ECE 491 FINAL PROJECT

**Multimedia Product Licensing**

**Computer and Communication Security**

**Fall 2015**

**Zachary Rosenthal**

**Table of Contents**

**Introduction……………………………………3**

**Project description and assumptions…….4**

**Project design…………………………………5**

**Project Analysis……………………………….8**

**Programming Code………………………….X**

**Conclusion….……………………………….X+1**

**Introduction**

Security is a never ending battle between those trying to protect data and those trying to breach data. The question of if something can be cracked or broken into is always “Yes, it can”. A better question is, “How easy can it be cracked?” and “How fast can it be cracked?”. A *good* answer to these question is very hard and a very long time, respectively. To measure the complexity and time of breaching a particular security measure depends on computational resources and ease of access. The proceeding is an exploration into the task of securing online multimedia, an ever daunting task in the security industry.

**Project description and assumptions**

The goal of this project is to design and implement a method of licensing online multimedia, much like software licensing. For simplicity I will make the product key readily available to the user. However, it is important to note that in a real implementation the product key would be purchased through some service. It is also important to note that in this particular implementation the initializing vector and encryption key are fixed. In a production implementation these values would need to be changed for each request to provide additional security.

This project serves as *proof of concept, but* is NOT production ready nor is it meant to be used in production. Many considerations need to be made for a production solution. Each encoded/scrambled file would be unique to specific user and most likely be downloaded. A client would most likely need to be created unless a web-based client can be perfected. The client would still need a server to validate product keys and potentially decoding/unscramblin. In its current state the user can still save a copy of the multimedia to their computer. There can be additional measures taken to further secure the browser based client and protect against the vast majority of users downloading content. As of today, there is not a fool proof way of ensuring multimedia served to a web browser cannot be downloaded. If it is in a user’s browser, it is local and they can get obtain it. However, it is possible to make this task difficult and undoable by the vast majority of users (95+% of typical internet users) using various browser *hacks.*  Taking the above into considerations, the project design will discuss the design, architecture, and implementation of this project.

**Project design**

This project will simulate a client-server interaction. The server will consist of VPS running Ubuntu 14.04 and an apache web server using PHP to handle the heavy lifting. I am running an ever so slightly modified version of ScotchBox (https://box.scotch.io/) using Vagrant (https://www.vagrantup.com/). The server will handle all licensing and product key distribution.

The server will also serve an interactive web-browser based application for the user to interact with. The client part of the application consists of a small amount of JavaScript, HTML, and CSS. The HTML and CSS are used to display a GUI. The JavaScript consists of the JQuery library (<https://jquery.com/)>, and three other functions. Two of these function make AJAX (asynchronous JavaScript and XML, part of the JQuery library) requests to the server and the other is used for displaying the multimedia. A typical client-server interaction will flow as follows:

1. Access application via web browser
   1. An html page will be served from the server and allow the user to interact with rest of the application
2. Obtain a product/license key for a specific software
   1. User selects a multimedia file
   2. A key is generated on the server and the multimedia file is encoded/scrambled using this key
   3. The scrambled file is saved in a scrambled directory
   4. A hexadecimal representation of the key is given to the user after being encrypted using AES-128-CBC encryption. This is the product key
      1. In a production implementation this would be purchased
3. Request usable multimedia
   1. User sends product key to server and asks for unscrambled file
   2. Server decrypts and validates product key
   3. Server decodes/unscrambles scrambled multimedia file
   4. Server returns the file contents to the browser
      1. The unscrambled file is deleted immediately after it is sent to the browser
      2. If the user tries to view the multimedia again it will not be accessible

The multimedia files are encoded using a key that consists of a 2 digits between three and nine. The two-digit key is prefixed with the name of the file that it will encode and suffixed with the current system time. The modified key is then encrypted using AES-128-CBC encryption and given to the user as a hexadecimal string.

The encoding/scrambling algorithm works by takes the beginning and ending of a line and swapping them. A Pair of lines (j, n-j) are processed together and then swapped, here n is number of lines in the file. This process is repeated 10 times by default and can be specified to any number. It is important to note that if either line j or line n-j is not of adequate length then the lines are not scrambled, but are still swapped with each other.

