Polymorphic Protection of 3D-Models

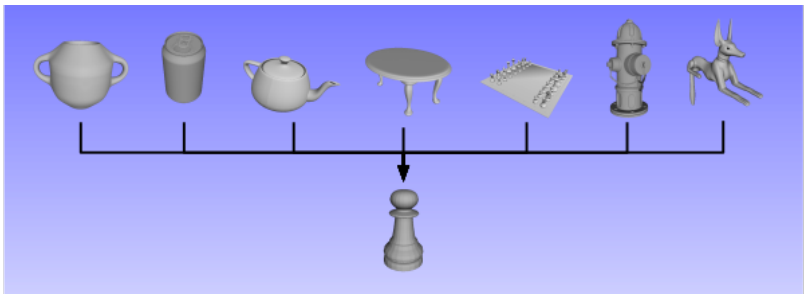

Figure 1: *Polymorphism for 3D object protection. The shape of the protected object is controlled by the user.*

Abstract

This paper introduces *Polymorphic Encryption* as a new way to encrypt 3D-models. Specifically, *Polymorphic Encryption* encrypts a 3D-model given in a specified format into another *chosen* 3D-model of *identical* format. Since the format is not disrupted, this makes it possible to integrate encryption into existing 3D-rendering systems that were not designed to support encryption. Furthermore, as the 3D-model representing the encrypted result can be freely chosen, this allows for application use cases that were previously not readily supported. Various approaches to *Polymorphic Encryption* are presented in the paper. Benchmarks on rendering impacts together with security analysis are also provided.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: 3D models, protection, security

1 Introduction

Progress in 3D-computational graphics and 3D-modeling around these last years have helped democratizing the use of 3D-objects and expanding their fields of application. The movie industry uses them as characters and objects for animated and real-life motion pictures whereas the medical industry uses detailed models of organs for simulations and research. Moreover, progress in virtual reality and augmented reality bring new real life applications through immersive experiences for video games, architecture building or museum virtual visits, online dressing to try clothes, etc. 3D-modeling applications also include 3D-printing technologies which transform virtual artwork into a real object. The demand for high resolution visualization has grown quickly as much as the value of these assets.

3D-object security is a legitimate concern and challenges regarding the protection of the intellectual property appears as the usages keep evolving. There is a strong need to maintain the confidentiality of 3D-objects against unauthorized uses. They may lead to uncomfortable situations such as the leak of unfinished assets or information about production content, counterfeit models, consumer products distribution using 3D printers or other intellectual property violations. Currently, research about 3D-object protection is mainly focused on 3D watermarking to trace the source of a leak. However, watermarking does not prevent unauthorized access to 3D-models. A common strategy to ensure the confidentiality of such information is encryption. A large set of encryption schemes exist among which some have been widely adopted all over the world like Advanced Encryption Standard [NIST 2001]. However, a simple encryption of a 3D-object binary file does not preserve backward compatibility and legacy rendering engines will therefore fail to parse the file which could lead to a critical crash or corrupt the whole 3D-scene. In order to satisfy production workflow needs, encrypted objects should be as usable as unencrypted 3D-object. In a video game production, for example, where a team of level designers, artists and programmers cooperate under the supervision of a producer, the design studio may not be willing to give access to the full scene to a particular team. However, this team may need to have access to visual clues in order to insert graphical elements without colliding with existing ones that would be invisible because of the protection framework. A solution consists in preserving the structure of the 3D-file while the structural elements are encrypted.

In cryptography, the terminology *Format-Preserving Encryption* (*FPE*) refers to ciphers that have the property of producing an output (the ciphertext) in the same format as the input (the plaintext) [Black and Rogaway 2002; Bellare et al. 2009]. The exact meaning of 'format' depends on the targeted application use case. One could for instance want to guarantee that:

- encrypting a 16-digits credit card number remains a 16-digit number after encryption;
- the encryption of a text made of English words results in another collection of English words;
- an address composed of a number, a street, a postcode and a town is preserved by the encryption process.

The idea is to extend this principle of *FPE* to 3D-objects. Thus, the aim is to transform a 3D-object into another 3D-object. Specifically, a protected 3D-object must remain a 3D-object, with correct 3D-properties (i.e. non-disturbing appearance).

FPE techniques have already been considered by introducing the concept of *Geometry-Preserving Encryption* (*GPE*) [Éluard et al. 2013; Jolfaei et al. 2015]. In those works, the solutions produce 3D-objects that yield big visual perturbations compared to the original object or to a 'classical' 3D-object (Figure 2).

Initial solutions simply relied on point or coordinate permutations. The advantage is that the coordinates of the points are not changed by an arithmetic operation which ensures that the deprotected object will be identical to the original object, and the protected object will not require more storage space than the initial object. On the other

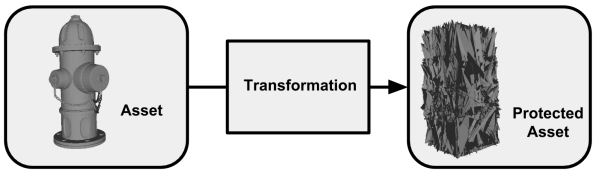

Figure 2: Geometry-Preserving Encryption *(GPE).*

hand, this simplicity does not allow to have a very robust solution against attacks (cryptographic or 3D).

