IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 20, 2021, accepted January 8, 2022, date of publication January 18, 2022, date of current version January 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3144533

The First Draco 3D Object Crypto-Compression Scheme

BIANCA JANSEN VAN RENSBURG[1,2], (Student Member, IEEE), WILLIAM PUECH[1] (Senior Member, IEEE), AND JEAN-PIERRE PEDEBOY[2]

[1]LIRMM, University of Montpellier, CNRS, 34090 Montpellier, France
[2]Stratégies, 94150 Rungis, France

Corresponding author: William Puech (william.puech@lirmm.fr)

ABSTRACT 3D objects have come to play an essential role in industry. They can also be very large, sometimes containing millions of vertices, and therefore it is vital that they are compressed, particularly when it comes to uploading them to the web and during real time rendering. The Draco 3D object compression system, proposed by Google, is becoming an industry standard for the compression of 3D objects structured as geometric meshes or point clouds. These 3D objects are important assets which also need to be secured. In this paper, we propose the first 3D object crypto-compression method, which integrates an encryption step into Google's Draco compression scheme, by adding an AES encryption step during Draco's entropy encoding step. Our proposed Draco 3D object crypto-compression scheme is format compliant, has no size expansion and does not require additional information such as an auxiliary file. After the entire decoding process (joint decryption and decompression), the reconstructed 3D object obtained is identical to the one after a standard compression by Draco. Experimental results on real 3D objects and a security analysis show our proposed method is efficient.

INDEX TERMS Multimedia compression, 3D security, crypto-compression, AES encryption, draco, compression.

I. INTRODUCTION

Over the last decade, 3D objects have become an essential part of everyday life. They are used in many different domains such as manufacturing, healthcare and the entertainment industry. 3D objects are often high quality and can be composed of several million vertices. They can therefore occupy a lot of space and need to be compressed during their network transmission and cloud storage. This is especially true when they are uploaded to the web, or need to be rendered in real time.

Over the past few years, many methods for 3D compression have been proposed. In 2017, Dong *et al.* described a progressive compression algorithm that uses the 3D object's attributes [1]. Recently, in 2021, Que *et al.* proposed an octree-based deep learning framework for point cloud compression [2]. In 2019, Doumanoglou *et al.* detailed a comparison of real-time 3D compression methods [3]. Then, in 2020, Liu *et al.* described a comparison of 3D point cloud compression methods [4]. Among these proposed methods is the 3D compression method Draco [5], this has been developed by Google and is quickly becoming the industry standard.

3D objects have become important assets and need to be secured as such, as they are often stored on the cloud or transferred many times during their existence. It is therefore important that they are both compressed and secured jointly. However, 3D object compression first and encryption afterwards is not efficient, because it is not format compliant and the trade off between the compression rate and the distortion is not optimized. This is the reason why over the years, several crypto-compression methods for multimedia have been proposed. These methods are beneficial as they combine the encryption and compression into a single process. This adds an extra layer of security as an additional level of complexity and uncertainty is added, making it more difficult for an attacker to break the system.

During the last twenty years, several JPEG image crypto-compression methods have been proposed where an encryption step is integrated into the JPEG compression method. Van Droogenbroeck and Benedett suggested encrypting only the AC coefficients after the Discrete

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

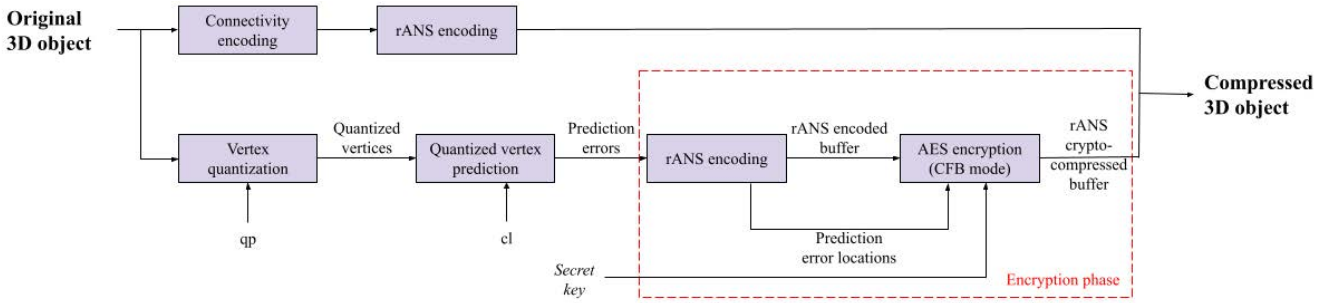


FIGURE 1. Overview of the proposed Draco 3D object crypto-compression method.

Cosine Transform (DCT) [6]. Puech and Rodrigues proposed encrypting the DC coefficients and the lowest frequency AC coefficients [7]. Gmira *et al.* proposed adding a dynamic Hill-Cipher encryption to the quantization step of JPEG compression [8]. Hajji *et al.* [9] and Dridi *et al.* [10] both proposed crypto-compression schemes for images based on chaos.

Crypto-compression methods have also been developed for videos. Dufaux and Ebrahimi described two scrambling methods to hide private data in regions of interest [11]. In 2011, Shahid *et al.* proposed an encryption method for H.264/AVC by selectively encrypting Context Adaptive Video Length Encoding (CAVLC) and Context Adaptive Binary Arithmetic Coding (CABAC) for P and I frames [12]. Then, in 2017, Hamidouche *et al.* proposed an encryption scheme based on the chaos system in the scalable extension of High-Efficiency Video Coding (HEVC) [13].

To the best of our knowledge, we are the first to propose a crypto-compression method for 3D objects. We note that a crypto-compression method based on Draco was requested on Github and was subsequently added by Google to their list of calls for Draco enhancements [14]. In this paper, we propose integrating an encryption step into Google's compression method Draco. During the range Asymmetric Numeral System (rANS) encoding step of the Draco compression scheme, we propose encrypting the encoded vertex prediction errors. This encryption is performed with the AES encryption scheme where the cipher feedback (CFB) mode is used. The proposed method is format compliant and has no size expansion in the encrypted or clear domain. It has a low complexity and does not require additional information such as an auxiliary file.