Algorithm:

By default: j = 0 and m = 10

Assume file with n lines: 1 through n

START:

for j to m:

line1 = j, line2 = n – j

if length of both lines are greater than x + y

for line1 and line2:

take first x characters and swap with last y characters

END

then swap lines j and n - j

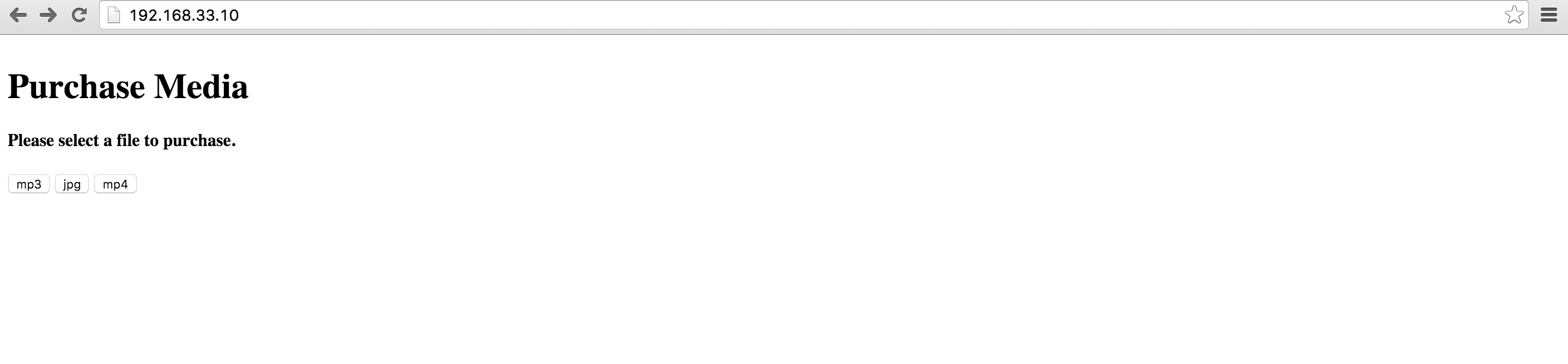
END

END

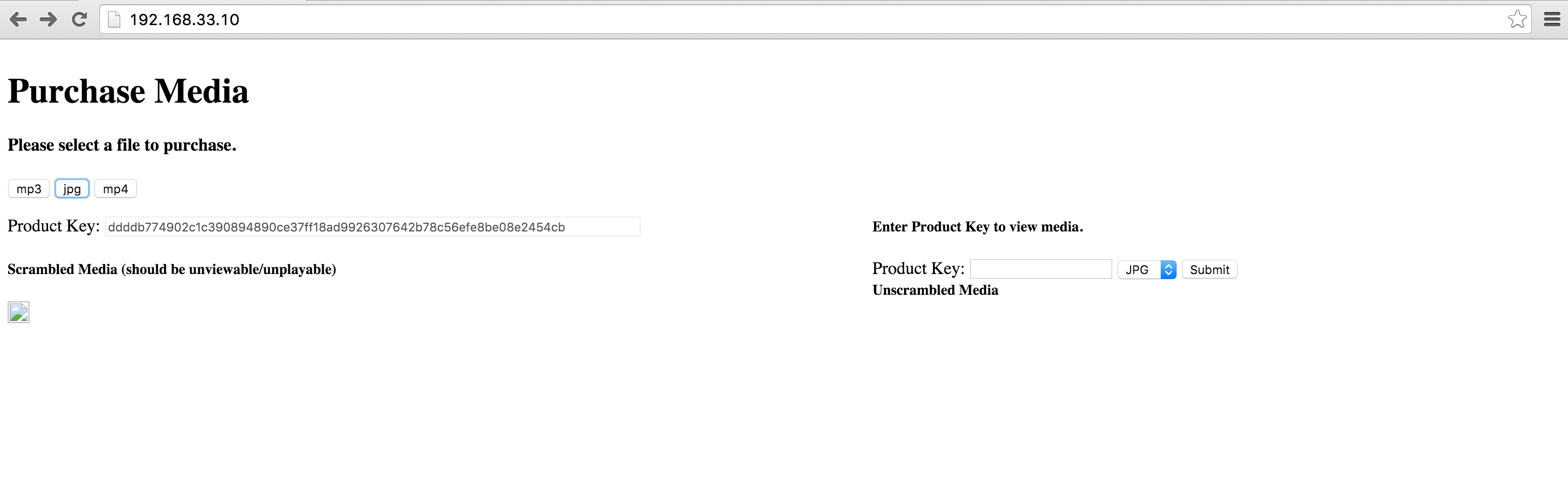
**Project Analysis**

This solution has been tested with jpg, mp3, and mp4 file types. All three of these file types were successful. The screenshots that proceed show a user flow through the application.