More sophisticated solutions have subsequently been proposed [Éluard et al. 2014; Jolfaei et al. 2015] that use arithmetic operations. The visual impact of these protections is still very high and it is not always possible to guarantee the previous two properties, i.e. strict recovery of the deprotected model and the occupied space preservation. However, one of the proposed solutions is completely different from others [Éluard et al. 2014]. In the case of `Fragment Scaling`, the protected object keeps a good topology, which allows controlling the visual impact. However, it is possible to recognize the original object.

In this paper, we will define a new *FPE* technique in which we are able to choose the aspect of the protected 3D-object. Figure 3 exemplifies what we call *Polymorphic Encryption* (*PE*).

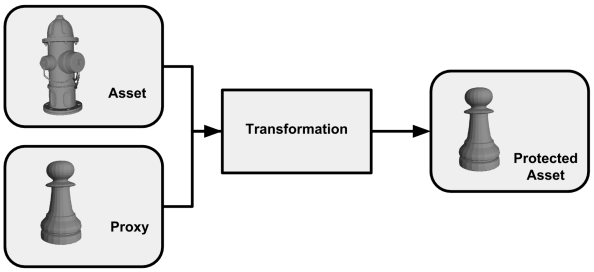

Figure 3: Polymorphic Encryption *(PE).*

The idea behind polymorphism is to constrain the result of the protection so that the protected object looks like an object we chose. The principle is to protect a 3D-object (called Asset) by impersonating another 3D-object (called Proxy) defined by the owner of the Asset. The resulting 3D-object looking like the Proxy, is a Protected Asset (Figure 3). More formally, we define *PE* with a function θ (Figure 4). This function takes an Asset and a Proxy as parameters and produces a Protected Asset which is visually identical to the Proxy. The inverse function takes a Protected Asset and retrieves the corresponding Asset.

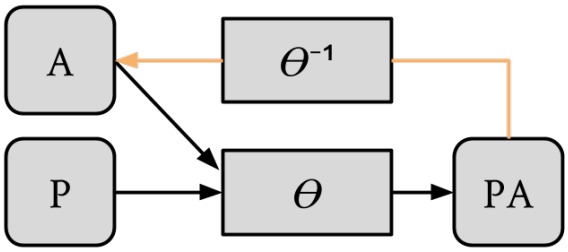

Figure 4: *Transformation schema with θ function.*

The purpose of this article is to demonstrate that this approach is possible. We first propose two approaches to implement this *Polymorphic Encryption* (Section 2). We then present an example θ function and discuss security (Section 3). We also show the visual impact and the impact on the rendering process using multiple measures (Section 4). We summarize the advantages and disadvantages of *Polymorphic Encryption* and give suggestions for improvement (Section 5).

2 3D Polymorphic Encryption

2.1 Surface-Based Approach (PE-S)

The shape of an object is given by the set of surfaces ($\mathscr{S}$) that composes it. The *Surface-Based Approach* uses this property to produce a Protected Asset that takes the form of Proxy (Figure 3).

2.1.1 Protection

The underlying principle is to take every point of the Asset and project them on Proxy surfaces. While effective, this approach is complex because we have to determine a point that exactly belongs to a surface or to choose a point on a given surface and determine the projection function. To simplify the approach, we assume that the Proxy is closed and transform the set of surfaces ($\mathscr{S}$) into a set of polyhedra ($\mathscr{V}$) (Figure 5). Now, we only have to project a point into a chosen volume. Classically, the Proxy can be represented as a set of voxels ($\mathscr{V}$). This transformation into voxels may differ from the classic one ([Kaufman et al. 1993; Nooruddin and Turk 2003]). For example, multiple surfaces can be found in the same voxel. To limit the number of voxels, it is also possible to consider 'polyhedron voxels' rather than cubic voxels.

Figure 5: *Illustration of the voxelization process of the Proxy.*

This transformation is configurable in order to control the 'deformation' that will be applied to the Proxy (Figure 5). The voxel size is indeed tied to the deformation. Bigger voxels approximate less accurately the surface of the Proxy than smaller voxels. There is more space available from the surface and thus more possible dispersion of the projected points.

The projection function (θ) is relatively simple, it takes a point and builds a new point in a constrained space. For the constrained space, the function chooses a voxel (we have to provide the description of the Proxy $\mathscr{V}$ or having the chosen voxel (v_i) as parameter). The inverse function (θ^{-1}) uses only a point of the Protected Asset to reconstruct the corresponding point of the Asset. Several functions can be used and one is described in Section 3.

The result of the projection is a point cloud characterized in that each point belongs to a voxel of the Proxy. As indicated in Section 1, the result must be a 3D-model, it is necessary to add surfaces on this point cloud. To do this, we use a surface reconstruction algorithm that takes our point cloud as input and computes a set of corresponding surfaces. The simplest technique is the convex hull. It allows very simply to get an envelope that cover all the points (Figure 6).

