This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia

Reversible Data Hiding in Encrypted 3D Mesh Models

Ruiqi Jiang, Hang Zhou, Weiming Zhang and Nenghai Yu

***Abstract*—Reversible data hiding in encrypted domain (RDH-ED) has greatly attracted researchers as the original content can be losslessly reconstructed after the embedded data are extracted, while the content owner's privacy remains protected. Most of the existing RDH-ED algorithms are designed for grayscale/color images, which cannot be directly applied to other carriers, such as three-dimensional (3D) meshes. With the rapid development of 3D related applications, 3D models have been widely used on the Internet, which motivated us to design a reliable RDH-ED scheme for 3D meshes. The proposed method maps decimals of the vertex coordinates into integers first, so that a bit-stream encryption technique can be executed. With a data-hiding key, several least-significant bits (LSBs) are operated to embed data. By using the encryption key, a receiver can roughly reconstruct the content of the mesh. According to the data-hiding key, with the aid of spatial correlation in natural mesh models, the embedded data can be successfully extracted and the original mesh can be perfectly recovered. Experiments show that the proposed method has a high data-embedding payload, maintains high values of the decrypted meshes, and has low computational complexity.**

***Index terms*—**3D mesh models, reversible data hiding, curvature, vertex partition.

I. Introduction

TWO classes of digital watermarking have been developed to protect copyright ownership and reduce counterfeiting of digital multimedia, which are in a way not immediately discernible but hard to reproduce. [1] Robust watermarking is designed to withstand attacks on a digital carrier, while fragile watermarking has a different application scenario that is designed to implement integrity authentication and annotation of confidential data [2]. Hereinto, reversible data hiding (RDH) is a branch of fragile watermarking, by which the original object can be losslessly recovered after the embedded message is extracted. It is widely used in the field of medical/military imagery, integrity verification [3], media asset management [4], copyright protection, and legal forensic. In addition, RDH cannot withstand geometric attack, noise attack, signal processing attack, and even minute modification such as modifying least-significant bits (LSBs). Steganography aspires to undetectability, which is used for covert communication. Properties desired in steganography, robust watermarking, and RDH are shown in Table I.

TABLE I
Properties Desired in Steganography, Robust Watermarking, and RDH

	Undetectability	Robustness	Reversibility
Steganography	√		
Robust watermarking		√	
RDH			√

In recent years, based on needs of cloud computing and various privacy-preserving applications, outsourced storage in the Cloud has become an increasingly popular service and has gained increasing attention, especially for multimedia files, such as images, videos, audio files, and three-dimensional (3D) mesh models, which require large volumes of storage space. By embedding a few additional data such as file category and notation information into the digital content, the Cloud server can manage the outsourced digital files and employ these data to identify the ownership [5] or verify the integrity of files. Obviously, the Cloud service provider has no right to introduce permanent distortion during data embedding into the outsourced files. Therefore, RDH technology is needed.

To this end, many RDH schemes have been proposed in the past. Essentially, all these methods can be viewed as a process of semantic lossless compression in which some spare space is vacated for embedding extra data by losslessly compressing, for example, an image [6]. Prediction error (PE) is widely employed to represent the the residuals of images, and can be easily compressed with small entropy. Almost all recent RDH methods first generate PEs as the host sequence [7]- [9], and then reversibly embed the data into the host sequence by modifying its histogram with methods such as histogram shifting (HS) [10] or into vacancies created by the expansion of difference values, such as difference expansion (DE) [11], [12]. Generating RDH codes according to the theoretical expressions of RDH [13] improves the embedding capacity. Zhang *et al.* proposed the optimal histogram modification algorithm [14], [15] for RDH by estimating the optimal modification probability [16], [17].

On the other hand, since the content of digital media can be easily copied, manipulated, and distributed in the digitized world nowadays, it is challenging to protect the privacy of uploaded content for outsourced Cloud storage. A recent example is that many private photos of Hollywood actresses were leaked from iCloud [21]. Despite the availability of RDH in managing the outsourced files, digital media content is vulnerable to attack, and thus encryption techniques are utilized for privacy protection. Along with increasing atten-

This work was supported in part by the Natural Science Foundation of China under Grant 61572452 and U1636201.

R. Jiang, H. Zhou, W. Zhang, and N. Yu are affiliated with CAS Key Laboratory of Electromagnetic Space Information, University of Science and Technology of China, Hefei 230026, China. (e-mail: jrq123@mail.ustc.edu.cn, zh2991@mail.ustc.edu.cn, zhangwm@ustc.edu.cn, ynh@ustc.edu.cn).

Corresponding author: Weiming Zhang.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia

tion paid to signal processing over encrypted domains, the investigation of embedding additional data in the encrypted domain in a reversible fashion has been triggered by such events, i.e., RDH in encrypted domain (RDH-ED). With a distinct consideration of compression, the existing RDH-ED methods can be classified into two categories [18]: "reserving room before encryption (RRBE)" and "vacating room after encryption (VRAE)".

The RRBE framework empties out room by using the RDH method in the plain images. After that, the image is encrypted and outsourced to the Cloud, and the Cloud server can freely embed data into the reserved room of the encrypted image. In the first method under the RRBE framework [18], the embedding room is created in digital images by embedding the least-significant bits (LSBs) of certain pixels into other pixels using a traditional RDH method, and then the image is encrypted. Thus, the positions of these LSBs in the encrypted image can be used to embed data. The method in [18] implies that the purpose of RDH-EI (reversible data hiding in encrypted image) can also be realized by RDH for plain-text images. Following this idea, Zhang *et al.* [19] reserved room in images by generating PEs and modifying the histogram of the PEs, which is the most popular technique used in RDH for plain-text images. Cao *et al.* [20] improved the methods of [18], [19] by patch-level sparse representation, which can yield PEs with smaller entropy and thus result in a large vacated room. Recently, Zhang *et al.* [21] proposed a novel framework for RDH-ED based on reversible image transformation, where the framework allows the user to transform the content of original image into the content of another target image of the same size. The transformed image, which looks like the target image, is used as the "encrypted image".

In VRAE methods, the original signal is encrypted directly by the content owner, and the Cloud server embeds the data by modifying some bits of the encrypted signal [22], [23]. Compression of encrypted data can be formulated as source coding with side information at the decoder [22], which is usually the correlations of plain-texts that are exploited for decompression by the decoder. The first method was proposed by Zhang for encrypted images [24], in which the Cloud server divides the encrypted images into blocks and embeds one bit into each block by flipping three LSBs of half the pixels in the block. On the receiver's side, after decrypting the marked encrypted image, the receiver flips the three LSBs of the pixels to form a new block and uses a function to estimate the texture complexity of each block. Owing to spatial correlation in natural images, the original block is presumed to be much smoother than the interfered block. Hong *et al.* [25] ameliorated Zhang's method at the decoder side by further exploiting the spatial correlation using a different estimation equation and side-match technique. Recently, in [26], a two-class support vector machine (SVM) classifier is adopted to distinguish encrypted and non-encrypted image blocks rather than the judging functions used in [24] and [25]. To separate the data extraction from image decryption, Zhang [26] emptied out space for data embedding by directly using the typical manner of cipher-text compression; that is, compressing the encrypted pixels in a lossless manner by using the syndromes of the parity-check matrix of channel codes. Qian *et al.* [28] improved the method of [27] by adopting a low-density parity check (LDPC) [29] code-based Slepian-Wolf encoder, which is also one of the most efficient methods for cipher-text compression.

The research on RDH for digital images has been ongoing for many years, and algorithm performance has already approached the theoretical optimal solution, but for other digital multimedia, such as audio, video, and especially for 3D models, it is still in the beginning stage. With the rapid development of 3D-related applications and their intrinsic large capacity, 3D models (mesh models or point cloud models) have been widely used on the Internet, which has also encouraged researchers to adopt them as the cover media for RDH. The former RDH for 3D models can be classified into three domains: the spatial domain, compressed domain, and transform domain. Spatial-domain-based RDH [30]- [32] modifies the vertex coordinates rather than the connectivity data, which has low computational complexity. Compressed-domain-based RDH [33], [34] has been proposed in the predictive vector quantization (VQ) domain to embed data into a mesh stream. Transform-domain-based RDH [35], [36] transforms the original models into a certain transform domain and then embeds data into the transform coefficients. In addition, it is noteworthy that many 3D steganography paradigms [44], [45], [46], [47] have been proposed.

Although the aforementioned RDH techniques in 3D models have developed over the years, research on encrypted 3D models has not yet begun. To our best of knowledge, this is the first work on reversible data hiding in encrypted 3D meshes. In this paper, we propose a RDH technique in encrypted 3D meshes. The proposed method maps decimals of the vertex coordinates into integers first, so that the bit-stream encryption technique can be executed. With a data-hiding key, several LSBs are operated to embed data. By using the encryption key, a receiver can roughly reconstruct the content of the mesh. According to the data-hiding key, with the aid of spatial correlation in natural mesh models, the embedded data can be successfully extracted and the original mesh can be perfectly recovered.

