the minimum value of the generated pseudo-random numbers, and b denotes the maximum value of the generated pseudo-random numbers. Here, a, b, c, d are all integers; *pr_number* is the generated pseudo-random number in the interval [a, b]; $floor$ is the downward rounding function; MT is a function that generates random numbers based on the Mersenne Twister; and *r_number* is a pseudo-random number generated by Formula (1) that maps real values in the interval [a, b] to the interval [c, d] by scaling and shifting.

Step 6: Looping is performed through Step 1 to 5 until a sufficient number of random numbers is obtained.

2.2. Difference Expansion (DE) Technique

The difference expansion (DE) technique was designed for and applied earlier in raster image reversible watermarking, where pairs of pixel values are used to calculate the difference values, and the reversibility of the scheme is achieved by embedding the watermark bits via difference expansion [28]. Considering the low correlation and redundancy between vertices for vector data, Peng et al. [29,30] explored and improved the difference expansion technique based on Wang [31], and embedded the watermark into the ratio set of the relative coordinates of all vertices of a 2D CAD engineering drawing to improve the watermarking capacity and imperceptibility. However, the watermarking data integrity is destroyed if the watermark is extracted by referring to the vertices in reverse order. Subsequent algorithms based on histogram shifting, prediction error expansion, least significant bit substitution (LSB), and wavelet transform combined with difference expansion were devoted to improving the watermark capacity and visual quality.

DE is a reversible watermarking technique. The principle of watermark embedding is to calculate the difference for each pair of neighboring elements and provide the watermark embedding space by expanding the difference to twice the original value. The smaller the difference value, the less the watermark embedding disturbs the data. Therefore, limiting the difference value to a small range can realize very small graphic distortion after watermark embedding. The fundamentals of the traditional DE technique are described in detail below.

During the watermark-embedding stage, firstly, given a pair of neighboring elements of highly correlated carrier data $(x_1, x_2)(x_1 > x_2)$, an integer transformation is defined to compute their difference d and integer mean m according to Equation (2). Then, a watermark bit w is embedded into the difference d according to Equation (3). Finally, the carrier data with embedded watermark (x_1^w, x_2^w) can be obtained from Equation (4).

$$\begin{cases} \text{d} = x_1 - x_2 \\ \text{m} = \text{floor}\left(\frac{x_1 + x_2}{2}\right) \end{cases} \quad (2)$$

$$\text{d}' = \text{d} \times 2 + \text{w} \quad (3)$$

$$\begin{cases} x_1^w = \text{m} + \text{floor}\left(\frac{\text{d}'+1}{2}\right) \\ x_2^w = \text{m} - \text{floor}\left(\frac{\text{d}'}{2}\right) \end{cases} \quad (4)$$

For watermark extraction, firstly, the difference d' and the integer mean m of the watermarked elements need to be computed according to Equation (5). Then, the original

difference d is recovered and the watermark w is extracted by Equation (6). Finally, the original pair of elements (x_1, x_2) is constructed based on Equation (7).

$$\begin{cases} d' = x_1^w - x_2^w \\ m = \text{floor}\left(\frac{x_1^w + x_2^w}{2}\right) \end{cases} \tag{5}$$

$$\begin{cases} d = \text{floor}\left(\frac{d'}{2}\right) \\ w = d' - d \times 2 \end{cases} \tag{6}$$

$$\begin{cases} x_1 = m + \text{floor}\left(\frac{d+1}{2}\right) \\ x_2 = m - \text{floor}\left(\frac{d}{2}\right) \end{cases} \tag{7}$$

2.3. Cosine-Transform-Based Chaotic System (CTBCS)

Chaotic systems are widely used in the fields of information hiding and cryptography due to their unpredictability and initial value sensitivity. CTBCS [32], with two chaotic mappings as seed mappings, is a composite chaotic system with lower computational complexity compared to high-dimensionality chaos, and is more secure and reliable compared to low-dimensionality chaos. Therefore, CTBCS can be used for chaotic dislocation during watermark information generation to further enhance the security of the algorithm. Here, we use the existing Logistic and Tent mappings as the seed mapping, which can be mathematically defined as

$$\text{Logistic mapping} : x_{i+1} = L(h, x_i) = 4hx_i(1 - x_i) \tag{8}$$

$$\text{Tent mapping} : x_{i+1} = T(h, x_i) = \begin{cases} 2hx_i & \text{if } x_i < 0.5 \\ 2h(1 - x_i) & \text{if } x_i \geq 0.5 \end{cases} \tag{9}$$

Here, the variable h is the control parameter of the Logistic and Tent mappings, $h \in [0, 1]$.

CTBCS can be mathematically expressed as

$$x_{i+1} = \cos(\pi(F(a, x_i) + G(b, x_i) + \beta)) \tag{10}$$

where $F(a, x_i)$ and $G(b, x_i)$ are two known chaotic seed mappings, a and b are the parameters of the seed mappings, and β is the transformation constant (in this paper, we set $\beta = -0.5$). We set parameter a = h and parameter b = 1 − h. h is the key parameter of the generated chaotic mapping.

3. The Proposed Method

3.1. Basic Idea

In this study, we devise a new coordinate scrambling encryption scheme and reversible watermarking scheme for vector map data based on virtual coordinates in the proposed CERW scheme. In the encryption scheme, we design a pseudo-random number generator by combining SHA-512 and Salsa20 and realize coordinate scrambling based on a virtual interval step and pseudo-random numbers. In the watermarking scheme, we improve the traditional DE technique by introducing virtual coordinates and finally realize reversible embedding of watermarks and lossless recovery of original data using the DE technique. As the relative position of each coordinate in the virtual interval remains unchanged in the coordinate-scrambling process, both the difference computed based on the coordinate relative position and the watermark embedded in the difference are unaffected by coordinate scrambling, so the encryption operation and the watermarking operation do not interfere

with each other. The combination of the two can build the CERW scheme. The framework of the proposed scheme is shown in Figure 2.

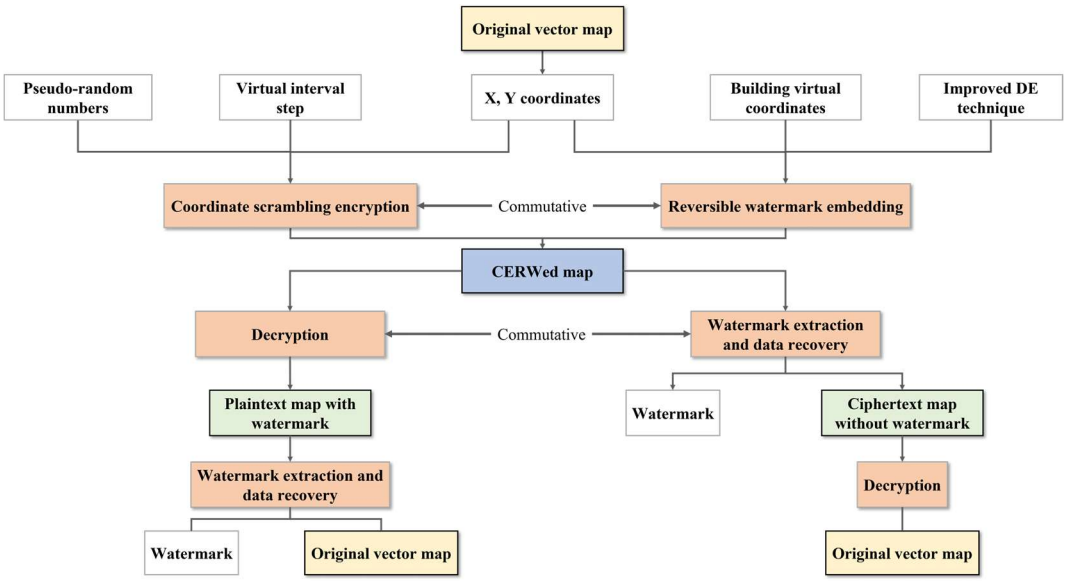


Figure 2. The framework of the proposed CERW scheme.

3.2. Map Coordinate Scrambling Encryption Scheme Based on Virtual Interval Step

In the map coordinate encryption scheme, we first design a pseudo-random number generator using the hash function and stream cipher ideas, as described in Section 2.1. Then, we use geometric scrambling for all x- and y-coordinates based on a certain integer virtual interval step and the pseudo-random numbers generated by the pseudo-random number generator, where the interval step and the seed of the pseudo-random number generator are used as the encryption key. The following subsections present the encryption method for the x-coordinates, and the encryption scheme for the y-coordinates is the same as that for the x-coordinates.

Scrambling encryption is an encryption technique that makes the original data difficult to decipher and use by changing the order or structure of the data. The underlying logic of vector map scrambling encryption is to destroy the data neighborhood correlation and spatial ordering [33]. Vector map data consist of spatial and attribute data that describe geographic entities in the form of geometric elements such as points, polylines, and polygons. The vector map scrambling encryption method is mainly used on the spatial coordinates.

