future internet

Article

Reversible Data Hiding in Encrypted 3D Mesh Models Based on Multi-Group Partition and Closest Pair Prediction

Xu Wang [1,2], Jui-Chuan Liu [2], Ching-Chun Chang [3] and Chin-Chen Chang [2,*]

1 School of Information Science and Engineering, University of Jinan, Jinan 250022, China; ise_wangx@ujn.edu.cn
2 Department of Information Engineering and Computer Science, Feng Chia University, 100 Wenhwa Road, Seatwen, Taichung 40724, Taiwan; p1200318@o365.fcu.edu.tw
3 Department of Computer Science, University of Warwick, Coventry CV47AL, UK; c.c.chang@warwickgrad.net
* Correspondence: alan3c@gmail.com

Abstract: The reversible data hiding scheme in the encrypted domain is a potential solution to the concerns regarding user privacy in cloud applications. The 3D mesh model is an emerging file format and is widely used in engineering modeling, special effects, and video games. However, studies on reversible data hiding in encrypted 3D mesh models are still in the preliminary stage. In this paper, two novel techniques, multi-group partition (MGP) and closest pair prediction (CPP), are proposed to improve performance. The MGP technique adaptively classifies vertices into reference and embeddable vertices, while the CPP technique efficiently predicts embeddable vertices and generates shorter recovery information to vacate more redundancy for additional data embedding. Experimental results indicate that the proposed scheme significantly improves the embedding rate compared to state-of-the-art schemes and can be used in real-time applications.

Keywords: reversible data hiding; 3D mesh models; cloud services; multi-group partition; closest pair prediction

check for updates

Citation: Wang, X.; Liu, J.-C.; Chang, C.-C.; Chang, C.-C. Reversible Data Hiding in Encrypted 3D Mesh Models Based on Multi-Group Partition and Closest Pair Prediction. *Future Internet* **2024**, *16*, 210. https://doi.org/10.3390/fi16060210

Academic Editor: Gianluigi Ferrari

Received: 18 April 2024
Revised: 12 June 2024
Accepted: 13 June 2024
Published: 15 June 2024

1. Introduction

With the rapid development of cloud services and computing, users' behavior has drastically altered. A massive amount of data has been stored and processed on cloud servers, and users can download required resources at any time through wired or wireless networks. However, the security vulnerability of the cloud is a critical issue, and user information disclosure events on the cloud emerge frequently. The data encryption technique [1], which can encrypt the plaintext into ciphertext, is one of the solutions proposed to protect the content security of users. However, cloud administrators usually need some functions, such as authentication, access control, and multimedia management. The additional data should be embedded into the multimedia files on the cloud. Therefore, the reversible data hiding technique [2–4] is introduced, and the reversible data hiding in the encrypted domain becomes a potential solution that has flourished in recent years. In the cloud, there are three roles in such applications: the content owner encrypts the multimedia files and uploads them into the cloud; the cloud server embeds some additional data into the encrypted files and stores them in the cloud; and an authorized third-party can acquire the embedded encrypted files and extract the embedded data or recover the original files according to different keys.

Most researchers focus on the digital image domain, a series of reversible data hiding schemes in encrypted images that have been proposed and can be classified into two categories: vacate the spare room after image encryption (VRAE) and reserve the spare room before image encryption (RRBE). In the conventional VRAE schemes, the content owner needs only to encrypt the image directly and the spare room is vacated on the fully encrypted image by the LSB flipping technique [5–7], LSB compression technique [8,9], prediction error detection [10], and pixel rotation [11]. However, because the fully encrypted image lacks redundancy, most of these schemes cannot totally extract the accurate embedded data, and all of them are unable to recover the original image without loss. Therefore, recent schemes have proposed some special image encryption methods, such as homomorphic encryption [12–14] and block based encryption [15–18], to recover the original image and further improve the embedding capacity. With the enhancement of the capabilities of personal computing devices, some RRBE schemes [19–25] have been proposed to preprocess the original image before image encryption to reserve redundancy by compressing the original image. As the original image has sufficient redundancy, the embedding capacities of RRBE schemes are higher than those in VRAE schemes.

Reversible data hiding in encrypted 3D mesh models finds practical applications across diverse fields to enhance their utility and security. In healthcare, patients' information can be embedded into encrypted MRI scans to ensure patient privacy during transmission and storage to facilitate secure remote consultations and diagnoses. In the manufacturing industry, CAD models of proprietary designs can incorporate watermarking to safeguard intellectual property rights and prevent unauthorized replication or modification of critical components. Moreover, 3D models of characters and environments can embed copyright information to ensure the protection of creative assets in virtual worlds and gaming environments in the field of digital entertainment. In the realm of augmented reality (AR) and virtual reality (VR), encrypted 3D models enriched with additional data can enable immersive training simulations for aviation and defense, where sensitive information needs to be securely integrated into training modules. Therefore, introducing reversible data hiding technique into encrypted 3D mesh models has broad application scenarios. As the 3D mesh model is an emerging file format, there are only a few schemes that have explored reversible data hiding in encrypted 3D mesh models. The first scheme was proposed in 2017 by Jiang et al. [26]. They converted the three-dimensional coordinates into integers and chose the central vertex for additional data embedding. Based on the integer coordinates, the idea of homomorphic encryption is further used in [27], and is improved in [28]. In [29], the 3D spatial subdivision and space encoding techniques are designed. Additionally, the MSB prediction technique is also introduced in [30]. The majority voting method is then used to predict multi-MSBs of the usable vertices [31]. To further investigate more usable vertices, half of the vertices are chosen as usable in [32] to improve the embedding rate. In [33], the ring Co-XOR encryption technique is proposed, utilizing the ring's center vertex to predict the ring's edge vertices for data hiding. However, the redundancy issue of these state-of-the-art schemes has not been effectively eliminated.

In this paper, two novel techniques, multi-group partition (MGP) and closest pair prediction (CPP), are proposed to further improve the performance of the reversible data hiding scheme in encrypted 3D mesh models. The MGP can better identify optimal grouping situations, while the CPP can effectively predict vertex coordinates. Using both techniques, we can efficiently compress 3D mesh models for abundant additional data embedding. Based on these two proposed techniques, the main contributions of our proposed scheme can be summarized into two aspects as follows:

(1) The MGP technique classifies more vertices as usable, and can adaptively choose different parameters for different models to find the optimal grouping strategy.
(2) The CPP technique chooses the closest reference vertex to predict the current usable vertex to improve the prediction accuracy and only pay a little cost of redundancy.