The potential application of RDH in encrypted 3D mesh is outsourced storage with preserved privacy. Compared to images, 3D meshes have larger sizes for outsourced storage. At present, RDH on encrypted 3D meshes adopts non-separable framework. The reversibility of encrypted 3D mesh enables Cloud server to manage the mesh by embedding some information (including the identity of the meshes owner, time stamps and other stored records) without leaking mesh privacy or introducing any permanent distortion on the mesh. Receiver can identify the owner and get these records when decrypting meshes, which can be used to trace the source and history of the data. In the future, we will extend the method to the separable case, where Cloud server could extract embedded data without decrypting the meshes. In this way, Cloud server could embed management information including the identity of the meshes owner and time stamps, to authenticate the integrity of encrypted meshes.

The rest of this paper is organized as follows. In Section

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia

II, we present the procedure of mesh preprocessing, mesh encryption, data embedding, mesh decryption, data extraction, and mesh recovery. In Section III, we present experimental results. The paper concludes with a discussion in Section IV.

II. Proposed Method

The proposed scheme comprises mesh preprocessing, mesh encryption, data embedding, mesh decryption, and data-extraction/mesh-recovery phases. The detailed procedures are as follows.

A. Preprocessing

Many mesh file formats (OFF, PLY, OBJ, VRML, X3D, etc.) are based on an indexed data structure. Triangle meshes are composed of two components: vertex data and connectivity (face) data. Vertex data include the positional coordinates of the vertices and, optionally, photometric information such as normal vectors or colors. The connectivity data supply the topological information that specifies which vertices belong to each triangle.

Let $\{\mathbf{v}_i\}_{i=0}^{N}$ represent the sequence of vertices encountered as a mesh is being traversed, where $\mathbf{v}_i = (v_{i,x}, v_{i,y}, v_{i,z})$ and N is the number of vertices. Note that each coordinate $v_{i,j} < 1, j \in \{x, y, z\}$. Uncompressed representations of mesh models typically specify each vertex coordinate as a 32-bit floating-point number, and the number of each vertex coordinate's significant digit is 6. For example, $\mathbf{v}_1 = (-0.202018, -0.0740184, 0.288808)$. We take a local region of a "cow" model for illustration in Fig. 1, and the corresponding file format is shown in Table II.

In 1995, Deering [37] advised that most applications do not require this level of precision and presented a lossy vertex data compression scheme. Positions are first normalized within an axis-aligned bounding box. The coordinates are then uniformly quantized to k bits of precision so they can be represented as integers between 0 and $2^k - 1$. Empirically, $k \in [1, 33]$. For each coordinate $v_{i,j}$ of a vertex, it is normalized to $v'_{i,j}$,

$$v'_{i,j} = \lfloor v_{i,j} \times 10^k \rfloor, \qquad j = x, y, z. \tag{1}$$

Therefore, we implement encryption, decryption, data hiding, and message extraction based on $v'_{i,j}$.

To generate processed meshes, we reversibly transform the processed integral coordinates of vertices $\bar{v}'_{i,j}$ to decimal coordinates $\bar{v}_{i,j}$,

$$\bar{v}_{i,j} = \bar{v}'_{i,j}/10^k, \qquad j = x, y, z. \tag{2}$$

Computer processors are often designed to process a data group into words of a given length of bits (8, 16, 32, 64 bits, etc.). When we manipulate integral preprocessed coordinates, the bit length of each coordinate defines how many memory locations can be independently addressed by the processor. It is clear that k affects the bit length of the coordinate l (bits), which is summarized by

$$l = \begin{cases} 8, & 1 \le k \le 2 \\ 16, & 3 \le k \le 4 \\ 32, & 5 \le k \le 9 \\ 64, & 10 \le k \le 33. \end{cases} \tag{3}$$

The value of k influences the time cost of each phase of the method, including encryption, data hiding, data extraction, decryption, and mesh recovery. Furthermore, the value of k determines whether a mesh can be losslessly recovered to its original version, and the threshold is denoted k_{th}. Both experimental analyses are implemented in Section III.

B. Encryption

We assume that we have preprocessed vertices, and denote the bits of a component of a vertex as $b_{i,j,0}, b_{i,j,1}, ..., b_{i,j,k}$,where $1 \le i \le N$ and $j \in \{x, y, z\}$. This implies that

$$b_{i,j,u} = \lfloor v'_{i,j}/2^u \rfloor \bmod 2, \qquad u = 0, 1, ..., k. \tag{4}$$

The content owner then chooses an encryption key K_1 to generate pseudo-random bits using a stream cipher function (e.g., RC4 or SEAL), and encrypts the bitstream of the preprocessed mesh model by

$$e_{i,j,u} = b_{i,j,u} \oplus k_{i,j,u}, \qquad u = 0, 1, ..., k, \tag{5}$$

where $k_{i,j,u}$ are the key stream bits, $e_{i,j,u}$ are the generated cipher text, and $\oplus$ denotes the exclusive OR.

Accordingly, the encrypted integral mesh model can be constructed by

$$E_{i,j} = \sum_{u=1}^{k} e_{i,j,u} \cdot 2^u, \tag{6}$$

where $E_{i,j}$ are the integral value of coordinates, $1 \le i \le N$ and $j \in \{x, y, z\}$. Note that the stream cipher in Eq. (4) only scrambles coordinate values, but does not shuffle coordinate locations.

C. Data Embedding

With the encrypted data, although a data hider does not know the original mesh content, he can embed an additional message into the mesh by modifying a small proportion of the encrypted data. The selection of mesh parts that are employed to embed messages is based on the following criterion. Since a vertex is contained within several triangles, once the vertex is modified to embed messages, the adjacent vertices should not be modified and are used to recover the central vertex by adjacent correlation at the receiver side. Therefore, for data hiders, they first divide the vertices into two sets: the "embedded" set and the "referenced" set.

1) Partition of Vertices. The "embedded" set is used for embedding data and the "referenced" set is used for recovering the adjacent "embedded" data at the receiver side. The concrete method is described below.

We traverse all the vertices on the connectivity data in ascending order. Let $\{\mathbf{f}_i\}_{i=0}^{M}$ represent the sequence of faces encountered as a mesh is being traversed, where $\mathbf{f}_i = (v_{i,x}, v_{i,y}, v_{i,z})$ and M is the number of faces. Let $\mathbf{f}_n = (v_{n,x}, v_{n,y}, v_{n,z})$ be the next face to be handled, $\mathcal{S}_e$ be the set of current "embedded" vertices, and $\mathcal{S}_n$ be the set of current "referenced" vertices. $\mathcal{S}_e$ and $\mathcal{S}_n$ are initialized with empty sets.

If no vertex in $\mathbf{f}_n$ exists in $\mathcal{S}_e$ or $\mathcal{S}_n$, we choose the first vertex $f_{n,x}$ in $\mathbf{f}_n$ to add to $\mathcal{S}_e$, and choose $f_{n,y}$ and $f_{n,z}$ to add to $\mathcal{S}_n$. As is shown in Fig. 1 and Table II, when we traverse the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia

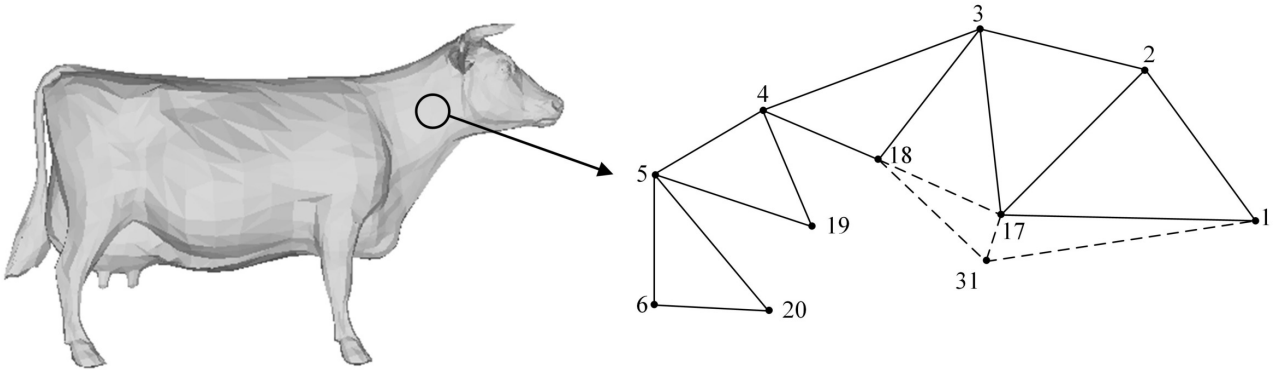


Fig. 1: Illustration of a mesh (cow).