Coordinate scrambling using the virtual interval step and pseudo-random numbers generated by a secure and efficient generation method is a feasible scrambling encryption method, which can achieve secure reversibility of the algorithm and smaller consumption (both in time and computation). This method moves the coordinates randomly by multiples of the virtual interval step. Take the x-coordinates as an example. First, virtual intervals are divided on the x-coordinate axis according to the virtual interval step l_t shown as Figure 3. Then, a certain pseudo-random number-generation method is applied to generate random numbers r_i. Finally, the coordinates are dislocated using Equation (11), where x_i is the original coordinate and x_i^* is the encrypted coordinate. The range of each coordinate move is a random multiple of the interval step. Equation (12) shows the process of coordinate recovery, which is the inverse operation of the disordered encryption mentioned above. The scrambling method for the y-coordinates is the same as for the x-coordinates.

$$x_i^* = x_i + r_i\, l_t \tag{11}$$

$$x_i = x_i^* - r_i\, l_t \tag{12}$$

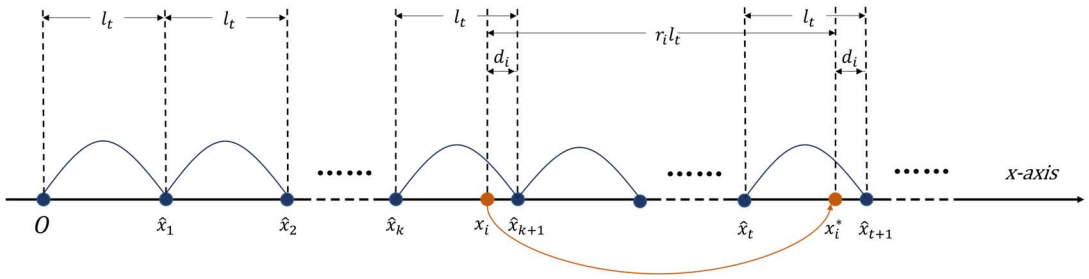


Figure 3. A schematic diagram of difference construction and coordinate scrambling based on virtual coordinates.

The preprocessing step defined by Equation (13) is performed on the original floating coordinate before encryption to obtain integer part x_i and decimal part f when implementing encryption operations to map coordinates.

$$\begin{cases} x_i = \text{floor}\left(x_i^o \times 10^P\right) \\ f = x_i^o \times 10^P - x_i \end{cases} \tag{13}$$

In Equation (13), x_i^o is the original floating coordinate. P is the number of digits that the decimal point shifts to the right. floor() is the downward rounding function.

The specific encryption operation is implemented on the coordinate's integer part x_i according to Equation (11). The final encrypted floating coordinate is formed by shifting the decimal point by P digits to the left after adding the decimal part f to the encrypted integer part x_i^*.

It is obvious that interval step l_t, the seed used for generating the random number r_i, and the value of P, are the essential keys of the encryption scheme in the above encryption process.

3.3. Reversible Watermarking Scheme Based on Improved Difference Expansion (IDE)

We utilize virtual coordinates to make improvements to the traditional DE technique. In our proposed IDE watermarking scheme, the watermark embedding domain consists of the difference values with virtual coordinates as references. Although both the watermarking scheme and the encryption scheme are based on coordinate modifications, the encryption scheme does not change the watermark-embedding domain, so the proposed watermarking scheme is commutative with the encryption scheme. The watermarking scheme includes watermark generation, watermark embedding, and watermark extraction and data recovery. In the watermark-generation stage, the original watermark sequence is scrambled before embedding using the CTBCS mapping mentioned in Section 2.3. In the watermark-embedding stage, the virtual coordinates are first generated based on the virtual interval step l_t, and then the differences are calculated based on the relative positions of map coordinates in virtual intervals. Then, the watermark is embedded in the difference using the DE technique to generate the watermark-containing map. In the watermark-extraction stage, the virtual coordinates consistent with those in the embedding stage are generated with the assistance of the watermarking key, and the differences are computed as in the watermark-embedding stage. The watermark bits are extracted from the differences according to the DE technique to verify the data source. Data recovery is based on the inverse process of difference expansion. The following is a detailed description of the watermarking scheme for the x-coordinates as an example, and the watermarking scheme for the y-coordinates is the same as that for the x-coordinates.

3.3.1. Differences Calculation Based on Virtual Coordinates

Vector maps consist of points, polylines, and polygons represented by coordinates. A prerequisite for using difference expansion is the high correlation between the neigh-

boring coordinates in order to obtain a small coordinate difference to ensure that the data distortions introduced by DE watermarking are insignificant. In practice, the neighboring coordinates of many kinds of maps are not highly correlated, leading to large coordinate differences and failed watermark embedding. Thus, we propose constructing virtual coordinates to overcome this problem. Taking the virtual coordinates as the reference points, the differences between the actual coordinates and the neighboring virtual coordinates are calculated. In this way, the differences can be controlled by adjusting the virtual interval step, based on which the virtual coordinates are constructed. The virtual coordinate construction and the coordinate difference calculation are shown in Figure 3.

Here, we use the x-coordinates as an example to illustrate the steps of difference calculation based on virtual coordinates.

Step 1: Construct virtual points $\hat{x}_i (i = 0, 1, ...)$ on the x-coordinate axis based on the virtual interval step l_t according to Equation (14). l_t is set to an odd integer.

$$\hat{x}_i = l_t \times i \tag{14}$$

Step 2: For any coordinate x_i, if x_i does not coincide with any virtual coordinate, then calculate the left adjacent virtual coordinate $\hat{x}_{left}$ and the right adjacent virtual coordinate $\hat{x}_{right}$, as shown in Equation (15).

$$\begin{cases} \hat{x}_{left} = l_t \times \left\lfloor \frac{x_i}{l_t} \right\rfloor \\ \hat{x}_{right} = l_t \times \left\lceil \frac{x_i}{l_t} \right\rceil \end{cases} \tag{15}$$

If x_i coincides with a certain virtual coordinate, then the left adjacent virtual coordinate $\hat{x}_{left}$ and the right adjacent virtual coordinate $\hat{x}_{right}$ are computed according to Equation (16).

$$\begin{cases} \hat{x}_{left} = l_t \times \left(\left\lfloor \frac{x_i}{l_t} \right\rfloor - 1 \right) \\ \hat{x}_{right} = l_t \times \left(\left\lceil \frac{x_i}{l_t} \right\rceil + 1 \right) \end{cases} \tag{16}$$

Step 3: Choose the virtual coordinate closest to x_i as the reference coordinate. Then, the difference between the actual coordinate x_i and the reference coordinate is calculated according to Equation (17). d_i is the direct carrier of the hidden watermark.

$$d_i = \begin{cases} 0 & \text{if } x_i - \hat{x}_{left} = \hat{x}_{right} - x_i \\ x_i - \hat{x}_{left} & \text{if } x_i - \hat{x}_{left} < \hat{x}_{right} - x_i \\ \hat{x}_{right} - x_i & \text{if } x_i - \hat{x}_{left} > \hat{x}_{right} - x_i \end{cases} \tag{17}$$

3.3.2. Reference Coordinate Flag Map

There are three cases for the reference coordinate of each actual coordinate x_i. Case 1: the reference coordinate is the left adjacent virtual coordinate of x_i; Case 2: the reference coordinate is the right adjacent virtual coordinate of x_i; Case 3: the reference coordinate is x_i itself if x_i coincides with a certain virtual coordinate. A reference coordinate flag map F is defined to record the reference coordinate of each actual coordinate.

$$F = \{f_i | f_i \in \{-1,0,1\}, i = 1,\ldots,n\} \tag{18}$$

$f_i = -1$ if the reference coordinate of x_i belongs to Case 1; $f_i = 1$ if the reference coordinate of x_i belongs to Case 2; $f_i = 0$ if the reference coordinate of x_i belongs to Case 3.

3.3.3. Watermark Generation

The original watermark information $W_o = \{b_1, b_2, \ldots\ldots, b_t\}$ is a binary sequence of length t. Thus, a one-dimensional chaotic sequence $W = \{w_1, w_2, \ldots\ldots, w_t\}$ is generated according to the CTBCS mapping mentioned in Section 2.3. This step is used to enhance the security of the watermark information. The key parameter for saving the CTBCS mapping is recorded as W_K_0, which can recover the original watermark information after watermark extraction. The chaotic sequence W is embedded into the map.

3.3.4. Watermark Embedding

Watermark embedding can be performed in both plaintext and ciphertext domains. The specific steps for watermark embedding are given as follows.

Step 1: Obtain all vertex coordinates of the map. For each coordinate, perform the preprocessing step defined in Equation (13) to obtain the integer coordinate x_i and the fractional part f. The subsequent watermark-embedding processes are based on the integer coordinate x_i. The fractional part f is not involved in watermark embedding.

Step 2: Based on the method described in Section 3.3.1, construct the virtual coordinates, determine the reference coordinate of x_i, and compute the difference d_i between the integer coordinate x_i and the reference coordinate.

Step 3: Assign the corresponding flag to f_i in F defined in Section 3.3.2 according to the determined reference coordinate of x_i.

Step 4: Embed one watermark bit into d_i using the DE technique described in Section 2.2 and obtain the watermark-containing difference d_i^w.

Step 5: The integer watermarked coordinate x_i^w is computed based on Equation (19).

$$x_i^w = \begin{cases} \hat{x}_{left} + d_i^w \text{ if } f_i = -1 \\ x_i + d_i^w \text{ if } f_i = 0 \\ \hat{x}_{right} - d_i^w \text{ if } f_i = 1 \end{cases} \quad (19)$$

If $f_i = 0$ and $d_i^w > 0$, reassign -1 to f_i after the bit insertion to amend the flag map.

