symmetry

MDPI

Article

A Novel Reversible Data Hiding Method for 3D Model in Homomorphic Encryption Domain

Ting Luo [1], Li Li [2,*], Shanqin Zhang [2], Shenxian Wang [2] and Wei Gu [3]

[1] Collage of Science and Technology, Ningbo University, Ningbo 315000, China; luoting@nbu.edu.cn
[2] Department of Computer Science, Hangzhou Dianzi University, Hangzhou 330018, China; sqzhang@hdu.edu.cn (S.Z.); wsx1131@hdu.edu.cn (S.W.)
[3] School of Computer Science and Technology, Anhui University, Hefei 230039, China; 09055@ahu.edu.cn
* Correspondence: lili2008@hdu.edu.cn; Tel.: +86-136-6663-2945

Abstract: Reversible data hiding in the encrypted domain (RDH-ED) is a technique that protects the privacy of multimedia in the cloud service. In order to manage three-dimensional (3D) models, a novel RDH-ED based on prediction error expansion (PEE) is proposed. First, the homomorphic Paillier cryptosystem is utilized to encrypt the 3D model for transmission to the cloud. In the data hiding, a greedy algorithm is employed to classify vertices of 3D models into reference and embedded sets in order to increase the embedding capacity. The prediction value of the embedded vertex is computed by using the reference vertex, and then the module length of the prediction error is expanded to embed data. In the receiving side, the data extraction is symmetric to the data embedding, and the range of the module length is compared to extract the secret data. Meanwhile, the original 3D model can be recovered with the help of the reference vertex. The experimental results show that the proposed method can achieve greater embedding capacity compared with the existing RDH-ED methods.

Keywords: 3D model; RDH-ED; homomorphic encryption; greedy algorithm; prediction error expansion

Citation: Luo, T.; Li, L.; Zhang, S.; Wang, S.; Gu, W. A Novel Reversible Data Hiding Method for 3D Model in Homomorphic Encryption Domain. *Symmetry* **2021**, *13*, 1090. https://doi.org/10.3390/sym13061090

Academic Editor: Aviv Gibali

Received: 18 May 2021
Accepted: 17 June 2021
Published: 19 June 2021

1. Introduction

Since three-dimensional (3D) models can intuitively display stereoscopic information about the real world, there are potential scenes in many applications, such as human, architectural, organ models, etc. [1–3]. With the increasing popularity of cloud-based storage, users often store their 3D models in a third party; thus, they can conveniently access the data. To manage the uploaded 3D model, the cloud administrator may embed additional information for media notation, content verification, and so on [4,5]. Moreover, the cloud administrator is not required to change the 3D models permanently when embedding data [6]. Since reversible data hiding (RDH) can completely recover the original media after the secret data are extracted, the RDH method has received much attention from researchers.

The reversible data hiding (RDH) method can be classified into three types: lossless compression [7], difference expansion (DE) [8], and histogram shifting (HS) [9]. Using lossless compression-based RDH, Fridrich et al. compressed an original image to empty the space in order to embed data [7], but the corresponding embedding capacity was small due to the low compression. Using DE-based RDH, Tian expanded the difference between two adjacent pixels to embed data [8]. However, two adjacent pixels sometimes are not close when they are located at the edges, and this leads to high image distortion. Using the histogram shifting-based RDH, Ni et al. built a pixel histogram, and then used the peak to embed data [9]. The pixel histogram is often the mean distribution and the peak is not sharp, so that the embedding performance is not satisfactory. As an extension of DE, Thodi presented a prediction error expansion (PEE)-based RDH method [10], which is effective.

In the following, many improvements are presented for the PEE-based RDH method, such as prediction accuracy, pairwise PEE, sorting, and so on [11–13].

The abovementioned RDH methods were designed for images and cannot be directly used for 3D models since the structures of the 3D model and the image are different. Dittmann et al. firstly presented an RDH method for authenticating a 3D model, wherein the concept of a distortion-free 3D model is introduced [14]. Wu et al. embedded secret data by modulating the distances between the face and the 3D model center, so that authentication can be obtained via extracting the data [15]. Moreover, the modulation information is stored in the Stego 3D model; thus, reversibility can be achieved. Jhou et al. selected the last few digits of the vertex coordinate value to embed data, and the original model can be perfectly recovered if it is intact [16]. Sun et al. and Lu et al. embedded data in a predictive vector quantization (PVQ) compressed domain by modifying the prediction mechanism during the compression process [17,18]. Wu et al. designed an RDH method based on DE, which modified differences between the adjacent vertex coordinates to embed data in order to keep the mesh topology unchanged [19]. Wu et al. predicted the vertex position by calculating the centroid of its traversed neighbors and then used PEE to embed data [20]. However, although their method has a large embedding capacity, the 3D model distortion is obvious. Jiang et al. built a three-dimensional prediction error histogram by using recursive construction coding due to the similarities between adjacent vertices, and they then modified the histogram to embed data [21]. Zhang et al. presented an RDH for 3D models based on hybrid prediction, which exploited vertex–vertex correlations to obtain high prediction accuracy to improve the data embedding performance [22].

In reality, before the media is transmitted into the cloud service, it will be encrypted in order to prohibit the cloud administrator from accessing the content of the 3D model [23]. Therefore, the RDH in the encrypted domain (RDH-ED) is more popular in real applications [24,25].

In this paper, an RDH-ED based on PEE is presented for managing the 3D model in the cloud service. Firstly, the user encrypts the 3D model by using the homomorphic Paillier cryptosystem for transmission to the cloud service. Secondly, for embedding data, a greedy algorithm is used to divide the vertices of the 3D model into reference and embedded sets in order to increase the embedding capacity. The reference vertex is employed to compute the prediction error for the embedded vertex, and the module length of the prediction error is expanded to embed data. In parallel with the data embedding, the data extraction in at the receiving end uses the range of the module length to be compared for extracting the secret data; meanwhile, the original 3D model is recovered by using the reference vertex completely. The experimental results show that the proposed method is superior to the existing method in relation to the embedding capacity. The main contributions of the paper are listed as follows.

1. A greedy algorithm is employed to classify the vertex into the embedded set and the reference set to increase the embedding capacity.
2. The module length of the prediction error is expanded to embed data in the encryption domain.
3. The embedding capacity of the proposed method is superior to that of the existing RDH-ED methods.

The rest of this article is organized as follows. The Paillier cryptosystem is briefly introduced in Section 2. The proposed method is described in detail in Section 3. Section 4 presents the experimental results. Section 5 provides the conclusions.

2. Related Work

In this section, we introduce the encryption technique and review related RDH-ED methods. Commonly used encryption algorithms include stream bit encryption and Paillier encryption. Stream bit encryption uses a stream ciphertext to encrypt the image, which has low complexity and no pixel overflow [26]. Since homomorphic encryption can directly process data in the encryption domain and there is no need to decrypt the cipher-texts before operating them, the Paillier cryptosystem is more practical [27,28]. For example,

Chen et al. encrypted an image through the Paillier cryptosystem and used the additive homomorphism of the Paillier encryption system to embed the secret data [29].

