symmetry

MDPI

Article

An Effective Authentication Algorithm for Polygonal Models with High Embedding Capacity

Yuan-Yu Tsai [1,2,*,†], Yu-Shiou Tsai [1] and Chia-Chun Chang [1]

[1] Department of M-Commerce and Multimedia Applications, Asia University, No. 500, Lioufeng Rd., Wufeng District, Taichung 41354, Taiwan; s0917251664@gmail.com (Y.-S.T.); ru8rmpgj9@gmail.com (C.-C.C.)

[2] Department of Medical Research, China Medical University Hospital, China Medical University, No. 2, Yude Road, North District, Taichung 40447, Taiwan

* Correspondence: yytsai@asia.edu.tw; Tel.: +886-4-23323456 (ext. 20034)

† This paper is an extended version of our paper published in 2018 IEEE International Conference on Applied System Innovation as titled "An Improved Region-based Tampered Detection Algorithm for 3D Polygonal Models".

Received: 20 June 2018; Accepted: 17 July 2018; Published: 18 July 2018

Abstract: Fragile watermarking algorithms for 3D models has attracted extensive research attention in recent years. Although many literatures have proposed lots of solutions on this issue, low embedding capacity and inaccurate located tampered region are still presented. Higher embedding capacity can reduce the probability of false alarms for authentication, while accurate tamper localization can detect all the modified vertices with fewer unaltered ones. This study proposes three strategies to enhance the embedding capacity and the detection accuracy of our previous algorithm. First, the modified message-digit substitution table can increase the embedding capacity by 11.5%. Second, the modified embedding ratio generation method can be integrated to raise the embedding capacity by about 47.74%. Finally, the strategy adopting a reduced number of reference vertices for authentication code generation accompanying the above two ones improves the embedding capacity up to 123.74%. Extensive experiments show that the proposed algorithm can achieve superior performance in terms of embedding capacity and tamper localization accuracy. Further, the model distortion between the original and the marked ones is small.

Keywords: tamper detection; authentication; polygonal models; fragile watermarking

1. Introduction

The rise of Internet changes human behavior significantly. Multimedia content can be rapidly shared and dispersed all over the world with the advanced network technology. However, Internet also breeds new information security problems, such as piracy, reproduction, and tampering. Copyright marking algorithms [1] can prevent the above problems from occurring, and robust watermarking and fragile watermarking are two main branches according to their different applications.

Robust watermarking algorithms [2–4] can detect copyright information and protect intellectual property rights even when the protected multimedia content is under different malicious attacks. However, fragile watermarking algorithms [5–7] can effectively detect and locate the tampered locations even if the protected multimedia content has minimum modifications. Some fragile watermarking algorithms even have the capacity for self-recovery. Recently, some authors [8] proposed a distributed and tamper-proof media transaction framework. The unique watermark information contains a cryptographic hash of the transaction histories in the blockchain and an image hash preserving retrievable original media content. The first part of the watermark is to retrieve the

historical transaction trail and the latter part is used to identify the tampered regions. Consequently, copyright marking is still a research topic that deserves our attention.

Whereas fragile watermarking has been extensively studied in images [9–14], videos [15,16], and audio [17,18], there are still few studies discussing its application in 3D models. According to the size of the tampered region localization, 3D fragile watermarking algorithms can be classified as either vertex-based or region-based ones. A vertex-based algorithm can exactly locate a modified vertex, whereas a region-based one can only locate a rough region with some unaltered vertices inside. However, a region-based fragile watermarking algorithm has the ability to detect the modification of the topological relationship between vertices.

The first fragile watermarking algorithm on 3D polygonal models was proposed by Yeo and Yeung [19]. They iteratively moved the vertices to new positions to make the location and value indexes, calculated from pre-defined hash functions, consistent. Lin et al. [20] proposed a local mesh parametrization approach to perturb the coordinates of invalid vertices to eliminate the causality problem caused by Yeo and Yeung. Molaei et al. [21] proposed a QIM-based fragile watermarking for 3D triangular mesh models. Watermark data is embedded into the middles of three sides of a marked triangle in the spherical coordinate system. The recognition accuracy and the embedding capacity can be enhanced by increasing the number of marked triangles. To solve the problem of low embedding capacity, Tsai et al. [6] proposed a low-complexity, high-capacity, distortion-controllable, region-based fragile watermarking algorithm with a high embedding rate for 3D polygonal models. A vertex traversal scheme subdivides the input polygonal model into several overlapping embedding blocks, each with one embeddable vertex and its reference vertices. The authentication code is generated by feeding the local geometrical properties of each block into a hash function. The embeddable vertex is then embedded with the authentication code and its vertex coordinates are modified based on a simple message-digit substitution scheme.

Tsai et al.'s proposed algorithm had a higher embedding rate and higher embedding capacity than up-to-date literature. However, the problems of low embedding capacity and inaccurately located tampered regions have not been resolved. The proposed algorithm comprehensively considers three issues that are not discussed in our previous scheme. Thus, the proposed algorithm offers three advantages over the previous method, including higher embedding capacity, higher embedding rate, and accurate tamper localization. In this paper, we employ modified message grabbing and modified embedding ratio generation methods to resolve the problem of low embedding capacity. In addition, we adopt a reduced number of reference vertices for authentication code generation, providing a significant improvement on embedding capacity and embedding rate. Furthermore, the accuracy of tamper localization is consequently raised. Finally, extensive experimental results demonstrate the feasibility of the proposed algorithm.

The rest of paper is organized as follows. In Section 2, Tsai et al.'s proposed algorithm is introduced. In Section 3, we give detailed descriptions of our three strategies to improve the embedding capacity. Section 4 shows the experimental evaluations; finally, conclusions and future studies are made in Section 5.