Depending on the chosen Proxy and the chosen surface reconstruction technique, the result can be more or less similar to the Proxy. There are more elaborate surface reconstruction techniques that

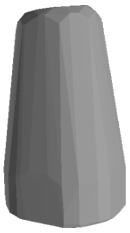

Figure 6: *Convex hull reconstruction on point cloud.*

give better results (Figure 7). Unfortunately, the result strongly depends on the shape of the point cloud (and on the parameters used). Thus, all the techniques do not give the same results for the same set of points, so we have to choose the most adapted method for the considered point cloud.

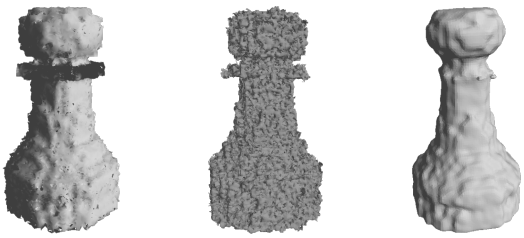

Figure 7: *Classical reconstruction techniques (Ball pivoting / Marching cube / Poisson) on the same point cloud of a Protected Asset.*

2.1.2 Deprotection

The deprotection process is relatively simple. First, we take the points of the Protected Asset and we apply the inverse of the transformation function to obtain the vertices of the Asset. Then we apply a surface reconstruction to retrieve the Asset.

Due to the surface reconstruction, it is possible that the deprotected object is not exactly the original object. If the application use case does not support this type of effect, we have to keep the original surfaces by recording the original connectivity with the corresponding projected points. In this case, there will be no surface error upon deprotection. However, it must be ensured that these surfaces remain inside the Protected Asset to avoid creating artefacts. With the use of surface reconstruction, we must ensure that the Proxy is convex or we must use a convex hull. Upon deprotection, it will remove surfaces added by the protection and keep those corresponding to the Asset.

This technique produces results regardless of the Asset and the Proxy but the results depends on the sizes of the two objects. On one side, we have an Asset composed with p points; on the other side, we have a Proxy described with v volumes. This relationship between p and v influences the result of protection. The ideal case is when p and v are similar; aberrations occur when p and v are very different. If $p \gg v$, a lot of vertices is projected into one voxel. In this case, we will have a very dense point cloud and local perturbations. In Figure 8, the Proxy is decomposed into 2146 voxels and we try to project Asset into it. The first version of the Asset has 75,121 vertices and the second version has 1,786,37 vertices. We see the difference of the aspect of the Protected Asset due to the difference of the number of vertices. Conversely, If $v \gg p$, we try to project few vertices in a large number of voxels. In this case, when we apply surface reconstruction techniques, we obtain object not close to the Proxy due to a lack of resolution.

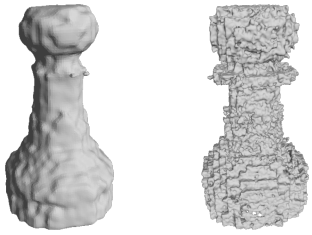

Figure 8: *Poisson reconstructions (75,121 vertices and 1,786,737 vertices)*

2.2 Volume-Based Approach (PE-V)

The *Surface-Based Approach* provides Protected Assets close to the surface of the Proxy with more perturbed surfaces. Indeed, the transformation of surfaces into voxels implies that many points of the Protected Asset lie outside of the chosen Proxy. It provides a degraded version of the 3D-object which can visually impact the scene. Based on this observation, another polymorphic strategy consists in placing the elements of the Asset inside the volume of the chosen Proxy.

2.2.1 Protection

First, we need to define the authorized volume for the transformation. Therefore, we define a set of disjoint parallelepipeds ($\mathscr{V}$) that are contained by the Proxy (Figure 9).

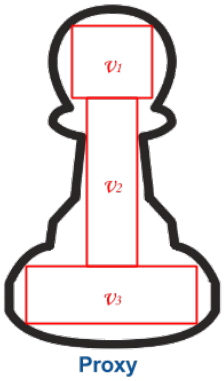

Figure 9: *Set of inscribed parallelepipeds*

We assume that a generic function V is able to compute the volume of the 3D-shapes passed as a parameter. The volume of the set of parallelepipeds is thus upper bounded by the volume of the Proxy.

$$V(\mathscr{V}) = \sum_{i=1}^{m} V(v_i)$$

The cover ratio (t) is defined as the ratio between these two volumes. This threshold is used to enforce a minimum volume available to store the points of the Asset. The closest the two volumes are, the better distributed will be the protected vertices within the Proxy.

$$\frac{V(\mathscr{V})}{V(Proxy)} \geq t$$

Then, as for *PE-S*, the transformation θ projects each point of the Asset into the set of volumes v_i. The result of the protection is a point cloud with the property that each point belongs to the inscribed volume of the Proxy ($\mathscr{V}$) (Figure 10).