Step 6: Add the original decimal part f to the watermarked integer part x_i^w and shift the decimal point by P digits to the left to form the final watermarked floating coordinate.

Step 7: Repeat Steps 1 to 6 until all the coordinates are watermarked.

The interval step l_t, the reference coordinate flag map F, and the value of P are kept as the watermarking key W_K_1 for subsequent watermark extraction and data recovery.

3.3.5. Watermark Extraction and Data Recovery

The watermark extraction and map decryption in the proposed CERW scheme are separable. Watermark extraction and data recovery can be performed in both ciphertext and plaintext domains. The watermark information is extracted directly from the watermarked difference computed based on the watermarked coordinate's relative position in a virtual interval. The original differences can be recovered based on the DE technique, and then the original coordinates can be recovered using the original differences. The steps for extracting the watermark from the watermarked coordinates are described in detail below.

Watermark extraction:

Step 1: Read W_K_1 to determine the virtual interval step l_t, the reference coordinate flag map F, and the value of P.

Step 2: Obtain all the coordinates of the watermarked map. For each coordinate, obtain the watermarked integer part x_i^w and the fractional part f like Step 1 in watermark embedding. Watermark extraction is performed on the integer coordinate x_i^w.

Step 3: Construct virtual coordinates based on virtual interval step l_t and determine the reference coordinate of x_i^w according to the reference coordinate flag map F. Calculate the watermarked difference d_i^w between the reference coordinate and x_i^w.

Step 4: Extract one watermark bit w from d_i^w according to Equation (20).

$$w = d_i^w \bmod 2 \tag{20}$$

Step 5: Repeat Step 2 to Step 4 until all the watermarked coordinates are used to extract the watermark bits.

Step 6: Arrange all the detected bits to form the detected chaotic sequence $W' = \{w'_1, w'_2 \ldots\ldots, w'_t\}$. Decrypt the chaotic sequence W' to obtain a sequence $W'_o = \{b'_1, b'_2, \ldots\ldots, b'_t\}$ using W_K_0. W'_o can be compared with the original watermark information W_o to verify map data consistency.

Data recovery:

Step 1: The original difference d_i can be recovered according to Equation (21).

$$d_i = \text{floor}\left(\frac{d_i^w}{2}\right) \tag{21}$$

Step 2: Then, the original integer coordinates x_i can be obtained based on Equation (22).

$$x_i = \begin{cases} \hat{x}_{left} + d_i & \text{if } f_i = -1 \\ x_i^w & \text{if } f_i = 0 \\ \hat{x}_{right} - d_i & \text{if } f_i = 1 \end{cases} \tag{22}$$

Step 3: Finally, the original floating coordinate x_i^o can be recovered according to Equation (23).

$$x_i^o = (x_i + f) \times 10^{-P} \tag{23}$$

Step 4: Repeat Steps 1 to 3 until all the original coordinates are recovered in a lossless manner.

4. Discussion of Some Parameters

4.1. Interval Step l_t

In the proposed scheme, the virtual interval step l_t is a decisive parameter for virtual coordinate construction and affects the magnitudes of data distortion introduced by DE watermarking. A larger l_t leads to bigger differences, resulting in more significant data distortions after watermarking. In order to ensure that the data distortions are within the data accuracy tolerance τ, l_t must not be too large.

To meet the data accuracy tolerance τ, the offset between the watermarked coordinate x_i^w and the original coordinate x_i should be not greater than $10^P\tau$; namely, $|x_i^w - x_i| \leq 10^P\tau$. In order to satisfy this inequality, according to the principle of DE watermarking, $|2d_i + w - d_i| \leq 10^P\tau$ can be set up. As $w \in \{0, 1\}$, it is enough to satisfy only $|d_i + 1| \leq 10^P\tau$. Furthermore, it is sufficient to satisfy $l_t/2+1 \leq 10^P\tau$ since $0 < d_i < l_t/2$. Thus, $l_t \leq 2\,(10^P\tau - 1)$ is obtained. Furthermore, l_t is an odd integer number with the minimum value of 1. This means that as long as the value of l_t is in the interval of $[1, 2(10^P\tau - 1)]$, all the coordinate differences d_i computed based on the virtual coordinates can be used to embed the watermark without affecting the usability of the watermarked data, and the data distortions introduced by watermarking are sure to be within the data accuracy tolerance τ.

4.2. Pseudo-Random Number r_i

There are two considerations for the pseudo-random number r_i. The first is that the range of the ciphertext map encrypted using r_i should be consistent with the range of the plaintext map for the sake of visual rationality. The second is that the differences between the encrypted coordinates and the original coordinates are big enough to destroy the data usability.

For the first consideration, $x_{min} \leq x_i^* \leq x_{max}$ can be set up; that is, $x_{min} \leq x_i + r_i l_t \leq x_{max}$. Furthermore, r_i needs to meet $\left\lceil \frac{x_{min}-x_i}{l_t} \right\rceil \leq r_i \leq \left\lfloor \frac{x_{max}-x_i}{l_t} \right\rfloor$. For the second consideration, the difference between x_i^* and x_i should be greater than the data accuracy tolerance τ; that is, $r_i\, l_t > 10^P\tau$ or $r_i\, l_t < -10^P\tau$. In other words, $r_i > \frac{10^P\tau}{l_t}$ or $r_i < -\frac{10^P\tau}{l_t}$ needs to be satisfied. To summarize, the pseudo-random number r_i generated by the SHA-512-Salsa20 generator should meet the two considerations for producing an effective ciphertext map.

5. Experiments and Results

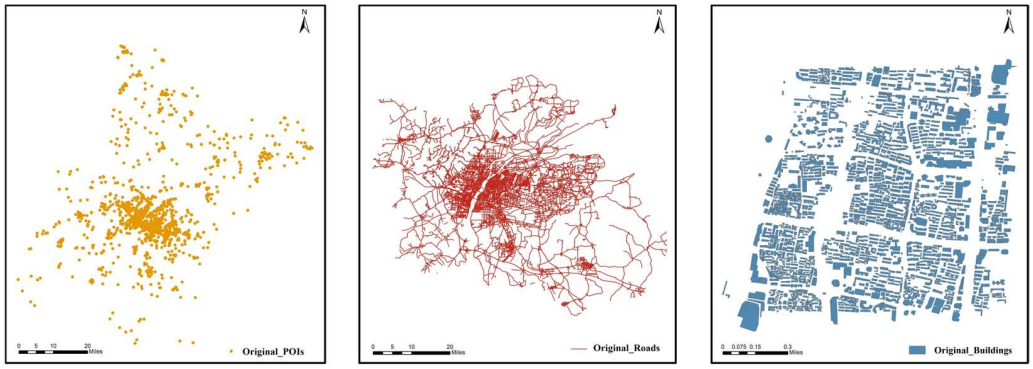
5.1. Experimental Data and Parameter Settings

The experiments were implemented in VS2013 on Windows 10 using Python language. Three shapefile datasets of different types were chosen as original vector maps, as listed in Table 1 and shown in Figure 4a–c. The original watermark image is a binary image of 64 × 64 pixels. The experimental parameters are set as follows: CTBCS mapping parameter h = 0.5, parameter P = 6, and virtual interval step $l_t = 1$. For the pseudo-random number generator, the seed is set to "666".

Table 1. Detailed information regarding the experimental data.

Maps	Data Types	Features	Number of Vertices	Scale	τ (m)
POIs	Point	2410	2410	1:10,000	1
Roads	Polyline	11,423	122,962	1:10,000	1
Buildings	Polygon	5660	20,325	1:1000	0.1

τ denotes the precision tolerance.



(a) (b) (c)

Figure 4. Original experimental data: (**a**) POIs, (**b**) roads, (**c**) buildings.

5.2. Resulting Map Visualization

The original vector maps are performed with commutative encryption and reversible watermarking (CERW) operations using the proposed algorithm. The resulting maps are shown in Figures 5 and 6. Figure 5 displays the CERWed maps. Figure 5a–c display the encrypted–watermarked (E-Wed) maps, which were obtained by performing the encryption operation first and reversible watermarking operation afterward on the original vector maps. Figure 5d–f display the watermarked–encrypted (W-Eed) maps obtained by performing the reversible watermarking operation first and then encryption operation on the original vector maps.

Figure 6 shows the recovered plaintext maps without watermarks. Figure 6a–c show the recovered–decrypted (R-Ded) maps obtained by performing watermark extraction and data recovery first and then decryption on the CERWed maps. Figure 6d–f show the decrypted–recovered (D-Red) maps obtained by performing decryption first and then watermark extraction and data recovery on the CERWed maps. As shown in Figure 5, there

is no clue regarding the original maps left in the CERWed maps. The recovered plaintext maps shown in Figure 6 are identical to the original maps visually.

