MSCS 630L 711 Security Algorithms & Protocols
Professor Kippins
Assignment: Writeup
Due: May 15, 2022
Kerry Lyon
Github: klyon0517

Abstract:

This project combines the AES encryption process with animation in the Babylon.js web rendering engine. Titled “BabylonAES”, the goal is to provide a real time 3D visualization of how AES encryption works from plaintext and system key inputs to the ciphertext output. Various plane meshes with dynamic textures displaying hex values are grouped and positioned around a virtual scene. They are updated and become visible depending on which function and method is occurring at that particular point in time. A camera is placed in the center of the scene allowing end users to rotate, zoom, and pan around the scene freely from any angle.

Introduction:

Having a great interest in game development, particularly in the tools and software, I wanted to see if cryptography could be combined with a real time rendering engine. As a full stack developer, I wanted to work with JavaScript and a web based engine to illustrate AES. I chose Babylon.js for this task.

Since I am a visual learner and have an art degree, this project is a good fit combining these fields of study. A simulation like this could prove useful to others learning about the inner workings of the encryption process. As an added benefit, there is a fun factor involved in the creation and final product.

Related Work:

There is an excellent AES Rijndael animation that breaks down the process in an easy to understand manner (Zabala, Enrique. 2017). This is a linear 2D animation. My work is a real time 3D animation, so it differs quite a bit in that the whole process is occurring as the viewer explores the scene. Nonetheless, it provided a great start of inspiration for development.

A slight improvement to the AES Rijndael animation would be to dive further into detail and add some length. Show the changing input and output states for more of the rounds and functions.

Methodology:

An iterative approach was taken in the development of this project. To begin, the semester’s labs were converted to JavaScript. Results were verified against the Java version and their test cases. The conversion process started with the simplest of labs and iterated to lab 5 which was AES encryption.

Once completed, it was time to start learning Babylon.js. Initial explorations involved the following: creating and positioning a camera, creating and positioning a mesh plane, adding a material, creating a dynamic texture with text, using a variable for the text, adding a texture

transparency, sizing and centering text on the mesh plane, setting up animation keyframes, and offsetting the keyframes.

After the basics were understood, work began on conceptualizing the animation process. This entailed starting with lab 1 and working up to lab 5 again. At first simple positional animations were used to illustrate the lab's process. A lot of research and development was spent on keyframes and the timing of movement. As things progressed, it became clear that more than positional movement was required which led to animating visibility.

With this breakthrough, work finally started on lab 5 and animating AES encryption. Function groupings of mesh planes and position in world space came together over time as work progressed. Much trial and error was spent on creating a style for the various methods. As well as on how to offset visibility for displaying a matrix. For loops quickly became essential to this process.

Then it was a time consuming process of running the animation, watching, fixing, and moving on to the next section. While this was going on, the structure and layout of the scene changed frequently, eventually ending up as seen in the video presentation.

Experiments:

Since this project dealt with visualization, the experimentation involved a creative process about the layout, style, and timing. (The input data was the same key and plaintext throughout.)

Experiments included: positioning of mesh planes, grouping the planes to represent various functions, and styling of the meshes and animation.Also, a large amount of time was spent on the keyframe timing. Specifically how to offset a mesh plane per set of animation. For example, displaying a four by four matrix (in the grid position) one mesh plane at a time over 16 seconds of animation. Then not displaying that matrix again until the next round after other functions had animated (which was 64 seconds later). To accomplish tasks such as this, there was lots of trial and error using for loops and integer incrementation as timing markers.

Analysis:

One of my favorite results is the animation of the calls to SBox. It is very interesting viewing the selections appear and remain.This provided a much closer look at how the SBox is used and how often it is called.

Another interesting result of the experiment is seeing the input state and return state appear correctly for each round of the AESNibbleSub, AESShiftRow, AESMixColumn, AESStateXOR loop. I set up the animation to display the input state parameter matrix with the output layer in front as it became visible for each function.

Conclusion:

Visualizing the AES encryption process divulged its complexity more clearly. It really highlights the order of the function calls and the amount of calculations that are involved. Then there is also the fun factor of being surrounded by the machinations in progress. With the ability to fly around and view the animation from any angle and distance. Overall, this was a challenging and entertaining project.

References:

Zabala, Enrique. (2017, November 27). AES Rijndael Cipher explained as a Flash animation. https://www.youtube.com/watch?v=gP4PqVGudtg

Zabala, Enrique. (2017, November 27). Rijndael Cipher (winner of the AES selection). https://formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng-html5.html