This paper is organised as follows. First, in Section II, we detail the proposed Draco 3D object crypto-compression method. Then, in Section III, we describe our experimental results and perform a security analysis. Finally, in Section IV, we conclude this paper.

II. PROPOSED DRACO 3D OBJECT CRYPTO-COMPRESSION METHOD

In this section, we detail the proposed Draco 3D object crypto-compression method. Fig. 1 illustrates an overview of the proposed Draco 3D object crypto-compression method. The encryption step is integrated in the rANS encoding of the prediction errors of the quantized vertices. First, in Section II-A, we give an overview of Google's compression method Draco, as well as the compression method Edgebreaker on which Draco is based. Then, in Section II-B, we describe the encryption step integration in the compression method Draco. Finally, in Section II-C, we describe the Draco decoding phase in which the decryption step is integrated.

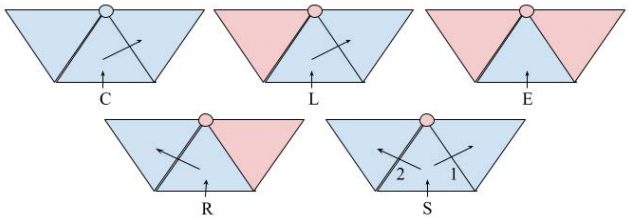


FIGURE 2. The CLERS string allocation for the Edgebreaker compression method.

A. DRACO 3D OBJECT COMPRESSION OVERVIEW

In Fig. 1 we can see the main steps of Google's compression method for 3D objects, Draco. This compression method relies heavily on the compression method Edgebreaker, proposed by Rossignac in 1999 [15]. The Edgebreaker method is used to encode the 3D object's connectivity by encoding each triangle of a manifold 3D object with less than 2 bits.

The Edgebreaker method consists of traversing the connectivity of the 3D object step-by-step by means of a depth-first spiraling triangle spanning tree. At each step, a new triangle is visited and encoded according to which of its neighbouring triangles have already been visited. The current triangle is then encoded with a single character C, L, E, R or S. These characters serve to describe how the current triangle can be reattached to set of already reconstructed triangles during the reconstruction.

Fig. 2 illustrates the CLERS string allocation, where the triangles or vertices in blue are not yet visited, while those in red have been visited, and the arrow indicates the next triangle to visit. If the vertex is not yet visited, then the

triangle is labeled 'C' and we move to the right. If the vertex is visited, then there are four possibilities. If the left triangle is visited, the current triangle is labeled 'L' and we move to the right. If the right triangle is visited, the current triangle is labeled 'R' and we move to the left. If neither are visited, the current triangle is labeled 'S' and we move to the right first and then recursively to the left. If both are visited, the current triangle is labeled 'E' and we end the current loop.

The Draco compression method has two main parameters which are used to compress a 3D object (Fig. 1). The quantization parameter qp, where $qp \in [0, 30]$, corresponds to the number of bits conserved per coordinate, except for $qp = 0$, which signifies that there is no quantization. The parameter cl corresponds to the compression level, where $cl \in [0, 10]$. The value $cl = 0$ corresponds to no compression, and $cl = 10$ to the highest level of compression. The vertex prediction step relies on this parameter. The value of qp is a trade-off between the compression rate and quality, whereas the value of cl is a trade-off between the compression rate and processing time. A vertex prediction step is then performed in order to transmit only the prediction errors. Depending on the compression level cl, the vertex prediction method used is either a delta prediction, a parallelogram prediction, or a constrained multi-parallelogram prediction.

Asymmetric Numeral System (ANS) encoding is an entropy encoding scheme proposed by Duda *et al.* in 2015 [16]. A range Asymmetric Numeral System (rANS) encoding is performed separately after the connectivity encoding and the quantized vertex prediction. A compressed 3D object is then obtained.

B. ENCRYPTION PHASE

In this paper, we propose integrating an encryption step during the rANS encoding of the prediction errors of the quantized vertices. As the proposed 3D object crypto-compression method must be format compliant we decide not to encrypt the CLERS string data. Otherwise this would make the 3D object crypto-compression illegible for the Edgebreaker decoder. Therefore the 3D object's connectivity is not modified.

As we propose only encrypting the prediction errors, then only the geometry is encrypted (Fig. 1). In order to compress the geometry, the vertices of the 3D object are then predicted according to a prediction scheme. This prediction scheme is determined according to the Draco compression level parameter cl, as detailed in Section II-A. The prediction errors, along with other information needed to decode the vertices, are encoded with the rANS encoding scheme, and then stored in the Draco output buffer. The rANS encoded prediction errors are then substituted with their encryption. We note that since the encryption takes place after the rANS entropy encoding, there is no size expansion and therefore Draco's original compression rate is maintained. This is because encryption eliminates redundancy and therefore increases the entropy.

To encrypt the prediction errors, we use the Advanced Encryption Standard (AES) algorithm, proposed by Daemen and Rijmen [17] in 1999. The AES encryption function consists of a set of processing operations which are performed iteratively on a block of 128 bits. The number of iterations, called rounds, depends on the size of the key. A 128 bit key has 10 rounds, and a 256 bit key has 14 rounds. In this paper, we use a 256 bit key. For each round, a round key is generated according to the original 256 bit key. Each processing operation depends on its corresponding round key. The AES algorithm supports many different block encryption modes such as Cipher Block Chaining (CBC), Cipher Feedback (CFB), Counter (CTR), ECB (Electronic Code Book), or Output Feedback (OFB). In the context of our application, the ECB mode should be avoided, because when two identical blocks are encrypted, their encryption is also identical. This is the reason why, as illustrated in Fig. 3 which represents the prediction error encryption step, in our method we use the CFB mode. In this case, only the AES encryption function, and not the decryption function, is used.