TABLE II
FILE FORMAT FOR FIG. 1

Vertex list				Face information	
Index of vertex	x-axis coordinate	y-axis coordinate	z-axis coordinate	Index of face	Elements in each face
1	$v_{1,x}$	$v_{1,y}$	$v_{1,z}$	1	(17,1,2)
2	$v_{2,x}$	$v_{2,y}$	$v_{2,z}$	2	(3,2,17)
3	$v_{3,x}$	$v_{3,y}$	$v_{3,z}$	3	(4,3,18)
4	$v_{4,x}$	$v_{4,y}$	$v_{4,z}$	4	(5,4,19)
5	$v_{5,x}$	$v_{5,y}$	$v_{5,z}$	5	(6,5,20)
6	$v_{6,x}$	$v_{6,y}$	$v_{6,z}$	...	...
...	...	...	...	16	(31,17,1)
17	$v_{17,x}$	$v_{17,y}$	$v_{17,z}$	17	(18,17,31)
18	$v_{18,x}$	$v_{18,y}$	$v_{18,z}$	...	...
19	$v_{19,x}$	$v_{19,y}$	$v_{19,z}$	241	(17,18,3)
20	$v_{20,x}$	$v_{20,y}$	$v_{20,z}$	...	...
...	...	...	...	...	...
31	$v_{31,x}$	$v_{31,y}$	$v_{31,z}$	...	...
...	...	...	...	...	...

first line in the face, $\mathcal{S}_e = \emptyset \bigcup \{17\} = \{17\}$ and $\mathcal{S}_n = \emptyset \bigcup \{1,2\} = \{1,2\}$. If there exists a vertex $f_{n,j}$ in $\mathbf{f}_n$ that does not appear in $\mathcal{S}_e$ but vertices in $\mathbf{f}_n$ that appear in $\mathcal{S}_e$, then vertex $f_{n,j}$ is added into $\mathcal{S}_n$. For example, as shown in Table II, when we traverse the second line in the face, $\mathcal{S}_e = \{17\} \bigcup \emptyset = \{17\}$ and $\mathcal{S}_n = \{1,2\} \bigcup \{3\} = \{1,2,3\}$. If no vertex in $\mathbf{f}_n$ exists in $\mathcal{S}_e$ and no less than one vertex $f_{n,j}$ exists in $\mathcal{S}_n$, the first vertex in $\mathbf{f}_n$ that has excluded $f_{n,j}$ is added into $\mathcal{S}_e$, and the other vertices in $\mathbf{f}_n$ that have excluded $f_{n,j}$ are added into $\mathcal{S}_n$. For example, as is shown in Table II, when we traverse the third line in the face, $\mathcal{S}_e = \{17\} \bigcup \{4\} = \{17,4\}$ and $\mathcal{S}_n = \{1,2,3\} \bigcup \{18\} = \{1,2,3,18\}$. Finally, $\mathcal{S}_e$ and $\mathcal{S}_n$ are obtained. The capacity of the embedded message is the number of vertices in the "embedded" set.

2) Data Embedding in "Embedded" Set. We take a data-hiding key K_2 to encrypt the embedded data. For each "embedded" vertex in the "embedded" set $\mathcal{S}_e$ in the encrypted integral mesh, if the additional bit b to be embedded is 0, no modification is done to the three coordinates; if the additional bit to be embedded is 1, we modify the m LSBs of the three coordinates $e_{i,j,u}$ to the opposite value:

$$e'_{i,j,u} = e_{i,j,u} \oplus b, \qquad v_{i,j} \in \mathcal{S}_e,\ j \in \{x,y,z\},\ \text{and}\ u = 0,1,...,m-1. \tag{7}$$

The other encrypted data are not changed. Note that m influences the quality of direct decryption of the mesh model and the average accuracy of the extracted message.

Here we use bits per vertex (bpv) to measure the embedding rate C, which is equal to the length of embedded bits dividing the amounts of vertices, and

$$C = \frac{\|\mathcal{S}_e\|}{\|\mathcal{S}_e\| + \|\mathcal{S}_n\|}. \tag{8}$$

Theoretical upper and lower bounds of the capacity are elaborated as follows. We denote the number of connected faces of a vertex by p. The topology of vertices and faces are studied first. To acquire the attribute of connectivity among vertices and faces, we randomly select 10 mesh models and give the distribution of the number of vertices with the corresponding faces' amount p, which is shown in Fig. 2(a). It is a symmetric distribution, where the symmetry axis is $p = 6$, indicating that most vertex units are hexagons. Also, $1 \le p \le 11$.

To analyze the upper bounds of the embedding capacity, Fig. 2(b) is presented as a special case. We denote "embedded" vertices by black spots and "referenced" vertices by white spots hereinafter. Under the rule of data embedding that two adjacent vertices shall be in different colors, a maximum C is equivalent to the largest proportion of black vertices. Therefore, the topology of Fig. 2(b) is the ultimate circumstance that each of the two "referenced" vertices is contained in the maximum $p = 11$ faces, while each of adjacent vertices is only

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia

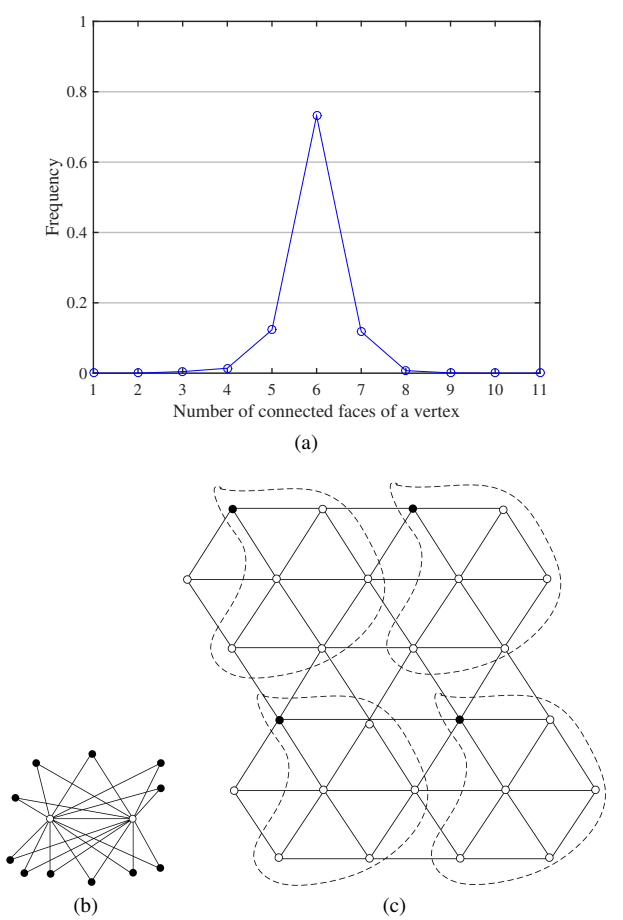


Fig. 2: (a) Distribution of the number of vertices with corresponding faces' amount. (b) Maximum embedding rate with a vertex connecting seven faces. (c) Minimum embedding rate with regular hexagon topological structure.

contained in the $p = 1$ face. Thus, by Eq. (9), the maximum embedding rate is obtained by $C_{maximum} = \frac{11}{11+2} = \frac{11}{13}$.

Fig. 2(c) shows the lower bounds of the embedding rate. A regular polygon with $p = 6$ of each vertex is shown. To enlarge the quantity of "referenced" vertices in a fixed unit, each pair of adjacent "embedded" vertices of a connection is bound to go through two "referenced" vertices, making the embedding rate the lowest. We segment the vertices into units containing one "embedded" vertex and five "referenced" vertices, which are encircled by short-dashed lines, as shown in Fig. 2(c). Therefore, the minimum embedding rate is obtained by $C_{minimum} = \frac{1}{6}$, and finally we obtain

$$\frac{1}{6} \leq C \leq \frac{11}{13}. \tag{9}$$

Since errors may occur, error correction codes (ECCs) are utilized and cause a drop in capacity. Before data hiding, ECCs are used to encode the plain message bits. Assuming that $[n, k]$ codes are used, a total of C_p plain bits can be embedded into the bitstream, where

$$C_p = \lfloor \|\mathcal{S}_e\| \cdot k/n \rfloor. \tag{10}$$

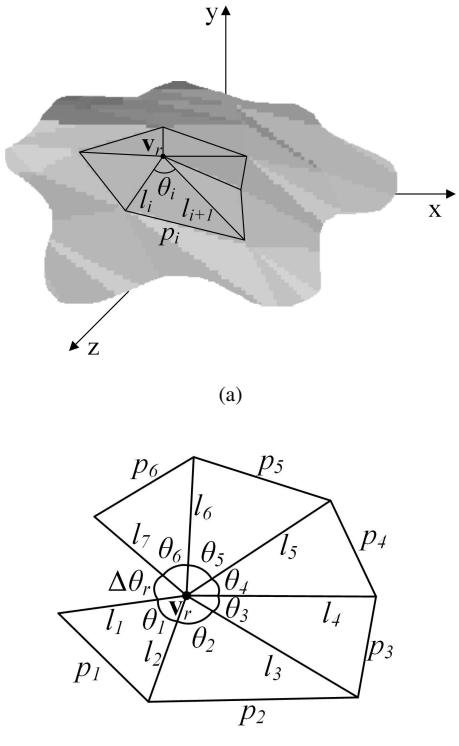


Fig. 3: Structure of a discrete point. (a) Surface piece-wise triangulation in Cartesian coordinates. (b) Angle deficit $\Delta\theta_r$ in a local region.

Therefore, the actual number of bits embedded into the bitstream is

$$C_e = \lfloor \|\mathcal{S}_e\| \cdot k/n \rfloor \cdot n/k. \tag{11}$$

We call C_e the *embedding capacity*, and C_p the *net capacity* that is the capacity of the plain message bits. Then, the plain bits $\mathbf{P} = [P_1, P_2, ..., P_{C_p}]$ are encoded into $\mathbf{T} = [T_1, T_2, ..., T_{C_e}]$:

$$\mathbf{T} = \text{ECC}(\mathbf{P}), \tag{12}$$

where ECC denotes an error correction function. Several ECC algorithms can be used to ensure correct extraction of the secret data, e.g., BCH codes, LDPC codes, and Golay codes.