2. Tsai et al.'s Proposed Algorithm

This section provides a review of Tsai et al.'s proposed algorithm, and its flowchart is shown in Figure 1. The proposed algorithm inputs a polygonal model with vertices and its topological information and then constructs a vertex neighboring table in the preprocessing process. The polygonal model is subdivided into lots of overlapping embedding blocks, each with one embeddable vertex and its reference vertices, based on an iterative vertex traversal mechanism. Each block is the basic element for the authentication code generation, embedding, reconstruction, and tamper localization processes. Taking the embedding block shown in Figure 2 for example, V_1 is the embeddable vertex and R_1, R_2, R_3, and R_4 are the reference vertices of V_1. G_1 is the center of the reference vertices of V_1.

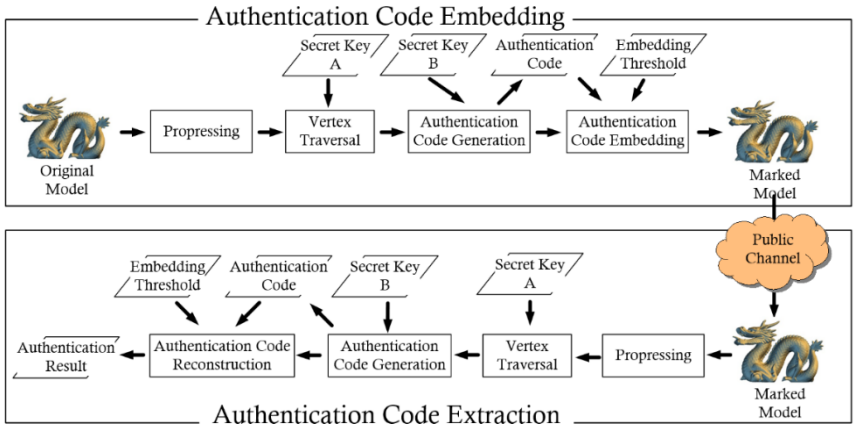


Figure 1. Flowchart of Tsai et al.'s proposed algorithm.

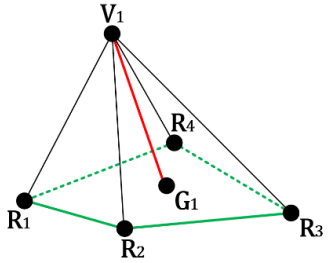


Figure 2. One embedding block of Tsai et al.'s proposed algorithm.

For each embedding block, the local geometrical property concerning the topological relationship is fed into the hash function to obtain the authentication code. Thereafter, the authors calculated the embedding ratio between the length of the embeddable vertex to the center of its reference vertices and the summation of all sides of the reference vertices for message embedding. Continuing the example in Figure 2, the embedding ratio A_{V_1} is shown in (1), where S_{V_1} is the summation of all sides $\|R_1R_2\|$, $\|R_2R_3\|$, $\|R_3R_4\|$, and $\|R_4R_1\|$.

$$A_{V_1} = \begin{cases} \|V_1G_1\|/S_{V_1} & if\ \|V_1G_1\| \le S_{V_1} \\ S_{V_1}/\|V_1G_1\| & if\ \|V_1G_1\| > S_{V_1} \end{cases} \quad (1)$$

Finally, the authors select parts of the authentication code to embed into above ratio. The data-embedded ratio A'_{V_1} is derived by modifying the next few decimal digits after the first non-zero one of A_{V_1} using a simple message-digit substitution mechanism shown in Table 1. The data-embedded vertex can be then obtained following the direction of $\overrightarrow{V_1G_1}$ with a data-embedded ratio. Figure 3 shows the modification example, where V'_1 is the vertex with the authentication code embedded. Experimental results demonstrate the feasibility of the proposed algorithm.

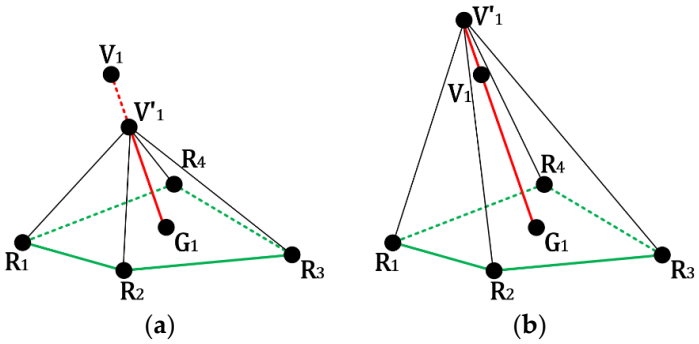


Figure 3. The modification example for the embeddable vertex after the authentication code is embedded: (**a**) $\|\overrightarrow{V_1G_1}\| \ge \|\overrightarrow{V'_1G_1}\|$; (**b**) $\|\overrightarrow{V_1G_1}\| < \|\overrightarrow{V'_1G_1}\|$.

Table 1. Tsai et al.'s message-digit substitution table.

Decimal Digit	Binary Authentication Code
0	000
1	001
2	010
3	011
4	100
5	101
6	1100
7	1101
8	1110
9	1111

3. The Proposed Algorithm

This section shows three improvement methods to provide higher embedding capacity than Tsai et al.'s proposed algorithm, including modified message grabbing, modified embedding ratio generation [22], and a reduced number of reference vertices for message embedding. The proposed algorithm can also support higher embedding rate and accurate tamper localization.