In addition to prediction errors, the prediction data contains other information needed to correctly decode the vertices. Markers are necessary to locate the prediction errors in the Draco output buffer (Fig. 3). The structure of the prediction data and therefore the location of the prediction errors depend on several parameters such as qp and cl, but also on the characteristics of the 3D object. From these markers, we are then able to encrypt the correct segment of information after the rANS encoding.

C. DECODING PHASE

In order to successfully decode the 3D object, the decryption is done jointly during the decompression. Fig 4 presents the vertex decoding, decryption and reconstruction process. In order to avoid adding to the output buffer and therefore increasing its size, decryption is performed jointly with the rANS decoding step. The AES decryption in CFB mode is carried out before prediction errors are decoded by rANS. During the rANS decoding step, the decoder is first initialised using prediction error data that remains in the clear domain. Once the decoder has extracted the prediction errors, they are decrypted using the AES algorithm using the CFB mode. The decrypted prediction errors can then be decoded by the rANS decoder in order to reconstruct the 3D object's vertices.

III. EXPERIMENTAL RESULTS

In this section, we detail our experimental results. In Section III-A, we describe our results when our proposed method is applied to the Stanford 3D dataset [18]. Then in Section III-B, we perform a security analysis on our proposed method by performing a differential analysis, comparing the correlation between the Draco compressed 3D object and the crypto-compressed 3D object and analysing the entropy.

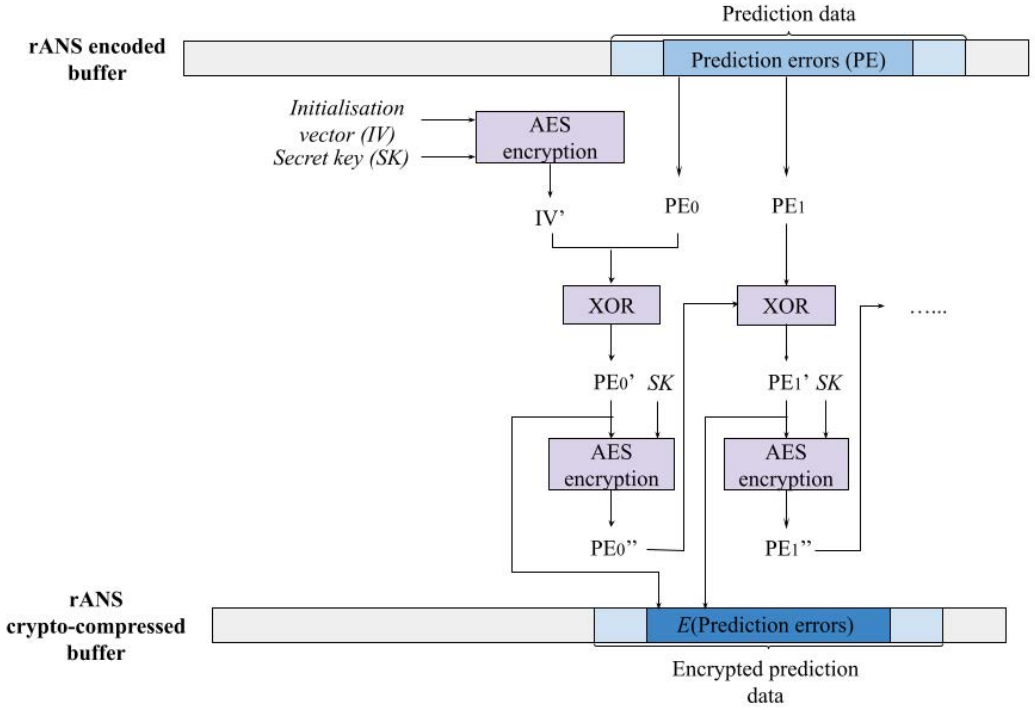


FIGURE 3. Encryption of prediction errors using the AES in CFB mode.

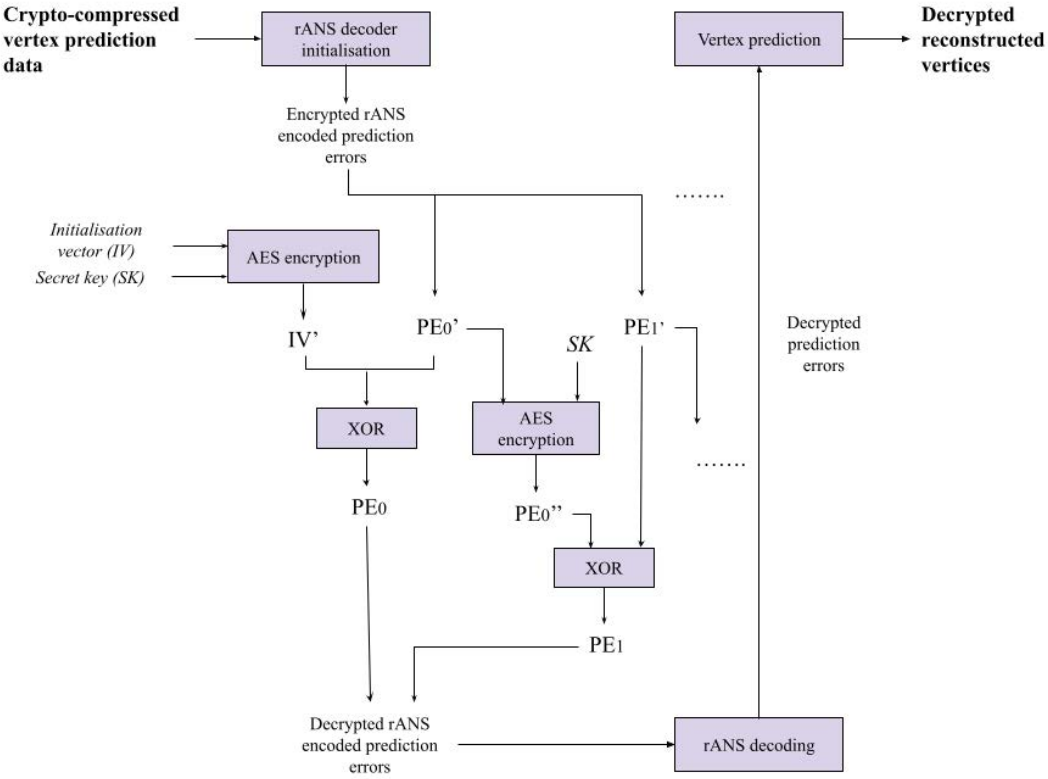


FIGURE 4. The vertex reconstruction process.