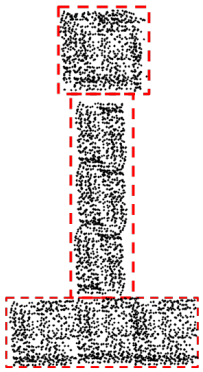

Figure 10: *Asset's points projection in Proxy's decomposition*

Finally, we add points and surfaces from the Proxy to the generated cloud of points to create our Protected Asset (Figure 11). The original surfaces could be conserved because they will lie inside the Proxy envelope.

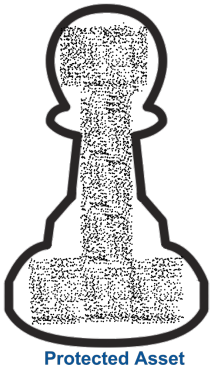

Figure 11: Volume-Based Approach *result*

2.2.2 Deprotection

Like for *PE-S*, we take the points of the Protected Asset and apply the inverse of the transformation function (θ^{-1}) to find the points of the original Asset. If the original surfaces have been kept during the protection process, applying the θ^{-1} function and striping the Proxy from the Protected Asset allows recovering the original Asset. On the other hand, if the original surfaces have not been recorded during the protection process, another step will be necessary after the inverse transformation function and the removal of the Proxy. Indeed, with these two steps, we only obtain the original points and we need to apply a surface reconstruction algorithm to recover a model which looks like the Asset.

2.2.3 Inscribed volumes

Defining a set of inscribed volumes in our Proxy is a simple concept but not an easy task. We decided to consider two methods to find inscribed volumes: by voxelization or by inflating spheres.

The first method is simpler. Voxels are generated on our Proxy and we only keep the voxels inscribed in our 3D-object. Then, in order to optimize the protection process, adjacent voxels may be merged together in order to obtain a smaller set of parallepipeds (Figure 12).

For the second method, we need a grid of volumes, for instance voxels of same size to seed spheres inside our 3D-object. The sphere packing problem has been well studied ([Hales 1992]) and tries to

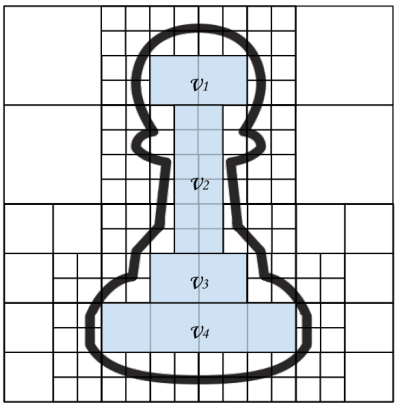

Figure 12: *Set of volume defined by voxelization*

spread an optimal number of identical spheres in the volume corresponding to an 3D-object. In our case, we need to cover a maximum volume with a minimal set of different spheres (with different radius). Once a sphere is seeded inside a voxel or a cell of our grid, we start to inflate this sphere until it meets any surface of the Proxy. When the sphere meets a surface or another sphere, we try to shift the center of the sphere in the opposite direction of the hit in order to allow further growth. We stop once there is no more solution to take more space inside the Proxy and start to inflate another sphere. We keep inflating spheres until the volume of the set of spheres reach our threshold (results will look like Figure 13).

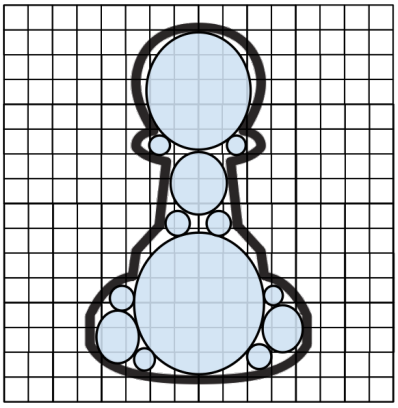

Figure 13: *Set of volume defined by inflating spheres*

Like for *PE-S*, this technique produces results regardless of the Asset and the Proxy but now the result depends on the relationship between the Proxy's inside volumes and the size of the Asset. If we try to project a lot of vertices in a small space with few volumes, we will have a very dense point cloud which could be easy to detect. Moreover, the Proxy selection is important for simplification and optimization of the protection application. The more convex the Proxy is the smaller the set of inscribed volumes will be.

3 Projection Function and Security Considerations

A first strategy to attack this protection is to attempt reversing the applied projection. The security level of our protection regarding this strategy is directly the security level of the projection function. In order to protect this solution against this kind of attacks and ensure a good security level, we can use a secret key to randomize the behavior of the projection function (Figure 14).

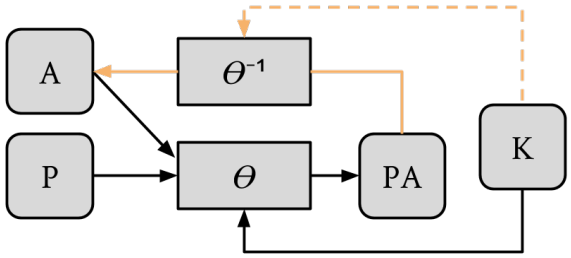

Figure 14: *Transformation function with secret key*

For illustration, we propose a definition for θ that is usable regardless of the approach (*PE-S* or *PE-V*). To simplify the definition of θ, we apply it on each point of the Asset. As a result, the first parameter of θ is now a point and not an Asset. First, we transform the set of volumes into spheres contained in these volumes (Figure 13). Thus, for each volume v_i, we have a center o_i and a radius r_i. We group these tuples in a set $\mathscr{C} = \bigcup_i (o_i, r_i)$. The idea is to define a function θ for projecting each point of the Asset in these various spheres (Algorithm 1).