Finally, the embedding rates of our proposed scheme have been significantly improved compared to other state-of-the-art schemes. Section 2 illustrates the proposed reversible data hiding scheme in encrypted 3D mesh models in detail. Section 3 presents the experimental results, and Section 4 provides conclusions.

2. Proposed Scheme

In this section, the proposed reversible data hiding scheme in encrypted 3D mesh models is detailed. The main techniques are based on the two techniques multi-group partition (MGP) and closest pair prediction (CPP). Figure 1 shows the schematic framework of our proposed scheme. First, the content owner detects the redundant room, which is reserved for additional data embedding in the original 3D mesh models using CPP and MGP before preprocessing, and then encrypts the 3D mesh models and uploads the encrypted models into the cloud. Then, the cloud server can embed some additional data into the encrypted models. When a third-party receives the embedded encrypted models, based on different keys, the embedded data can be extracted or the original 3D mesh models can be recovered. The details of preprocessing, multi-group partition, closest pair prediction, 3D mesh model encryption and uploading on the content owner side, data embedding on the cloud server side, data extraction, and 3D mesh model recovery on the third-party side are described in the following subsections. The symbols used in this paper are listed in Table 1.

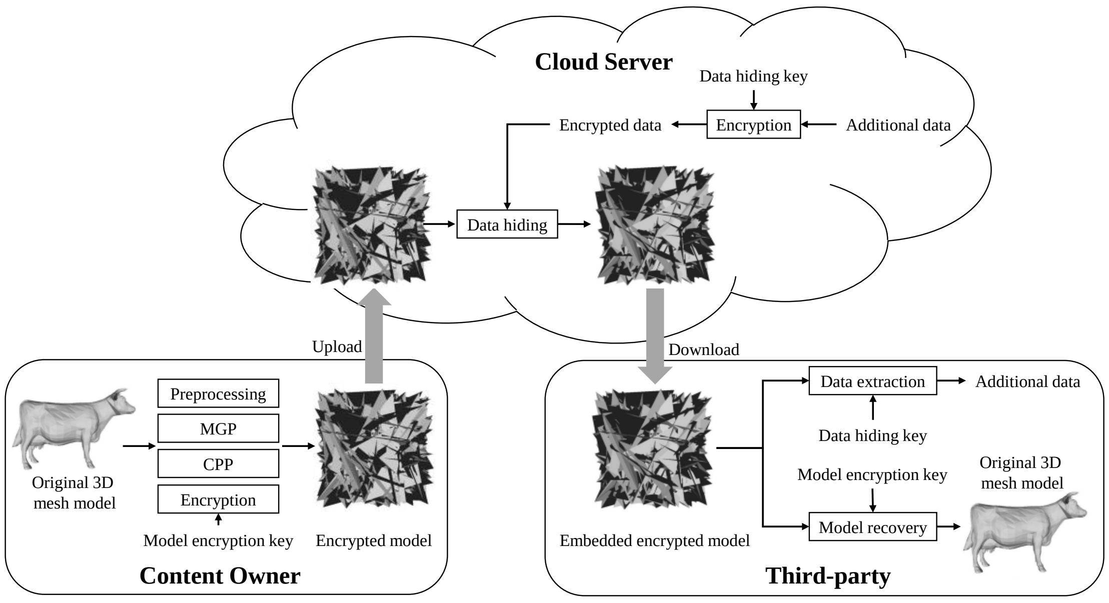

Figure 1. Schematic workflow.

Table 1. The List of Symbols.

Symbol	Definition	Symbol	Definition
V/v	Vertex set/vertex	F/f	Face set/face
(x, y, z)	Three-dimensional coordinate	n_V/n_F	Number of vertex/face
i/j	Index of vertex/face	σ	Number of preserving valid bits
$\mathbb{v}$	Integer vertex	$\hat{v}$	Recovered vertex
G/g	Vertex group/each group	α	Number of vertices in each group
$\mathbb{v}g$	Vertex in each group	$\mathbb{v}u$	Usable vertex
$\mathbb{v}r$	Reference vertex	R	Reference vertex set
n_r	Number of reference vertices	l	Index of R
d	Distance between $\mathbb{v}u$ and $\mathbb{v}r$	$\mathbb{v}r^c$	Closest reference vertex
d^c	Closest distance	β	Maximum length of three-dimensional closest distance
θ	Bits used to record β	$\mathbb{R}$	Recovery information sequence
δ	Length of binary bits	K_m/K_h	Model encryption/data hiding key
$\overline{\mathbb{R}}$	Encrypted $\mathbb{R}$	γ	Embedding rate

2.1. On Content Owner Side

On the content owner side, the original 3D mesh model is first preprocessed and divided into several groups. In each group, the first vertex is denoted as reference and then the closest pair prediction is proposed to predict rest vertices using the reference vertices. By doing so, the original 3D mesh model can be compressed to vacate the spare room before model encryption. Finally, the content owner encrypts the processed model and uploads it to the cloud.

2.1.1. Preprocessing

Similar to the related schemes, only OFF format 3D mesh files are discussed in this paper. The 3D mesh model M contains vertices and faces in the OFF file. The vertex set and the face set are represented by:

$$V = \left\{ v^i_{(x,\, y,\, z)} \middle| 0 < i \le n_V \right\}, \tag{1}$$

$$F = \left\{ f^j \middle| 0 < j \le n_F \right\}, \tag{2}$$

where i and j are the indexes of vertex and face, and n_V and n_F are the numbers of vertex and face. The $(x,\ y,\ z)$ indicates the three-dimensional coordinate of each vertex, and each coordinate has 6 valid bits, i.e., is represented by a 32-bit floating point number. Each face contains three vertices and is not modified in this paper. Standing on the shoulders of giants, we compress the 3D coordinate of each vertex by preserving σ valid bits and transform it into an integer by:

$$\mathbb{v}^i_{(x,\, y,\, z)} = \left\lfloor v^i_{(x,\, y,\, z)} \times 10^{\sigma} \right\rfloor. \tag{3}$$

Then, the integer coordinate can be used for further processes. The floating coordinate can be recovered after the rest processing by:

$$\hat{v}^i_{(x,\, y,\, z)} = \frac{\mathbb{v}^i_{(x,\, y,\, z)}}{10^{\sigma}}. \tag{4}$$