3.1. Modified Message Grabbing Method

The authentication code embedding process in Tsai et al.'s proposed algorithm used a simple message-digit substitution scheme shown in Table 1 to modify the embedded ratio. The distance with the center of its reference vertices for each embeddable vertex is consequently modified. Three-bit authentication code is grabbed one time and one more bit can be grabbed again if the above three-bit code equals 110 or 111. In our test, the embedding capacity rises effectively if a four-bit authentication code is firstly grabbed and the last bit is left for the next iteration if the grabbed four-bit binary code is larger than one-bit decimal digit. Table 2 shows our message-digit substitution table. The bit with a strikeout sign means that the bit should be left for next iteration because the decimal value for the grabbed four-bit binary code is larger than nine.

Table 2. Our proposed message-digit substitution table.

Decimal Digit	Binary Authentication Code
0	0000
1	0001
2	0010/~~1~~010
3	0011/~~1~~011
4	0100/~~1~~100
5	0101/~~1~~101
6	0110/~~1~~110
7	0111/~~1~~111
8	1000
9	1001

3.2. Modified Embedding Ratio Generation Method

From the above message-digit substitution table, we can know one decimal digit can have a three- to four-bit authentication code embedded. However, from the experimental results in Tsai et al.'s proposed algorithm, the number of most used decimal digits is three. It means most embedding ratios equal to 0.0XEEEN, where X is the first non-zero digit, E is the decimal digit that can have the authentication code embedded, and N is the digit without a message embedded for avoiding

extraction error. If we can change the embedding ratio with the value of 0.0XEEEN to 0.XEEEEN, one more decimal digit can be used to further raise the embedding capacity.

We then modify the embedding ratio generation method shown in (1) from the summation of all sides of the reference vertices S_{V_1} to maximum, minimum, and average side length (see blue line in Figure 4) between the reference vertices for each embeddable vertex. The experimental results show that the embedding ratio using minimum side length can achieve highest embedding capacity.

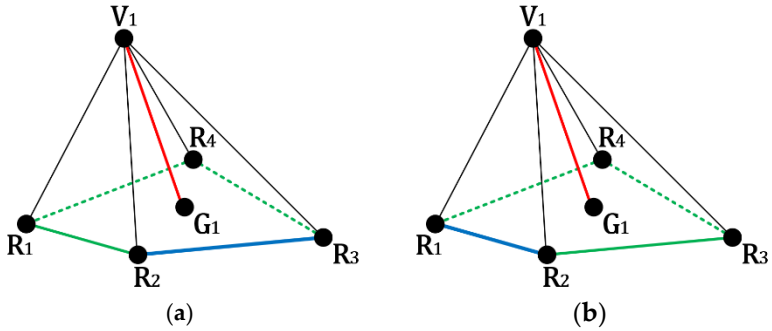


Figure 4. Modified embedding ratio generation method: (**a**) Maximum side length; (**b**) Minimum side length.

3.3. Reduced Number of Reference Vertices for Authentication Code Generation

Tsai et al.'s proposed algorithm uses all the reference vertices for authentication code generation. Therefore, only 24 to 30 percent of vertices [6] can be message-embedded. This improvement adopts a user-defined embedding parameter p within 0 and 1 to control the ratio of used reference vertices for each embeddable vertex. Equation (2) shows the controlling mechanism, where ON_{V_i} and FN_{V_i} are the original and final number of used reference vertices for the embeddable vertex V_i. We then randomly obtain FN_{V_i} reference vertices for the remaining process.

$$FN_{V_i} = \lfloor p \times ON_{V_i} \rfloor \quad (2)$$

However, not each vertex pair in the original input model has one edge to connect them. For the example shown in Figure 2, assume that R_2, R_3, and R_4 are the selected reference vertices for the embeddable vertex V_1. Comparing Figure 2 with Figure 5, no edges are presented to connect the vertices R_2 and R_4. To make the following processes correctly performed, we construct the 'virtual' edge to connect the corresponding two unconnected vertices. G_1 becomes the center of the used reference vertices R_2, R_3, and R_4. Further, the minimum number of used reference vertices is set as 3.

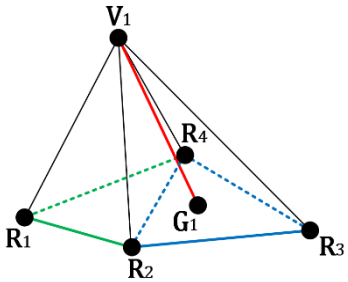


Figure 5. One example of a reduced number of reference vertices for authentication code generation.

4. Experimental Evaluations

This section presents the experimental results obtained from twenty-five 3D polygonal models shown in Figure 6, where N_V and N_F mean the number of vertices and faces separately. Microsoft Visual C++ programming language was used to implement the proposed algorithm on a personal

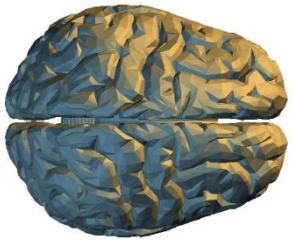
computer with an Intel Core i7-6700K 4.00 GHz processor and 16 GB RAM. The distortion between the original and marked models was measured by normalized root-mean-squared error, which is derived from dividing the root-mean-squared error by the model size.