We note an original 3D object O, O' corresponds to the decoded 3D object after a standard Draco compression, and O'_e corresponds to the decoded 3D object after a Draco crypto-compression.

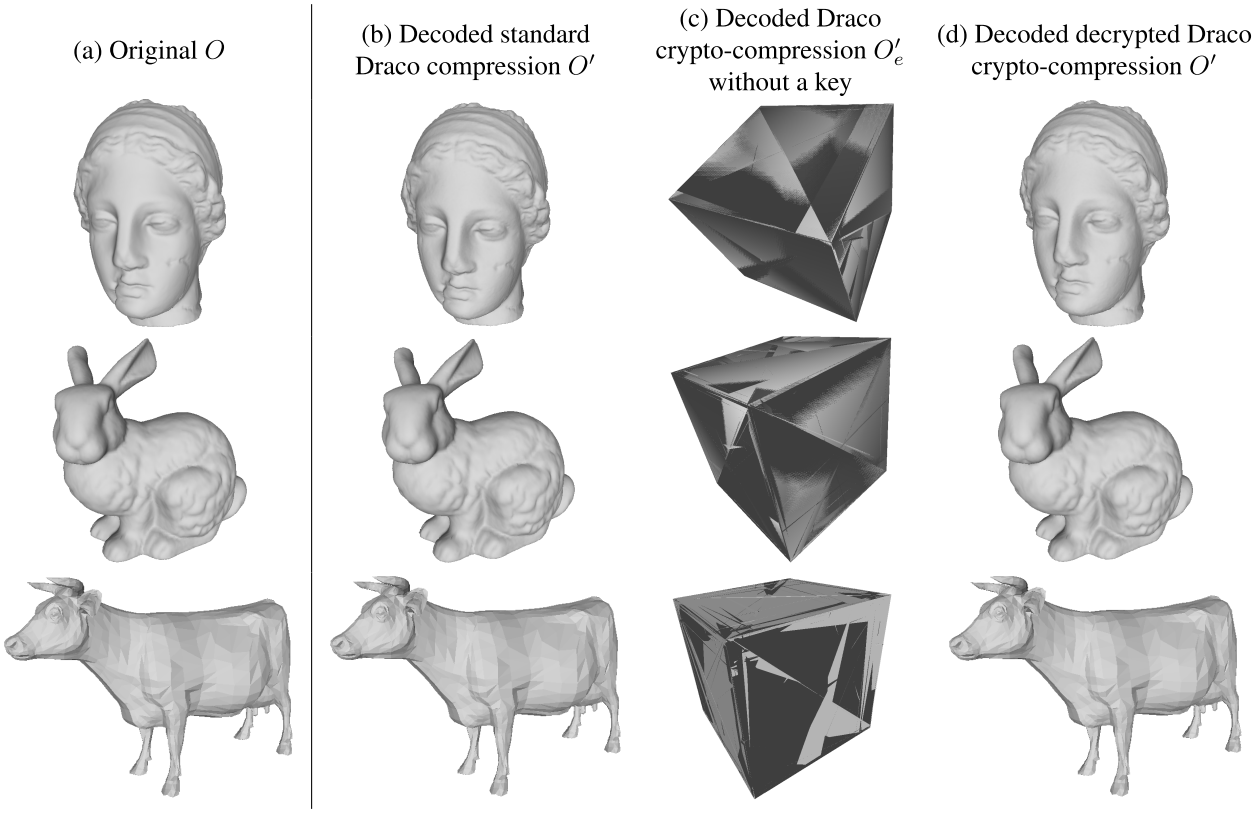


FIGURE 5. Illustrations of the Draco 3D object crypto-compression scheme: a) The original 3D object O, b) the decoded standard Draco compressed 3D object O', c) the decoded Draco crypto-compressed 3D object O'_e without a key, and d) the decoded decrypted 3D object O'. The default Draco parameters $cl = 7$ and $qp = 11$ are used.

A. CRYPTO-COMPRESSION RESULTS

Our experimentation was performed on the Stanford 3D dataset [18]. Each combination of cl and qp is applied to the 11 objects, resulting in a dataset of 3630 different 3D objects. Each 3D object is encrypted with a different pseudo randomly generated key.

Fig. 5 illustrates three examples of our proposed method, *Venus* (100, 759 vertices), *Bunny* (35, 947 vertices) and *Cow* (2,904 vertices). Fig 5.a shows the original 3D object O, then the decoded standard Draco compressed 3D object O' is shown in Fig 5.b, followed by the decoded Draco crypto-compressed 3D object O'_e without a key in Fig 5.c. Then, the decoded decrypted Draco crypto-compressed 3D object O' is shown in Fig 5.d. For these examples, the default parameters provided by Google for Draco are used: $cl = 7$ and $qp = 11$.

Our proposed method is fully reversible. There is no size expansion or data loss. The root mean squared error (RMSE) between a decoded standard Draco compressed 3D object O' and its corresponding decoded decrypted Draco crypto-compressed 3D object O'_e is always 0. This means that these obtained 3D objects are identical and our proposed crypto-compression method is fully reversible in relation to the Draco compression scheme. Therefore, we note both the decoded standard Draco compressed 3D object and its corresponding decoded decrypted Draco crypto-compressed 3D object O'.

TABLE 1. Compression rates and hausdorff distances (HD) between the original 3D objects of the Stanford dataset O and their decoded standard Draco compression O' and decoded Draco crypto-compression O'_e.