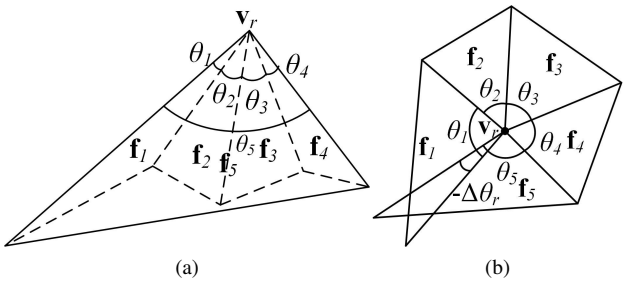
The net embedding rate C_n is the actual embedding rate after ECCs,

$$C_n = \frac{C_p}{\|\mathcal{S}_e\| + \|\mathcal{S}_n\|}. \tag{13}$$

D. Direct Decryption

For an encrypted mesh containing embedded data, a receiver first generates $k_{i,j,u}$ according to the encryption key K_1, and calculates the exclusive OR of the received data and $k_{i,j,u}$ to decrypt the mesh model. We denote the decrypted bits as $b'_{i,j,u}$. Clearly, the original $32 - m$ most-significant bits (MSBs) are retrieved correctly. For a certain vertex, if the embedded bit in the local region including the vertex is zero and the vertex belongs to $\mathcal{S}_e$, the data hiding does not affect any encrypted bits of the vertex in the local region. Thus, the u decrypted LSBs must be the same as the original LSBs, implying that the decrypted integral value of the mesh coordinate is correct.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia



(a) (b)

Fig. 4: (a) Illustration of a non-convex polygon. (b) Angle deficit $\Delta\theta_r$ that appears to be negative.

On the other hand, if the embedded bit in the vertex is 1 and the vertex belongs to $\mathcal{S}_n$, the decrypted LSB is

$$b'_{i,j,u} = \overline{b_{i,j,u}}, \qquad u = 0, 1, ..., m-1, \tag{14}$$

and

$$b'_{i,j,u} + b_{i,j,u} = 1, \qquad u = 0, 1, ..., m-1. \tag{15}$$

In our 3D case, the geometrical distortion of a mesh model including N vertices after adding some noise to the mesh content can be measured by the the signal-to-noise ratio (SNR), which is defined as [37]

$$\mathrm{SNR} = 10\lg \frac{\sum_{i=1}^{N}[(v_{i,x}-\bar{v}_x)^2 + (v_{i,y}-\bar{v}_y)^2 + (v_{i,z}-\bar{v}_z)^2]}{\sum_{i=1}^{N}[(g_{i,x}-v_{i,x})^2 + (g_{i,y}-v_{i,y})^2 + (g_{i,z}-v_{i,z})^2]}, \tag{16}$$

where $\bar{v}_x, \bar{v}_y, \bar{v}_z$ are the mean coordinates of models, $v_{i,x}, v_{i,y}, v_{i,z}$ are the original coordinates of v_i , and $g_{i,x}, g_{i,y}, g_{i,z}$ are the modified coordinates of v_i.

We denote the SNR between the direct decrypted mesh model and the original mesh model by SNR_d, and the SNR between the recovered mesh model and the original mesh model by SNR_r. Note that k and m influence the quality of direct decryption of the mesh model.

E. Data Extraction and Lossless recovery

If the receiver has the data-hiding key K_2, the receiver will extract the embedded bits and recover the original mesh content from the direct decrypted mesh.

We exploit spatial correlation between neighboring coordinates to achieve a high embedding rate. Since the kind of 3D meshes constitutes a series of flat triangles adjacent to each other around the inspection point, as shown in Fig. 3, angle deficit curvature is adopted to implement curvature measurement, which makes it unnecessary to use explicit derivative estimates [38]. Meanwhile, the constraint of side length is taken as a part of the measurement of spatial correlation.

1) Angle Deficit Curvature. A given surface can be approximated to a desired accuracy by arbitrarily fine triangulations using pairwise linear manifolds [39], [40]. To form a surface patch, the Zucker-Hummel operator [41] may be applied at the inspection point. On the plane perpendicular to the normal vector, orthographic projections from several evenly distributed points are made onto the object surface. The corresponding five or six points on the surface will serve, along with the inspection point, as vertices of triangular tiles. Each angle θ_i (Fig. 3) at the inspection point $\mathbf{v}_r$ $(1 \le r \le N)$ is computed as

$$\theta_i = \arccos\left[\frac{l_i^2 + l_{i+1}^2 - p_i^2}{2l_i l_{i+1}}\right], \qquad i \in [1, L], \tag{17}$$

where l_i and l_{i+1} represent the adjacent two sides of the specified angle of the ith triangle, p_i is the side opposite of the angle, and L is the number of adjacent triangles, where $l_1 = l_{L+1}$.

Since each triangle is flat, it is clear that all the curvatures are concentrated at the vertices of the triangles (the inspection point). We define a quantity called the angle deficit $\Delta\theta_r$ [39],

$$\Delta\theta_r = 2\pi - \sum_{i=1}^{L}\theta_i. \tag{18}$$

The angle deficit curvature at a point is calculated based on angle deficit as

$$K(\mathbf{v}) = \Delta\theta_r \cdot \delta(\mathbf{v} - \mathbf{v}_r), \tag{19}$$

where $\delta(\cdot)$ is the Dirac delta function. The Gaussian curvature is calculated as

$$G(\mathbf{v}) = \frac{\Delta\theta_r}{\frac{1}{3}\sum_{i=1}^{L} S_i} \cdot \delta(\mathbf{v} - \mathbf{v}_r), \tag{20}$$

where the area of a triangle $S_i = \sqrt{q_i(q_i - l_i)(q_i - l_{i+1})(q_i - p_i)}$ and the semi-perimeter $q_i = (l_i + l_{i+1} + p_i)/2$.

If a local region is smooth, then $K(\mathbf{v}_r)$ tends to be a small positive value converging to zero. Furthermore, the computational complexity is low because no function fitting is involved. We do not take the Gaussian curvature as the criterion of smoothness because a bulge from a smooth surface caused by the modification of coordinates would invalidate the measurement.

2) Constraint of Side Length of Triangles. We observe that non-convex polygons may occur occasionally, resulting in the nullification of the curvature criterion. This is shown in Fig. 4. $\Delta\theta_r$ calculated from Eq. (13) might be a negative number under this situation. Therefore, we add another constraint, side length, to measure the smoothness of local regions.

If the coordinates of triangulations in the mesh models have been modified, the shapes and sizes of the triangles will differ from the original ones. As shown in Fig. 5, most of the maximum lengths of a triangle from an original mesh are smaller than that of the modified mesh of the "cow" to some extent, implying another classification standard for distinguishing "modified" vertices and "non-modified" vertices. Thus, we add another criteria $E(\mathbf{v})$,

$$E(\mathbf{v}) = \sum_{i=1}^{L} \max\{l_i, l_{i+1}, p_i\} \cdot \delta(\mathbf{v} - \mathbf{v}_r). \tag{21}$$

3) Smooth Function. With the same role in the partition of vertices as in the preceding subsection, the receiver segments the vertices into two sets, the "embedded" set $\mathcal{S}_e$ and the "referenced" set $\mathcal{S}_n$. For each decrypted "embedded" vertex

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia

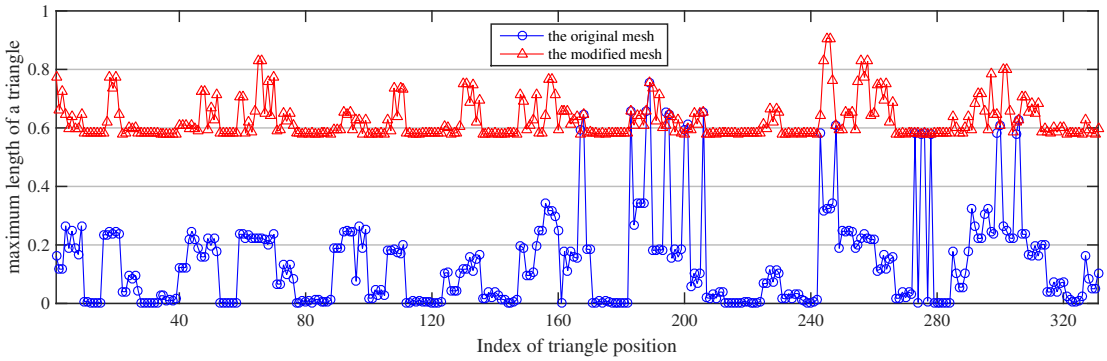


Fig. 5: Comparison of maximum length of a triangle between the original mesh and the modified mesh of "mushroom".

$(v_{i,x}, v_{i,y}, v_{i,z})$ in $\mathcal{S}_e$, the receiver denotes the local region centered on $(v_{i,x}, v_{i,y}, v_{i,z})$ by H_0. At the same time, the receiver flips all the m LSBs of all three integral coordinates of $(v_{i,x}, v_{i,y}, v_{i,z})$ to form a new local region, and denotes the local region by H_1. It must be that either H_0 or H_1 is the original local region of the mesh, and another one is more seriously interfered due to the LSB flip operation. For the two local regions structured by a mutual central point $(v_{i,x}, v_{i,y}, v_{i,z})$ and several faces $\mathbf{F}_i = \{f_j | j = 1, ..., r\}$, where $f_j = (v_{j,x}, v_{j,y}, v_{j,z})$ and r is the number of faces sharing the same vertex $(v_{i,x}, v_{i,y}, v_{i,z})$, a smooth function that is used to measure the fluctuation is considered as

$$M(\mathbf{v}_r) = K(\mathbf{v}_r) + \mu E(\mathbf{v}_r)^{\gamma}, \tag{22}$$

where μ and γ are the balancing weights.