(**a**) Armadillo

N_V: 172974, N_F: 345944

(**b**) Ateneam

N_V: 7546, N_F: 15014



(**c**) Brain

N_V: 18844, N_F: 36752

(**d**) Brain2

N_V: 294012, N_F: 588032

(**e**) Bunny

N_V: 34834, N_F: 69451

(**f**) Cow

N_V: 46433, N_F: 92864

(**g**) Dinosaur

N_V: 56194, N_F: 112384

(**h**) Dragon

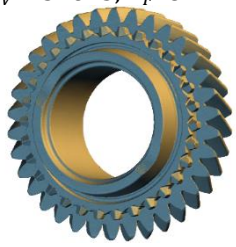
N_V: 437645, N_F: 871414

(**i**) DragonFusion

N_V: 450227, N_F: 900490

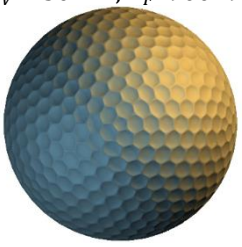
(**j**) Elephant

N_V: 19753, N_F: 39290



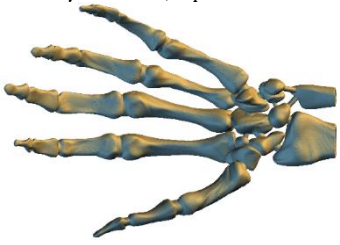
(**k**) Gear

N_V: 250000, N_F: 500000



(**l**) Golfball

N_V: 122882, N_F: 245760



(**m**) Hand

N_V: 327323, N_F: 654666

(**n**) HappyBuddha

N_V: 543652, N_F: 1087716

(**o**) Hippo

N_V: 23105, N_F: 46202

(**p**) Horse

N_V: 48485, N_F: 96966

(**q**) Lion

N_V: 183408, N_F: 367277

(**r**) Lucy

N_V: 262909, N_F: 525814

(**s**) Maxplanck

N_V: 49132, N_F: 98260

(**t**) Rabbit

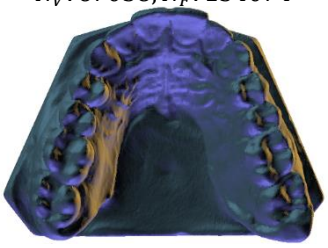
N_V: 67038, N_F: 134074

(**u**) RockerArm

N_V: 40177, N_F: 80354

(**v**) Screw

N_V: 44511, N_F: 89018



(**w**) Teeth

N_V: 116604, N_F: 233204

(**x**) VenusBody

N_V: 19847, N_F: 43357

(**y**) VenusHead

N_V: 134345, N_F: 268686

Figure 6. Visual effects of our twenty-five test models.

First, this section presents the embedding capacity comparison for each improved strategy with Tsai et al.'s proposed algorithm. We also show the model distortion for different embedding ratio generation methods. Finally, we present the experimental results for tamper detection to demonstrate the feasibility of our proposed algorithm.

Table 3 shows the embedding capacity comparison under different embedding thresholds between Tsai et al.'s and our message grabbing method. The experimental results show that our message grabbing method can effectively raise the embedding capacity, exceeding 11.5% on average for all embedding thresholds.

Table 3. Experimental results for our message grabbing method.

Model \ Capacity	[6]			Proposed		
	$T = 1$	$T = 2$	$T = 3$	$T = 1$	$T = 2$	$T = 3$
Armadillo	414,512	269,128	147,280	462,775	300,427	164,377
Ateneam	18,161	12,046	6231	20,263	13,449	6941
Brain	53,982	37,438	21,010	60,231	41,792	23,501
Brain2	766,719	482,906	284,062	855,862	539,170	317,113
Bunny	79,115	49,020	31,137	88,273	54,709	34,704
Cow	108,200	62,873	45,415	120,695	70,198	50,650
Dinosaur	141,047	90,351	50,857	157,553	100,938	56,807
Dragon	1,199,940	806,740	435,755	1,337,835	899,538	485,751
DragonFusion	1,177,286	785,816	440,208	1,313,028	876,273	490,908
Elephant	46,107	30,401	15,883	51,461	33,992	17,758
Gear	625,266	409,512	219,449	697,417	456,854	244,805
Golfball	352,285	227,500	124,838	392,886	253,655	139,112
Hand	868,284	580,290	315,408	968,111	646,857	351,560
HappyBuddha	1,484,268	996,728	534,871	1,656,155	1,111,984	596,324
Hippo	58,474	38,900	20,092	65,037	43,206	22,305
Horse	117,450	75,291	43,598	131,063	83,978	48,625
Lion	430,769	278,123	157,266	480,906	310,553	175,720
Lucy	652,499	433,616	224,752	727,550	483,643	250,656
Maxplanck	117,011	75,540	41,556	130,232	84,161	46,302
Rabbit	162,576	102,799	60,551	181,361	114,605	67,502
RockerArm	98,948	63,072	36,546	110,314	70,262	40,665
Screw	121,398	81,681	44,396	135,366	91,167	49,568
Teeth	278,982	175,831	105,332	311,230	196,141	117,493
VenusBody	45,654	30,275	16,018	51,088	33,817	17,872
VenusHead	314,810	197,871	120,740	351,333	220,870	134,751
Average Improvement				11.56%	11.56%	11.53%

Table 4 shows the embedding capacity comparison for different embedding ratio generation methods, including maximum (MAX), minimum (MIN), and average (AVE) of all the side lengths with the embedding threshold $T = 1$. The embedding capacity using the summation (SUM) method for each model is shown in Table 3. Obviously, the embedding ratio generation method using the minimum side length can have the maximum embedding capacity, improving by about 47.74% on average, with our modified message grabbing method. From the experimental results, we also found that the highest embedding capacity for each embeddable vertex in each 3D polygonal model is between 14 and 16. Four decimal digits of the calculated embedding ratio are used for data embedding, each with three or four bits. Table 5 shows the model distortion for each test model under different embedding ratio generation methods. The model distortion using the MIN method is only 0.009% of the model size, on average.

Table 4. Experimental results for our embedding ratio generation method.