	Compression rate	HD $(O, O')(10^{-4})$	HD (O, O'_e)
Mean	40.808	6.116	1.148
Std. dev	19.645	2.892	0.642
Min	14.399	0.805	0.100
Max	68.816	11.712	1.942

Table 1 presents compression rates and hausdorff distances (HD) when comparing the original 3D objects of the Stanford dataset and their corresponding decoded standard Draco compressed O' and decoded Draco crypto-compressed O'_e 3D objects. We define the compression rate as the ratio between the size of the original 3D object and the size of the crypto-compressed 3D object. We note that our proposed Draco crypto-compression scheme has no size expansion in relation to the standard Draco compression scheme. Therefore the compression rate remains constant for O' and O'_e. The HD between O and O' remains low with an order of 10^{-4}.

We note that the proposed encryption method is efficient, as the AES encryption scheme is linear. The average time[1] of the standard Draco compression encoding process of the 11

[1]Algorithms were tested on a 6-core work station with 32 GB of RAM and a Windows based operating system.

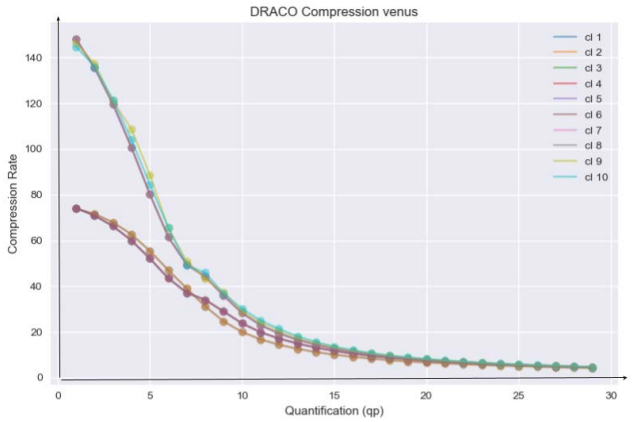


FIGURE 6. Compression rate according to the quantization parameter *qp* for each compression level *cl* for both the decoded standard Draco compression and the decoded Draco crypto-compression of the 3D object *Venus*.

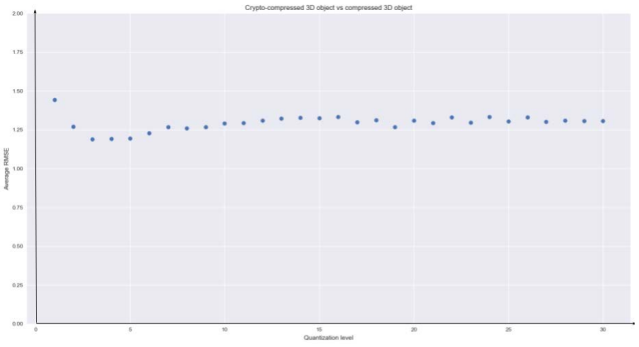


FIGURE 7. Average RMSE between the decoded standard Draco compressed 3D objects of the Stanford datset O' and their corresponding decoded crypto-compression O'_e according the quantization level *qp*.

3D objects is 157.455 *ms*, while that of the proposed Draco crypto-compression method is 157.545 *ms*. The average time of the standard Draco decoding process is 33.636 *ms*, while the average decoding and decryption time of the proposed Draco crypto-compression method is 34.818 *ms*. Therefore we conclude that the time added by the AES encryption and decryption is negligible.

Fig. 6 illustrates a full example of the compression rate according to the quantization level *qp* for each compression level *cl* for the 3D object *Venus*. The curve represents the decoded standard Draco compressed 3D object O', while the points represent the decoded Draco crypto-compressed 3D object. We observe that there are two major trends according to *cl*. This is explained by the type of encoding method used during Edgebreaker. For the parameters $cl \in [1-5]$ where the compression rate starts at about 75, the standard Edgebreaker encoding is used, while for parameters $cl \in [6-10]$, a valence encoding is used and the compression rate starts at about 150. We can observe that our proposed crypto-compression method has absolutely no size expansion, which means that the compression rate is preserved.

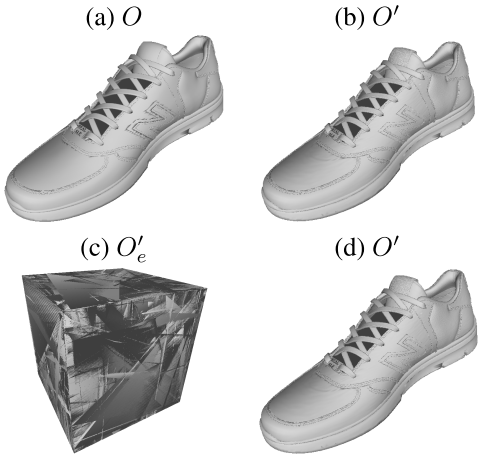


FIGURE 8. Results of the proposed Draco crypto-compression method applied to a manufactured object *Shoe* (Provided by STRATEGIES (https://www.romans-cad.com/)): a) The original manufactured 3D object O, b) The decoded standard Draco compressed 3D object O', c) The decoded Draco crypto-compressed 3D object O'_e, d) The decoded decrypted Draco crypto-compressed 3D object O'.

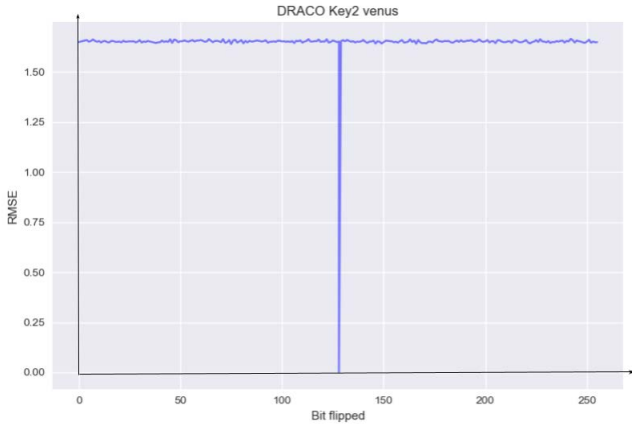


FIGURE 9. Differential analysis of the 3D object *Venus*.

