Rendering Computer Animated Films on Fully Homomorphic Encrypted Data
6.UAP Final Report

Naim J. Lujan

naim@mit.edu

Supervisor: Fredo Durand

Department of Electrical Engineering and Computer Science

May 5, 2015

1 Introduction

The process of rendering computer animated films is computationally expensive given the complex variety of physical processes being simulated. To render a film animation studios need to have dedicated render farms consisting of a large array of powerful computers who's sole task is to render each frame of the film. As a result of the significant amount of computing resources it takes as well as the constant maintenance and update these rendering farms require, they are very expensive. Large studios such as Pixar have the resources to have their own rendering farms to produce their films but smaller studios need to turn to outside cloud computing to render their animated movies. Cloud computing services allow you to only pay for the compute power you need as you need it.

The problem with using cloud computing services is that studios are very secretive so they don't want the service to be able to see any information of their upcoming movie when they send it to them. They want to be able to use send their data to outside sources to do their rendering while also keeping their information secret. I believe that the solution to this problem is to utilize an encryption scheme called homomorphic encryption. Using homomorphic encryption, specific types of computations can be carried out on ciphertext, and generate an encrypted result which, when decrypted, matches the result of operations performed on the plaintext. This allows for the sending of data across different services without exposing the data to each of those services. In theory, this would allow an animation studio to send their encrypted film to be rendered by a cloud service and when sent back the studio can decrypt the result and obtain their film ensuring that their film remains secret throughout.

In this paper I will given some background on homomorphic encryption and detail a promising C++ library called HElib which is an implementation of the Brakerski-Gentry-Vaikuntanathan scheme. I will then detail a basic rendering method called triangle rasterization and detail my proof of concept implementation of HElib to perform this computation to show the feasibility of using homomorphic encryption

2 Homomorphic Encryption

In this section I will talk a little bit about homomorphic encryption and the advances in the technology. Why haven't we used this before? Is it feasible now? More background on HE.

2.1 HElib (Brakerski-Gentry-Vaikuntanathan scheme)

Here I will get into the specific of the library I am using and some detail about the specifics of the encryption scheme.

3 Proof of Concept and Analysis

I will entail here what I want to achieve using a simple example of animation called triangle rasterization

3.1 Triangle Rasterization

Explain Triangle Rasterization

3.2 Implementation

Explain Implementation. Show some code.

3.3 Analysis

Say what worked well. What could be improved.

4 Conclusion

You can keep a lot of it secret but not as much of the good stuff as you'd like.