The RDH-ED method can be divided into two categories. The first category is reserving room before encryption (RRBE), in which the room for data embedding is created before encrypting the original image [30–32]. Zhang et al. embedded one bit into each encrypted image block by flipping three least significant bits (LSB), and the data were extracted through the texture evaluation [33]. On the basis of Zhang's study, Hong et al. increased the embedding capacity by using a public key modulation mechanism [34]. However, in these two methods, the image needs to be decrypted before data extraction. Zhang compressed the LSBs of the encrypted image to provide embedding room, and they separated data extraction and image recovery [35]. Xiang et al. designed an RDH-ED method using the properties of the Paillier cryptosystem [36], where data can be extracted from the decrypted domain and the encrypted domain, respectively. In summary, the RRBE method requires a great deal of work before encryption, and reliable interaction with the cloud administrator must be established.

Compared with RRBE, vacating room after encryption (VRAE) as the second category is more practical [37,38]. Xiang et al. selected two pixels as a group for encryption, and data were embedded by shifting the histogram of the absolute difference between pairs of pixels [39]. However, the embedding capacity was insufficient. Xiong et al. embedded two bits into groups of three pixels to increase the embedding capacity [40]. However, these RDH-ED methods are designed for images; to date, only a few RDH-ED methods have been studied for 3D models. Jiang et al. proposed an RDH-ED based on stream bit encryption, and they embedded data by flipping several LSBs of vertex coordinates [41]. With the help of spatial correlation of the 3D model, the receiver extracted data by using the smooth function. Li et al. divided the 3D model into non-overlapping patches, and the vertex in each patch was encrypted using the Paillier cryptosystem [42]. In this method, three directions of each patch are computed to build the corresponding histogram for embedding data. However, each patch only can hold one bit, and the embedding capacity cannot be increased further. Moreover, the cipher mapping table should be transmitted to the receiver for data decryption. In order to increase the embedding capacity, Li et al. encrypted the 3D model using the Paillier cryptosystem and used the dyeing algorithm to classify vertices into the embedded set and the reference set for data embedding [43]. To embed more than one bit into the embedded vertex, each bit is mapped to the direction of the vertex, which should be recorded in a mapping table. However, after the dyeing algorithm, the number of embedded vertices is low, which affects the embedding capacity. Furthermore, the mapping table as the additional information should be transmitted for data extraction.

Finally, to better distinguish the proposed method from some related methods, some differences are listed in Table 1. Similar to Jiang's [41] and Li's [43] methods, the vertex is classified, but the proposed method uses a greedy algorithm to improve the embedding capacity. Similarly to Li's [43] method, the proposed method uses PEE but expands the module length of the prediction error to embed data. Compared with Jiang's [41] and Li's [42] methods, the proposed method embeds more than one bit for one vertex. Moreover, the proposed method only transmits secret keys as the additional information for data extraction and uses the Paillier cryptosystem to encrypt the 3D model.

Table 1. Differences among related RDH-ED methods.

	Jiang [41]	**Li [42]**	**Li [43]**	**Proposed**
Vertex Classification	Yes	No	Yes (Dyeing algorithm)	Yes (Greedy algorithm)
PEE	No	No	Yes (Angle)	Yes (Module length)
Embedded bits in one vertex	One	Less than one	More than one	More than one
Side information	Small	Huge	Huge	Small
Encryption	Stream bit encryption	Paillier	Paillier	Paillier

3. Background

The Paillier cryptosystem is an additive homomorphic encryption method [27] that is often used to process ciphertext in the encrypted domain. Its homomorphism shows that the ciphertext can be arithmetically operated after encryption and the operation result is consistent with the corresponding operation result in the plaintext field. The Paillier cryptosystem mainly includes key generation, encryption, and decryption, which are depicted here in detail.

Key generation is used to select two large primes, a_1 and a_2, randomly at first. Then, $N = a_1 \times a_2$ and $\gamma = f_1(a_1, a_2)$, where $f_1(\bullet)$ returns the smallest common multiple. Afterwards, we select an integer $g \in Z^*_{N^2}$, where $Z_{N^2} = \{0, 1, 2, \ldots, N^2 - 1\}$ and $Z^*_{N^2}$ are the numbers in Z_{N^2}, which are prime with N^2. Moreover, g satisfies:

$$f_2\left(f_3\left(\mathrm{mod}\left(g^{\gamma}, N^2\right)\right)\right) = 1, \tag{1}$$

where $f_2(\bullet)$ returns the greatest common divisor, mod($\bullet$) returns the remainder after division, and $f_3(x) = (x - 1)/N$. Finally, we obtain the public key (N,g) and the private key γ.

In the process of encryption, we select an integer $t \in Z^*_{N^2}$ randomly, and the plaintext $s \in Z_{N^2}$ is encrypted with the public key (N, g) using Equation (2):

$$c = \mathrm{En}(s, t) = \mathrm{mod}\left(g^s \times t^N, N^2\right), \tag{2}$$

where En($\bullet$) denotes the encryption function, and c is the ciphertext. Owing to the nature of the Paillier cryptosystem, it ensures the security of the ciphertext. Specifically, different ciphertexts are computed with different t for the same s, and, after decryption, different ciphertexts can be restored to the same s with the secret key γ:

$$s = \mathrm{De}(c) = \mathrm{mod}\left(\frac{f_3(\mathrm{mod}(c^{\gamma}, N^2))}{f_3(\mathrm{mod}(g^{\gamma}, N^2))}, N\right), \tag{3}$$

where De($\bullet$) denotes the decryption function. In the following, homomorphism and homomorphic extension of subtraction are described for the Paillier cryptosystem. For the homomorphism, let two plaintexts be s_1 and s_2, respectively, and $\forall t_1, t_2 \in Z^*_{N^2}$. The two ciphertexts are computed as:

$$\begin{aligned} c_1 &= \mathrm{En}(s_1, t_1), \\ c_2 &= \mathrm{En}(s_2, t_2). \end{aligned} \tag{4}$$

The original Paillier cryptosystem has addition homomorphism and multiplication homomorphism:

$$c_1 \times c_2 = \mathrm{En}(s_1, t_1) \times \mathrm{En}(s_2, t_2) = \mathrm{mod}\left(g^{s_1+s_2} \times (t_1 \times t_2)^N, N^2\right), \tag{5}$$

$$\mathrm{De}(c_1 \times c_2) = \mathrm{De}\left(\mathrm{mod}\left(\mathrm{En}(s_1, t_1) \times \mathrm{En}(s_2, t_2), N^2\right)\right) = \mathrm{mod}(m_1 + m_2, N). \tag{6}$$

The subtraction homomorphism can be obtained through modular multiplication inverse (MMI). In order to compute $s_1 - s_2$ in the Paillier cryptosystem, first, the negative number $-s_2$ is represented by N–s_2. Then, the Euclidean technique is used to compute the ciphertext of N–s_2. Suppose that $\mathrm{En}(s_2)^{-1}$ is the cipertext of N–m_2; thus, in the Paillier cryptosystem, s_1+($-s_2$) is computed using $\mathrm{mod}(\mathrm{En}(s_1, t_1) \times \mathrm{En}(s_2, t_2)^{-1}, N^2)$.

4. The Proposed Method

Using the Paillier cryptosystem, the 3D model is encrypted and then uploaded to the cloud service. In the cloud service, the corresponding administrator embeds data into the encrypted domain, as illustrated in Figure 1. First, the vertices are preprocessed for

the Paillier cryptosystem, and then the greedy algorithm is used to classify the vertices into reference vertices and embedded vertices in order to increase the embedding capacity. Then, the prediction error is computed for each embedded vertex, and data are embedded using PEE. On the receiver side, the data can be extracted and the original 3D model can be restored by comparing the module length range of the prediction error, which is symmetric to the data embedding. In the following, the preprocessing, encryption, data embedding, and data extraction are depicted in detail.