Fig. 7 illustrates the average RMSE between the Draco compressed 3D objects O' of the Stanford dataset and their corresponding crypto-compression O'_e according to the quantization level *qp*. We observe that apart from the low quantization levels where the 3D object is unrecognizable, the average RMSE remains similar around 1.25 when *qp* increases. We note that the RMSE does not vary with the compression level *cl*.

Fig. 8 illustrates the results of the proposed Draco crypto-compression method on a manufactured 3D object *Shoe*[2] composed of 976, 943 vertices. Fig. 8.a presents the original manufactured 3D object and Fig. 8.b the decoded standard Draco compressed 3D object O'. Fig. 8.c illustrates the decoded Draco crypto-compressed 3D object O'_e and Fig. 8.d the decoded decrypted Draco crypto-compressed 3D object O'. There is a compression rate of 34.75 and the RMSE between O' and O'_e is 273.817.

[2]Provided by STRATEGIES (https://www.romans-cad.com/)

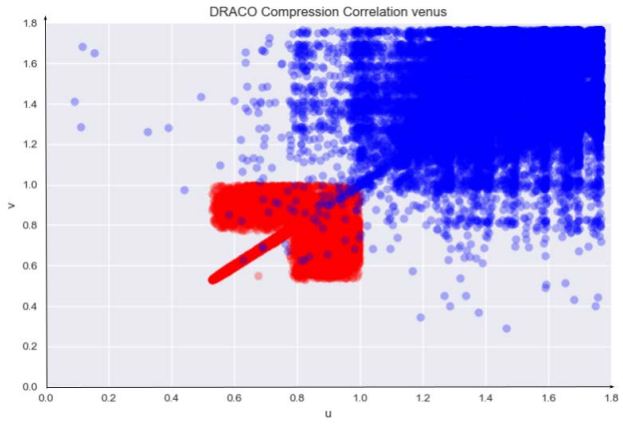


FIGURE 10. Correlation of the decoded standard Draco compressed 3D object *Venus* (red), and the correlation of the corresponding decoded Draco crypto-compressed 3D object (blue) with the parameters $cl = 7$ and $qp = 11$.

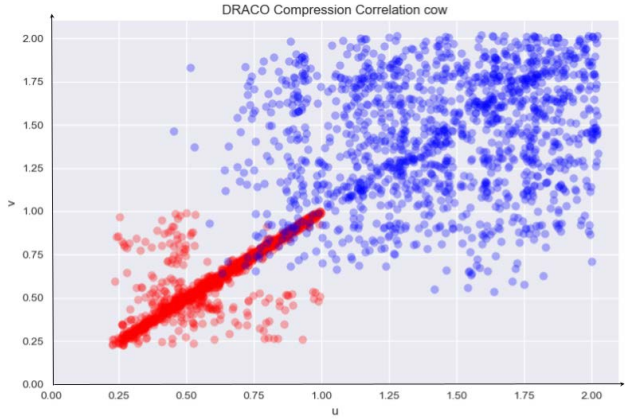


FIGURE 11. Correlation of the decoded standard Draco compressed 3D object *Cow* (red), and the correlation of the corresponding decoded Draco crypto-compressed 3D object (blue) with the parameters $cl = 7$ and $qp = 11$.

B. SECURITY ANALYSIS

In this Section, we examine the security of the proposed crypto-compression scheme in terms of confidentiality. We first perform a differential analysis and measure the 3D object's correlation. Then we study the entropy of the decoded standard Draco compressed 3D object O' and the decoded Draco crypto-compressed 3D object O'_e.

Fig. 9 illustrates the differential analysis for the 3D object *Venus* with the default parameters $cl = 7$ and $qp = 11$. A Key_2 test is performed, where the 3D object is decrypted using keys where only a single bit of the correct 256 bit private key is flipped. The x axis represents the index of the flipped bit in the private key, and the y axis represents the RMSE between the resulting 3D object and the decoded standard Draco compressed 3D object. We note that the key used in position 128 is the correct private key.

We observe from Fig. 9 that while the correct key results in an RMSE of 0, and therefore a perfect decryption, any incorrect key results in an encrypted 3D object with an RMSE of around 1.65. This stable value means that an attacker

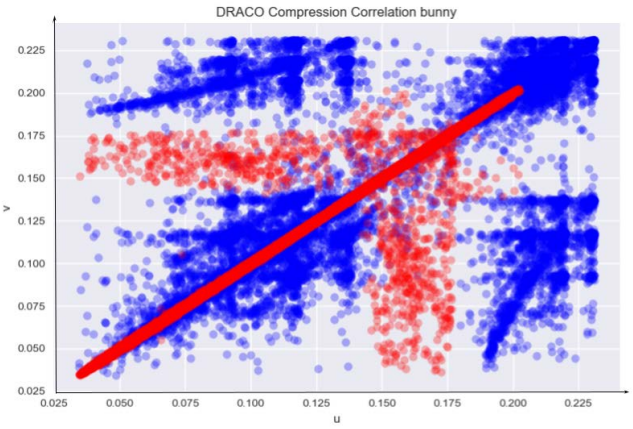
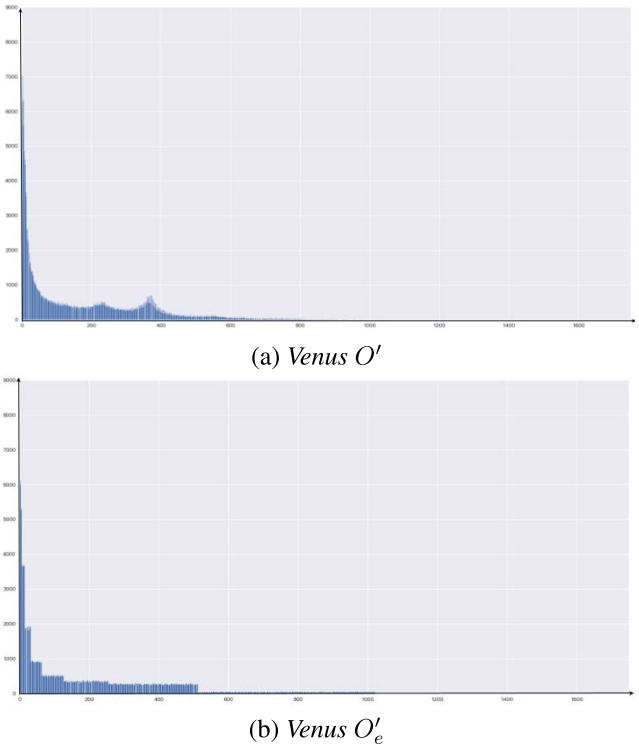