Model	MAX	Ratio	AVE	Ratio	MIN	Ratio
Armadillo	592,331	42.90%	605,771	46.14%	616,170	48.65%
Ateneam	24,908	37.15%	26,248	44.53%	26,987	48.60%
Brain	72,116	33.59%	73,231	35.66%	72,569	34.43%
Brain2	1,007,684	31.43%	1,070,707	39.65%	1,191,820	55.44%
Bunny	114,739	45.03%	117,149	48.07%	120,612	52.45%
Cow	152,848	41.26%	156,472	44.61%	168,732	55.94%
Dinosaur	191,920	36.07%	200,944	42.47%	209,391	48.45%

Dragon	1,660,464	38.38%	1,705,093	42.10%	1,656,703	38.07%
DragonFusion	1,663,575	41.31%	1,688,508	43.42%	1,671,791	42.00%
Elephant	64,613	40.14%	67,376	46.13%	69,126	49.93%
Gear	873,125	39.64%	910,416	45.60%	932,665	49.16%
Golfball	471,018	33.70%	505,152	43.39%	532,600	51.18%
Hand	1,235,389	42.28%	1,250,350	44.00%	1,246,830	43.60%
HappyBuddha	2,062,769	38.98%	2,121,269	42.92%	2,067,096	39.27%
Hippo	80,343	37.40%	84,051	43.74%	85,824	46.77%
Horse	169,728	44.51%	173,291	47.54%	176,905	50.62%
Lion	621,351	44.24%	634,770	47.36%	643,836	49.46%
Lucy	912,968	39.92%	951,119	45.77%	971,868	48.95%
Maxplanck	158,746	35.67%	164,168	40.30%	168,967	44.40%
Rabbit	237,878	46.32%	241,277	48.41%	246,362	51.54%
RockerArm	142,768	44.29%	145,529	47.08%	148,664	50.24%
Screw	171,429	41.21%	172,941	42.46%	171,053	40.90%
Teeth	406,523	45.72%	413,817	48.33%	423,267	51.72%
VenusBody	63,749	39.64%	66,681	46.06%	68,058	49.07%
VenusHead	459,561	45.98%	469,800	49.23%	480,740	52.71%
Average		40.27%		44.60%		47.74%

Table 5. Model distortion between the original and marked models.

Model	SUM	MAX	AVE	MIN
Armadillo	0.003%	0.006%	0.005%	0.004%
Ateneam	0.066%	0.032%	0.027%	0.026%
Brain	0.080%	0.062%	0.043%	0.047%
Brain2	0.004%	0.003%	0.003%	0.004%
Bunny	0.008%	0.013%	0.011%	0.007%
Cow	0.003%	0.002%	0.002%	0.003%
Dinosaur	0.003%	0.004%	0.005%	0.004%
Dragon	0.006%	0.006%	0.005%	0.004%
DragonFusion	0.006%	0.006%	0.005%	0.004%
Elephant	0.014%	0.027%	0.019%	0.011%
Gear	0.004%	0.007%	0.006%	0.004%
Golfball	0.004%	0.005%	0.007%	0.005%
Hand	0.005%	0.005%	0.004%	0.003%
HappyBuddha	0.005%	0.006%	0.004%	0.003%
Hippo	0.019%	0.026%	0.019%	0.013%
Horse	0.007%	0.011%	0.009%	0.006%
Lion	0.018%	0.022%	0.017%	0.013%
Lucy	0.003%	0.005%	0.004%	0.003%
Maxplanck	0.005%	0.006%	0.007%	0.007%
Rabbit	0.005%	0.011%	0.009%	0.006%
RockerArm	0.008%	0.014%	0.012%	0.009%
Screw	0.016%	0.014%	0.011%	0.011%
Teeth	0.005%	0.010%	0.008%	0.006%
VenusBody	0.039%	0.029%	0.020%	0.015%
VenusHead	0.003%	0.008%	0.007%	0.005%
Average	0.014%	0.014%	0.011%	0.009%

This section shows the experimental results for adopting a reduced number of reference vertices during the authentication code generation. When not all reference vertices are used, the residual unused reference vertices may be the embeddable ones in some iterations. Because the number of

embedded vertices is increased, and the total embedding capacity can be effectively raised. Table 6 shows the embedding rate under different embedding parameters for each test model. Obviously, the embedding rate can be effectively raised from 23.93~31.25% to 35.32~42.80%. Tables 7–9 show the embedding capacity comparison for each model under different embedding parameters. When the value of the embedding parameter p is smaller, it means the ratio of used reference vertices is decreased. The total embedding capacity is increased with more embeddable vertices. The improved ratio, on average, can be improved from 47.74% (see Table 4), 94.13%, or 121.37%, to 123.74% with different embedding parameters p using the minimum side length for authentication code generation.

Table 6. Embedding rate under different embedding parameters for each test model.

Model	Embedding Rate			
	$p = 1.00$	$p = 0.75$	$p = 0.50$	$p = 0.25$
Armadillo	26.18%	32.73%	36.13%	36.54%
Ateneam	24.98%	33.46%	38.15%	38.88%
Brain	26.68%	33.07%	36.06%	36.82%
Brain2	29.74%	33.59%	35.55%	35.62%
Bunny	27.45%	33.24%	35.59%	35.73%
Cow	30.08%	33.62%	35.17%	35.32%
Dinosaur	27.85%	37.30%	42.48%	42.56%
Dragon	27.58%	36.05%	40.21%	40.55%
DragonFusion	27.50%	35.53%	40.02%	40.13%
Elephant	24.38%	32.28%	35.70%	36.15%
Gear	26.79%	34.71%	38.93%	39.28%
Golfball	31.25%	30.91%	34.57%	36.91%
Hand	27.42%	35.38%	39.30%	39.41%
HappyBuddha	27.63%	36.16%	40.30%	40.65%
Hippo	26.10%	32.83%	36.21%	37.17%
Horse	27.40%	37.23%	41.91%	41.97%
Lion	26.32%	32.77%	35.98%	36.39%
Lucy	26.17%	32.92%	36.52%	37.37%
Maxplanck	26.01%	32.70%	35.99%	36.40%
Rabbit	27.60%	37.41%	42.35%	42.41%
RockerArm	27.73%	38.30%	42.65%	42.73%
Screw	27.53%	35.44%	39.56%	39.93%
Teeth	27.62%	36.39%	42.73%	42.80%
VenusBody	23.93%	33.06%	38.02%	38.64%
VenusHead	27.62%	37.97%	42.62%	42.66%
Average	27.18%	34.60%	38.51%	38.92%

Table 7. The embedding capacity comparison for each model under $p = 0.75$.