2.1.2. Multi-Group Partition

As the coordinates of the adjacent vertices are very similar, after obtaining the integer coordinates of all vertices, the vertices can be classified into two sets: embeddable set and reference set. The reference vertices are used for predicting coordinates of embeddable vertices, which have not made any changes. In the conventional schemes, several reference vertices are used for predicting one embeddable vertex, so the redundancy in these reference vertices is not efficiently vacated. In this paper, the vertices are first classified into groups. Based on the index of each vertex, all α vertices are gathered into one group to form vertex group $G = \left\{ g^k \middle| 0 < k \le n_V/\alpha \right\}$. Then, for each group, the first vertex $\mathbb{v}g^k_1$ is chosen as reference vertex, and the rest vertices $\mathbb{v}g^k_2$ to $\mathbb{v}g^k_\alpha$ are embeddable. When setting the α equal to 2, it can be regarded as classifying vertices by odd-even indexes. In this paper, we mainly discuss the value of α equal to 2, 3, and 4, because with the value of α growing, the prediction accuracy by our proposed CPP will undoubtedly be decreasing, so that it is not conductive for redundancy vacating.

2.1.3. Closest Pair Prediction

After vertices grouping and classification, each embeddable vertex is predicted by the closest reference vertex. However, before prediction, each embeddable vertex should be further classified into usable and unusable sets judged by:

$$\mathbb{v}g \text{ is} \begin{cases} \text{usable}, & \text{if } \mathbb{v}g \text{ connects at least one reference vertex}, \\ \text{unusable}, & \text{otherwise}. \end{cases} \tag{5}$$

Only usable vertices are processed by CPP, and unusable vertices are preserved unchanged. It should be noted that the proposed CPP will not change the vertex index, so the usable and unusable vertices can be also judged before CPP. Therefore, no additional location map is used for recording usability.

For each usable vertex, denoted as $\mathbb{v}u_{(x,\, y,\, z)}$, the corresponding reference vertex set $R = \left\{ \mathbb{v}r^l_{(x,\, y,\, z)} \middle| 0 < l \leq n_r \right\}$ is generated, where n_r denotes the number of reference vertices connected to the current usable vertex. The distance between $\mathbb{v}u_{(x,\, y,\, z)}$ and $\mathbb{v}r^l_{(x,\, y,\, z)}$ is calculated by:

$$d^l = \sum_{x,y,z} |\mathbb{v}u_{(x,\, y,\, z)} - \mathbb{v}r^l_{(x,\, y,\, z)}|. \tag{6}$$

Next, these distances are compared and the reference vertex with the minimum distance is chosen as the closest reference vertex, denoted as $\mathbb{v}r^c$. Then, the three-dimensional distance between the usable vertex and its closest reference vertex can be obtained by:

$$d^c_{(x,\, y,\, z)} : \left(d^c_x, d^c_y, d^c_z\right) = \mathbb{v}u_{(x,\, y,\, z)} - \mathbb{v}r^c_{(x,\, y,\, z)}. \tag{7}$$

If the closest vertex can be distinguished and the three-dimensional distance between them can be acquired, the original usable vertex can be recovered by:

$$\mathbb{v}u_{(x,\, y,\, z)} = \mathbb{v}r^c_{(x,\, y,\, z)} + d^c_{(x,\, y,\, z)}. \tag{8}$$

Therefore, we record the index of the closest vertex and the three-dimensional distance using binary bits. First, the index of the closest vertex is recorded by $\lceil log_2^{n_r} \rceil$ bits. Second, each decimal distance is converted into binary, and the maximum length of them is denoted as β. Third, β by θ bits is recorded and 3β bits are used to record three-dimensional distance. It should be declared that $\theta = 4$ is more than enough to record θ for most of 3D mesh models. In some extreme cases, the abnormal vertices with higher β can be recorded in advance using a location map, preserving these vertices unchanged. Finally, $\lceil log_2^{n_r} \rceil + \left\lceil log_2^{\beta} \right\rceil + 3\beta$ binary bits are generated for recording recovery information of one usable vertex. By concatenating all recovery information of each usable vertex, the whole recovery information binary sequence $\mathbb{R}$ is obtained.

A simple example of our proposed scheme combining MGP and CPP is illustrated in Figure 2. Assuming there are a dozen vertices indexed from 1 to 12, the MGP technique is initially employed to form four groups of three vertices. The first vertex in a group serves as the reference vertex, and the remaining two vertices are embeddable. Using the example in Figure 2, the third vertex (index 3) is connected to three reference vertices: 1, 4, and 7. The distance between the fourth vertex (index 4) and the third vertex is minimal, so the fourth vertex is selected as the closest reference and utilized for predicting the vertex 3 in our proposed CPP technique. As the reference vertices of the third vertex remain constant (1, 4, and 7), the CPP only needs 2 bits "01" to record the index of the closest reference. Consequently, the embeddable vertex 3 can be completely recovered using reference vertex 4 after recording the distance between vertices 3 and 4.

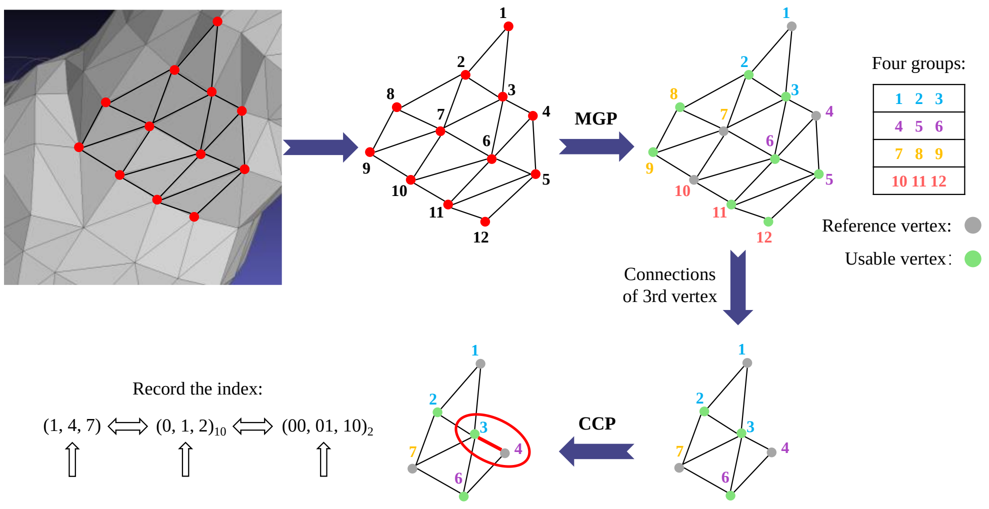

Figure 2. An example of our proposed MGP and CPP.