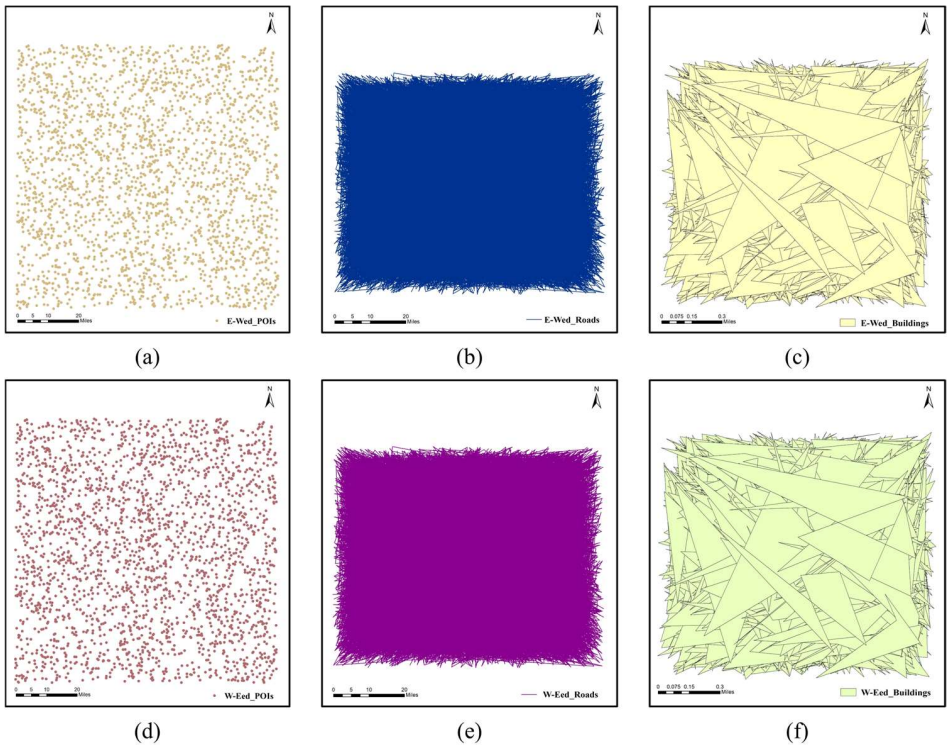


Figure 5. The visualization of CERWed maps: (**a**) E-Wed POIs; (**b**) E-Wed roads; (**c**) E-Wed buildings; (**d**) W-Eed POIs; (**e**) W-Eed roads; (**f**) W-Eed buildings.

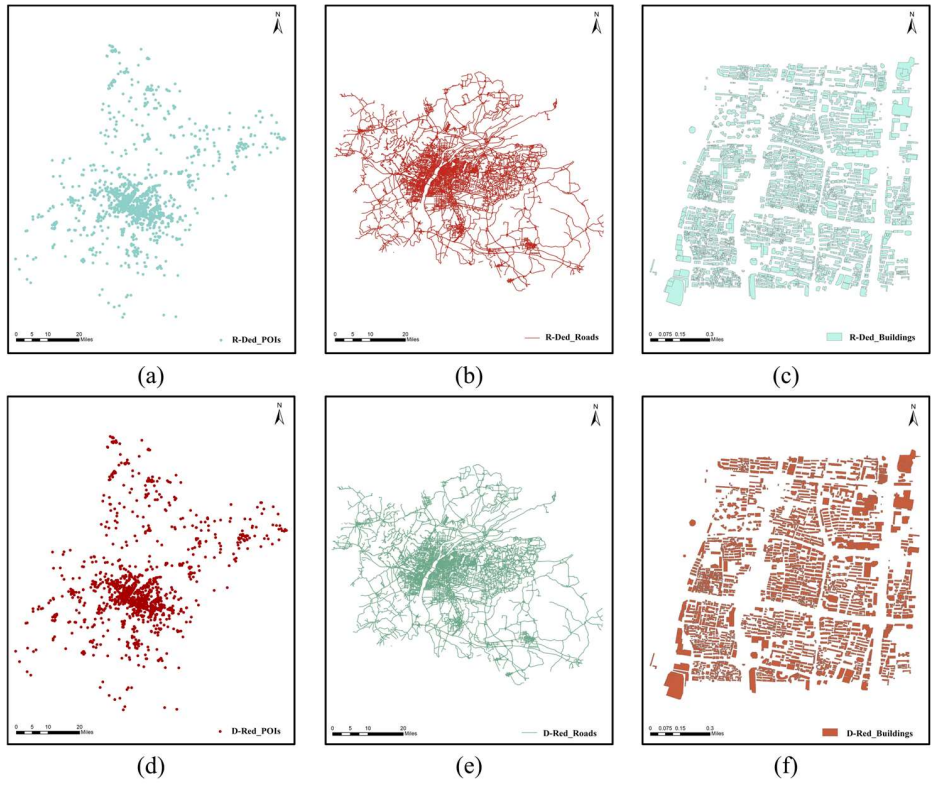


Figure 6. The visualization of the recovered plaintext maps: (**a**) R-Ded POIs; (**b**) R-Ded roads; (**c**) R-Ded buildings; (**d**) D-Red POIs; (**e**) D-Red roads; (**f**) D-Red buildings.

5.3. Commutativity

In order to verify that the encryption process and the reversible watermarking process in the proposed scheme are commutative, it is necessary to compare the E-Wed maps produced by the operation sequence of encryption and watermarking with the W-Eed maps produced by the operation sequence of watermarking and encryption. If the two kinds of maps are consistent, then the commutativity between the encryption process and the reversible watermarking process is proved.

Similarly, the commutativity between the watermark-extraction and data-recovery process and the decryption process in the proposed scheme can be verified by evaluating the consistency between the R-Ded maps produced by watermark extraction and data recovery first and decryption afterward, and the D-Red maps produced by decryption first and watermark extraction and data recovery afterward. Moreover, the watermarks extracted before and after decryption are compared. If the R-Ded maps and the D-Red maps are consistent, and the watermarks extracted before and after decryption are identical, then commutativity between the watermark-extraction and data-recovery process and the decryption process is proved.

In order to evaluate the difference between the E-Wed map and the W-Eed map and the difference between the R-Ded map and the D-Red map, the root mean square error (RMSE) was introduced, as shown in Equation (24). If the value of the RMSE is closer to 0, the two compared maps are more similar. Table 2 displays the results of a comparison between the E-Wed maps and the W-Eed maps. Table 2 shows that all the vertices in both maps are identical, and their RMSEs are 0, which suggests that the E-Wed maps are exactly the same as the W-Eed maps. Therefore, it is obvious that the encryption process and the reversible watermarking process in the proposed scheme are commutative.

$$\mathrm{RMSE} = \frac{1}{n} \parallel V - V' \parallel = \frac{1}{n} \sqrt{\sum_{i=1}^{n} [\, V_i - V'_i \,]^2} \tag{24}$$

Table 2. The results of a comparison between the E-Wed maps and the W-Eed maps.

Datasets	Number of Consistent Vertices	Number of Inconsistent Vertices	RMSE
E-Wed and W-Eed POIs	2410	0	0
E-Wed and W-Eed roads	122,962	0	0
E-Wed and W-Eed buildings	20,325	0	0

Table 3 displays the results of a comparison between the R-Ded maps and the D-Red maps. Table 3 also shows that the R-Ded maps and the D-Red maps are exactly the same. Moreover, the bit error ratio (BER) of the extracted watermark information is also introduced to measure the difference between the original watermark and the extracted watermark. The definition of BER is shown in Equation (25).

$$\mathrm{BER} = \frac{n_e}{L_w} \tag{25}$$

where n_e denotes the number of extracted wrong watermark bits, and L_w is the watermark length.

Table 3. The results of a comparison between the R-Ded maps and the D-Red maps.

Datasets	Number of Consistent Vertices	Number of Inconsistent Vertices	RMSE
R-Ded and D-Red POIs	2410	0	0
R-Ded and D-Red roads	122,962	0	0
R-Ded and D-Red buildings	20,325	0	0

The BERs of the watermarks extracted before and after decryption for the three test datasets were computed. All the BER values obtained were 0, which indicates that the watermarks extracted before decryption are exactly the same as the watermarks extracted after decryption. Table 3 and the obtained BER results jointly prove commutativity between the watermark-extraction and data-recovery process and the decryption process in the proposed scheme.

5.4. Encryption Performance Evaluation

5.4.1. Effectiveness of the Encryption Scheme

To verify the security of the proposed encryption scheme, we compared the experimental results before and after encryption.

A reliable scheme needs to demonstrate the unavailability of the data after encryption as well as the availability after decryption. The encryption effect is shown in Figure 7a–c, where the spatial relationships of the original maps are totally destroyed and the encrypted maps are completely unavailable, making it difficult to obtain any plaintext information directly from the ciphertext map. Figure 7d–f show that the decrypted maps are visually consistent with the original maps, indicating a good decryption effect.