Next, an evaluation criterion is implemented:

$$\lambda(\mathbf{v}_r) = \frac{M(\mathbf{v}_r|H_1)}{M(\mathbf{v}_r|H_0)} \underset{H_1}{\overset{H_0}{\gtrless}} 1. \tag{23}$$

Because of spatial correlation in 3D meshes, the fluctuation function of an original local region is generally lower than that of a seriously interfered version. Thus, the receiver can perform data extraction and mesh recovery by computing $\lambda(\mathbf{v}_r)$. If $\lambda(\mathbf{v}_r) \geq 1$, regard H_0 as the original content of the local region and let the extracted bit be 0. Otherwise, regard H_1 as the original content of this local region and extract a bit 1. Finally, concatenate the extracted bits to retrieve the additional message and collect the recovered local region to form the original mesh.

III. Experiments and Analysis

A. Setups

Our proposed method processed 3D models and operated RDH in an experimental environment in MATLAB R2015b under Windows 7, and we downloaded 3D mesh models in OFF format from the *Princeton Shape Retrieval and Analysis Group* [1] and those in OBJ format from *The Stanford 3D Scanning Repository* [2]. The experiment server configuration was E7-4820 and 32-GB RAM. We implemented the following experiments with 1,815 meshes from the Princeton Shape Retrieval and Analysis Group database.

Fig. 6 shows a group of experimental results of visual effects with different meshes. The phases from left to right are original meshes, encrypted meshes, data-embedded meshes, direct-decrypted meshes, and recovered meshes. Encrypted meshes and data-embedded meshes cannot be recognized by eavesdroppers. Direct-decrypted meshes can be roughly identified, and recovered meshes can be almost clearly recognized. With ECCs, recovered meshes can be further perfected to the original version.

B. Choice of Digits of Decimal Reservation k

The choice of digits of decimal reservation k influences the quality of decrypted meshes and the times costs. We encrypted the source meshes and directly decrypted the encrypted version. We varied k from 1 to 15 to observe the phenomenon.

Fig. (7a) manifests that the threshold $k_{th} = 6$, which is a truncation point that either enables most of the meshes to losslessly recover the original meshes nor restrains most of the meshes to losslessly recover the original meshes. As mentioned previoujsly, the vertex coordinates are stored in the format of floating numbers and the number of significant digits is 6; thus, a truncation threshold smaller than 6 will cause permanent distortion, indicating the impossibility of recovery of the original mesh. Apart from the meshes that can be losslessly decrypted, we average the SNR of the lossy decrypted meshes, which is shown in Fig. (7b). As is shown, the average SNR monotonically increases with increasing k, which manifests the fact that, to acquire a highly similar decrypted version, a better k is preferred. However, as shown in Fig. (7c), k influences the memory overhead of the process. Meanwhile, the shape of the curve, which fits a step function, corresponds to Eq. (3). Thus, the generation and storage of coordinate l are the leading sources of overhead time.

To balance the quality of the decrypted mesh and the overhead time, we take $k = 4$ in the following experiments.

C. Choice of Number of Embedded LSBs m

Based on the RDH in encrypted images proposed in [24], Zhang *et. al.* took three LSBs of the encrypted pixel to embed a one-bit message. In encrypted 3D meshes, we varied the number of embedded LSBs m to observe the experimental phenomena.

[1] http://shape.cs.princeton.edu/benchmark/index.cgi
[2] http://graphics.stanford.edu/data/3Dscanrep/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia

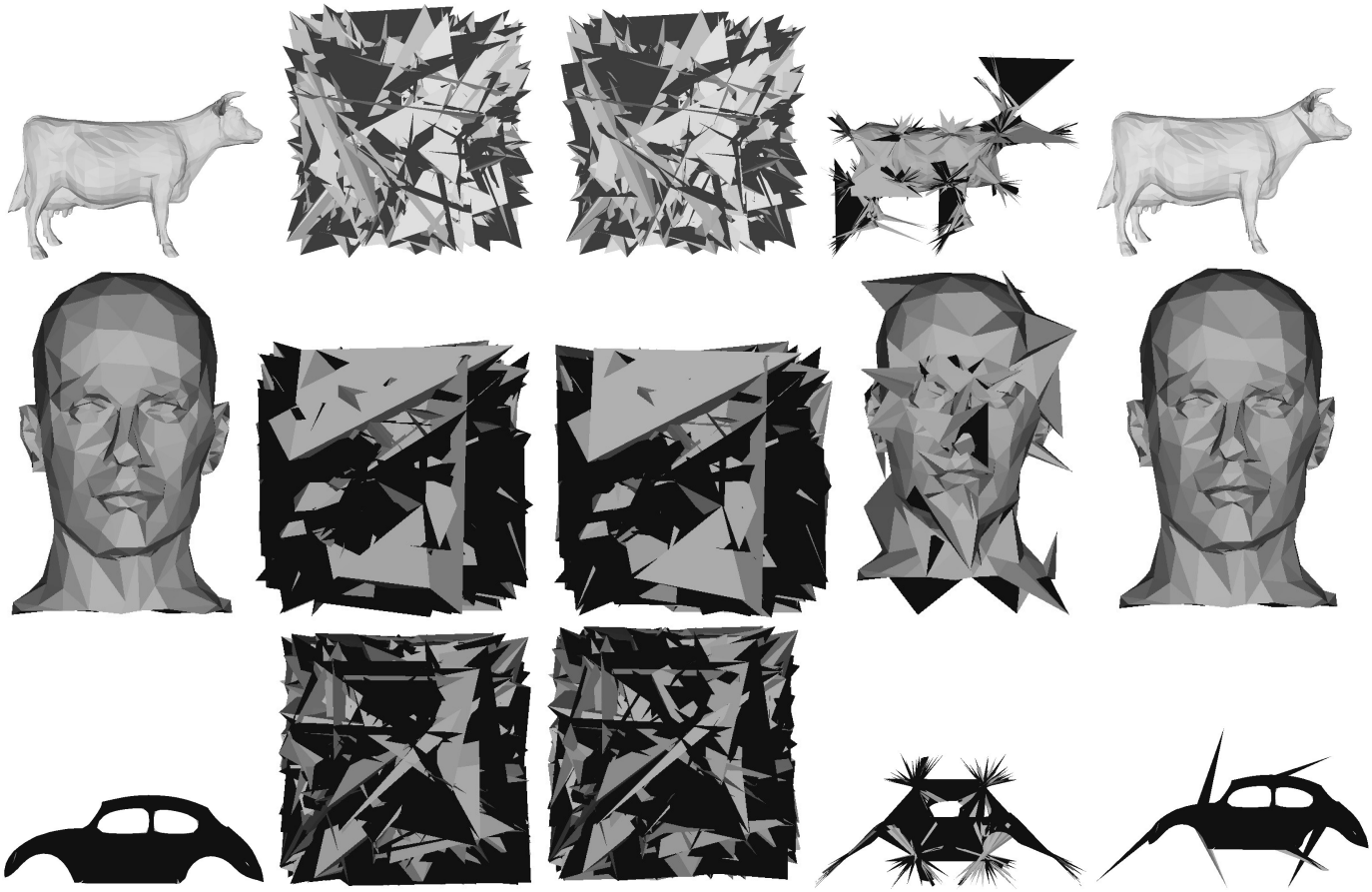
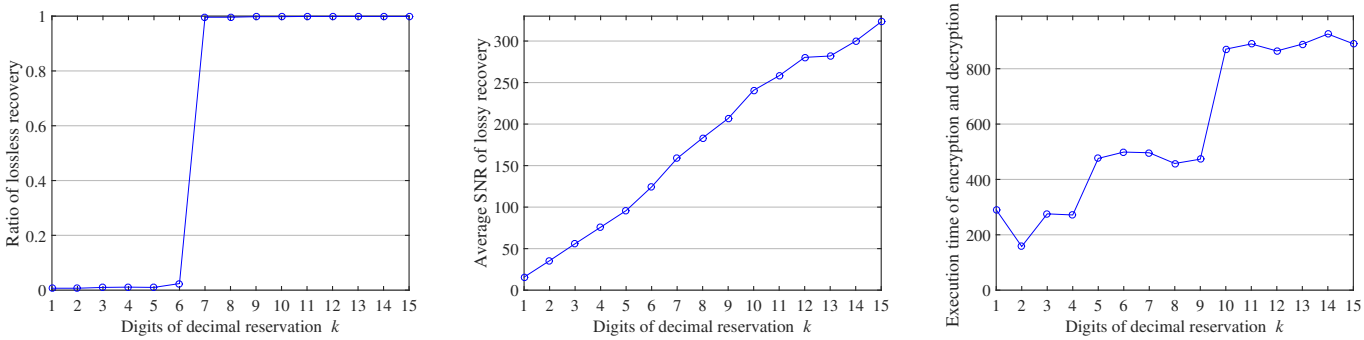