2.1.4. 3D Mesh Model Encryption and Uploading

The last procedure on the content owner side is to encrypt the 3D mesh model. First, the three-dimensional coordinate of each vertex needs to be converted into the binary system, and the length of the binary bits is judged by:

$$\delta = \begin{cases} 8, & 1 \leq \sigma \leq 2, \\ 16, & 3 \leq \sigma \leq 4. \\ 32, & 5 \leq \sigma \leq 9 \\ 64, & 10 \leq \sigma \leq 32 \end{cases} \tag{9}$$

As each coordinate of the OFF format 3D mesh model only has 6 valid bits, uniformly, we convert coordinates into the 32 bits binary system and preserve 5 valid bits in the experimental results. Then, for each reference vertex and unusable vertex, the model encryption key K_m with random binary bits is used to encrypt three-dimensional coordinates by excursive or operation. According to the descriptions in Sections 2.1.2 and 2.1.3, reference vertices and unusable vertices are predefined, and no additional records are necessary. Next, all binary bits of usable vertices can be vacated for embedding recovery information $\mathbb{R}$. Before embedding, a recovery information encryption key K_r is generated to encrypt $\mathbb{R}$ by excursive or operation. When embedding the encrypted recovery information $\overline{\mathbb{R}}$, the first usable vertex is used to record the length of $\overline{\mathbb{R}}$, and then replace all bits of following usable vertices' three-dimensional coordinates. As the distance between closest pair vertices is short, β is far less than 32 bits, so the usable vertex can be efficiently compressed. Therefore, after embedding $\overline{\mathbb{R}}$, the rest bits of the usable vertices' coordinates can be fully vacated for additional data embedding on the cloud server side, but these bits should be replaced by random bits to ensure the security. Finally, the encrypted 3D mesh model $\mathbb{M}$ can be obtained and uploaded to the cloud.

2.2. *On Cloud Server Side*

To effectively manage the saved encrypted 3D mesh models, the cloud server usually embeds some additional data, such as watermarks or timestamps, into these models. When embedding additional data, they can acquire the length of the recovery information from the first usable vertex. To ensure the security of the additional data, a data hiding key K_h is used to encrypt additional data before embedding. Then, except for the recovery information, all bits of the usable vertices' coordinates can be totally replaced by the encrypted additional data. The embedding rate per vertex (bpv: bits-per-vertex) is the number of bits that can be embedded in a vertex in an encrypted 3D mesh model, and it can be calculated by:

$$\gamma = \frac{32 \times 3 \times (n_u - 1) - \left|\overline{\mathbb{R}}\right|}{n_V}, \tag{10}$$

where n_u denotes the number of the usable vertices and $\left|\overline{\mathbb{R}}\right|$ is the length of the recovery information. After additional data embedding, the encrypted 3D model with additional data is saved in the cloud.

2.3. On Third-Party Side

When an authorized third-party downloads the encrypted 3D model, with different keys, the embedded data can be extracted or the original 3D mesh mode can be recovered without any losses.

2.3.1. Data Extraction

If the third-party has the data hiding key K_h, the embedded additional data can be extracted. As the embedded additional data are located behind the recovery information, the length of the recovery information should be extracted from the first usable vertex to determine the breakpoint between the recovery information and additional data. Next, except for the recovery information, all bits of the usable vertices' coordinates can be extracted. Then, the additional data can be obtained by decrypting these bits using the data hiding key.

D Mesh Model Recovery

If the third-party has the model encryption key K_m and the recovery information encryption key K_r, the original 3D mesh model can be recovered. The detailed steps are as follows.

Step 1: Extract the length of the recovery information from the first usable vertex to determine the recovery information part.

Step 2: Decrypt the reference vertices using the model encryption key K_m, and convert the coordinates into decimal.

Step 3: Extract all bits of the usable vertices' coordinates from the recovery information, and decrypt them using the recovery information encryption key K_r.

Step 4: Recover each usable vertex using the following steps.

Step 4.1: According to the number n_r of the reference vertices which are connected with the current usable vertex, extract $\lceil log_2^{n_r} \rceil$ bits to confirm its closest reference vertex.

Step 4.2: Extract θ bits to obtain the value of β.

Step 4.3: Extract 3β bits to obtain the binary three-dimensional distance between the current vertex and its closest reference vertex, and convert it into decimal to obtain $d^c_{(x,\,y,\,z)} : \left(d^c_x, d^c_y, d^c_z\right)$. According to the coordinate of the closest reference vertex, the current original usable vertex can be recovered into the integer by Equation (8), and can be further converted into a floating coordinate by Equation (4).

Step 5: Process all usable vertices according to **Step 4** to recover the original 3D mesh model.

3. Experimental Results

To evaluate the performance of our proposed scheme, a famous 3D mesh model dataset, Princeton Shape Retrieval and Analysis Group (PSR&AG) [34], was used in this paper. The four models of the dataset, dog, face, spaceship, and car, were analyzed as special cases, shown in Figure 3. The 3D model processing software utilized in this paper is MATLAB R2023b, and the software for 3D model visualization is MeshLab v2022. 02. The encrypted data streams employed are generated using MATLAB's built-in pseudo-random functions. We first evaluated the embedding rate of four test models. The embedding rate mentioned in Equation (10) denotes that the capability of each vertex to carry how many additional bits. The security of the encrypted 3D model and the embedded encrypted 3D model were measured qualitatively and quantitatively. Finally, the proposed scheme was compared with other state-of-the-art schemes to demonstrate the superiority.

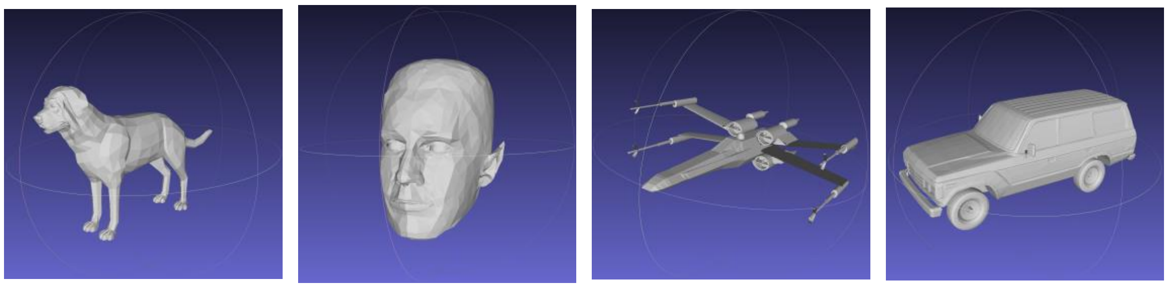
(**a**) Dog (**b**) Face (**c**) Spaceship (**d**) Car

Figure 3. Four test 3D mesh models.