Model	$p = 0.75$							
	SUM	Ratio	MAX	Ratio	AVE	Ratio	MIN	Ratio
Armadillo	620,651	49.73%	771,150	86.04%	789,114	90.37%	801,150	93.28%
Ateneam	28,848	58.85%	33,833	86.29%	35,245	94.07%	36,113	98.85%
Brain	77,514	43.59%	87,689	62.44%	89,430	65.67%	89,719	66.20%
Brain2	1,113,234	45.19%	1,323,805	72.66%	1,364,716	77.99%	1,402,490	82.92%
Bunny	126,784	60.25%	154,571	95.38%	159,829	102.02%	161,943	104.69%
Cow	172,822	59.72%	213,148	96.99%	217,408	100.93%	220,346	103.65%
Dinosaur	226,448	60.55%	270,542	91.81%	279,177	97.93%	284,736	101.87%
Dragon	1,897,529	58.14%	2,189,923	82.50%	2,233,088	86.10%	2,230,750	85.91%
DragonFusion	1,910,059	62.24%	2,247,132	90.87%	2,277,575	93.46%	2,284,915	94.08%
Elephant	73,733	59.92%	88,401	91.73%	90,172	95.57%	91,357	98.14%

Gear	989,284	58.22%	1,173,128	87.62%	1,211,001	93.68%	1,238,561	98.09%
Golfball	418,875	18.90%	502,197	42.55%	522,089	48.20%	537,986	52.71%
Hand	1,332,624	53.48%	1,607,963	85.19%	1,629,149	87.63%	1,638,186	88.67%
HappyBuddha	2,364,308	59.29%	2,729,198	83.88%	2,784,286	87.59%	2,784,237	87.58%
Hippo	86,695	48.26%	104,253	78.29%	106,992	82.97%	108,839	86.13%
Horse	193,485	64.74%	232,812	98.22%	241,042	105.23%	244,307	108.01%
Lion	654,428	51.92%	816,358	89.51%	835,455	93.95%	846,985	96.62%
Lucy	978,076	49.90%	1,172,694	79.72%	1,211,375	85.65%	1,239,243	89.92%
Maxplanck	175,509	49.99%	215,308	84.01%	221,682	89.45%	227,816	94.70%
Rabbit	263,592	62.13%	312,833	92.42%	327,974	101.74%	334,099	105.50%
RockerArm	162,807	64.54%	196,435	98.52%	202,342	104.49%	205,530	107.72%
Screw	191,189	57.49%	221,447	82.41%	224,922	85.28%	225,048	85.38%
Teeth	452,926	62.35%	545,497	95.53%	566,572	103.09%	575,483	106.28%
VenusBody	76,738	68.09%	89,767	96.62%	92,430	102.46%	94,199	106.33%
VenusHead	517,689	64.44%	627,006	99.17%	651,104	106.82%	660,952	109.95%
Average		55.68%		86.01%		91.29%		94.13%

Table 8. The embedding capacity comparison for each model under $p = 0.50$.

Model	$p = 0.50$							
	SUM	**Ratio**	**MAX**	**Ratio**	**AVE**	**Ratio**	**MIN**	**Ratio**
Armadillo	779,943	88.16%	885,585	113.65%	890,101	114.73%	894,890	115.89%
Ateneam	35,858	97.45%	40,180	121.24%	40,925	125.35%	41,456	128.27%
Brain	90,430	67.52%	96,232	78.27%	97,276	80.20%	97,200	80.06%
Brain2	1,339,809	74.75%	1,455,695	89.86%	1,480,357	93.08%	1,492,992	94.72%
Bunny	155,324	96.33%	175,873	122.30%	177,556	124.43%	177,989	124.98%
Cow	213,088	96.94%	232,068	114.48%	234,133	116.39%	234,979	117.17%
Dinosaur	282,584	100.35%	338,080	139.69%	342,089	142.54%	343,994	143.89%
Dragon	2,264,305	88.70%	2,485,317	107.12%	2,510,444	109.21%	2,491,536	107.64%
DragonFusion	2,323,078	97.32%	2,567,574	118.09%	2,585,866	119.65%	2,581,596	119.28%
Elephant	91,355	98.14%	99,234	115.23%	100,394	117.74%	101,377	119.87%
Gear	1,222,781	95.56%	1,366,716	118.58%	1,386,046	121.67%	1,399,906	123.89%
Golfball	540,521	53.43%	592,094	68.07%	602,896	71.14%	608,555	72.75%
Hand	1,636,256	88.45%	1,829,274	110.68%	1,844,593	112.44%	1,849,223	112.97%
HappyBuddha	2,819,907	89.99%	3,094,219	108.47%	3,127,214	110.69%	3,107,199	109.34%
Hippo	105,985	81.25%	117,936	101.69%	119,370	104.14%	120,406	105.91%
Horse	239,368	103.80%	289,708	146.66%	291,669	148.33%	292,486	149.03%
Lion	823,877	91.26%	935,470	117.16%	939,658	118.14%	944,095	119.17%
Lucy	1,210,175	85.47%	1,347,293	106.48%	1,366,173	109.38%	1,381,562	111.73%
Maxplanck	219,920	87.95%	250,033	113.68%	252,351	115.66%	254,049	117.12%
Rabbit	335,830	106.57%	401,153	146.75%	404,071	148.54%	405,735	149.57%
RockerArm	204,968	107.15%	243,940	146.53%	245,722	148.33%	246,705	149.33%
Screw	227,399	87.32%	250,753	106.55%	252,551	108.04%	251,622	107.27%
Teeth	591,592	112.05%	705,657	152.94%	711,382	154.99%	714,163	155.99%
VenusBody	95,500	109.18%	105,800	131.74%	107,417	135.28%	108,608	137.89%
VenusHead	666,516	111.72%	813,406	158.38%	817,330	159.63%	819,809	160.41%
Average		92.67%		118.17%		120.39%		121.37%

Table 9. The embedding capacity comparison for each model under $p = 0.25$.