FIGURE 12. Correlation of the decoded standard Draco compressed 3D object *Bunny* (red), and the correlation of the corresponding decoded Draco crypto-compressed 3D object (blue) with the parameters $cl = 7$ and $qp = 11$.



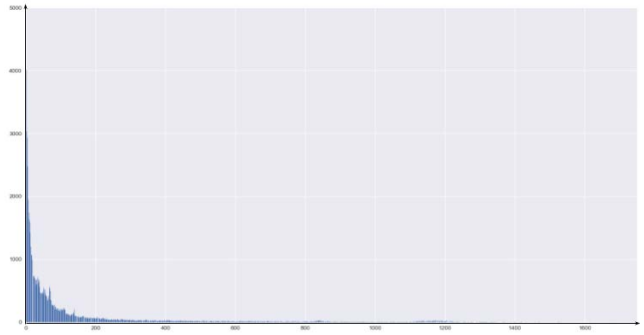
(a) *Venus* O'

(b) *Venus* O'_e

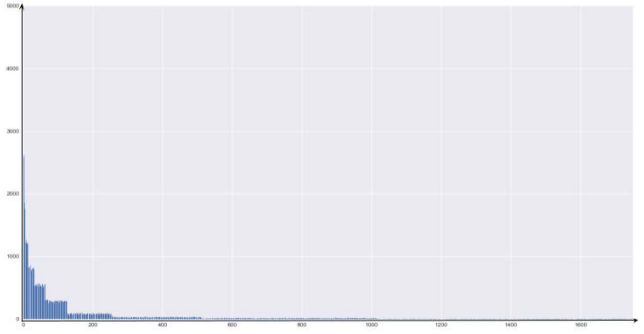
FIGURE 13. Prediction error distribution for: a) The decoded standard Draco compressed 3D object *Venus* O', b) The corresponding crypto-compressed 3D object O'_e (parameters $cl = 7$ and $qp = 16$).

cannot try to converge to the correct value. Therefore an incorrect key which has only a single false bit does not reveal any information about the 3D object. This is to be expected, since the AES encryption scheme is used.

Fig. 10 illustrates the correlation of the euclidean norms of vertices belonging to the decoded standard Draco compressed 3D object *Venus* in red, and the correlation of the euclidean norms of the vertices belonging to the decoded Draco crypto-compressed 3D object *Venus* in blue. Fig. 11 presents the same for the 3D object *Cow* and Fig. 12 the same for *Bunny*.



(a) *Bunny* O'



(b) *Bunny* O'_e

FIGURE 14. Prediction error distribution for: a) The decoded standard Draco compressed 3D object *Bunny* O', b) The corresponding crypto-compressed 3D object O'_e (parameters $cl = 7$ and $qp = 16$).

We note that the default Draco parameters of $cl = 7$ and $qp = 11$ were used. The correlation was performed between pairs of neighbouring vertices, where the neighbours are defined by the Edgebreaker scheme and are separated into two groups u and v. We note that this division into groups was performed in the same manner for both the decoded standard Draco compressed 3D object O' and the decoded Draco crypto-compressed 3D object O'_e.

We observe that there is a strong correlation between neighbouring vertices in the decoded standard Draco compressed 3D object O'. However, we can also note that there is a large portion of uncorrelated data. This can be explained by the vertex order defined by the the Edgebreaker scheme. Once a loop is completed, there can be a jump to the start of the next loop. In comparison, we observe that when the 3D object is crypto-compressed, the neighbouring vertices are no longer correlated.

The Pearson correlation coefficient between the vertices of O' and O'_e is 0.004 for *Venus*, 0.043 for *Cow*, and 0.510 for *Bunny*. These results show that the Draco crypto-compressed 3D objects O'_e is not correlated with its corresponding 3D object O' in the clear domain.

Fig. 13.a presents the prediction error distribution of the decoded standard Draco compressed 3D object *Venus* while Fig. 13.b presents the prediction error distribution of the decoded Draco crypto-compressed 3D object *Venus*, where the parameters are $cl = 7$ and $qp = 16$. Fig. 14.a shows the prediction error distribution of the decoded standard Draco compressed 3D object *Bunny* and Fig. 14.b shows the prediction error distribution of the decoded Draco crypto-compressed 3D object *Bunny*, where the parameters are $cl = 7$ and $qp = 16$.

We observe in Fig. 13.b and Fig. 14.b that after the 3D object has been encrypted, we obtain a histogram which is uniform within intervals of powers of 2. The entropy between the interval $2^8 + 1$ and 2^9 is 7.79 bits/prediction error for the standard Draco compressed 3D object *Venus* and 7.90 for *Bunny*, and a near maximum value of 7.99 bits/prediction error for the Draco crypto-compressed 3D object *Venus* and 7.97 for *Bunny*.

From this security analysis, we can conclude that our proposed Draco crypto-compression method is secure in terms of confidentiality.