3.1. Performance of Our Proposed Scheme

In this paper, we only considered the compression of the 3D models by preserving 5 valid bits of each coordinate, i.e., using 32 binary bits to represent one x, y, or z coordinate, and 96 binary bits for a vertex. The compression rate only changes the value, but does not change the trend of the embedding rate. We first used the test 3D mesh model "dog" to visually display some processing results of our proposed scheme, as shown in Figure 4. Then, we set every two vertices into one group to test the performance as listed in Table 2. For each model, half of the vertices were embeddable, resulting in the total number of embeddable bits being equal to twice the number of embeddable vertices. As the recovery information is used for usable vertices recovery, $\left|\overline{\mathbb{R}}\right|$ bits should be used for recovery information embedding. In addition to the first usable vertex, used for $\left|\overline{\mathbb{R}}\right|$ recording, the remaining bits can be totally vacated for additional data embedding as listed in the second-to-last column. The embedding rate of each model in the last column can be obtained by dividing the vacated bits by the number of vertices. Therefore, the embedding rate is directly proportional to the vacated bits and inversely proportional to the number of vertices.

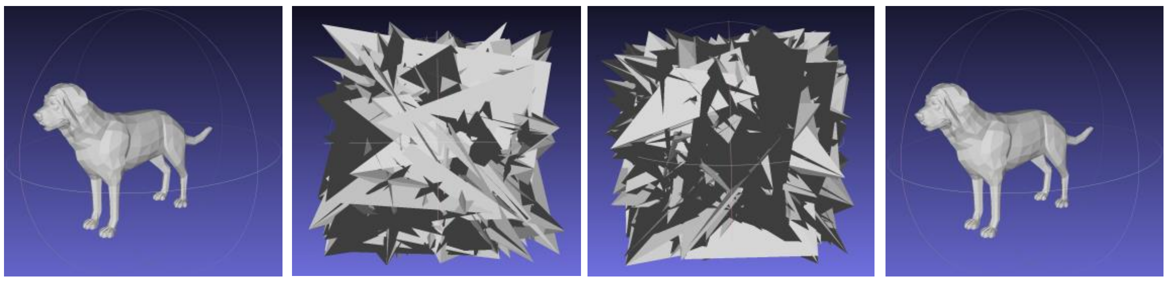
Original Encrypted Embedded Recovered

Figure 4. Visual results of dog in different phases.

Table 2. Relevant Data of Four Test Models When $\alpha = 2$.

3D Models	Number of Faces	Number of Vertices	Total Embeddable Bits	$\left\|\overline{\mathbb{R}}\right\|$	Vacated Bits	Embedding Rate γ (bpv)
Dog	3230	1618	77,664	32,382	45,186	27.93
Face	3604	1884	90,432	39,735	50,601	26.86
Spaceship	9528	5986	287,328	101,049	186,183	31.10
Car	11,230	5911	283,728	169,176	114,456	19.36

.

As the embedding rate was mainly affected by the number of vertices in each group of our proposed MGP technique, we further tested the embedding rates when setting different values of α. As shown in Figure 5, except for the model dog, the remaining models achieved the best embedding rate when setting $\alpha = 3$. The underlying reason is the reference vertices were far more than required when $\alpha = 2$. On the other hand, the reference is insufficient when $\alpha = 4$, as the prediction distance becomes larger, and more bits are needed to record the recovery information. When the number of vertices within the group is too small, it results in excessive vertices being classified as reference vertices. As the coordinates of reference vertices will be retained, this leads to significant spatial wastage. However, when the number of vertices within the group is too large, it results in a considerable distance between usable vertices and reference vertices, causing a decrease in prediction accuracy and, consequently, affecting compression efficiency. Therefore, the proposed MGP can adaptively find the optimal number of vertices in a group for different 3D models to achieve the best embedding rate.

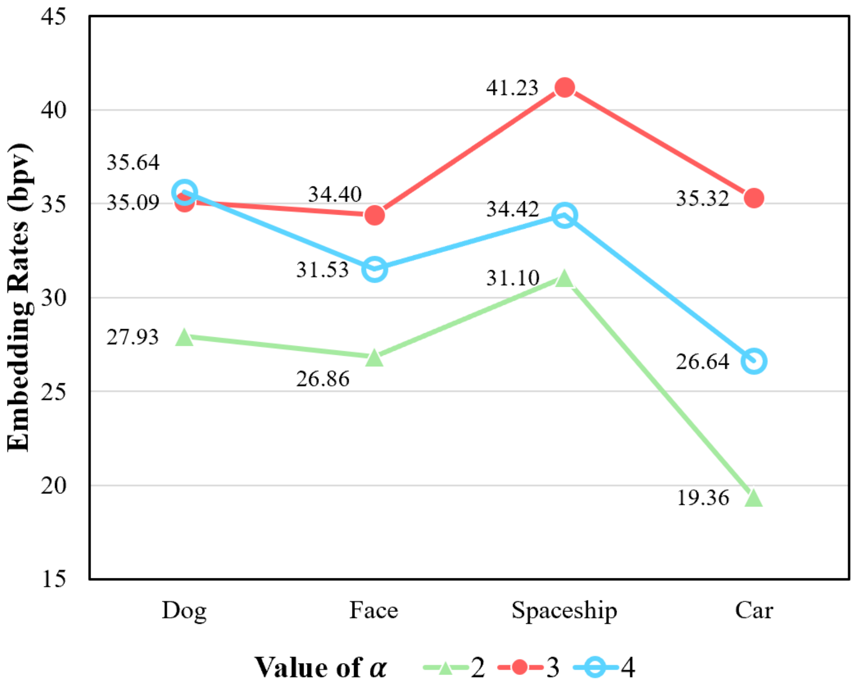

Figure 5. Embedding rates of four test models under different α.