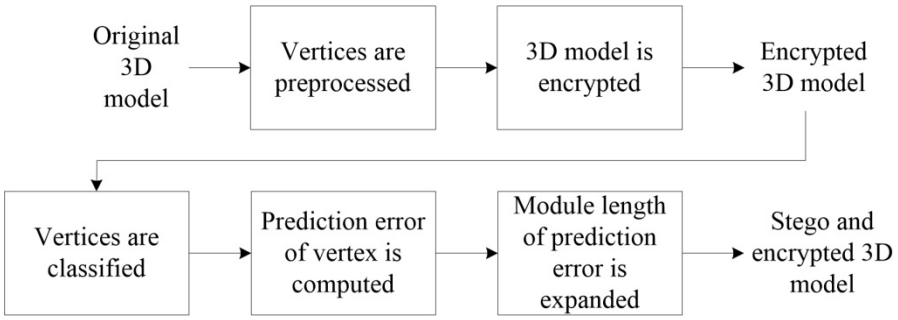


Figure 1. Flowchart of proposed RDH-ED method.

4.1. Preprocessing and Encryption

Generally, uncompressed vertices of the 3D model are 32-bit floating point numbers. However, the Paillier cryptosystem requires positive integers for the input; thus, the vertex coordinates are required to be converted from the decimal to the positive integer.

The 3D model mainly consists of vertex data and face data. The vertex data include coordinates of each vertex, and the face data supply the topological information that reflects the connection between vertices. Let $\{v_i\}_{i=0}^{N_v}$ represent the sequence of vertices after scanning a 3D model, where N_v is the number of vertices and $v_i = \{v_{i,x}, v_{i,y}, v_{i,z}\}$ as shown in Table 2. Note that each coordinate of the vertex $|v_{i,j}| < 1$, $j \in \{x, y, z\}$ and its significant digit is 6.

Table 2. File format of the 3D model.

Vertex List				**Face Information**	
Index of Vertex	***X*-Axis**	***Y*-Axis**	***Z*-Axis**	**Index of Face**	**Elements in Each Face**
1	$v_{1,x}$	$v_{1,y}$	$v_{1,z}$	1	(2, 3, 1)
2	$v_{2,x}$	$v_{2,y}$	$v_{2,z}$	2	(5, 1, 4)
3	$v_{3,x}$	$v_{3,y}$	$v_{3,z}$	3	(3, 6, 7)
4	$v_{4,x}$	$v_{4,y}$	$v_{4,z}$	4	(5, 2, 1)
5	$v_{5,x}$	$v_{5,y}$	$v_{5,z}$	5	(3, 10, 7)
...	...	...	...	...	...

Since the 3D model can be accurately represented by the first four significant digits of the vertex coordinates, the vertex coordinates are converted into an integer with four significant digits by using the following formula:

$$v'_{i,j} = \left\lfloor v_{i,j} \times 10^4 \right\rfloor. \tag{7}$$

Then, all coordinates of the vertices are changed to the positive integer:

$$v'_{i,j} = v'_{i,j} + 10^4. \tag{8}$$

After preprocessing, $v'_{i,j}$ is encrypted as

$$c_{i,j} = En\left(v'_{i,j}, t_{i,j}\right) = \mathrm{mod}\left(g^{v'_{i,j}} \times {t_{i,j}}^N, N^2\right), \quad (9)$$

where $t_{i,j}$ is a small random integer for $v'_{i,j}$, and it is less than N.

4.2. Processes of Data Hiding

After encryption, data are embedded to manage the encrypted 3D model. Firstly, the vertex is classified into the embedded vertex set and reference vertex set using the greedy algorithm. Then, the prediction error of the embedded vertex is calculated by using the reference vertex. Finally, the module length of the prediction error is expanded to embed the secret data in the encrypted domain.

When the 3D model is encrypted and uploaded to the cloud, the cloud administrator still can embed the secret data without knowing the original content. Some vertices are chosen to embed data and their adjacent vertices are unmodified as references for prediction. In the following, vertex classification, prediction error computation, data embedding, and the parameter d are described in detail.

4.2.1. Vertex Classification

First, the 1-ring neighborhood and 2-ring neighborhood of a vertex are defined. For two vertices v_i and v_p, where $i,p \in \{1,2, \ldots, N_v\}$, if they are connected by an edge, v_i is an adjacent neighbor of v_p. All adjacent neighbors of v_i constitute the 1-ring neighborhood, and the neighbors of all 1-ring neighborhoods constitute its 2-ring neighborhood. Vertices are classified into the embedded set and the reference set, which are used for embedding data and predicting data, respectively. Specifically, one vertex in the embedded set is utilized to embed data, and its 1-ring neighborhood is not changed for prediction. Let the embedded set and the referenced set be $\mathbf{S}e$ and $\mathbf{S}r$, respectively. In order to increase the embedding capacity, the maximum of $\mathbf{S}e$ must be reached.

Vertices of $\mathbf{S}e$ are not connected each other, and if the graph is bipartite, the polynomial algorithm can identify the largest set of nonadjacent vertices. However, there are three loops in the 3D model and the patch structure of the 3D model cannot be represented by the bipartite graph. Thus, NP represents a significant challenge in obtaining the maximum value of $\mathbf{S}e$ in the 3D model, and the greedy algorithm is employed to obtain embedded vertices.

When a vertex is added to $\mathbf{S}e$, the 1-ring neighborhood of this vertex is added to $\mathbf{S}r$. According to the local optimal solution of the greedy algorithm, the values of $\mathbf{S}r$ and $\mathbf{S}e$ should be small and large, respectively, by selecting the vertex with the smallest number of the 1-ring neighborhood to be added to $\mathbf{S}e$. The specific steps of vertex classification are listed as follows.

Step a-1. Compute the vertex number of the 1-ring neighborhood for all vertices in the 3D model; we define it as the degree of each vertex.

Step a-2. Suppose that k is the minimum of all degrees and choose vertices with the degree of k to form a set, denoted as $\mathbf{V}_k$. For example, $\mathbf{V}_3$ consists of vertices with the degree of 3.

Step a-3. One vertex of $\mathbf{V}_k$ is added to $\mathbf{S}e$, and the 1-ring neighborhood of this vertex is added to $\mathbf{S}r$.

Step a-4. Remove this vertex and its 1-ring neighborhood from the 3D model, and the degrees of vertices in the 2-ring neighborhood are updated.

Step a-5. If all vertices of the 3D model are assigned to $\mathbf{S}e$ or $\mathbf{S}r$, the vertex classification is finished. Otherwise, return to Step a-2 until all vertices are assigned.

In order to describe the processes clearly, an example is illustrated in Figure 2. First, we scan v_1, v_2, ... , v_{10} and v_{11} and compute their degrees. We obtain $\mathbf{V}_2 = \{v_4, v_6, v_{11}\}$, $\mathbf{S}e = \{v_4\}$, and $\mathbf{S}r = \{v_1, v_5\}$. We remove these vertices from the 3D model and update the degrees of corresponding vertices. At this time, the minimum degree is 1, $\mathbf{V}_1 = \{v_2\}$,

S*e* = {v_4} ∪ {v_2} = {v_4, v_2} and **S***r* = {v_1,v_5} ∪ {v_3} = {v_1,v_5,v_3}. We update the degrees of corresponding vertices after removing assigned vertices and then continue to classify the remaining vertices. Lastly, **S***e* = {v_4, v_2, v_6, v_{10}, v_{11}} and **S***r* = {v_1, v_5, v_3, v_7, v_9, v_8}.