For each point p, we pseudo-randomly choose a sphere i within which it will be projected (with the key K). Then, we pseudo-randomly determine a distance d between the center of the sphere and the projected point (smaller or equal to the radius). We place the new point p' on the segment connecting p and the center of the sphere o_i using the distance generated d. Finally, we compute the ratio α between the two collinear vectors. The result is a tuple with the point p' and the ratio α.

Algorithm 1 Protection

procedure $\theta(p,\mathscr{C},K)$
  $i \leftarrow$ *key-based random* $(0, size(\mathscr{C}) - 1, K)$
  $d \leftarrow$ *random* $(0, r_i)$
  $p' \leftarrow p' \in [p, o_i] \wedge ||\overrightarrow{o_i, p'}|| = d$
  $\alpha \leftarrow \overrightarrow{o_i, p} = \alpha \overrightarrow{o_i, p'}$
  return(p', α)

To easily incorporate this information in the Proxy to obtain the Protected Asset, the points p' are added to the list of vertices and we set α as new length of the normal at the point p'. For the sphere centers o_i, we add them as new points in the list of vertices.

Remark: it is possible the unlink p' and the corresponding α by reordering the list of α with a key-based sort algorithm before modifying the normals. To unprotect, we first re-sort the list of α with the same key K.

To unprotect the object, we just have to retrieve for each point p' the corresponding sphere o_i and compute p with the set of normals ($\mathscr{N} = \bigcup_i n_i$).

Algorithm 2 Unprotection

procedure $\theta^{-1}(p', \mathscr{C}, \mathscr{N}, K)$
  $i \leftarrow$ *key-based-random* $(0, size(\mathscr{C}) - 1, K)$
  $p.x \leftarrow (n_i.length \times \overrightarrow{o_i, p'}).x + o_i.x$
  $p.y \leftarrow (n_i.length \times \overrightarrow{o_i, p'}).y + o_i.y$
  $p.z \leftarrow (n_i.length \times \overrightarrow{o_i, p'}).z + o_i.z$
  $n_i.length \leftarrow 1$
  return(p)

In this case, the overhead is one point per volume used to model the Proxy. If normals for points are not present in the Asset, we can use w[1] for example in order to store α. In this case, the overhead is one point per volume used to model the Proxy and one coordinate per point. If the Proxy is kept during the protection process, its definition should be added to the overhead estimation.

Inspired from previous paper ([Éluard et al. 2013]), similar attacks have been experimented. The goal of all of these attacks is to retrieve or identify the Asset with using only the Protected Asset and functions based on the geometry of this Protected Asset. Results were good on many *GPE* protections such as `Point-Shuffling`, `CoordinateShuffling` and also `Dithering` depending of the distortion parameters. The results are more mitigated on these polymorphic protections, the only available information corresponding to the geometry of the Proxy and not of the Asset.

Statistical analysis have also been proposed [Jolfaei et al. 2015] to evaluate the strength of 3D protections. We used the *Similarity Analysis* approach to realize our attacks on *GPE* and *PE* by analyzing the coordinates frequency and distribution over a database of plaintext models and protected models. On *PE*, we only were able to recognize the Proxy used. An incomplete analysis has been made for the other 3 strategies. However, from our empirical observations, the *Plaintext Sensitivity Analysis* is not relevant because similar outputs can be obtained with the same Proxy and key but with two different Assets having similar number of vertices. For the *Spatial Randomness*, it depends of θ used but with our example explained before in the same section, our points are randomly distributed in defined volumes.

In cryptography there are establyshed benchmark that allow to effectively test and validate the security level of newly proposed security primitives. In 3D, this kind of benchmark does not exist. The different attacks that have been implemented for the previous protections algorithms gives no result on this polymorphic protection. Consequently, this lack of formal framework prevents us to conclude on the security level of our polymorphic protections. We can however be legitimately confident since the Protected Asset only gives information regarding the Proxy and not the Asset.

4 Rendering complexity benchmarking

3D-protection like *GPE* or *PE* significantly alter the nature of 3D-objects. Related works on *GPE* introduced studies on performance degradation and highlighted the consequences on the rasterisation process [Foley et al. 1993; Dachsbacher et al. 2009], rendering time and frame rate. For instance, *Geometry-Preserving Encryption* protections impact rasterisation with more overdraw of facet due to z-buffer overwrite and much larger facets to draw.

Two metrics have been introduced in precedent works on *GPE* algorithms to measure the impact of introducing 3D-object protection [Éluard et al. 2013; Éluard et al. 2014].