3.2. Embedding Rate Comparisons

In this section, the embedding rate performances of our proposed scheme and four state-of-the-art schemes, Shah et al. [27], Rensburg et al. [28], Tsai [29], Xu et al. [30], Yin et al. [31], and Lyu et al. [32], are compared. As scheme [32] also gathers every two vertices into one group, we first compare the efficiency of our proposed closest pair prediction and the relevant techniques in [32]. Table 3 lists the embedding rates of two schemes when $\alpha = 2$.

Table 3. Embedding Rates of Two Schemes When $\alpha = 2$.

3D Models	[32]	The Proposed Scheme	Increment
Dog	24.22	27.93	3.71
Face	23.05	26.86	3.81
Spaceship	28.99	31.10	2.11
Car	16.06	19.36	3.30

Obviously, the embedding rates of our proposed scheme are higher than those in [32], and the average increment achieves to 3.23 bpv. In [32], the majority voting method is used to predict binary three-dimensional coordinates of each usable vertex by its connected reference vertices. As the reference vertices connected to the currently useable vertex are not necessarily the closest to it, but each reference vertex has the same weight when majority voting, the result is some usable vertices in the edge area have poor prediction accuracy. For the CPP technique of our proposed scheme, we use only the closest reference vertex to predict corresponding usable vertex and only pay a little cost to record which the closest vertex is. Therefore, the prediction accuracy of our proposed scheme is higher and the embedding rates have superiority.

Next, because our proposed multi-group partition technique will further improve the embedding rate, we adaptively chose a different value of α for different 3D mesh models to achieve best performance for the rest comparisons. As shown in Figure 6, four test models were first used for comparison. In the first two schemes, homomorphic encryption was adopted, ensuring constant embedding rates. The performances of rest schemes were affected by the complexity of each 3D mesh models. The MSB prediction method was used in [30–32]. The scheme [30] only vacated the redundancy of the first MSB in each vertex, so that the embedding rates were constrained below 3 bpv. The majority voting method was used in [31,32], and only the number of usable vertices was different in the two schemes. However, the majority voting method did not seem to be optimal for three-dimensional coordinate prediction, and the reference vertices seemed superfluous. In our proposed scheme we first improved the prediction method and gathered more vertices into one group to further vacate more spare room. Therefore, the embedding rate of our proposed scheme has evident superiority compared to other state-of-the-art schemes.

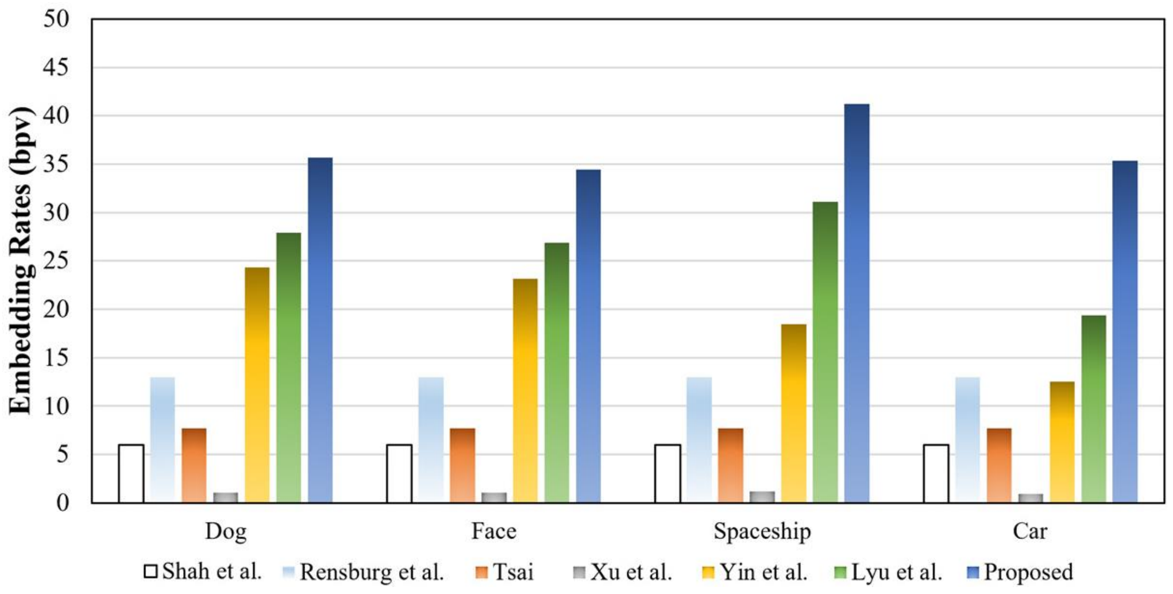

Figure 6. Embedding rate comparison of four test models [27–32].

We also evaluated the embedding rate on the whole PSR&AG dataset to test the practicality of our proposed scheme. As shown in Figure 7, the average embedding rates of our proposed scheme and state-of-the-art schemes are given. The embedding rates of only the proposed scheme exceed 35 bpv, and the increment rises to 10.31 bpv compared to the suboptimal scheme.

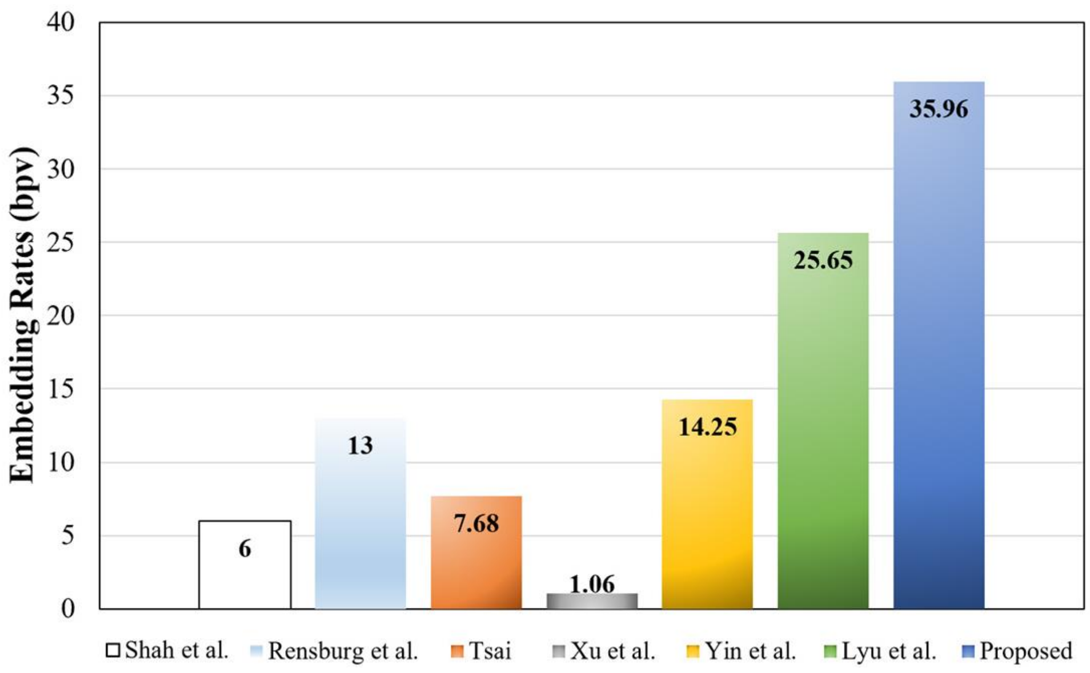

Figure 7. Embedding rate comparison on the dataset [27–32].