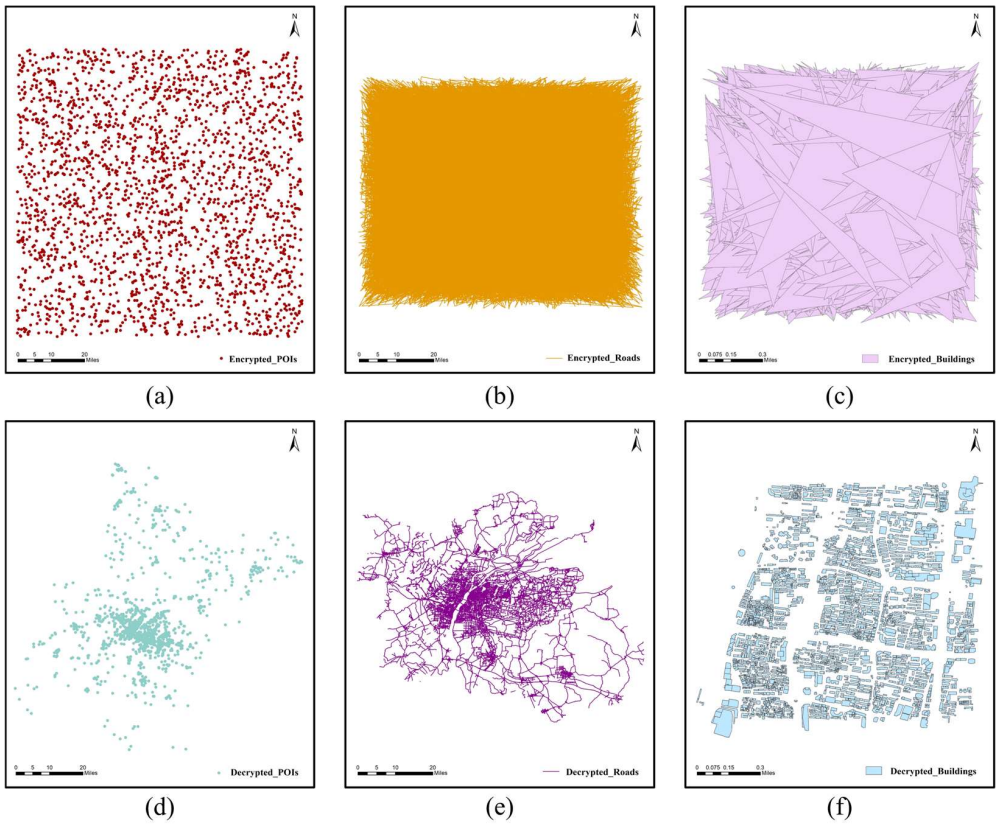


Figure 7. Visualization of the encryption and decryption results: (**a**) encrypted POIs; (**b**) encrypted roads; (**c**) Encrypted buildings; (**d**) Decrypted POIs; (**e**) Decrypted roads; (**f**) Decrypted buildings.

In addition to visual verification, RMSE is used as a quantitative index to illustrate the encryption and decryption effect. The encrypted maps and decrypted maps are compared with the original maps, and the RMSEs between them are calculated. The results are displayed in Table 4. Table 4 shows that, due to the coordinate geometrical scrambling encryption, the RMSEs between the encrypted maps and the original maps are very large, which suggests a pretty good encryption effect. The RMSEs between the decrypted maps and the original maps are 0, as shown in Table 4, quantitatively proving the effectiveness of

the proposed decryption algorithm. Therefore, the effectiveness of the proposed encryption scheme is proved visually and quantitatively.

Table 4. The RMSE values of the encrypted and decrypted maps.

Datasets	RMSE between Encrypted Map and Original Map (m)	RMSE between Decrypted Map and Original Map (m)
POIs	448,973.8688	0
Roads	48,368.7397	0
Buildings	15,216.4770	0

5.4.2. Key Space

The key space is the set of all possible values of a key used in cryptography to encrypt and decrypt data. The size of the key space affects the security of a cryptosystem. A larger key space usually implies greater security, and the size of the key space should be larger than the standard requirement (i.e., 2^{100}) [19]. The encryption key for the proposed algorithm, which is also the decryption key, includes the parameter P, the virtual interval step l_t, and the seed of the SHA-512-Salsa20 generator. The maximum value of the parameter P is set to 6 in this paper. So, there are seven possible values for P. The value range for l_t is $[1, 2(10^P \tau - 1)]$, suggesting that there are $2(10^P \tau - 1)$ possible values for l_t. For the seed of the SHA-512-Salsa20 generator, its length is 512 bits, indicating that there are 2^{512} possible values for it. Based on the possible values for P, l_t and the seed, the key space of the proposed scheme is given as KeySpace$= 7 \times 2(10^P \tau - 1) \times 2^{512} > 2^{512}$, which is sufficiently larger than the standard requirement. Therefore, the key space of the proposed scheme is large enough to resist the brute force attack and has good security.

5.4.3. Key Sensitivity

High sensitivity of a key is of great importance for encryption algorithms, which means that small adjustments can make a huge difference, thus making it impossible for an illegal user to decipher the unreadable data. To further illustrate the impact of key changes, we experimentally verified the key sensitivity of the proposed algorithm using Keys 1 and 2. Specifically, Key 1 $= (P = 6, l_t = 3,$ seed $=$ "666"), Key 2 $= (P = 6, l_t = 5,$ seed $=$ "666"). Figure 8a–c show the encrypted maps using different keys. Table 5 shows the RMSE and Max-R between the two encrypted maps obtained using the two keys, indicating that there is a large gap between the corresponding encrypted coordinates in the two ciphertext maps. The encryption results using the two keys are significantly different, although the two keys are only slightly different.

Table 5. The difference between the encryption results of Keys 1 and 2.

Datasets	RMSE (m)	Max-R (m)
POIs	4322.2641	15,318.6846
Roads	615.6829	2711.3073
Buildings	437.2614	1757.5051

We also analyzed the effect of a key change on the decryption results. When using Key 1 to decrypt the map encrypted with Key 2, Figure 8d–f show that the decryption fails. This indicates that the ciphertext map cannot be decrypted correctly when the key is modified, even with minimal changes.

All the experiments show that both the encryption and decryption results are sensitive to key change. Therefore, the proposed algorithm has strong key sensitivity.

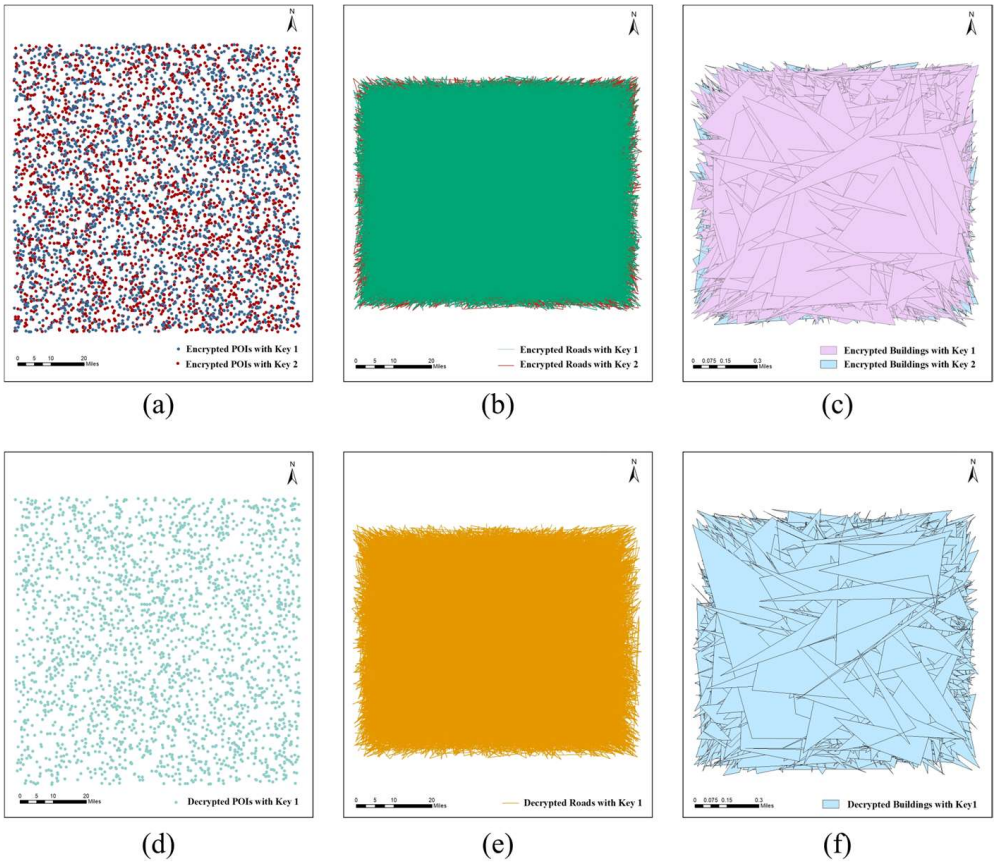


Figure 8. The encrypted maps using slightly different keys and decrypted maps using slightly modified keys: (**a**) encrypted POIs using Key 1 and Key 2; (**b**) encrypted roads using Key 1 and Key 2; (**c**) encrypted buildings using Key 1 and Key 2; (**d**) the result of using Key 1 to decrypt the POIs encrypted with Key 2; (**e**) the result of using Key 1 to decrypt the roads encrypted with Key 2; (**f**) the result of using Key 1 to decrypt the buildings encrypted with Key 2.

5.4.4. Encryption Efficiency Evaluation

The efficiency of the encryption scheme in the proposed CERW method was evaluated using a running time comparison for encryption and decryption, respectively. Wu et al. [18] proposed robust vector map watermarking in the encrypted domain in which the vector map is encrypted using homomorphic encryption. Li et al. [20] encrypt vector maps using a permutation-based encryption scheme in their proposed CEW method. Ren et al. [21] encrypt vector maps based on the congruence relationship in the proposed CEW scheme. The above three algorithms are the latest related algorithms and were selected as the comparison algorithms.