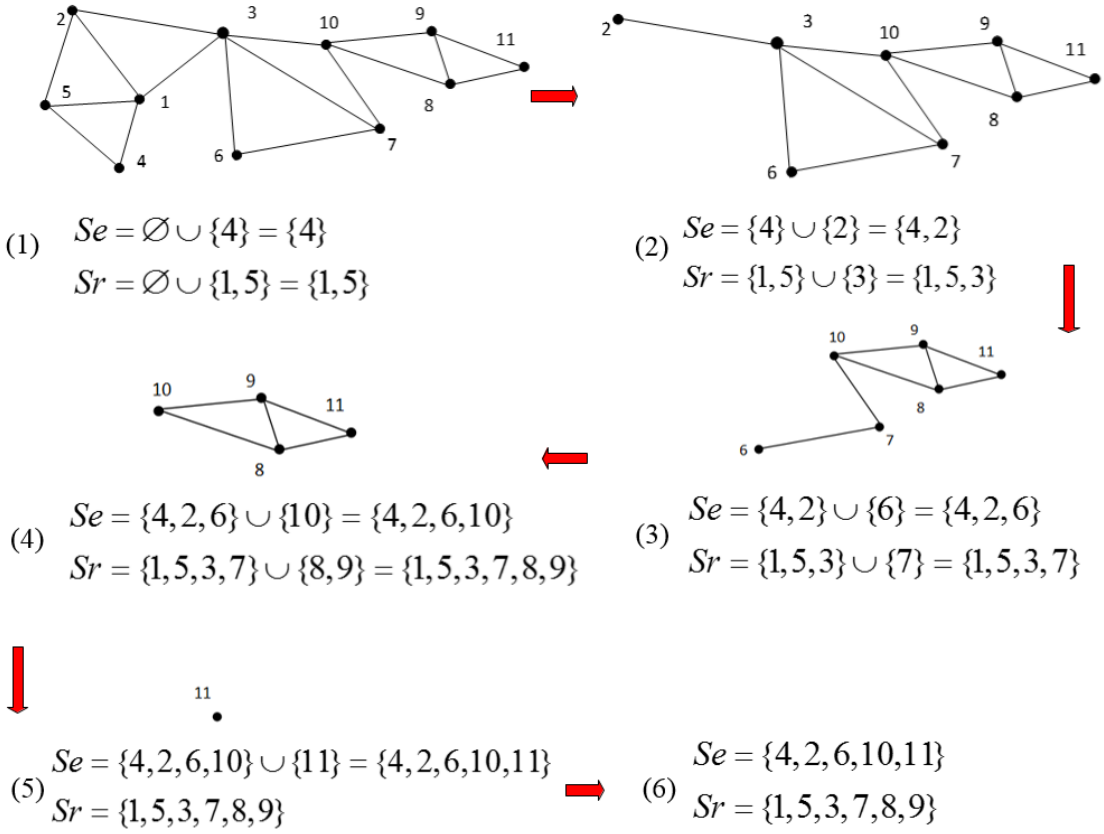


Figure 2. Illustration of vertex classification.

4.2.2. Prediction Error Computation

The 1-ring neighborhood is used to predict the current vertex in **S***e*, and the prediction value of v'_i is computed as

$$\overline{v_i} = \frac{1}{N_i} \sum_{l=1}^{N_i} v'_l, \tag{10}$$

where v'_l is the 1-ring neighborhood vertex of v'_i from **S***r*, and N_i is the number of vertices in the 1-ring neighborhood. In the spatial domain, $\overline{v_i}$ is close to v'_i and the prediction error is computed as:

$$\Delta v_i = v'_i - \overline{v_i}. \tag{11}$$

Δv_i includes three directional values, and its module length is often small. Suppose that D is the maximum module length for all prediction errors, and $|\Delta v_i|$ satisfies:

$$|\Delta v_i| \in [0, D). \tag{12}$$

4.2.3. Data Embedding

Before data embedding, the secret data are divided evenly into several groups; suppose that $w_0, w_1, \ldots, w_{n\text{-}1}$ are bits in each group, where n is the number of bits. The weight of each bit is computed as:

$$s_w = \sum_{i=0}^{n-1} w_i \times 2^i, s_w \in [0, 2^n - 1]. \tag{13}$$

v'_i is modified to embed data:

$$v''_i = v'_i + s_w \times d \times \vec{b} = \left(v'_{i,x} + s_w \times d, v'_{i,y} + s_w \times d, v'_{i,z} + s_w \times d\right), \tag{14}$$

where d is the secret key for data embedding, $\vec{b} = (1,1,1)$, and the corresponding module length of Δv_i is changed. In the encryption domain, data are embedded as:

$$c'_{i,j} = c_{i,j} \times c_{s_w} = \mathrm{mod}\left(\mathrm{En}\left(v'_{i,j}, r_{i,j}\right) \cdot \mathrm{En}(s_w \times d, r_{s_w}), N^2\right), \tag{15}$$

where r_{s_w} is the random integer, and c_{s_w} is the encrypted ciphertext of $s_w \cdot d$.

After embedding data, the prediction error is modified as:

$$\Delta v'_i = \Delta v_i + s_w \times d \times \vec{b}. \tag{16}$$

When the directions of Δv_i and $\vec{b}$ are the same, the module length $|\Delta v'_i|$ is the maximum. If the directions of Δv_i and $\vec{b}$ are opposite, $|\Delta v'_i|$ is the minimum, which is

$$\begin{cases} |\Delta v'_i|_{max} = s_w \times \sqrt{3}d + |\Delta v_i|, if\ \Delta v_i = \frac{|\Delta v_i|}{\sqrt{3}} \times \vec{b} \\ |\Delta v'_i|_{min} = s_w \times \sqrt{3}d - |\Delta v_i|, if\ \Delta v_i = -\frac{|\Delta v_i|}{\sqrt{3}} \times \vec{b} \end{cases}. \tag{17}$$

From Equation (17), it can be concluded that the value of $|\Delta v'_i|$ is the range of $s_w \times \sqrt{3}d - |\Delta v'_i|$ to $s_w \times \sqrt{3}d + |\Delta v'_i|$.

4.2.4. Parameter d

In order to increase the correctness of data extraction, their ranges of module lengths should not be overlapped after embedding the secret data. After embedding s_w + 1, its minimum module length should be greater than the maximum module length after embedding s_w; that is,

$$(s_w + 1) \times \sqrt{3}d - |\Delta v_i| \geq s_w \times \sqrt{3}d + |\Delta v_i|. \tag{18}$$

From Equation (18), we can obtain $d \geq 2 \times |\Delta v_i| / \sqrt{3}$. If $d = 2 \times D / \sqrt{3}$. The ranges of module lengths are not overlapped for different s_w, and $|\Delta v'_i| \in \left[0, \sqrt{3}d/2\right)$. After data embedding, the range of $|\Delta v'_i|$ is defined as:

$$\begin{array}{ll} |\Delta v'_i| \in \left[(s_w - 0.5) \times \sqrt{3}d, (s_w + 0.5) \times \sqrt{3}d\right), & if\ s_w \in [1, 2^n - 1] \\ |\Delta v'_i| \in \left[0, \frac{\sqrt{3}}{2}d\right], & if\ s_w = 0 \end{array} \tag{19}$$

Figure 3 shows the value of $|\Delta v'_i|$ after '0' or '1' is embedded, and, obviously, the module length of embedding '1' is greater than that of embedding '0'.