3.3. *Reversibility, Security, and Complexity*

We further evaluated the reversibility, security, and complexity of our proposed scheme in this section. The project was run on a Mac mini device with a M2 chip using MATLAB R2023b. In Table 4, there are three Hausdorff distances [35]; the Hausdorff distances[1] represent the distances between the original 3D model and the recovered 3D model. It can be observed that the values are all equal to 0, which indicates the original 3D mesh models can be totally recovered in our scheme. The Hausdorff distances[2] and Hausdorff distances[3] were calculated using the original 3D model with the encrypted 3D model and the embedded 3D model, respectively. These distances were too large, so the shapes of original 3D models could be well protected to ensure security. In addition, the distances under different α values show no noticeable difference, which means that the security is not affected by the number of vertices in a group. The run time refers to the processing time on the content owner side. It can be seen that the run times of different models are mainly affected by the number of vertices, and the 3D models can be processed in a short time to adapt the real-time application. Additionally, as α increases and more usable vertices need to be processed, it results in a linear growth in processing time.

Table 4. Relevant Data of Four Test Models.

3D Models	Number of Faces	Number of Vertices	α	Hausdorff Distance[1]	Hausdorff Distance[2]	Hausdorff Distance[3]	Run Time (s)
Dog	3230	1618	2	0	18.57	18.67	2.03
			3	0	18.45	18.73	3.50
			4	0	18.61	18.52	4.69
Face	3604	1884	2	0	15.98	15.95	2.14
			3	0	15.81	16.03	3.71
			4	0	15.92	15.85	5.26
Spaceship	9528	5986	2	0	36.66	36.90	5.86
			3	0	36.85	36.71	11.20
			4	0	36.97	36.82	16.64
Car	11,230	5911	2	0	33.71	33.57	6.54
			3	0	33.77	33.69	12.51
			4	0	33.58	33.62	18.48

3.4. Over Performance Comparison

We comprehensively compare the performance of our proposed scheme with seven latest state-of-the-art schemes in Table 5. It can be observed that most schemes other than [29] use bit-XOR encryption, and all schemes achieve model recovery without loss and data extraction. For schemes that use XOR encryption, the model is processed vertex by vertex, resulting in time complexities of O(n). Our proposed scheme has a significant improvement in embedding rate compared to others, indicating that our scheme has advantages to extend the privacy and copyright protection of 3D models to a wider range of applications.

Table 5. Comparison of Over Performance.

	Shah et al. [27]	**Rensburg et al. [28]**	**Tsai [29]**	**Xu et al. [30]**	**Yin et al. [31]**	**Lyu et al. [32]**	**Qu et al. [33]**	**Proposed**
Encryption technique	Homomorphic	Homomorphic	XOR	XOR	XOR	XOR	RCXOR	XOR
Error-free in model recovery	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Error-free in data extraction	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Time complexity	O(f(n))	O(f(n))	O(n)	O(n)	O(n)	O(n)	O(n^2)	O(n)
Embedding capacity	Low (<10 bpv)	Medium (10–20 bpv)	Low (<10 bpv)	Low (<10 bpv)	Medium (10–20 bpv)	High (20–30 bpv)	High (20–30 bpv)	Very High (>30 bpv)

4. Conclusions

In this paper, we propose a novel reversible data hiding scheme in encrypted 3D mesh models in cloud applications. Two efficient techniques, MGP and CPP, are proposed to improve the embedding rates. After preprocessing, the content owner first gathers vertices into groups and classifies them into reference vertices and embeddable vertices using MGP. Then, the CPP technique is used to predict embeddable vertices by the connected reference vertices. As more vertices are embeddable and the embeddable vertices can be accurately predicted, more spare room can be vacated for additional data embedding on the cloud server side. On the authorized third-party side, the embedded data can be totally extracted and the original 3D mesh model can be recovered exactly according to different keys, separately. Experimental results demonstrate that the proposed scheme achieves the best embedding rate and prominent improvement compared to state-of-the-art schemes. In the future, we will explore the efficient compression of vertices using deep learning techniques to further improve performance.

Author Contributions: Methodology, X.W.; Software, X.W. and J.-C.L.; Validation, J.-C.L.; Formal analysis, J.-C.L.; Writing—original draft, X.W.; Writing—review & editing, X.W. and C.-C.C. (Ching-Chun Chang); Supervision, C.-C.C. (Ching-Chun Chang) and C.-C.C. (Chin-Chen Chang); Project administration, C.-C.C. (Ching-Chun Chang). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data can be shared up on request.

Conflicts of Interest: The authors declared that they have no conflicts of interest with this work.

References

1. Davis, R. The data encryption standard in perspective. *IEEE Commun. Soc. Mag.* **1978**, *16*, 5–9. [CrossRef]
2. Ni, Z.; Shi, Y.-Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
3. Celik, M.U.; Sharma, G.; Tekalp, A.M.; Saber, E. Reversible data hiding. In Proceedings of the International conference on image processing, Rochester, NY, USA, 22–25 September 2002; Volume 2, p. II.
4. Peng, F.; Li, X.; Yang, B. Improved PVO-based reversible data hiding. *Digit. Signal Process.* **2014**, *25*, 255–265. [CrossRef]
5. Zhang, X. Reversible data hiding in encrypted image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [CrossRef]