- *Average surface per facet*: the idea is to record the surface of each facet and compute the average. Larger facets implies more time needed to draw them. Increasing the facets surface average thus increase the rendering time.
- *Average surface intersections*: the idea is to trace rays in the bounding box, count each intersection with a face per ray and compute the average. The average number of intersections per ray represents the *z-buffer* refresh ratio. More overdraws

[1] w is 4th coordinate of a point, usualy used in Game Engine. w is a normalization of z (depth) and defined by $w = 1/z$.

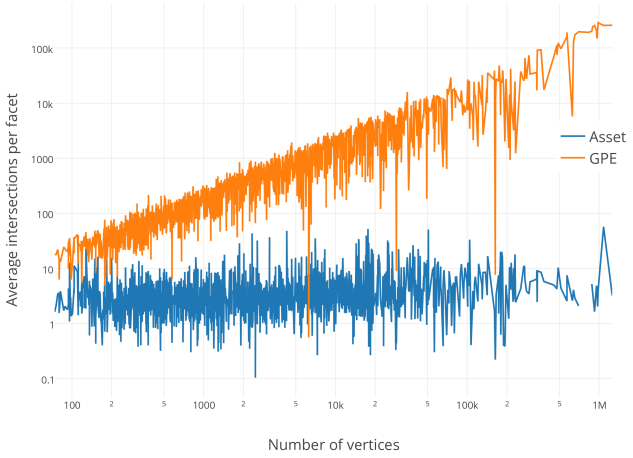

Figure 15: GPE *evaluation by average surface intersections*

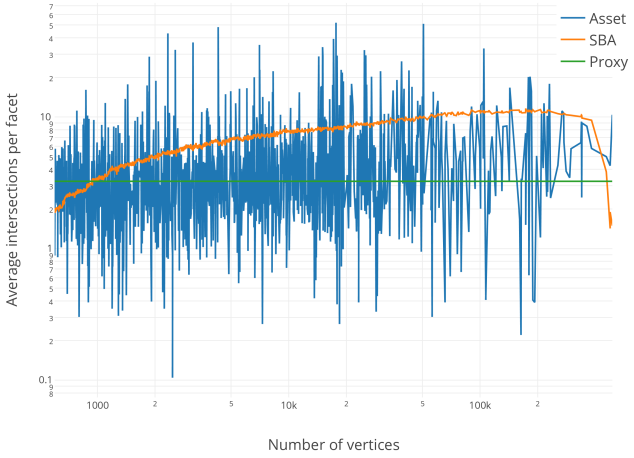

Figure 17: PE-S *evaluation by average surface intersections*

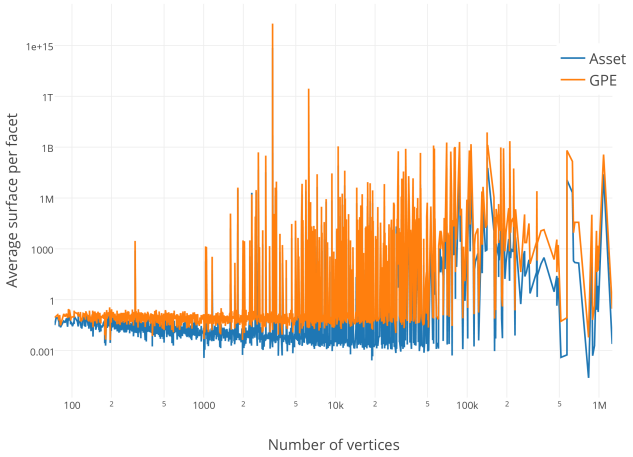

Figure 16: GPE *evaluation by average surface per facet*

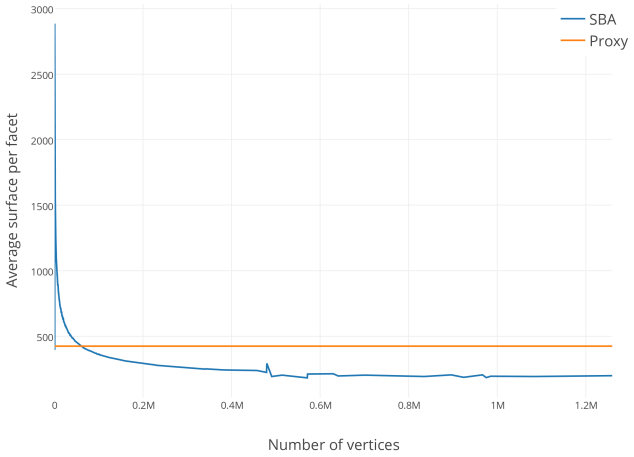

Figure 18: PE-S *evaluation by average surface per facet*

implies more time to render the 3D-object. Increasing the intersections per ray increase the rendering time.

Adopting these two metrics, we benchmarked our two *PE* approaches over a collection of more than 2,000 3D-objects of various shapes and complexity (100 to 1,000,000 vertices). *PE* approaches will be compared to two *GPE* protections: `PointShuffling` and `CoordinateShuffling` techniques.