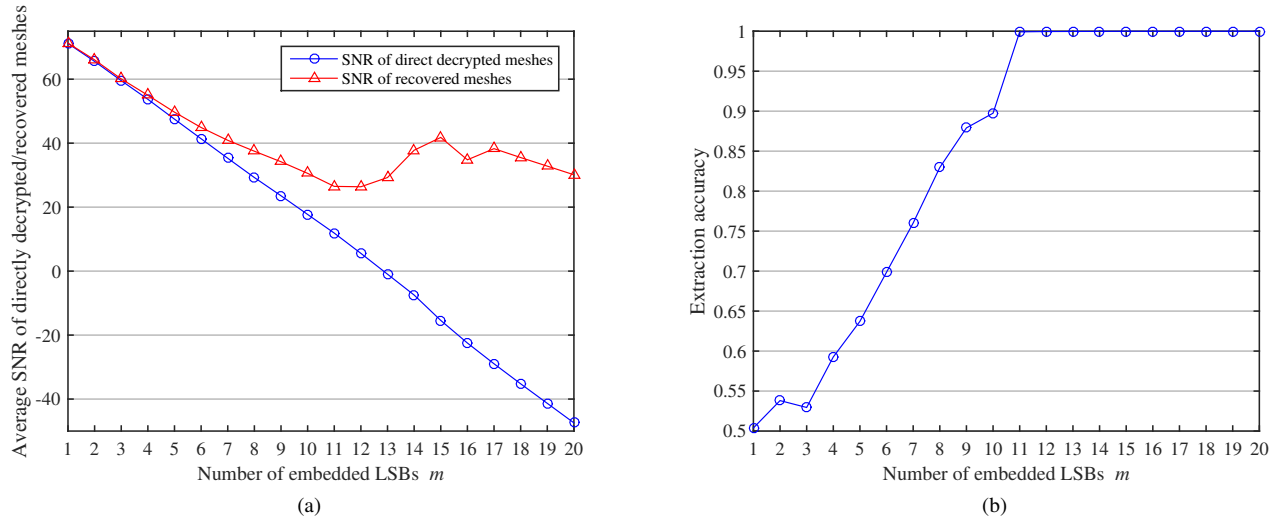
Fig. 6: Illustrative examples showing the appearance of the model of each phase from left to right, *i.e.* original meshes, encrypted meshes, data-embedded meshes, direct-decrypted meshes, and recovered meshes (without ECCs). From the first row to the fifth row, the exemplified models are "cow", "mannequin," and "beetle". Note that after error correction, more meshes can be recovered losslessly.



(a) (b) (c)

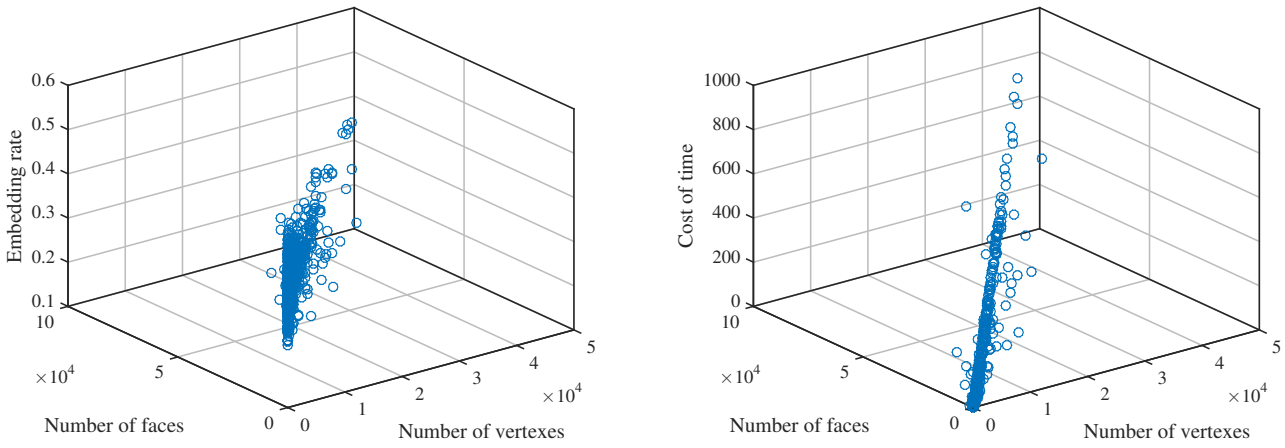
Fig. 7: (a) Ratio of lossless recovery under varying digits of decimal reservation k. (b) Average SNR of lossy recovery under varying digits of decimal reservation k. (c) Average execution time of encryption and decryption under varying digits of decimal reservation k.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia



(a) (b)

Fig. 8: (a) Comparison between the average SNR_d of direct-decrypted meshes with source meshes and the average SNR_r of recovered meshes with source meshes under a varying number of embedded LSBs m. (b) Extraction accuracy under a varying number of embedded LSBs m.



(a) (b)

Fig. 9: (a) Distribution of the embedding rate under a varying number of vertices and number of faces. (b) Distribution of the time cost under a varying number of vertices and a varying number of faces.

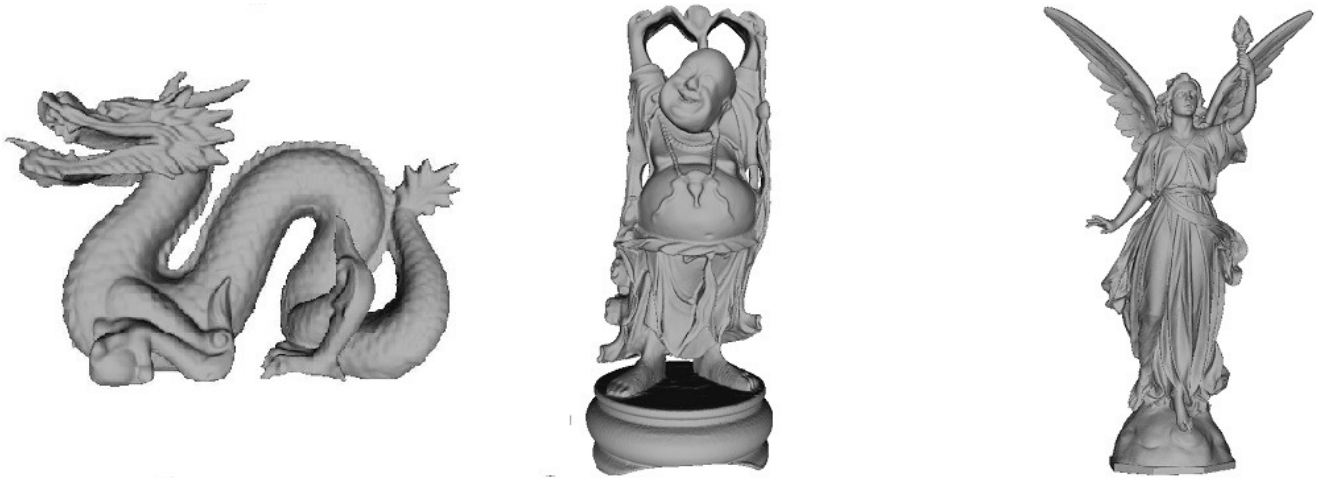
TABLE III
AVERAGE ERROR RATE FOR VARYING μ AND VARYING γ

μ	γ								
	0	0.5	1	1.5	2	2.5	3	3.5	4
0	4.43%	4.43%	4.43%	4.43%	4.43%	4.43%	4.43%	4.43%	4.43%
5	4.43%	4.40%	4.38%	4.32%	4.24%	4.10%	4.02%	3.91%	3.85%
10	4.43%	4.37%	4.32%	4.21%	4.08%	3.96%	3.88%	3.81%	3.79%
15	4.43%	4.35%	4.24%	4.11%	3.97%	3.89%	3.80%	3.77%	3.75%
20	4.43%	4.35%	4.10%	4.04%	3.92%	3.83%	3.74%	3.74%	3.74%
25	4.43%	4.32%	4.14%	3.99%	3.88%	3.77%	3.72%	3.72%	3.73%
30	4.43%	4.29%	4.10%	3.96%	3.84%	3.74%	3.71%	3.72%	3.72%
35	4.43%	4.26%	4.07%	3.92%	3.81%	3.71%	3.68%	3.72%	3.71%
40	4.43%	4.24%	4.02%	3.90%	3.77%	3.69%	3.69%	3.70%	3.69%

TABLE IV
PERFORMANCE OF REVERSIBLE DATA HIDING BETWEEN DEFICIT CURVATURE AND GAUSSIAN CURVATURE

Curvature methods	Average embedding rate	Average error rate	Average SNR_d	Average SNR_r
Deficit curvature	0.3692 bpv	4.22%	5.3579	31.9723
Gaussian curvature	0.3692 bpv	32.89%	0.3289	9.4480

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia



(a) (b) (c)

Fig. 10: Sophisticated meshes. (a) Dragon. (b) Happy Buddha. (c) Thai Statue.