6. Qin, C.; Zhang, X. Effective reversible data hiding in encrypted image with privacy protection for image content. *J. Vis. Commun. Image Represent.* **2015**, *31*, 154–164. [CrossRef]
7. Hong, W.; Chen, T.-S.; Wu, H.-Y. An improved reversible data hiding in encrypted images using side match. *IEEE Signal Process. Lett.* **2012**, *19*, 199–202. [CrossRef]
8. Zhang, X. Separable reversible data hiding in encrypted image. *IEEE Trans. Inf. Forensics Secur.* **2011**, *7*, 826–832. [CrossRef]
9. Qian, Z.; Zhang, X. Reversible data hiding in encrypted images with distributed source encoding. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *26*, 636–646. [CrossRef]
10. Wu, X.; Sun, W. High-capacity reversible data hiding in encrypted images by prediction error. *Signal Process.* **2014**, *104*, 387–400. [CrossRef]
11. Wang, X.; Chang, C.-C.; Lin, C.-C.; Chang, C.-C. Reversal of pixel rotation: A reversible data hiding system towards cybersecurity in encrypted images. *J. Vis. Commun. Image Represent.* **2022**, *82*, 103421. [CrossRef]
12. Xiong, L.; Dong, D. Reversible data hiding in encrypted images with somewhat homomorphic encryption based on sorting block-level prediction-error expansion. *J. Inf. Secur. Appl.* **2019**, *47*, 78–85. [CrossRef]
13. 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]
14. Ke, Y.; Zhang, M.-Q.; Liu, J.; Su, T.-T.; Yang, X.-Y. Fully homomorphic encryption encapsulated difference expansion for reversible data hiding in encrypted domain. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 2353–2365. [CrossRef]
15. Huang, F.; Huang, J.; Shi, Y.-Q. New framework for reversible data hiding in encrypted domain. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 2777–2789. [CrossRef]
16. Ge, H.; Chen, Y.; Qian, Z.; Wang, J. A high capacity multi-level approach for reversible data hiding in encrypted images. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *29*, 2285–2295. [CrossRef]
17. Liu, Z.-L.; Pun, C.-M. Reversible data-hiding in encrypted images by redundant space transfer. *Inf. Sci.* **2018**, *433*, 188–203. [CrossRef]
18. Wang, X.; Chang, C.-C.; Lin, C.-C. Reversible data hiding in encrypted images with block-based adaptive MSB encoding. *Inf. Sci.* **2021**, *567*, 375–394. [CrossRef]
19. Ma, K.; Zhang, W.; Zhao, X.; Yu, N.; Li, F. Reversible data hiding in encrypted images by reserving room before encryption. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 553–562. [CrossRef]
20. Cao, X.; Du, L.; Wei, X.; Meng, D.; Guo, X. High capacity reversible data hiding in encrypted images by patch-level sparse representation. *IEEE Trans. Cybern.* **2015**, *46*, 1132–1143. [CrossRef]
21. Yi, S.; Zhou, Y. Separable and reversible data hiding in encrypted images using parametric binary tree labeling. *IEEE Trans. Multimed.* **2018**, *21*, 51–64. [CrossRef]
22. Yin, Z.; Xiang, Y.; Zhang, X. Reversible data hiding in encrypted images based on multi-MSB prediction and Huffman coding. *IEEE Trans. Multimed.* **2019**, *22*, 874–884. [CrossRef]
23. Mohammadi, A.; Nakhkash, M.; Akhaee, M.A. A high-capacity reversible data hiding in encrypted images employing local difference predictor. *IEEE Trans. Circuits Syst. Video Technol.* **2020**, *30*, 2366–2376. [CrossRef]
24. Puteaux, P.; Puech, W. A recursive reversible data hiding in encrypted images method with a very high payload. *IEEE Trans. Multimed.* **2020**, *23*, 636–650. [CrossRef]
25. Wang, X.; Chang, C.-C.; Lin, C.-C.; Chang, C.-C. On the multi-level embedding of crypto-image reversible data hiding. *J. Vis. Commun. Image Represent.* **2022**, *87*, 103556. [CrossRef]
26. Jiang, R.; Zhou, H.; Zhang, W.; Yu, N. Reversible data hiding in encrypted three-dimensional mesh models. *IEEE Trans. Multimed.* **2017**, *20*, 55–67. [CrossRef]
27. Shah, M.; Zhang, W.; Hu, H.; Zhou, H.; Mahmood, T. Homomorphic encryption-based reversible data hiding for 3D mesh models. *Arab. J. Sci. Eng.* **2018**, *43*, 8145–8157. [CrossRef]
28. Van Rensburg, B.J.; Puteaux, P.; Puech, W.; Pedeboy, J.-P. Homomorphic two tier reversible data hiding in encrypted 3d objects. In Proceedings of the 2021 IEEE International Conference on Image Processing (ICIP), Virtual, 19–22 September 2021; pp. 3068–3072.
29. Tsai, Y.-Y. Separable reversible data hiding for encrypted three-dimensional models based on spatial subdivision and space encoding. *IEEE Trans. Multimed.* **2020**, *23*, 2286–2296. [CrossRef]
30. Xu, N.; Tang, J.; Luo, B.; Yin, Z. Separable reversible data hiding based on integer mapping and MSB prediction for encrypted 3D mesh models. *Cogn. Comput.* **2022**, *14*, 1172–1181. [CrossRef]
31. Yin, Z.; Xu, N.; Wang, F.; Cheng, L.; Luo, B. Separable reversible data hiding based on integer mapping and multi-MSB prediction for encrypted 3D mesh models. In Proceedings of the Pattern Recognition and Computer Vision: 4th Chinese Conference, PRCV 2021, Beijing, China, 29 October–1 November 2021; Proceedings, Part II 4. pp. 336–348.
32. Lyu, W.-L.; Cheng, L.; Yin, Z. High-capacity reversible data hiding in encrypted 3D mesh models based on multi-MSB prediction. *Signal Process.* **2022**, *201*, 108686. [CrossRef]
33. Qu, L.; Lu, H.; Chen, P.; Amirpour, H.; Timmerer, C. Ring Co-XOR encryption based reversible data hiding for 3D mesh model. *Signal Process.* **2024**, *217*, 109357. [CrossRef]

34. Shilane, P.; Min, P.; Kazhdan, M.; Funkhouser, T. The Princeton Shape Benchmark. In Proceedings of the Proceedings Shape Modeling Applications, Genova, Italy, 7–9 June 2004; pp. 167–178.
35. Dubuisson, M.-P.; Jain, A.K. A modified Hausdorff distance for object matching. In Proceedings of the 12th International Conference on Pattern Recognition, Jerusalem, Israel, 9–13 October 1994; Volume 1, pp. 566–568.