The encryption and decryption programs were run on 64-bit Windows 10 with an Intel Core i5-1135G7 CPU @2.40 GHz, 16 GB of RAM, and a 120 GB HDD. The running time for encryption and decryption of the compared algorithms and the proposed algorithm is listed in Table 6. As seen in Table 6, the method of Wu et al. [18] evidently has the slowest processing speed in encryption and decryption, as it uses homomorphic encryption. The difference in the processing speed among the algorithms created by Li et al. [20] and Ren et al. [21], and the proposed algorithm, is not very large. Table 6 shows that the encryption efficiency of the proposed algorithm is obviously better than that of Wu et al. [18] and at the same level as that of Li et al. [20] and Ren et al. [21].

Table 6. A running time comparison (in milliseconds) of the proposed algorithm and state-of-the-art algorithms for the three test datasets.

Datasets	Operations	Wu et al. [18]	Li et al. [20]	Ren et al. [21]	The Proposed
POIs	Encryption	1672	1048	1372	1353
	Decryption	1565	1005	1348	1290
Buildings	Encryption	4977	1861	2460	2389
	Decryption	5974	1670	2418	2327
Roads	Encryption	10663	2230	2910	2610
	Decryption	7004	2184	2879	2489

5.5. Watermarking Performance Evaluation

The watermarking performance under different interval steps l_t, including watermark capacity, watermark invisibility, and reversibility, is discussed in this section. Some existing related algorithms were selected for watermarking performance comparison. The approach of Wang et al. [31] is a reversible watermarking algorithm for vector maps based on the traditional DE technique. Wang et al. [34] proposed a reversible vector map watermarking algorithm based on virtual coordinates, but this differs from the proposed algorithm in the virtual coordinate construction method and watermark-embedding mechanism. Peng et al. [30] proposed an improved algorithm based on that generated by Wang et al. [34] for 2D CAD engineering graphics.

5.5.1. Analysis of Watermark Capacity

The watermark capacity is the upper limit of the number of watermark bits that can be embedded in a dataset, and the watermark capacity can be expressed in terms of the average number of watermark bits embedded in each vertex, which is also known as the embedding rate (bits/vertex). In the proposed watermarking algorithm, as long as the value of l_t is in the interval $[1, 2(10^P\tau - 1)]$, then each coordinate can be embedded with one watermark bit. Thus, the embedding rate reaches 2 bits/vertex. Moreover, the watermark capacity can be stably achieved for any map type, including maps with a sparse vertex distribution, in which many reversible watermarking algorithms often fail to embed watermarks. For example, Wang et al. [31] failed to embed a watermark in POI data, and Peng et al. [30] failed to embed a watermark in POI and buildings data, as shown in Table 7.

Table 7. A comparison of the watermarking capacity of different algorithms (bits/vertex).

Datasets	Wang et al. [31]	Wang et al. [34]	Peng et al. [30]	The Proposed Algorithm
POIs	-	1.9887	-	2
Roads	0.2790	1.9761	1.9761	2
Buildings	0.1480	1.9568	-	2

Table 7 demonstrates the watermarking capacity of the compared algorithms and the proposed algorithm. Overall, the capacity of the proposed algorithm is stabilized at 2 bits/vertex across different datasets, which is more than 7 times higher than the traditional DE method outlined by Wang et al. [31]. Theoretically, the number of iterations in the algorithm proposed by Peng et al. [30] can tend to infinity, which makes the watermarking capacity of Peng et al. [30] much higher than that of the proposed algorithm without considering other watermark performances. However, it was found that when the algorithm of Peng et al. [30] was applied to large-scale maps in this study, the increase in the number of iterations affects the watermark invisibility, and the computational complexity increases significantly. Thus, the number of iterations in the comparison experiments is set to 1.

Under the same watermark embedding strength, the number of coordinates for embedding watermark bits in the algorithm of Wang et al. [34] is the same as in that of Peng et al. [30], and therefore, the watermark capacity is the same. The watermarking capacity of the proposed algorithm is slightly higher than in the algorithms of Wang et al. [34] and Peng et al. [30], which is because the method used by Wang et al. [34] and Peng et al. [30] cannot embed the watermark into the reference coordinates, while the algorithm in this study allows all vertices to participate in watermark embedding.

In summary, we propose a watermarking algorithm that can achieve a much larger watermark capacity relative to the traditional DE method and significantly reduces the impact of vertex correlation on the watermarking capacity.

5.5.2. Analysis of Watermark Invisibility

Invisibility means that the watermark will not be noticed by users after it is embedded. In this experiment, watermark invisibility was assessed visually and quantitatively. To ensure watermark security and data usability, the coordinate offsets of the watermarked map should be invisible. In the proposed CERW scheme, the coordinate offsets of the decrypted–watermarked (D-Wed) map obtained by decrypting the CERWed map should satisfy the invisibility property.

To visually assess the watermark invisibility of the proposed watermarking scheme, the watermarked map was compared with the original map overlay. There was no significant data distortion or other problems found when checking the topology in ArcGIS. Roads can serve as an example. Figure 9 displays the overlay effect of the watermarked data and the original data, indicating that the watermarked data and the original data are almost the same without zooming in, and zooming in on the details reveals that the coordinates have a slight offset, but overall distortion does not occur, and the element topology is maintained well.

$$\text{Max-R}\ (V,V^w) = \max(\| v_i - v_i^w \|) \tag{26}$$

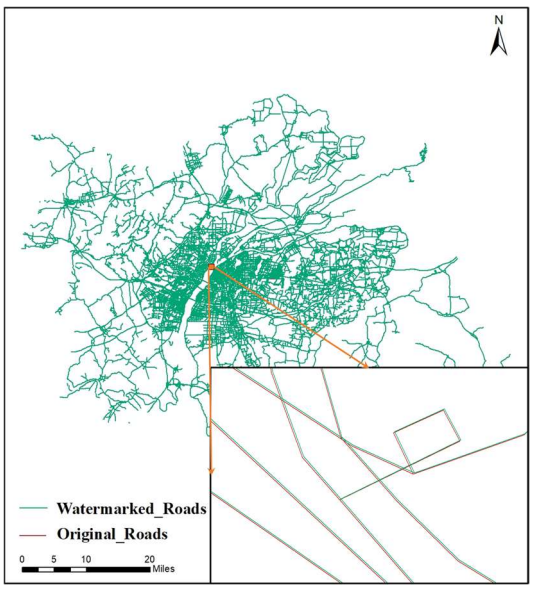


Figure 9. An overlay of the watermarked and original data (roads).

The RMSE and Max-R between the watermarked map and the original map were also calculated to quantitatively measure the watermark invisibility. The smaller the values of RMSE and Max-R, the better the watermark invisibility. In the proposed watermarking scheme, the magnitude of the RMSE and Max-R is affected by the interval step size l_t and the p-value. The relationship between Max-R and l_t and between the RMSE and l_t under different p-values is shown in Figures 10 and 11, respectively.

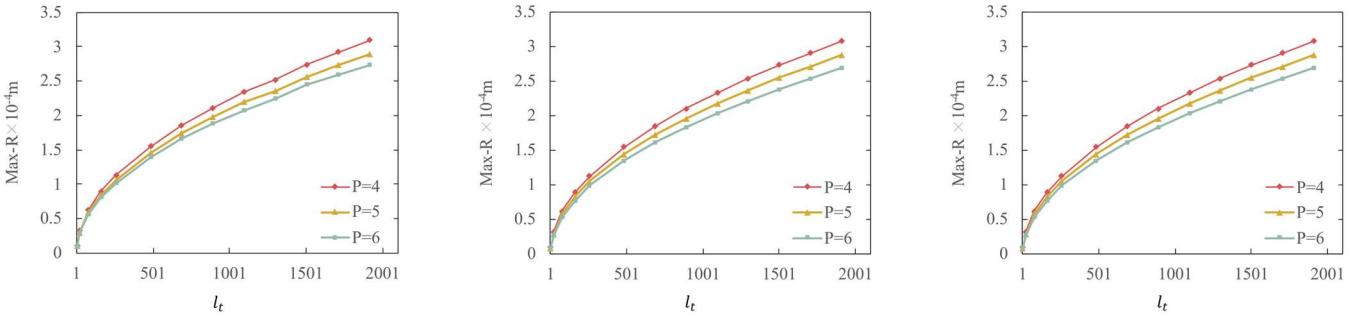


Figure 10. Max-R versus interval step l_t: (**a**) POIs; (**b**) roads; (**c**) buildings.

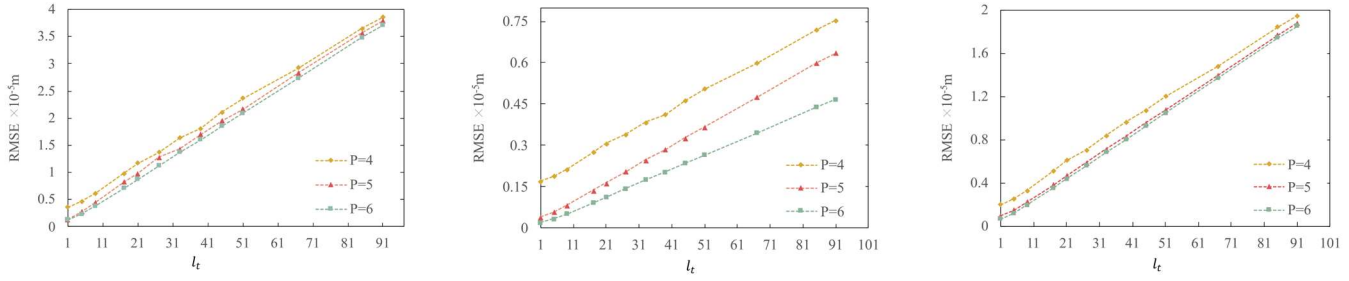


Figure 11. The RMSE versus interval step l_t: (**a**) POIs; (**b**) roads; (**c**) buildings.