-Landing Page: Note: the IP address does not change because AJAX is used to make all request.

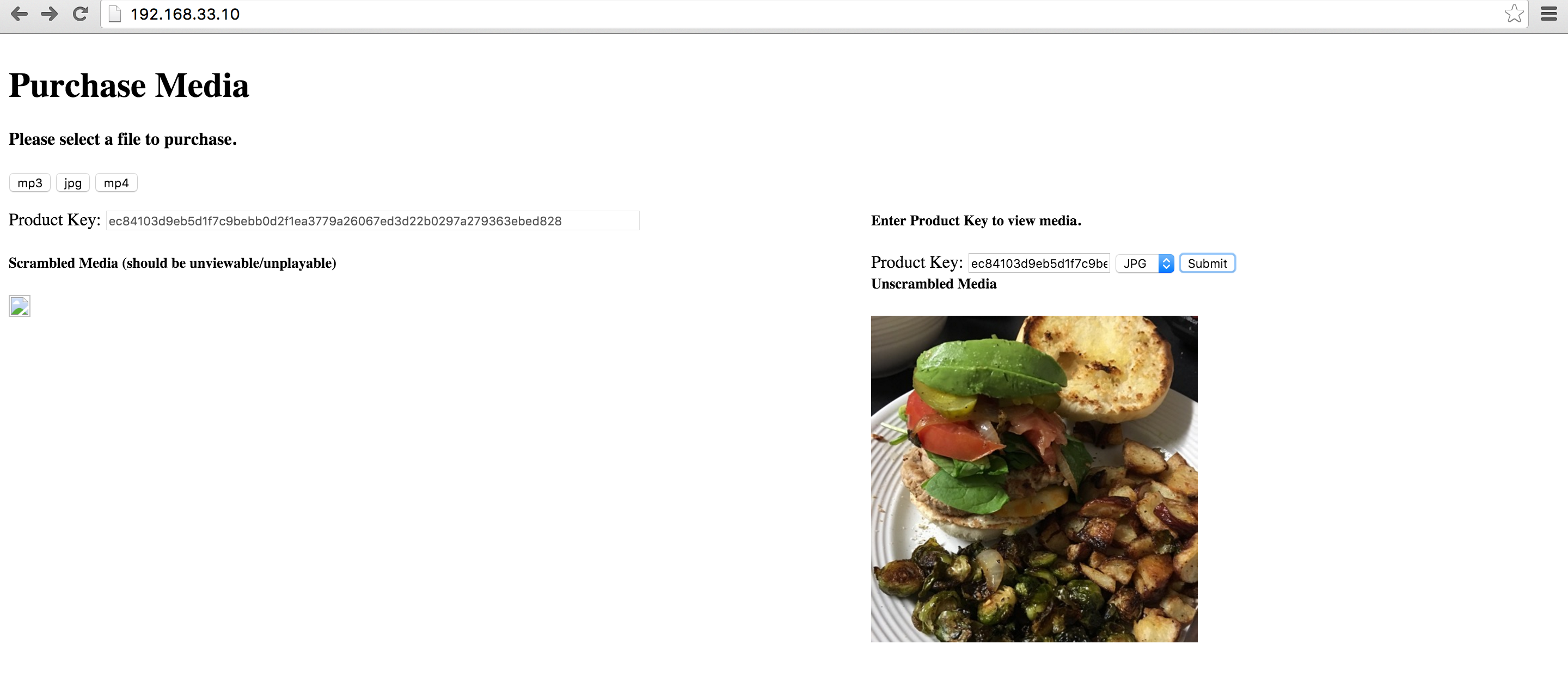


-Click on file type (each button corresponds to a test file of the specified type):