Model	$p = 0.25$							
	SUM	**Ratio**	**MAX**	**Ratio**	**AVE**	**Ratio**	**MIN**	**Ratio**
Armadillo	790,660	90.74%	898,171	116.68%	902,038	117.61%	905,992	118.57%
Ateneam	36,753	102.37%	40,922	125.33%	41,663	129.41%	42,195	132.34%
Brain	92,907	72.11%	98,315	82.13%	99,335	84.02%	99,268	83.89%
Brain2	1,342,154	75.05%	1,458,624	90.24%	1,483,496	93.49%	1,496,203	95.14%
Bunny	156,111	97.32%	176,644	123.27%	178,348	125.43%	178,757	125.95%
Cow	213,704	97.51%	233,022	115.36%	235,147	117.33%	235,891	118.01%
Dinosaur	283,536	101.02%	339,191	140.48%	343,144	143.28%	345,000	144.60%

Dragon	2,298,323	91.54%	2,509,489	109.13%	2,532,754	111.07%	2,512,309	109.37%
DragonFusion	2,336,787	98.49%	2,576,554	118.86%	2,594,226	120.36%	2,589,329	119.94%
Elephant	92,850	101.38%	100,541	118.06%	101,598	120.35%	102,602	122.53%
Gear	1,242,282	98.68%	1,382,372	121.09%	1,400,308	123.95%	1,413,421	126.05%
Golfball	579,062	64.37%	633,105	79.71%	643,223	82.59%	649,481	84.36%
Hand	1,643,661	89.30%	1,835,045	111.34%	1,849,972	113.06%	1,854,574	113.59%
HappyBuddha	2,863,070	92.89%	3,124,863	110.53%	3,155,207	112.58%	3,133,523	111.12%
Hippo	109,637	87.50%	121,174	107.23%	122,571	109.62%	123,613	111.40%
Horse	239,887	104.25%	290,235	147.11%	292,155	148.75%	292,969	149.44%
Lion	835,199	93.89%	948,496	120.19%	952,092	121.02%	955,782	121.88%
Lucy	1,246,947	91.10%	1,382,862	111.93%	1,400,097	114.57%	1,414,295	116.75%
Maxplanck	223,092	90.66%	253,473	116.62%	255,685	118.51%	257,157	119.77%
Rabbit	336,766	107.14%	402,232	147.41%	405,113	149.18%	406,737	150.18%
RockerArm	205,505	107.69%	244,484	147.08%	246,315	148.93%	247,260	149.89%
Screw	231,044	90.32%	253,407	108.74%	255,109	110.14%	254,139	109.34%
Teeth	593,427	112.71%	707,872	153.73%	713,451	155.73%	716,165	156.71%
VenusBody	97,584	113.75%	107,681	135.86%	109,314	139.44%	110,436	141.90%
VenusHead	667,585	112.06%	814,909	158.86%	818,767	160.08%	821,202	160.86%
Average		95.35%		120.68%		122.82%		123.74%

Finally, Table 10 shows the number of detected suspicious vertices for each model under different embedding parameters. We randomly add or subtract 0.01% of the length, width, and height of the bounding volume of input model from the x, y, and z coordinate values of 50 vertices separately. The proposed algorithm is then used to perform tamper localization. Recall that a region-based tamper detection algorithm can only locate a rough region with some unaltered vertices inside. Therefore, any one vertex that is altered in the embedding block may lead to all the vertices within the block being regarded as suspicious. As the value of embedding parameter is decreased, the embedding block is smaller with fewer vertices. Thus, the number of suspicious vertices is decreased and the accuracy of the located tampered region is higher. The only exception is the results of embedding parameter 0.50 and 0.25 because their numbers of reference vertices for each embeddable vertex are similar.

Table 10. Number of suspicious vertices detected for each model under different embedding parameters.

Vertex Number / **Model**	**Embedding Parameter *p***			
	1.00	**0.75**	**0.50**	**0.25**
Armadillo	617	396	277	272
Ateneam	540	365	265	254
Brain	588	417	281	269
Brain2	597	357	254	263
Bunny	561	358	245	254
Cow	636	367	257	272
Dinosaur	565	414	353	353
Dragon	1011	714	545	488
DragonFusion	553	419	346	339
Elephant	532	356	274	268
Gear	592	373	284	293
Golfball	706	411	247	249
Hand	587	398	311	303
HappyBuddha	685	488	418	429
Hippo	613	426	274	251
Horse	543	356	277	277
Lion	578	418	282	276
Lucy	570	375	271	277
Maxplanck	553	385	267	263
Rabbit	583	393	330	321
RockerArm	586	369	279	283

Screw	500	329	288	275
Teeth	513	368	295	289
VenusBody	498	356	290	290
VenusHead	594	379	286	286

5. Conclusions and Future Studies

This paper proposes three strategies to improve Tsai et al.'s proposed algorithm. We firstly modified the message grabbing method to raise the embedding capacity above 11.5%. Further, we also modified Tsai et al.'s embedding ratio generation method, increasing the embedding capacity to 47.74%. Finally, the strategy adopting a reduced number of reference vertices for message embedding accompanying the above two ones improves the embedding capacity up to 123.74%. The experimental results demonstrate the feasibility of the proposed algorithm with higher embedding capacity, higher embedding rate, and accurate tamper localization. Further, the model distortion between the original and the marked ones is small.