Figures 10 and 11 show that the RMSE and Max-R of all three datasets increase with the increase in l_t and decrease with the increase in the *p*-value when l_t is unchanged. Figure 10 shows that the Max-R curves of all three datasets are almost the same. This is because the theoretical value of Max-R is determined only by the *p*-value and l_t, and is not affected by map characteristics. Consequently, although the three datasets have completely different characteristics, the Max-R curves of the three datasets present the same pattern. This is where the proposed watermarking method outperforms the traditional DE watermarking method, in which the Max-R value is affected by the map characteristics. In fact, the traditional DE watermarking method is unsuitable for maps with a sparse vertex distribution because the Max-R and RMSE values are too large. In contrast, the proposed improved DE watermarking method in this study suits all types of maps.

According to the RMSE curves, as shown in Figure 11, the roads dataset has the best invisibility, and the POIs have the relatively worst invisibility among the three datasets. This indicates that the watermark invisibility of a map with a dense vertex distribution is often better than that of a map with a less dense vertex distribution if watermarked using the proposed watermarking method, which is the common characteristic of DE watermarking methods.

To comprehensively demonstrate the watermark invisibility of the proposed algorithm, some related algorithms were selected for comparison. The results of the comparison are listed in Table 8 and shown in Figure 12. In particular, the vertical axis in Figure 12 uses a logarithmic scale, where the longer the length of the bar, the closer it is to 0.0001, indicating a smaller value. Conversely, the shorter the length of the bar, the closer it is to 1, indicating a larger value. Table 8 and Figure 12 show that the RMSE and Max-R of the proposed algorithm are significantly lower than that of the other three algorithms. Moreover, the watermark invisibility of the proposed algorithm is more stable than that of the other three algorithms when applied to different types of maps, which can be proved by Table 8 and Figure 12. Figure 12 also shows that the RMSE and Max-R values of the algorithm of Wang et al. [31], when applied to the three different datasets, are significantly different.

For POIs, the algorithm of Wang et al. [31] even fails to embed watermarks due to the RMSE value being too large. Meanwhile, the algorithm of Peng et al. [30] only succeeds in embedding watermarks in the dataset of roads and fails in the other two datasets. Although the algorithm of Wang et al. [34] successfully embeds watermarks in the three datasets, there is a relatively large difference in the watermark invisibility for the three datasets. It is worth noting that the proposed algorithm not only succeeds in embedding watermarks in the three different datasets, but also achieves approximate Max-R values for the three datasets. In summary, the proposed algorithm suits various kinds of maps and has much better and more stable watermark invisibility.

Table 8. The RMSE and Max-R values of different algorithms (P = 6) (10^{-3} m).

Datasets		Wang et al. [31]	Wang et al. [34]	Peng et al. [30]	The Proposed Algorithm
POIs	RMSE	-	0.25920	-	0.00131
	Max-R	-	0.47274	-	0.01367
Roads	RMSE	0.62505	0.25107	0.34937	0.00020
	Max-R	0.78143	0.33009	0.39582	0.01373
Buildings	RMSE	0.41708	0.26517	-	0.00066
	Max-R	0.50781	0.39302	-	0.01395

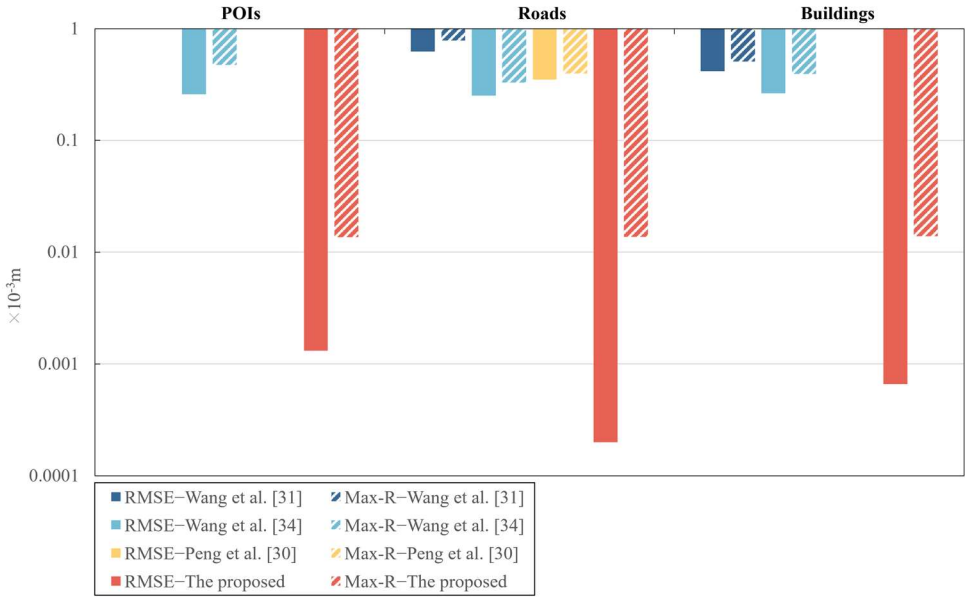


Figure 12. A comparison of the RMSE and Max-R of different reversible algorithms [30,31,34].

In this experiment, when the map scale is 1:1000, the Max-R of the proposed algorithm is 0.00001395 < 0.30 m [18], which is far below the threshold of the actual production standards. The watermarked map accuracy of other test datasets also meets the application requirements. Therefore, the data distortions caused by the watermark embedding in the proposed algorithm do not affect the usability of the vector map.

5.5.3. Analysis of Watermark Reversibility

The watermarking algorithm proposed in the CERW scheme is fully reversible. The data-recovery operation in the CERWed map can be performed either before or after decryption. The encrypted–recovered (E-Red) map without a watermark can be obtained when the data-recovery operation is performed on the CERWed map before decryption, and the decrypted–recovered (D-Red) map without a watermark can be obtained when the data-recovery operation is performed on the CERWed map after decryption. In order to verify the reversibility of the watermarking scheme, the E-Red map was compared with the

original encrypted map, and the D-Red map was compared with the original plaintext map. The results of the comparison are listed in Table 9. As Table 9 shows, all the RMSE and Max-R values of E-Red maps and D-Red maps are 0, which demonstrates the reversibility of the proposed watermarking scheme.

Table 9. A comparison of the reversibility of different algorithms (P = 6) (m).

Datasets		Wang et al. [31]	Wang et al. [34]	Peng et al. [30]	The Proposed Algorithm	
					E-Red Map	D-Red Map
POIs	RMSE	-	6.90900×10^{-8}	-	0	0
	Max-R	-	1.40915×10^{-7}	-	0	0
Roads	RMSE	0	4.06783×10^{-8}	2.45678×10^{-8}	0	0
	Max-R	0	1.02359×10^{-7}	1.04865×10^{-7}	0	0
Buildings	RMSE	0	9.89177×10^{-8}	-	0	0
	Max-R	0	2.33080×10^{-7}	-	0	0

The reversibility of the three compared algorithms was also assessed using the RMSE and Max-R computed for the recovered map and the original map. As listed in Table 9, the RMSE and Max-R values of Wang et al. [31] are all 0, and the RMSE and Max-R values of the algorithms of Wang et al. [34] and Peng et al. [30] are very small floating numbers. Table 9 shows that the reversibility of the algorithm of Wang et al. [31] is the same as the proposed algorithm, while the reversibility of the algorithms of Wang et al. [34] and Peng et al. [30] is relatively worse than that of the proposed scheme. This is determined by the underlying watermarking mechanism. The watermark map data of the proposed algorithm and that of Wang et al. [31] use the DE technique, which shifts the binary form of the difference value one bit to the left to embed the watermark and shifts one bit back to the right to recover data. There is no precision loss in the binary difference shifting operations. In the algorithms of Wang et al. [34] and Peng et al. [30], the watermark embedding and data recovery are achieved by building reversible mapping between original data and watermarked data. Precision loss is caused by floating operations during the mapping process from original data to watermarked data, and inverse mapping from watermarked data to original data. This is why the RMSE and Max-R values of the algorithms of Wang et al. [34] and Peng et al. [30] are not equal to 0. Moreover, the three compared algorithms can only be implemented in the plaintext domain, and the proposed watermarking algorithm can be implemented in both plaintext and ciphertext domains, thus having a higher practical value.

6. Conclusions

In this study, a CERW algorithm for vector maps based on virtual coordinates was proposed. In the proposed coordinate-scrambling encryption scheme, the coordinates are moved by random multiples of the virtual interval step. The random number is generated with a pseudo-random number generator designed using the hash function SHA-512 and the Salsa20 stream cipher algorithm. In the proposed reversible watermarking scheme, the coordinate differences are first calculated with virtual coordinates as references, and then the DE technique is utilized to reversibly embed the watermark. As the coordinate-scrambling encryption does not impact the coordinate differences in which the watermark is embedded, the commutativity between the encryption and reversible watermarking operation is achieved. The operation order of encryption and watermark embedding does not change the final results, and watermark extraction and data recovery can be performed before or after decryption. The main contributions of this study are as follows:

(1). Although there are a few existing CEW algorithms for vector maps, the original map cannot be recovered from the watermarked version in these algorithms, which makes the algorithms unsuitable for applications that require high data precision. The proposed CERW algorithm not only achieves commutativity between encryption and

watermarking but also achieves watermark reversibility, which makes the proposed algorithm suitable for more kinds of applications than the existing CEW algorithms.