The ray-tracing complexity (Figure 15) of the *GPE* Protected Assets is much bigger than for the Asset results. With these techniques, facets tend to overlap each others and the number of faces hits per ray grow quickly with the number of facets. With shuffling techniques, on a set of points for example, there are few possibilities to obtain Protected Asset without additionnal intersections than the original one.

For the surfaces complexity (Figure 16), the *GPE* result evolve following the Asset results due to the preservation of the space occupied by the Asset.

4.1 *Surface-Based Approach* Evaluation

The *PE-S* produces 3D-object with a number of vertices close to the Asset when the faces are obtained with a surface reconstruction algorithm. Normally, we should obtain ray-tracing results close to the Proxy due to the distribution of the Asset's vertices close to the Proxy's facets. The facets surfaces of the Protected Asset will decrease with the increase of the number of vertices in the Asset because the Protected Asset will have more points in a same space for the same Proxy.

The ray-tracing complexity (Figure 17) of the *PE-S* results are of the same order of magnitude than the results for the Asset. There is a lower growth of the number of faces hits per ray than *GPE* techniques. The ray-tracing complexity is constrained by the shapes chosen to describe the Proxy. The vertices of the protected Asset are close to the Proxy's skin and, after surface reconstruction, the results could be better than the Asset itself but should be lower bounded by the Proxy. However, results may vary depending on the surface reconstruction technique. For example, the reconstruction was not smooth or perfectly complete without a sufficient amount of points and some points were ignored with very dense clouds. These two facts explain the strange behavior (lower results than the Proxy) at the beginning and the end of the curve.

For the surfaces complexity (Figure 18), the *PE-S* results in a logarithmic way with the increase of the number of Asset's vertices. Indeed, vertices from the protected Asset are randomly distributed over the surface of the Proxy. The more vertices there are in the Asset, the more vertices will be in the same space in the protected Asset which explains the decrease of the average surface ratio.

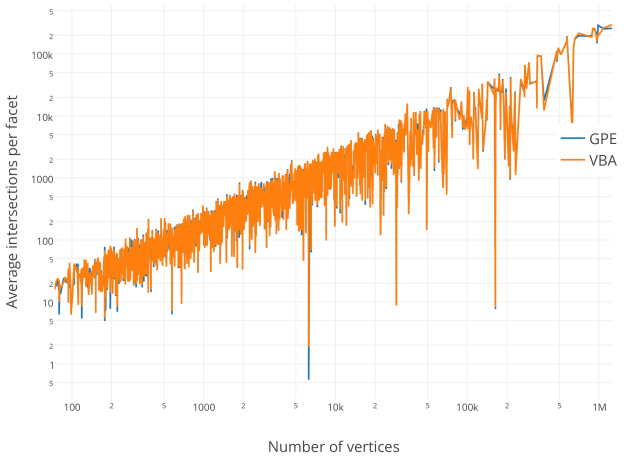

Figure 19: PE-V *evaluation by average surface intersections*

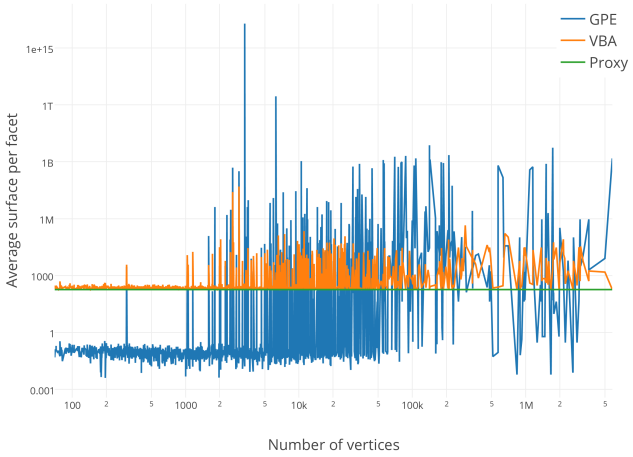

Figure 20: PE-V *evaluation by average surface per facet*

To conclude, Assets protected with *PE-S* are better than *GPE* results in terms of complexity with those two metrics and the evolution of the complexity is guided by the choice of the Proxy.

4.2 *Volume-Based Approach* Evaluation

PE-V produces Protected Assets with the vertices from the Asset moved inside the chosen Proxy (which is also kept in the Protected Asset). Intuitively, we should obtain ray-tracing and facets surfaces results which is similar to the ones obtained with *GPE* protections since the original facets are kept inside the Proxy hull.

The ray-tracing complexity (Figure 19) of the *PE-V* results are very much aligned with the results for *GPE* protection and much bigger than the Asset complexity. However, thanks to the fact that all the facets are inside the Proxy, the resulting protected Asset will normally affect less the rasterisation process than *GPE* results. The rasterisation will find the Proxy's facets (with the Proxy's description defined in first or last position) and the *z-buffer* will not be overwritten.

For the surfaces complexity (Figure 20), the *PE-V* results are heavily constrained by the Proxy and, in comparison to *GPE* results, they remain stable over the increase of the number of vertices.