IV. CONCLUSION

In this paper, we propose a format compliant crypto-compression method for 3D objects based on Google's compression method Draco. We have integrated an AES encryption step using the CFB mode during the rANS encoding step, where we encrypted the vertex prediction errors. This encryption step is performed jointly with the 3D object compression. The proposed crypto-compression scheme has no size expansion and is completely reversible. To the best of our knowlege, we are the first to propose a crypto-compression method for 3D objects. We have shown with our experimental results that the proposed method is efficient in terms of time and compression rate. Based on an AES encryption with a 256 bit key, we have also shown with a security analysis that the proposed method is secure in terms of confidentiality. In future work, we want to develop a selective crypto-compression based on Draco.

REFERENCES

[1] Y. Dong, X. Yu, and P. Li, "3D model progressive compression algorithm using attributes," in *Proc. 4th Int. Conf. Smart Sustain. City (ICSSC)*. Edison, NJ, USA: IET, 2017, pp. 1–5.

[2] Z. Que, G. Lu, and D. Xu, "VoxelContext-Net: An octree based framework for point cloud compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6042–6051.

[3] A. Doumanoglou, P. Drakoulis, N. Zioulis, D. Zarpalas, and P. Daras, "Benchmarking open-source static 3D mesh codecs for immersive media interactive live streaming," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 190–203, Mar. 2019.

[4] H. Liu, H. Yuan, Q. Liu, J. Hou, and J. Liu, "A comprehensive study and comparison of core technologies for MPEG 3-D point cloud compression," *IEEE Trans. Broadcast.*, vol. 66, no. 3, pp. 701–717, Sep. 2020.

[5] Google. (2014). *Draco 3D Graphics Compression*. [Online]. Available: https://google.github.io/draco/

[6] M. Van Droogenbroeck and R. Benedett, "Techniques for a selective encryption of uncompressed and compressed images," in *Advanced Concepts for Intelligent Vision Systems (ACIVS)*. Ghent, Belgium: Springer, 2002, pp. 90–97.

[7] W. Puech and J. M. Rodrigues, "Crypto-compression of medical images by selective encryption of DCT," in *Proc. 13th Eur. signal Process. Conf.*, Sep. 2005, pp. 1–4.

[8] F. Gmira, S. Hraoui, A. Saaidi, A. J. Oulidi, and K. Satori, "Securing the architecture of the JPEG compression by an dynamic encryption," in *Intelligent Systems and Computer Vision (ISCV)*. Fes, Morocco: IEEE, 2015, pp. 1–6, doi: 10.1109/ISACV.2015.7106192.

[9] J. Hajji, M. A. Ben Farah, M. Samet, and A. Kachouri, "Crypto-compression of images based on chaos," in *Proc. 6th Int. Conf. Sci. Electron., Technol. Inf. Telecommun. (SETIT)*, Mar. 2012, pp. 344–350.

[10] M. Dridi, B. Bouallegue, and A. Mtibaa, "Crypto-compression of medical image based on DCT and chaotic system," in *Proc. Global Summit Comput. Inf. Technol. (GSCIT)*, Jun. 2014, pp. 1–6.
[11] F. Dufaux and T. Ebrahimi, "Scrambling for privacy protection in video surveillance systems," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 8, pp. 1168–1174, Aug. 2008.
[12] Z. Shahid, M. Chaumont, and W. Puech, "Fast protection of H.264/AVC by selective encryption of CAVLC and CABAC for I and P frames," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 565–576, May 2011.
[13] W. Hamidouche, M. Farajallah, N. Sidaty, S. El Assad, and O. Deforges, "Real-time selective video encryption based on the chaos system in scalable HEVC extension," *Signal Process., Image Commun.*, vol. 58, pp. 73–86, Oct. 2017.
[14] (2018). *Please Offer an Option for Encryption*. [Online]. Available: https://github.com/google/draco/issues/376
[15] J. Rossignac, "3D compression made simple: Edgebreaker with Zipand-Wrap on a corner-table," in *Proc. Int. Conf. Shape Model. Appl.*, 2001, pp. 278–283.
[16] J. Duda, K. Tahboub, N. J. Gadgil, and E. J. Delp, "The use of asymmetric numeral systems as an accurate replacement for Huffman coding," in *Proc. Picture Coding Symp. (PCS)*, 2015, pp. 65–69.
[17] J. Daemen and V. Rijmen, *The Design of Rijndael*, vol. 2. Berlin, Germany: Springer, 2002.
[18] M. Levoy, J. Gerth, B. Curless, and K. Pull. (2005). *The Stanford 3D Scanning Repository*. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/

BIANCA JANSEN VAN RENSBURG (Student Member, IEEE) received the M.S. degree in computer science with a specialization in image, games and intelligent agents from the University of Montpellier, France, in 2020. She is currently pursuing the Ph.D. degree with the Laboratory of Informatics, Robotics and Microelectronics of Montpellier, and Stratégies, Rungis, France. Her current interests include multimedia security, computer graphics, and 3D data hiding.

WILLIAM PUECH (Senior Member, IEEE) received the M.S. degree in electrical engineering from the University of Montpellier, France, in 1991, and the Ph.D. degree in signal-image-speech from the Polytechnic National Institute of Grenoble, France, in 1997, with research activities in image processing and computer vision. He worked as a Visiting Research Associate with the University of Thessaloniki, Greece. From 1997 to 2008, he was an Associate Professor with the University of Montpellier, where he has been a Full Professor in image processing, since 2009. His current interests include image forensics and security for safe transfer, storage and visualization by combining data hiding, compression, cryptography, and machine learning. He is the Head of the ICAR Team (Image and Interaction), LIRMM. He has published more than 45 journal articles and 140 conference papers. He is an associate editor for five journals, such as JASP, SPIC, SP, JVCIR, and IEEE TDSC, in the areas of image forensics and security. Since 2017, he has been the General Chair of the IEEE Signal Processing French Chapter. He was a member of the IEEE Information Forensics and Security TC, between 2018 and 2020. Since 2021, he has been a member of the IEEE Image, Video and Multidimensional Signal Processing TC.

JEAN-PIERRE PEDEBOY is currently the Head of the company STRATEGIES for 35 years. He is also an Engineer and he is very well known in 3D modeling of manufactured objects.