(2). The traditional DE watermarking algorithm often has a small watermark capacity and large data distortions when used in vector map watermarking. Specifically, the watermarking operation often fails due to the too-large data distortions introduced by watermarking when it is used to watermark a map with sparse vertex distribution. The watermarking performance of the traditional DE algorithm is affected significantly by map characteristics. The improved DE (IDE) watermarking algorithm proposed in this paper overcomes the above-mentioned problems of traditional algorithms by introducing virtual coordinates as references in the coordinate difference calculation and significantly improves the watermark capacity, which is more than 7 times higher than traditional DE methods. Meanwhile, through adjusting the virtual interval step, the proposed IDE algorithm can achieve very small data distortions after watermarking. Furthermore, the watermarking performance of the proposed IDE method is not affected by map characteristics and is very stable for different kinds of maps.

(3). The encryption method of the proposed CERW scheme was used to design an SHA-512-Salsa20 random number generator to implement coordinate scrambling and achieved a good encryption effect and high encryption security, as analyzed in Section 5.4. It is worth noting that the SHA-512-Salsa20 generator can be replaced with any arbitrary existing pseudo-random number-generation method in the proposed encryption scheme.

The proposed algorithm has a relatively poor performance in watermark robustness. The map-translation operation does not impact the coordinate differences; thus, the proposed algorithm can resist the map-translation attack. However, the map-scaling and map-rotation operations change the coordinate differences, so the proposed algorithm cannot resist these map-scaling and map-rotation attacks. Therefore, improving the robustness of the proposed CERW scheme will be a future line of research. Exploring more secure and effective CERW mechanisms for vector maps over the cloud will be our future research focus, including building commutative secret sharing and reversible watermarking for vector maps over the cloud.

Author Contributions: Qianyi Dai and Baiyan Wu conceived and designed the experiments; Qianyi Dai implemented the methodology; Qianyi Dai and Fanshuo Liu completed the computer code and the implementation of the supporting algorithms; Qianyi Dai performed the analyses and prepared the original draft; Baiyan Wu, Fanshuo Liu, Zixuan Bu, and Haodong Zhang reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research Foundation of the Department of Natural Resources of Hunan Province, China, grant number 20230126XX.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author. The data are not publicly available due to spatial sensitivity.

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

References

1. Da, Q.; Sun, J.; Zhang, L.; Kou, L.; Wang, W.; Han, Q.; Zhou, R. A novel hybrid information security scheme for 2D vector map. *Mob. Netw. Appl.* **2018**, *23*, 734–742. [CrossRef]
2. Zhu, C. Research progresses in digital watermarking and encryption control for geographical data. *Acta Geod. Cartogr. Sin.* **2017**, *46*, 1609–1619. [CrossRef]
3. Wang, Y.; Yang, C.; Zhu, C.; Ding, K. An efficient robust multiple watermarking algorithm for vector geographic data. *Information* **2018**, *9*, 296. [CrossRef]
4. Broumandnia, A. Designing digital image encryption using 2D and 3D reversible modular chaotic maps. *J. Inf. Secur. Appl.* **2019**, *47*, 188–198. [CrossRef]
5. 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. (TIIS)* **2020**, *14*, 323–343. [CrossRef]

6. Pham, G.N.; Ngo, S.T.; Bui, A.N.; Tran, D.V.; Lee, S.H.; Kwon, K.R. Vector map random encryption algorithm based on multi-scale simplification and Gaussian distribution. *Appl. Sci.* **2019**, *9*, 4889. [CrossRef]
7. Higgins, S. The lifecycle of data management. *Manag. Res. Data* **2012**, 17–46. [CrossRef]
8. Shi, Y.Q.; Li, X.; Zhang, X.; Wu, H.T.; Ma, B. Reversible data hiding: Advances in the past two decades. *IEEE Access* **2016**, *4*, 3210–3237. [CrossRef]
9. Ma, K.; Zhang, W.; Zhao, X.; Yu, N.; Li, F. Reversible data hiding in encrypted images by reserving room before encryption. *IEEE Trans. Inf. Foren. Sec.* **2013**, *8*, 553–562. [CrossRef]
10. Zhang, X. Reversible data hiding in encrypted image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [CrossRef]
11. Puech, W.; Chaumont, M.; Strauss, O. A reversible data hiding method for encrypted images. In *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*; SPIE: San Jose, CA, USA, 2008; Volume 6819, pp. 534–542. [CrossRef]
12. Zhang, X.; Long, J.; Wang, Z.; Cheng, H. Lossless and reversible data hiding in encrypted images with public-key cryptography. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 1622–1631. [CrossRef]
13. Peng, F.; Lin, Z.X.; Zhang, X.; Long, M. Reversible data hiding in encrypted 2D ve-ctor graphics based on reversible mapping model for real numbers. *IEEE Trans. Inf. Foren. Sec.* **2019**, *14*, 2400–2411. [CrossRef]
14. Peng, F.; Jiang, W.Y.; Qi, Y.; Lin, Z.X.; Long, M. Separable robust reversible watermarking in encrypted 2D vector graphics. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 2391–2405. [CrossRef]
15. Jang, B.J.; Lee, S.H.; Lee, E.J.; Lim, S.; Kwon, K.R. A crypto-marking method for secure vector map. *Multimed. Tools Appl.* **2017**, *76*, 16011–16044. [CrossRef]
16. Jiang, L.; Xu, Z.; Xu, Y. Commutative encryption and watermarking based on orthogonal decomposition. *Multimed. Tools Appl.* **2014**, *70*, 1617–1635. [CrossRef]
17. Lian, S. Quasi-commutative watermarking and encryption for secure media content distribution. *Multimed. Tools Appl.* **2009**, *43*, 91–107. [CrossRef]
18. Wu, B.; Dai, Q.; Peng, Y.; Wang, W. Robust vector map watermarking algorithm in homomorphic encrypted domain. *J. Geo-Inf. Sci.* **2022**, *24*, 1120–1129. [CrossRef]
19. Ren, N.; Zhu, C.; Tong, D.; Chen, W.; Zhou, Q. Commutative encryption and watermarking algorithm based on feature invariants for secure vector map. *IEEE Access* **2020**, *8*, 221481–221493. [CrossRef]
20. 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. [CrossRef]
21. 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*, 105009. [CrossRef]
22. 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. [CrossRef]
23. Guo, S.; Zhu, S.; Zhu, C.; Ren, N.; Tang, W.; Xu, D. A robust and lossless commutative encryption and watermarking algorithm for vector geographic data. *J. Inf. Secur. Appl.* **2023**, *75*, 103503. [CrossRef]
24. Tan, T.; Zhang, L.; Zhang, M.; Wang, S.; Wang, L.; Zhang, Z.; Wang, P. Commutative encryption and watermarking algorithm based on compound chaotic systems and zero-watermarking for vector map. *Comput. Geosci.* **2024**, *184*, 105530. [CrossRef]
25. Deng, L.Y.; Bowman, D. Developments in pseudo-random number generators. *Wires Comput. Stat.* **2017**, *9*, e1404. [CrossRef]
26. Coron, J.S.; Dodis, Y.; Malinaud, C.; Puniya, P. Merkle-Damgård revisited: How to construct a hash function. In Proceedings of the Advances in Cryptology–CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 430–448. [CrossRef]
27. Ding, L. Improved related-cipher attack on salsa20 stream cipher. *IEEE Access* **2019**, *7*, 30197–30202. [CrossRef]
28. Tian, J. Reversible watermarking by difference expansion. In *Proceedings of Workshop on Multimedia and Security*; ACM: New York, NY, USA, 2002; pp. 19–22.
29. Peng, F.; Lei, Y.Z.; Long, M.; Sun, X.M. A reversible watermarking scheme for two-dimensional CAD engineering graphics based on improved difference expansion. *Comput. Aided Des.* **2011**, *43*, 1018–1024. [CrossRef]
30. Peng, F.; Long, Q.; Lin, Z.X.; Long, M. A reversible watermarking for authenticating 2D CAD engineering graphics based on iterative embedding and virtual coordinates. *Multimed. Tools Appl.* **2019**, *78*, 26885–26905. [CrossRef]
31. Wang, X.; Shao, C.; Xu, X.; Niu, X. Reversible data-hiding scheme for 2-D vector maps based on difference expansion. *IEEE Trans. Inf. Foren. Sec.* **2007**, *2*, 311–320. [CrossRef]
32. Hua, Z.; Zhou, Y.; Huang, H. Cosine-transform-based chaotic system for image encryption. *Inf. Sci.* **2019**, *480*, 403–419. [CrossRef]
33. 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. [CrossRef]
34. Wang, N.; Zhang, H.; Men, C. A high capacity reversible data hiding method for 2D vector maps based on virtual coordinates. *Comput. Aided Des.* **2014**, *47*, 108–117. [CrossRef]