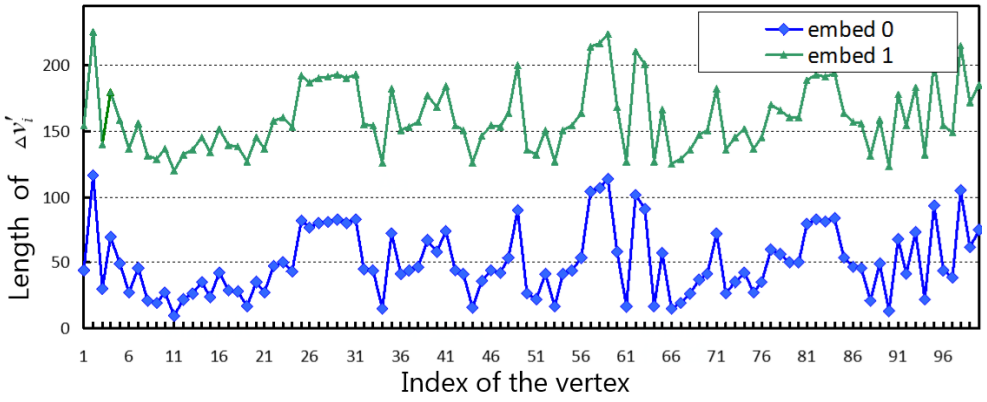


Figure 3. The module length after data embedding.

4.3. Data Extraction and Model Recovery

In the receiving side, when the Stego and encrypted 3D model is received, it can be decrypted via the private key γ. The decrypted vertex is computed as:

$$v''_{i,j} = \mathrm{De}\left(c'_{i,j}\right) = \mathrm{mod}\left(\frac{f_3\left(\mathrm{mod}\left(c'^{\gamma}_{i,j}, N^2\right)\right)}{f_3(\mathrm{mod}(g^{\gamma}, N^2))}, N\right). \tag{20}$$

Since the coordinates of a few vertices are modified, the decrypted 3D model is similar to the original 3D model. After decryption, the data extraction and 3D model recovery are processed:

- Step b-1. All vertices are classified into the embedded set and the reference set according to the greedy algorithm.
- Step b-2. The prediction error $\Delta v'_i$ of v_i" is computed.
- Step b-3. Equation (21) is used to obtain the range of s_w:

$$\frac{|\Delta v'_i|}{\sqrt{3}d} - 0.5 < s_w < \frac{|\Delta v'_i|}{\sqrt{3}d} + 0.5. \tag{21}$$

From Equation (21), only one integer is in the range, and s_w can be extracted correctly. s_w is converted to the binary bits as:

$$w_i = \mathrm{mod}\left(\left\lfloor \frac{s_w}{2^i}, 2 \right\rfloor\right). \tag{22}$$

- Step b-4. Via d and s_w, the original 3D model is recovered as:

$$v'_i = v''_i - \mathrm{s}_w \times d \times \vec{b}. \tag{23}$$

5. Experimental Results and Discussion

The experiments were implemented in MATLAB R2016b under Windows 7 Pro. To demonstrate the effectiveness of the proposed method, one hundred 3D models were used for testing, and six 3D models are illustrated in Figure 4.

The *SNR* was used to evaluate the quality of the Stego 3D model.

$$SNR = 10\mathrm{lg}\frac{\sum_{i=1}^{N_v}\left((v_{i,x} - \overline{v_x})^2 + (v_{i,y} - \overline{v_y})^2 + (v_{i,z} - \overline{v_z})^2\right)}{\sum_{i=1}^{N_v}\left((g_{i,x} - v_{i,x})^2 + (g_{i,y} - v_{i,y})^2 + (g_{i,z} - v_{i,z})^2\right)}, \tag{24}$$

where N_v is the number of vertices, $v_{i,x}$, $v_{i,y}$, and $v_{i,z}$ are the vertex coordinates of the original 3D model, $g_{i,x}$, $g_{i,y}$, and $g_{i,z}$ are the vertex coordinates of the Stego 3D model, and $\overline{v_x}$, $\overline{v_x}$, and $\overline{v_x}$ are the mean of $v_{i,x}$, $v_{i,y}$, and $v_{i,z}$, respectively.

Suppose that SNR_D is the *SNR* of the decrypted 3D model, and SNR_R is the *SNR* of the recovered 3D model. *BER* is used to evaluate the correctness of the data extraction. For the parameter D, we compute the maximum module lengths for the 3D models, and it is concluded that $D < 250$ [43].

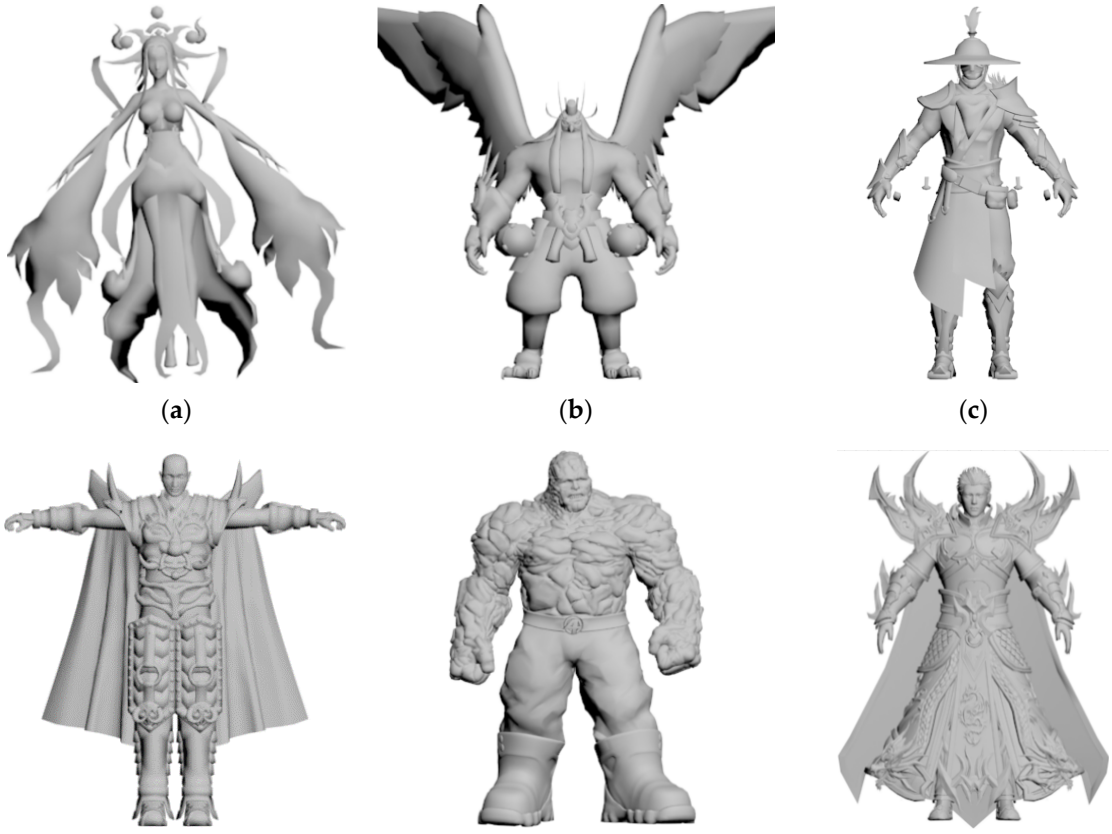


Figure 4. Original 3D model. (**a**) Fairy, (**b**) Boss, (**c**) Solider, (**d**) Devil, (**e**) Thing, (**f**) Lord.