****

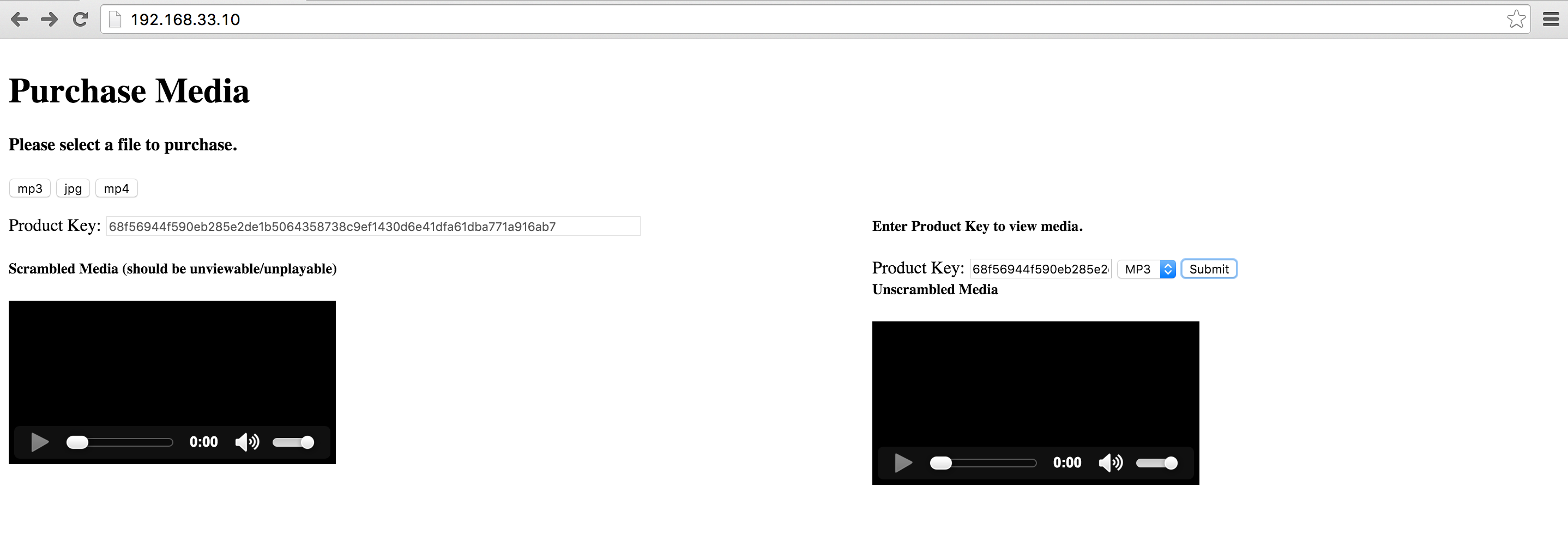
-In the above view the jpg button was clicked and the product key is displayed to the user along with the scrambled jpg file, which is not viewable.

-After entering the product key into the input box, selecting JPG and clicking submit we are shown the unscrambled jpg image.