To conclude, Assets protected with *PE-S* have a complexity closed to *GPE* protection and worse than the *PE-S* results in terms of complexity. However, they will have more natural shapes with the preservation of the Proxy which will also decrease the impact on rasterisation. To avoid the degradation of those two metrics, the original surfaces could be striped and a surface reconstruction technique could be applied to the resulting cloud of points before the integration of the Proxy. This way, the metrics results can be better but it could aslo create artefacts. Moreover, the connectivity will be lost and the 3D-object recovered, after surface reconstruction, will be a degraded version of the Asset.

5 Conclusion and futur work

Research on the protection of 3D-object is relatively new and few solutions are available by legacy renderers. It is always possible to protect a 3D-object with a conventional encryption algorithm but the result is not usable. Innovative solutions have been proposed by defining *GPE* solutions ([Éluard et al. 2013; Éluard et al. 2014; Jolfaei et al. 2015]). These solutions are used to keep protected 3D-object in 3D scenes but the visual impact and the impact on rendering performances is important. A solution was proposed (`Fragment Scaling`) which maintains a correct topology but also a similar appearance to the original 3D-object([Éluard et al. 2014]).

Our approach is similar to `Fragment Scaling` in that sense that it keeps a good topology that allows good results in term of rendering. It changes the appearance of the 3D-object to anonymize and disguise it so that it looks like another. The difficulty of this approach lies in the definition of the projection function θ. There is not universal θ function that would have all the good properties to solve this problem. Each context, each Asset, each Proxy, each environment comes with specific constraints that must be taken into account in the definition of θ. Typically, with the proposed θ function, we are subject to calculation errors during protection and unprotection. Even if the errors are small, it means that the reconstructed Asset is not exactly the original Asset. This is a real problem in precision engineering but may not be an issue in games for example. Likewise, depending on the shape of the Proxy, we may be unable to keep the surfaces of the Asset. In this case, we are subject to errors related to the reconstruction technique used.

In this paper, we focused on the graphical result of a Protected Asset more than on an analysis to quantify the difference between the original Asset and the recovered one. We ensured that the differences were in the last decimal in the points coordinates, but we have not developed a methodology to quantify formally this difference. An analysis of the original/recovered Asset distortion could be made using the same approach as *Key Sensitivity Analysis* [Jolfaei et al. 2015] by using the pairwise Euclidean distance between the corresponding points of the original and the deprotected Asset. Once θ uses arithmetic operations, the result is subject to calculation errors and a function to quantify the differences will be mandatory in order to compare two different θ functions during the implementation of another method.

On the other hand, it may be possible to construct some θ functions without using arithmetic operation, which would ensure that the points of the reconstructed Asset are the same as those of the original Asset. But in this case, the available operations are more limited and it is possible that the Proxy can not be chosen by the user, but imposed by the projection technique (θ). The basis for polymorphic protection solutions have been laid down and it remains now to define the most appropriate tools for protection, adapted to the context of use as well as tools that compare these solutions together.

References

BELLARE, M., RISTENPART, T., ROGAWAY, P., AND STEGERS, T. 2009. Format-preserving encryption. In *Proceedings of Selected Areas in Cryptography*, vol. 5867 of *Lecture Notes in Computer Science*, 295–312.

BLACK, J., AND ROGAWAY, P. 2002. Ciphers with arbitrary finite domains. In *Proceedings of of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, vol. 2271 of *Lecture Notes in Computer Science*, 114–130.

DACHSBACHER, C., SLUSALLEK, P., DAVIDOVIC, T., ENGELHARDT, T., PHILLIPS, M., AND GEORGIEV, I. 2009. 3D rasterization - Unifying rasterization and ray casting. Technical report, VISUS/University Stuttgart and Saarland University.

ÉLUARD, M., MAETZ, Y., AND DOËRR, G. 2013. Geometry-preserving encryption for 3D meshes. In *COmpression et REprsentation des Signaux Audiovisuels*, 7–12.

ÉLUARD, M., MAETZ, Y., AND DOËRR, G. 2014. Impact of geometry-preserving encryption on rendering time. In *IEEE International Conference on Image Processing*, 4787–4791.

FOLEY, J. D., VAN DAM, A., FEINER, S. K., HUGHES, J. F., AND PHILLIPS, R. L. 1993. *Introduction to Computer Graphics.* Addison-Wesley Professionnal, Aug.

HALES, T. C. 1992. The sphere packing problem. *Journal of Computational and Applied Mathematics 44*, 41–76.

JOLFAEI, A., WU, X.-W., AND MUTHUKKUMARASAMY, V. 2015. A 3D object encryption scheme which maintains dimensional and spatial stability. *IEEE Transactions on Information Forensics and Security*, 409–422.

KAUFMAN, A., COHEN, D., AND YAGEL, R. 1993. *Volume Graphics*. IEEE Computer Society Press.

NIST. 2001. Advanced encryption standard. *NIST FIPS PUB 197*.

NOORUDDIN, F. S., AND TURK, G. 2003. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics 9*, 191–205.