TABLE V
PERFORMANCE OF REVERSIBLE DATA HIDING IN THE DATASET OF PRINCETON SHAPE RETRIEVAL AND ANALYSIS GROUP

Embedding rate	Ratio of lossless recovery	Error rate	Average SNR_d	Average SNR_r	ECC methods	Net embedding rate	Ratio of lossless recovery with ECC	Error rate with ECC
0.3692 bpv	79.8%	4.22%	5.3579	31.9723	BCH	0.2461 bpv	89.6%	2.11%
					LDPC	0.2293 bpv	95.1%	1.65%
					Golay	0.1766 bpv	94.4%	1.78%

In Fig. (8a), we analyze the average SNR_d of direct-decrypted meshes with source meshes and the average SNR_r of recovered meshes with source meshes under varying m. When $m \leq 13$, SNR of direct decrypted meshes decreases in a linear relation with increasing m, indicating the decrease of quality of decrypted meshes; when $m > 13$, the curve fluctuates up and down in a small range and no longer decreases. This can be attributed to the boost of recovery ability with increasing m, because when m is small the dissimilarity between H_1 and H_0 is minute, leading to a weak recovery ability; when m is larger, the dissimilarity is greater, making the judgment of H_1 and H_0 easier. Thus, the SNR between the recovery mesh and the original mesh maintains a certain balance when m is larger than 12. In the meantime, as shown in Fig. (8b), the extraction accuracy of messages increases gradually with increasing m. When $m \geq 9$, ECCs are capable of checking and correcting the error bits at the cost of capacity reduction to losslessly extract data and recover the mesh. Taking the above two factors into consideration, a moderate m is suitable for the experimental setup. Unless stated, m is set to 12 as the default in the following implementation.

A scatter diagram of embedding rate wit a varying number of vertices and faces is shown in Fig. (9a). As the figure shows, the boundary of the embedding rate fits a parabola. It is apparent that the scatter points can be well fitted by the theoretical limit of the embedding rate. Fig. (9b) is the scatter diagram of the time cost with a varying number of vertices and faces.

D. μ and γ

To balance the effect of μ and γ on the recovery accuracy of mesh models, we varied μ from 0 to 40 with an interval of 5 and varied γ from 0 to 4 with an interval of 0.5 to find the best μ and γ. In our implementation, digits of decimal reservation k were set to 4 and number of embedded LSBs m to 12. The average error rate is shown in Table III. It can be seen that $\mu = 35$ and $\gamma = 3$ provide the best performance with a minimum average error rate of 3.68%. Thus, the two parameters μ and γ were set to 35 and 3, respectively, as the defaults in our implementation.

E. Comparison Between Deficit Curvature and Gaussian Curvature

With the same experimental setups ($k = 4$ and $m = 12$), the deficit curvature has the lowest average error rate under $\mu = 35$ and $\gamma = 3$ and the Gaussian curvature with $\mu = 10$ and $\gamma = 1$. Evaluations of the selection of curvature functions are shown in Table IV.

It is obvious that the deficit curvature has a much better capability of recovering the original version of a decrypted 3D model, and thus we take deficit curvature as the criteria of smoothness.

F. Performance Analysis on Datasets

First, we appraised the proposed RDH method with the dataset of the *Princeton Shape Retrieval and Analysis Group*. As shown in Table V, the average embedding rate is 0.3692 bpv, and the capacity increases monotonically with the number of vertices. The average error rate is 4.22%, which can be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2017.2723244, IEEE Transactions on Multimedia

TABLE VI
PERFORMANCE OF REVERSIBLE DATA HIDING IN DENSE MESHES

Meshes	#Triangles	#Vetices	Embedding rate	Error rate	SNR_d	SNR_r
Dragon	871,414	437,645	0.3472 bpv	10.13%	-10.3974	-3.4610
Happy Buddha	5,500,000	543,652	0.3485 bpv	11.39%	-11.3478	-4.9186
Thai statue	7,500,000	4,999,996	0.3508 bpv	9.72%	-9.9223	-2.8495

reduced to zero with ECCs. The SNR_d is much smaller than SNR_r, which verifies that the quality of direct-decrypted meshes is much worse than that of recovered meshes in terms of visual quality. We took BCH code, LDPC code, and Golay code as ECCs to implement experiments to compare the performance of net embedding rate C_n and error rates. We assumed $[15, 5]$ codes are used in BCH code, $[576, 288]$ codes are used in LDPC code, and $[23, 12]$ codes are used in Golay code. Considering the results above, all the three ECC algorithms are capable of reducing the error rates, and yet at the same time the actual embedding rate (i.e., net embedding rate) has dropped slightly compared to the embedding rate without using ECC algorithms. The ratio of lossless recovery has been boosted from the original 79.8% to higher than 89% with ECCs. Sacrificing more net embedding capacity will earn better accuracy, which depends on actual demand.

Second, we performed experiments aimed at dense meshes consisting of tens of millions or hundreds of millions of triangles, which are widely used in 3D printing. We took several meshes from *The Stanford 3D Scanning Repository* to test the performance of the proposed method, and the results are shown in Fig. 10. In Table VI, the error rates of all the three dense meshes can be seen to be fairly low, which can be further corrected with ECCs. The experiments show that the proposed RDH method is applicable to sophisticated models.

Third, since CAD/CAM applications have been widely developed, massive meshes with irregular triangles are being created and used. We experimented with a few meshes of the type collected from hand-made CAD meshes, which are shown in Fig. 11. As shown in Table VII, error rates of the four hand-made meshes are all rather low ($< 2\%$), confirming the feasibility of the proposed method for meshes from CAD/CAM applications.

The results indicate that the proposed RDH method is of a certain commonality and typicality, and will have some practical significance and application prospects.

G. Non-separable Versus Separable

The framework of the proposed scheme adopts non-separable solutions. To ensure embedding security, the existing RDH methods all contain data-hiding keys, which need to be shared and managed between the date hider and the recipient. However, the key management functions, e.g., key generation, activation, deactivation, suspension, expiration, destruction, archiving, and revocation, are difficult to reliably implement within such a distributed infrastructure [42]. It manifests that a RDH scheme without a secret data-hiding key still ensures that only the party with the secret encryption key can disclose the embedded message, which could be very valuable in practice.

It is intuitive that the security offered by the encryption key may be appropriately extended to protect the data embedding [26]. In the encrypted-domain secure RDH scheme without a data-hiding key [26], the possibility of eliminating the data-hiding key is applicable to our non-separable RDH scheme. Here, some design goals are slightly different from those of the existing solutions, due to the elimination of the data-hiding key. The requirement of the direct-decrypted version of meshes is invalid since we only have a single encryption key, making the decryption and data extraction naturally tied together.

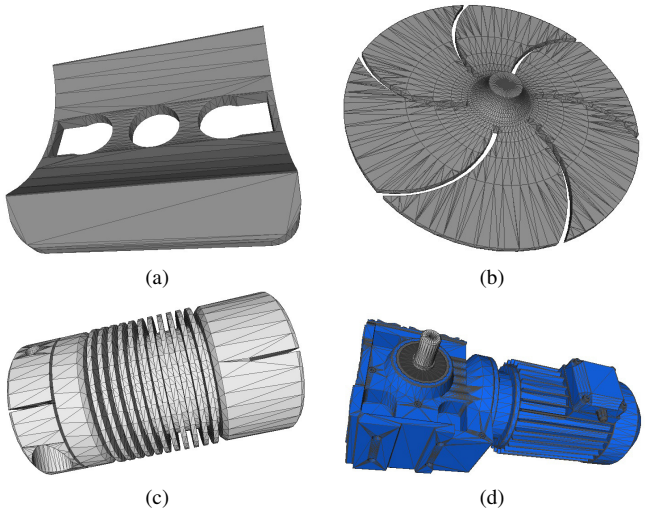


Fig. 11: Hand-made CAD meshes. (a) m1. (b) m2. (c) m3. (d) m4.

IV. CONCLUSIONS

Currently, 3D models are widely used on the Internet. A new RDH-ED technique for 3D meshes is of great significance since it may have a series of applications. With the "VRAE" framework, we first mapped decimals of the vertex coordinates into integers, so that a bit-stream encryption technique could be executed. With a data-hiding key, several LSBs were operated to embed data. By using the encryption key, a receiver can roughly reconstruct the content of the mesh. According to the data-hiding key, with the aid of spatial correlation in natural mesh models, the embedded data can be successfully extracted and the original mesh can be perfectly recovered. Experimental results on the two datasets demonstrated that our average error rate is 4.22%, which can be further reduced with ECCs. The performance analysis implies that our proposed method has very good potential for practical applications.

For future work, aimed at the specific applications of separable RDH schemes, we plan to expand the investigation from non-separable RDH schemes to separable RDH schemes.

TABLE VII
PERFORMANCE OF REVERSIBLE DATA HIDING IN CAD MESHES

Meshes	#Triangles	#Vetices	Embedding rate	Error rate	SNR_d	SNR_r
m1	524	258	0.3527 bpv	1.10%	15.0858	35.2048
m2	3,656	1,830	0.3809 bpv	0.72%	7.8812	25.8820
m3	4,920	2,452	0.3413 bpv	0.12%	11.9705	39.9634
m4	19,294	9,633	0.3528 bpv	0.74%	7.9331	25.0314

REFERENCES

[1] H. S. Malvar and D. A. Florêncio, "Improved spread spectrum: A new modulation technique for robust watermarking," *IEEE transactions on signal processing*, vol. 51, no. 4, pp. 898–905, 2003.

[2] K. Wang, G. Lavoué, F. Denis, and A. Baskurt, "Three-dimensional meshes watermarking: Review and attack-centric investigation," in *International Workshop on Information Hiding*, pp. 50–64, Springer, 2007.

[3] X. Zhang and S. Wang, "Fragile watermarking with error-free restoration capability," *IEEE Transactions on Multimedia*, vol. 10, pp. 1490–1499, Dec 2008.