5.1. Discussion of Parameters d and n

In order to show the feasibility of the proposed data hiding method, 1024 bits were embedded in the 3D models, and the data could be extracted completely. Taking 'Solider' as an example, the 'solider' encrypted using the Paillier cryptosystem cannot be recognized, as illustrated in Figure 5b. Without data extraction, the 3D model can be decrypted as illustrated in Figure 5d, and the decrypted 3D model is similar to the original 3D model. The 3D model can be recovered completely and data can be extracted without any error, as illustrated in Figure 5e,f, respectively.

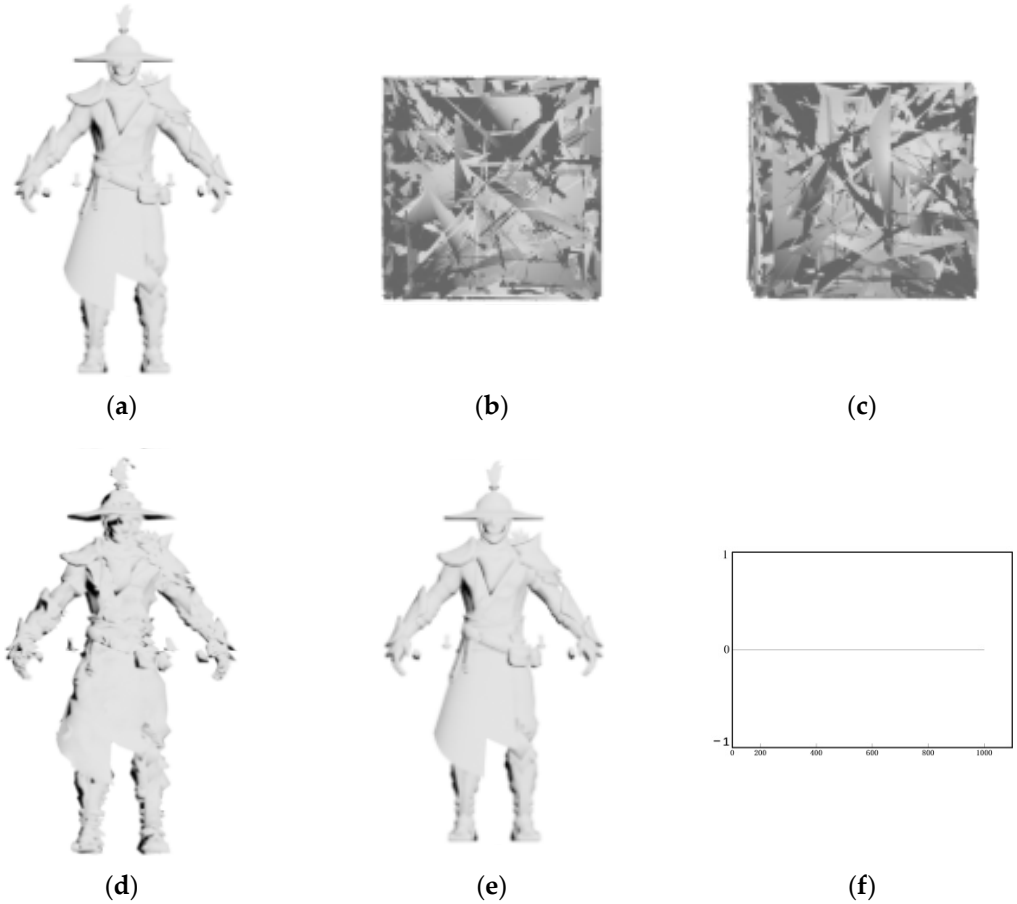


Figure 5. Data embedding and extraction of 'Solider'. (**a**) Original 3D model, (**b**) encrypted 3D model, (**c**) Stego and encrypted 3D model, (**d**) decrypted and Stego 3D model, (**e**) recovered 3D model (SNR = +∞), (**f**) *BER* of extracted data (BER = 0).

The parameters d and n are related to the 3D model's quality and the correctness of the data extraction. When d is low, *BER* is increased and the 3D model distortion is small, as illustrated in Figure 6a. If d is high, *BER* is decreased and the distortion of the 3D model is increased.

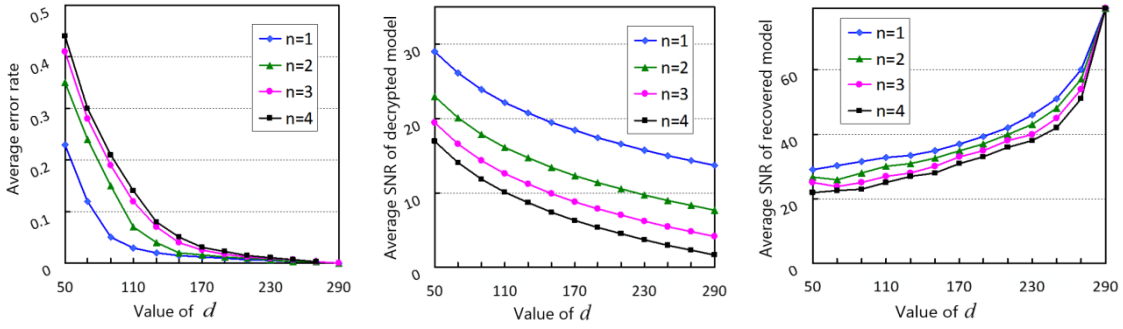
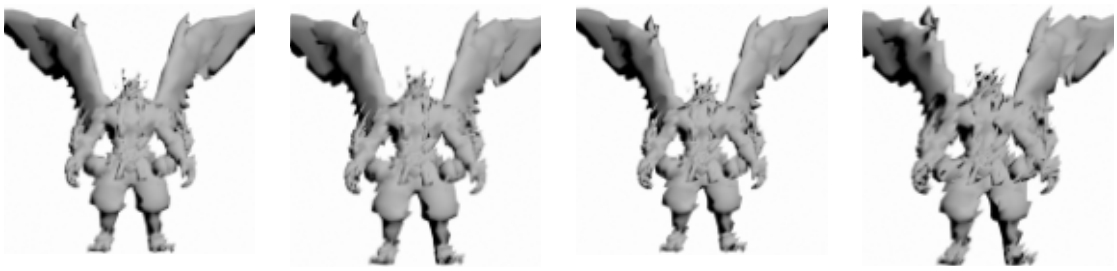


Figure 6. The data hiding performance versus different values of d. (**a**) Average *BER*, (**b**) average SNR_D, (**c**) average SNR_R.