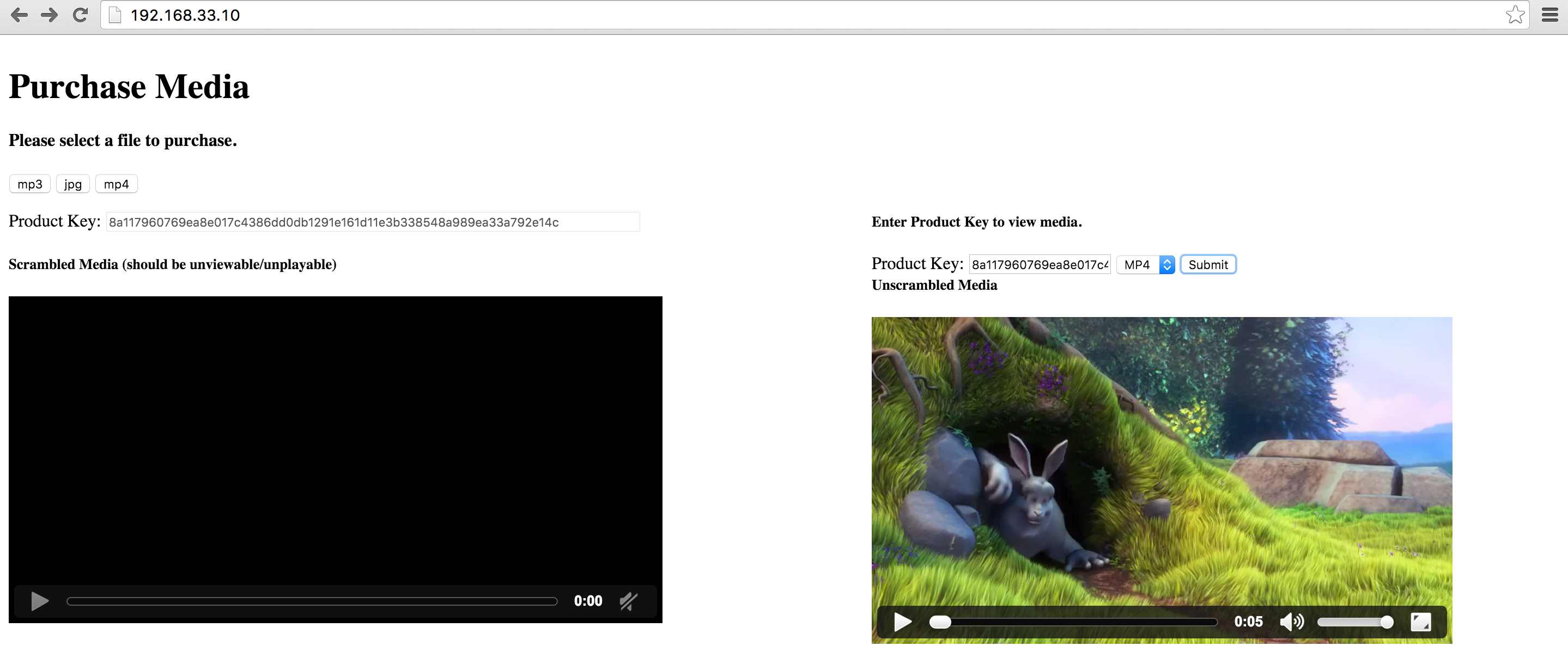


-The application works the same way for mp3 and mp4 file types as well.

-mp3: the mp3 on the left will play while the one the right will not.

****

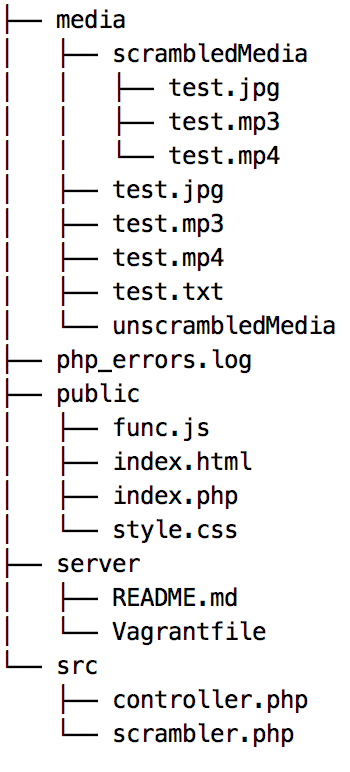
-mp4



**Programming Code**

Can also be viewed at: <https://github.com/zRosenthal/multiMediaLicensing>

File Sturcture:



**Conclusion**

This paper explores the task of securing online multimedia by using a scrambling technique to make multimedia files unaccusable. While the solution presented is incomplete, it represents proof of concept. The solution is not meant to be deployable or perfected, but rather a first cut at the topic. Taking into consideration the assumptions made and the various security flaws mentioned, this solution can readily be cracked. However, with some fine tuning it can be perfected.

A proper solution should most definitely be written in a language such as C or C++, for performance and security reasons. The concepts apply the same to any language and the takeaways are the same. It is possible to create a layer of security for multimedia licensing. I hope this can serve as a building block in future explorations. In the never ending battle of protecting data, multimedia security is a key concern and will continue to draw more attention in the future.