[4] C. D. Vleeschouwer, J. F. Delaigle, and B. Macq, "Circular interpretation of bijective transformations in lossless watermarking for media asset management," *IEEE Transactions on Multimedia*, vol. 5, pp. 97–105, March 2003.

[5] K. Hwang and D. Li, "Trusted cloud computing with secure resources and data coloring," *IEEE Internet Computing*, vol. 14, no. 5, pp. 14–22, 2010.

[6] F. Willems, D. Maas, and T. Kalker, "Semantic lossless source coding," in *Proc. 42nd Annu. Allerton Conf. Communication, Control and Computing*, 2004.

[7] V. Sachnev, H. J. Kim, J. Nam, S. Suresh, and Y. Q. Shi, "Reversible watermarking algorithm using sorting and prediction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 989–999, 2009.

[8] B. Ou, X. Li, Y. Zhao, R. Ni, and Y.-Q. Shi, "Pairwise prediction-error expansion for efficient reversible data hiding," *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 5010–5021, 2013.

[9] I.-C. Dragoi and D. Coltuc, "Local-prediction-based difference expansion reversible watermarking," *IEEE Transactions on image processing*, vol. 23, no. 4, pp. 1779–1790, 2014.

[10] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on circuits and systems for video technology*, vol. 16, no. 3, pp. 354–362, 2006.

[11] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 13, no. 8, pp. 890–896, 2003.

[12] Y. Hu, H. K. Lee, K. Chen, and J. Li, "Difference expansion based reversible data hiding using two embedding directions," *IEEE Transactions on Multimedia*, vol. 10, pp. 1500–1512, Dec 2008.

[13] X. Zhang, "Reversible data hiding with optimal value transfer," *IEEE Transactions on Multimedia*, vol. 15, pp. 316–325, Feb 2013.

[14] W. Zhang, X. Hu, X. Li, and N. Yu, "Recursive histogram modification: establishing equivalency between reversible data hiding and lossless data compression," *IEEE Transactions on Image Processing*, vol. 22, no. 7, pp. 2775–2785, 2013.

[15] X. Hu, W. Zhang, X. Li, and N. Yu, "Minimum rate prediction and optimized histograms modification for reversible data hiding," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 653–664, 2015.

[16] X. Hu, W. Zhang, X. Hu, N. Yu, X. Zhao, and F. Li, "Fast estimation of optimal marked-signal distribution for reversible data hiding," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 5, pp. 779–788, 2013.

[17] W. Zhang, X. Hu, X. Li, and Y. Nenghai, "Optimal transition probability of reversible data hiding for general distortion metrics and its applications," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 294–304, 2015.

[18] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Transactions on information forensics and security*, vol. 8, no. 3, pp. 553–562, 2013.

[19] W. Zhang, K. Ma, and N. Yu, "Reversibility improved data hiding in encrypted images," *Signal Processing*, vol. 94, pp. 118–127, 2014.

[20] X. Cao, L. Du, X. Wei, D. Meng, and X. Guo, "High capacity reversible data hiding in encrypted images by patch-level sparse representation," *IEEE transactions on cybernetics*, vol. 46, no. 5, pp. 1132–1143, 2016.

[21] W. Zhang, H. Wang, D. Hou, and N. Yu, "Reversible data hiding in encrypted images by reversible image transformation," *IEEE Transactions on Multimedia*, vol. 18, pp. 1469–1479, Aug 2016.

[22] M. Johnson, P. Ishwar, V. Prabhakaran, D. Schonberg, and K. Ramchandran, "On compressing encrypted data," *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2992–3006, 2004.

[23] W. Liu, W. Zeng, L. Dong, and Q. Yao, "Efficient compression of encrypted grayscale images," *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 1097–1102, 2010.

[24] X. Zhang, "Reversible data hiding in encrypted image," *IEEE Signal Processing Letters*, vol. 18, no. 4, pp. 255–258, 2011.

[25] W. Hong, T.-S. Chen, and H.-Y. Wu, "An improved reversible data hiding in encrypted images using side match," *IEEE Signal Processing Letters*, vol. 19, no. 4, pp. 199–202, 2012.

[26] J. Zhou, W. Sun, L. Dong, X. Liu, O. C. Au, and Y. Y. Tang, "Secure reversible image data hiding over encrypted domain via key modulation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 3, pp. 441–452, 2016.

[27] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 826–832, 2012.

[28] Z. Qian and X. Zhang, "Reversible data hiding in encrypted images with distributed source encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 4, pp. 636–646, 2016.

[29] W.E.Ryan, "An introduction to LDPC codes," in *CRC Handbook for Coding and Signal Processing for Recoding Systems*, B. Vasic, Ed. Boca Raton, FL, USA: CRC Press, 2004.

[30] D. Chou, C.-Y. Jhou, S.-C. Chu, *et al.*, "Reversible watermark for 3d vertices based on data hiding in mesh formation," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 7, pp. 1893–1901, 2009.

[31] H.-t. Wu and J.-L. Dugelay, "Reversible watermarking of 3d mesh models by prediction-error expansion," in *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, pp. 797–802, IEEE, 2008.

[32] H.-T. Wu and Y.-m. Cheung, "A reversible data hiding approach to mesh authentication," in *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pp. 774–777, IEEE, 2005.

[33] Z. Sun, Z.-M. Lu, and Z. Li, "Reversible data hiding for 3d meshes in the pvq-compressed domain," in *2006 International Conference on Intelligent Information Hiding and Multimedia*, pp. 593–596, IEEE, 2006.

[34] Z.-M. Lu and Z. Li, "High capacity reversible data hiding for 3d meshes in the pvq domain," in *International Workshop on Digital Watermarking*, pp. 233–243, Springer, 2007.

[35] H. Luo, Z.-M. Lu, and J.-S. Pan, "A reversible data hiding scheme for 3d point cloud model," in *2006 IEEE International Symposium on Signal Processing and Information Technology*, pp. 863–867, IEEE, 2006.

[36] H. Luo, J.-S. Pan, Z.-M. Lu, and H.-C. Huang, "Reversible data hiding for 3d point cloud model," in *2006 International Conference on Intelligent Information Hiding and Multimedia*, pp. 487–490, IEEE, 2006.

[37] M. Deering, "Geometry compression," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 13–20, ACM, 1995.

[38] E. M. Stokely and S. Y. Wu, "Surface parametrization and curvature measurement of arbitrary 3-d objects: five practical methods," *IEEE Transactions on pattern analysis and machine Intelligence*, vol. 14, no. 8, pp. 833–840, 1992.

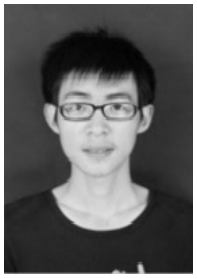
[39] C. Lin and M. Perry, "Shape description using surface triangulation," in *Proceedings of the Workshop on Computer Vision: Representation and Control*, pp. 38–43, 1982.

[40] P. J. Besl and R. C. Jain, "Invariant surface characteristics for 3d object recognition in range images," *Computer vision, graphics, and image processing*, vol. 33, no. 1, pp. 33–80, 1986.

[41] S. W. Zucker and R. A. Hummel, "A three-dimensional edge operator," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 3, pp. 324–331, 1981.

[42] R. Chandramouli, M. Iorga, and S. Chokhani, "Cryptographic key management issues and challenges in cloud services," in *Secure Cloud Computing*, pp. 1–30, Springer, 2014.
[43] Z. Qian, X. Zhang, and S. Wang, "Reversible data hiding in encrypted jpeg bitstream," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1486–1491, 2014.
[44] F. Cayre and B. Macq, "Data hiding on 3-d triangle meshes," *IEEE Transactions on signal Processing*, vol. 51, no. 4, pp. 939–949, 2003.
[45] C.-M. Wang and Y.-M. Cheng, "An efficient information hiding algorithm for polygon models," in *Computer Graphics Forum*, vol. 24, pp. 591–600, Wiley Online Library, 2005.
[46] Y.-M. Cheng and C.-M. Wang, "A high-capacity steganographic approach for 3d polygonal meshes," *The Visual Computer*, vol. 22, no. 9, pp. 845–855, 2006.
[47] M.-W. Chao, C.-h. Lin, C.-W. Yu, and T.-Y. Lee, "A high capacity 3d steganography algorithm," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 2, pp. 274–284, 2009.

Ruiqi Jiang received his B.S. degree in 2011 from Harbin Institute of Technology, M.S. degree in 2013 from The George Washington University (USA). He is currently pursuing the Ph.D. degree in electronic engineering in University of Science and Technology of China. His research interests including multimedia security and information hiding.



Hang Zhou received the B.S. degree in School of Communication and Information Engineering, Shanghai University, in 2015. He is currently pursuing the Ph.D. degree in Electronic Engineering in University of Science and Technology of China (USTC). His research interests include information hiding, image processing and computer graphics.

Weiming Zhang received his M.S. degree and Ph.D. degree in 2002 and 2005 respectively from the Zhengzhou Information Science and Technology Institute, P.R. China. Currently, he is a professor with the School of Information Science and Technology, University of Science and Technology of China. His research interests include information hiding and multimedia security.

Nenghai Yu received his B.S. degree in 1987 from Nanjing University of Posts and Telecommunications, M.E. degree in 1992 from Tsinghua University and Ph.D. degree in 2004 from University of Science and Technology of China, where he is currently a professor. His research interests include multimedia security, multimedia information retrieval, video processing and information hiding.