Future studies could fruitfully explore the proposed algorithm further for point geometries without topological relationship between vertices. Thereafter, providing self-recovery ability is another important research issue to be discussed.

Author Contributions: Y.-Y.T. supervised the whole work, designed the experiments, analyzed the performance of the proposed idea and wrote the paper. Y.-S.T. and C.-C.C. perform experiments simulations and collect the results.

Funding: This research was funded by Ministry of Science and Technology of Taiwan under the grant numbers MOST 106-2221-E-468-022, MOST 106-2218-E-468-001, and MOST 106-2632-E-468-003.

Acknowledgments: The authors would like to thank the editor and three anonymous reviewers for their constructive suggestions in improving the paper significantly.

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

References

1. Petitcolas, F.A.P.; Anderson, R.J.; Kuhn, M.G. Information Hiding—A Survey. *Proc. IEEE* **1999**, *87*, 1062–1078.
2. Bors, A.G.; Luo, M. Optimized 3D Watermarking for Minimal Surface Distortion. *IEEE Trans. Image Process.* **2013**, *22*, 1822–1835.
3. Chen, H.K.; Chen, W.S. GPU-accelerated Blind and Robust 3D Mesh Watermarking by Geometry Image. *Multimedia Tools Appl.* **2016**, *75*, 10077–10096.
4. Hou, J.U.; Kim, D.G.; Lee, H.K. Blind 3D Mesh Watermarking for 3D Printed Model by Analyzing Layering Artifact. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 2712–2725.
5. Chen, T.Y.; Hwang, M.S.; Jan, J.K. Adaptive Authentication Schemes for 3D Mesh Models. *Int. J. Innov. Comput. Inf. Control* **2009**, *5*, 4561–4572.
6. Tsai, Y.Y.; Chen, T.C.; Huang, Y.H. A Low-complexity Region-based Authentication Algorithm for 3D Polygonal Models. *Secur. Commun. Netw.* **2017**, *2017*, 1096463.
7. Wang, J.T.; Chang, Y.C.; Yu, C.Y.; Yu, S.S. Hamming Code Based Watermarking Scheme for 3D Model Verification. *Math. Probl. Eng.* **2014**, *2014*, 241093.
8. Bhowmik, D.; Feng, T. The Multimedia Blockchain: A Distributed and Tamper-proof Media Transaction Framework. In Proceedings of the 22nd International Conference on Digital Signal Processing, London, UK, 23–25 August 2017.
9. Fan, M.; Wang, H. An Enhanced Fragile Watermarking Scheme to Digital Image Protection and Self-Recovery. *Signal Process. Image Commun.* **2018**, *66*, 19–29.
10. Huynh-The, T.; Banos, O.; Lee, S.; Yoon, Y.; Le-Tien, T. Improving Digital Image Watermarking by Means of Optimal Channel Selection. *Expert Syst. Appl.* **2016**, *62*, 177–189.
11. Lo, C.C.; Hu, Y.C. A Novel Reversible Image Authentication Scheme for Digital Images. *Signal Process.* **2014**, *98*, 174–185.
12. Qin, C.; Zhang, X.; Dong, J.; Wang, J. Fragile Image Watermarking with Pixel-wise Recovery Based on Overlapping Embedding Strategy. *Signal Process.* **2017**, *138*, 280–293.

13. Singh, D.; Singh, S.K. DCT Based Efficient Fragile Watermarking Scheme for Image Authentication and Restoration. *Multimedia Tools Appl.* **2017**, *76*, 953–977.
14. Wang, X.Y.; Zhang, J.M. A Novel Image Authentication and Recovery Algorithm Based on Chaos and Hamming Code. *Acta Phys. Sin.* **2014**, *63*, 020701.
15. Asikuzzaman, M.; Pickering, M.R. An Overview of Digital Video Watermarking. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, accepted.
16. Fallahpour, M.; Shirmohammadi, S.; Semsarzadeh, M.; Zhao, J. Tampering Detection in Compressed Digital Video Using Watermarking. *IEEE Trans. Instrum. Meas.* **2014**, *63*, 1057–1072.
17. Qian, Q.; Wang, H.X.; Hu, Y.; Zhou, L.N.; Li, J.F. A Dual Fragile Watermarking Scheme for Speech Authentication. *Multimedia Tools Appl.* **2016**, *75*, 13431–13450.
18. Renza, D.; Ballesteros, L.D.M.; Lemus, C. Authenticity Verification of Audio Signals Based on Fragile Watermarking for Audio Forensics. *Expert Syst. Appl.* **2018**, *91*, 211–222.
19. Yeo, B.L.; Yeung, M.M. Watermarking 3-D Objects for Verification. *IEEE Comput. Graph. Appl.* **1999**, *19*, 36–45.
20. Lin, H.Y.S.; Liao, H.Y.M.; Lu, C.S.; Lin, J.C. Fragile Watermarking for Authenticating 3-D Polygonal Meshes. *IEEE Trans. Multimedia* **2005**, *7*, 997–1006.
21. Molaei, A.M.; Ebrahimnezhad, H.; Sedaaghi, M.H. A Blind Fragile Watermarking Method for 3D Models Based on Geometric Properties of Triangles. *3D Res.* **2013**, *4*, 4.
22. Tsai, Y.Y.; Tsai, Y.S.; Chang, C.C. An Improved Region-based Tampered Detection Algorithm for 3D Polygonal Models. In Proceedings of the 4th IEEE International Conference on Applied System Innovation, Chiba, Japan, 13–17 April 2018.