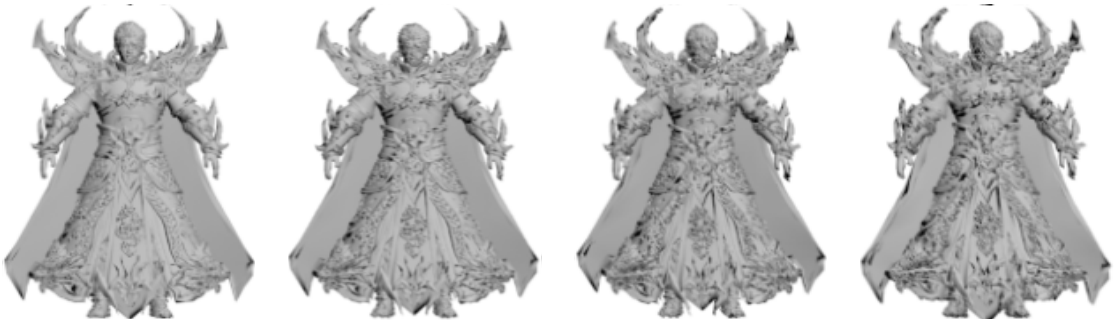
If n is large, s_w is large, and the embedding capacity and the corresponding model distortion are increased, as illustrated in Figure 6. For instance, when n = 1 and d = 110, the corresponding *BER* = 2.89%, SNR_D = 15.34 dB, and SNR_R = 33.89 dB. From the experiments, we can conclude that when d = 150 and n = 3, the model distortion is small and *BER* is

small as well. Figure 7 shows the visual quality of the 3D models for different values of d when $n = 3$, and Figure 8 shows the visual quality of the decrypted 3D models for different values of n when $d = 150$. It was also found that the decrypted 3D model was similar to the original 3D model.



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

Figure 7. Decrypted 3D models for different values of d when $n = 3$. (**a**) $d = 110$, (**b**) $d = 130$, (**c**) $d = 150$, (**d**) $d = 170$.



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

Figure 8. Decrypted 3D models for different values of n when $d = 150$. (**a**) $n = 1$, (**b**) $n = 2$, (**c**) $n = 3$, (**d**) $n = 4$.

5.2. Vertex Classification Comparison

In order to show the effectiveness of the proposed method for classifying the vertex into the embedded set and the reference set using the greedy algorithm, Jiang's [41] and Li's methods were used for comparison, as shown in Table 3. In Jiang's [41] method, the number of vertices in $\mathbf{S}_e$ is large, but some vertices are connected, which will lead to high *BER* of data extraction. The proposed method classifies the vertices more efficiently by using the greedy algorithm compared with Jiang's [41] method since the number of connected vertices is 0. The number of vertices in $\mathbf{S}_e$ is greater than in Li's [43] method, which indicates that the proposed method can embed more bits.

Table 3. Comparisons of the vertex number in $\mathbf{S}_e$.

3D Model	N_v	Proposed		Jiang [41]		Li [43]	
		$\mathbf{S}_e$	Connected Vertex	$\mathbf{S}_e$	Connected Vertex	$\mathbf{S}_e$	Connected Vertex
Fairy	4252	1389	0	1356	312	1086	0
Boss	10,663	3412	0	3515	948	2498	0
Devil	27,872	8640	0	8357	2146	6944	0
Thing	110,812	34,905	0	35,820	9420	23,368	0
Lord	250,343	80,109	0	81,521	21,321	53,548	0

5.3. Performance Comparison

In order to show the effectiveness of the proposed method, Li's [43] method was first used for comparison. We embedded one bit in each vertex, and, as shown in Table 4, the proposed method performed better than Li's [43] in relation to the embedding capacity and the 3D model quality. This is mainly because the greedy algorithm for vertex classification is superior to the dyeing algorithm. This also shows that the proposed method successfully achieves low embedding capacity.

Table 4. Performance comparison with Li's [43] method.

	SNR_D	SNR_R	Embedding Capacity (.bpp)
Proposed	16.65	∞	0.324
Li's [43]	15.24	∞	0.242

In Table 5, Jiang's [41] and Li's [42] methods are used for comparison. Compared with Jiang's [41] method, the embedding capacity of the proposed method is nearly three times higher, at 0.972 bpp, as shown in Table 5. The corresponding *BER* is also lower than that of Jiang's method and the quality is also higher. Compared with Li's [42] method, although the image quality is slightly lower, the embedding capacity is still much higher. Moreover, when d is adjusted to 289, the image quality of the proposed method can be increased significantly. Thus, in summary, the proposed method is superior to the other two methods, with proven effectiveness.

Table 5. Performance comparison with Jiang's [41] and Li's [42] methods.

	BER	SNR_R	Embedding Capacity (.bpp)
Proposed (d = 150)	3.34%	34.14	0.972
Proposed (d = 289)	0	∞	0.972
Jiang [41]	4.22%	31.97	0.369
Li [42]	0	∞	0.396

6. Conclusions

In this paper, a reversible data hiding in encrypted domain (RDH-ED) based on prediction error expansion (PEE) has been presented. Before being sent to the cloud for storage, the 3D model is encrypted using the homomorphic Paillier cryptosystem so that the content cannot be accessed by the cloud administrator. In the cloud service, data are embedded into the encrypted domain for management of the 3D model. Specifically, the vertices of the 3D models are divided into embedded and reference sets using the greedy algorithm, and the maximum vertex number is obtained for the embedded set in order to increase the embedding capacity. The reference vertex is used to compute the prediction value of the embedded vertex, and the corresponding prediction error is calculated and expanded to embed data. On the receiving side, the secret data are extracted by comparing the module length of the prediction error, and the original 3D model can be reconstructed. The experimental results show that the proposed method is superior to the existing RDH-ED methods. However, when the embedding capacity is high, the visual quality is reduced significantly; we will attempt to resolve this in future studies.

Author Contributions: The conceptualization and funding acquisition are credited to L.L. The writing—review and editing are credited to T.L. The methodology and experiments are credited to S.W. The conceptualization and supervision are credited to S.Z. and W.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the National Natural Science Foundation of China (No. 61971247, No. 61370218) and the Public Welfare Technology and Industry Project of Zhejiang Provincial Science Technology Department (No. LGG19F020016).y.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors are very thankful to the editor and referees for their valuable comments and suggestions for improving the paper.

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

References

1. He, X.W.; Huang, T.T.; Bai, S.; Bai, X. View N-Gram network for 3D object retrieval. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 7515–7524.
2. Gao, Z.; Li, T.M.; Wan, S.H. Exploring deep learning for view-based 3D model retrieval. ACM Transactions on Multimedia Computing. *Commun. Appl.* **2020**, *16*, 1–21.
3. Zhang, Q.; Im, J.; Park, M.; Choi, M.; Kim, C.H.; Shim, Y. Shrubbery-shell inspired 3D model stylization. *Comput. Graph.* **2019**, *82*, 13–21. [CrossRef]
4. Lee, C.F.; Shen, J.J.; Agrawal, S.; Li, Y.H. High-capacity embedding method based on double-layer octagon-shaped shell matrix. *Symmetry* **2021**, *13*, 583. [CrossRef]
5. Zafeiriou, S.; Tefas, A.; Pitas, I. Blind robust data hiding schemes for copyright protection of 3D mesh objects. *IEEE Trans. Vis. Comput. Graph.* **2005**, *11*, 596–607. [CrossRef] [PubMed]
6. Bhardwaj, R.; Aggarwal, A. Hiding clinical information in medical images: An encrypted dual-image reversible data hiding algorithm with base-3 numeral framework. *Optik* **2019**, *181*, 1099–1112. [CrossRef]
7. Fridrich, J.; Goljan, M.; Du, R. Lossless data embedding for all image formats. In Proceedings of the SPIE 4675, Security and Watermarking of Multimedia Contents IV, San Jose, CA, USA, 19–23 January 2002; pp. 572–583.
8. Tian, J. Reversible Data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [CrossRef]
9. Ni, Z.; Shi, Y.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
10. Thodi, D.M.; Rodríguez, J.J. Expansion embedding techniques for reversible data hiding. *IEEE Trans. Image Process.* **2007**, *16*, 721–730. [CrossRef] [PubMed]
11. Luo, T.; Jiang, G.; Yu, M.; Zhong, C.; Xu, H.; Pan, Z. Convolutional neural networks-based stereo image reversible data hiding method. *J. Vis. Commun. Image Represent.* **2019**, *61*, 61–73. [CrossRef]
12. Ou, B.; Li, X.; Zhao, Y.; Ni, R.; Shi, Y.Q. Pairwise prediction-error expansion for efficient reversible data hiding. *IEEE Trans. Image Process.* **2018**, *22*, 5010–5021. [CrossRef] [PubMed]
13. Qin, J.; Huang, F. Reversible data hiding based on multiple two-dimensional histograms modification. *IEEE Signal Process. Lett.* **2019**, *26*, 843–847. [CrossRef]
14. Dittmann, J.; Benedens, O. Invertible authentication for 3D meshes. In Proceedings of the SPIE 5020, Security and Watermarking of Multimedia Contents V, Santa Clara, CA, USA, 20–24 January 2003; pp. 653–664.
15. Wu, H.; Ming, Y. A Reversible Data Hiding Approach to Mesh Authentication. In Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, Compiegne, France, 19–22 September 2005; pp. 774–777.
16. Jhou, C.; Pan, J.; Chou, D. Reversible data hiding base on histogram shift for 3D vertex. In Proceedings of the Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007), Kaohsiung, Taiwan, 26–28 November 2007; pp. 365–370.
17. Sun, Z.; Lu, Z.; Li, Z. Reversible data hiding for 3D meshes in the PVQ-compressed domain. In Proceedings of the IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Pasadena, CA, USA, 18–20 December 2006; pp. 593–596.
18. Lu, Z.; Li, Z. High capacity reversible data hiding for 3D meshes in the PVQ domain. In Proceedings of the International Workshop on Digital Data Hiding, Sana Barbara, CA, USA, 19–21 May 2008; pp. 233–243.
19. Wu, D.; Wang, G. A reversible watermarking scheme for 3D meshes. In Proceedings of the International Conference on Active Media Technology, Beijing, China, 22–24 October 2009; pp. 513–521.
20. Wu, H.; Dugelay, J. Reversible watermarking of 3D mesh models by prediction-error expansion. In Proceedings of the IEEE 10th Workshop on Multimedia Signal Processing, Queensland, Australia, 8–10 October 2008; pp. 797–802.
21. Jiang, R.; Zhang, W.; Hou, D.; Wang, H.; Yu, N. Reversible data hiding for 3D mesh models with three-dimensional prediction-error histogram modification. *Multimed. Tools Appl.* **2018**, *77*, 5263–5280. [CrossRef]
22. Zhang, Q.; Song, X.; Wen, T.; Fu, C. Reversible data hiding for 3D mesh models with hybrid prediction and multilayer strategy. *Multimed. Tools Appl.* **2019**, *78*, 29713–29729. [CrossRef]
23. Elsheh, E.; Hamza, A.B. Secret sharing approaches for 3D object encryption. *Expert Syst. Appl.* **2011**, *38*, 13906–13911. [CrossRef]
24. Huang, D.; Wang, J. High-capacity reversible data hiding in encrypted image based on specific encryption process. *Signal Proces. Image Commun.* **2020**, *80*, 115632. [CrossRef]
25. Zhou, N.; Zhang, M.; Wang, H.; Ke, Y.; Di, F. Separable reversible data hiding scheme in homomorphic encrypted domain based on ntru. *IEEE Access* **2020**, *8*, 81412–81424. [CrossRef]

26. Zhou, J.; Sun, W.; Dong, L.; Liu, X.; Au, O.C.; Tang, Y.Y. Secure reversible image data hiding over encrypted domain via key modulation. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *26*, 441–452. [CrossRef]
27. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
28. Brakershi, Z.; Vaikuntanathan, V. Effificient fully homomorphic encryption from (standard) LWE. *Siam J. Comput.* **2014**, *43*, 831–871. [CrossRef]
29. Chen, B.; Wu, X.; Lu, W.; Ren, H. Reversible data hiding in encrypted images with additive and multiplicative public-key homomorphism. *Signal Process.* **2019**, *164*, 48–57. [CrossRef]
30. Zhang, W.; Ma, K.; Yu, N. Reversibility improved data hiding in encrypted images. *Signal Process.* **2014**, *94*, 118–127. [CrossRef]
31. Shiu, C.W.; Chen, Y.C.; Hong, W. Encrypted image-based reversible data hiding with public key cryptography from difference expansion. *Signal Process. Image Commun.* **2015**, *39*, 226–233. [CrossRef]
32. Cao, X.; Du, L.; Wei, X.; Meng, D. High capacity reversible data hiding in encrypted images by patch-level sparse representation. *IEEE Trans. Cybern.* **2016**, *46*, 1132–1143. [CrossRef] [PubMed]
33. Zhang, X. Reversible data hiding in encrypted image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [CrossRef]
34. Hong, W.; Chen, T.S.; Wu, H. An improved reversible data hiding in encrypted images using side match. *IEEE Signal Process. Lett.* **2012**, *19*, 199–202. [CrossRef]
35. Zhang, X. Separable reversible data hiding in encrypted image. *IEEE Trans. Inf. Forensics Secur.* **2012**, *16*, 826–832. [CrossRef]
36. Xiang, S.; Luo, X. Reversible data hiding in homomorphic encrypted domain by mirroring ciphertext group. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *28*, 3099–3110. [CrossRef]
37. Zhang, X.; Qian, Z.; Feng, G. Effificient reversible data hiding in encrypted images. *J. Vis. Commun. Image Represent.* **2014**, *25*, 322–328. [CrossRef]
38. Qian, Z.; Zhang, X. Reversible data hiding in encrypted image with distributed source encoding. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 636–646. [CrossRef]
39. Xiang, S.; Luo, X. Efficient reversible data hiding in encrypted image with public key cryptosystem. *Eurasip J. Adv. Signal Process.* **2017**, *2017*, 59. [CrossRef]
40. Xiong, L.; Dong, D.; Xia, Z.; Chen, X. High-capacity reversible data hiding for encrypted multimedia data with somewhat homomorphic encryption. *IEEE Access* **2018**, *6*, 60635–60644. [CrossRef]
41. Jiang, R.; Zhang, W.; Yu, N. Reversible data hiding in encrypted three-dimensional mesh models. *IEEE Trans. Multimed.* **2018**, *20*, 55–67. [CrossRef]
42. Li, L.; Wang, S.; Zhang, S.; Luo, T. Homomorphic encryption-based robust reversible data hiding for 3D model. *Symmetry* **2020**, *12*, 347. [CrossRef]
43. Li, L.; Wang, S.; Luo, T.; Chang, C.C.; Zhou, Q.; Li, H. Reversible Data Hiding for Encrypted 3D Model Based on Prediction Error Expansion. *J. Sens.* **2020**, 8851999. [CrossRef]