****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/16/2022** | **Kennedy Uzoho** | **dev and security** |

## Client



## Developer

Kennedy Uzoho

## 1. Algorithm Cipher

## Encryption Algorithm Recommendation: AES-256

Advanced Encryption Standards (AES) is one of the Secure Encryption Algorithm available. AES-256 is a very patented cryptographic encryption function that can encrypt a digital value or data that is 256 bits long. The most important feature of AES-256 is the resilience it has in protecting data from unintended access or hackers. AES-256 has been used in some of the most popular authentication and encryption protocols, including bank transaction encryptions, classified data, online shopping data, and social media apps data. AES-256 is symmetric cryptographic function that allows the use of the same key to decrypt an encrypted data, unlike RSA that is asymmetric. RSA is asymmetric it uses two different but linked keys (public and private keys) to encrypt.

**Hashing Data Algorithm Recommendation: SHA-1 OR SHA-256**

SHA-256 does not belong to the encryption algorithm category because it is a hash function. it belongs to the family of Hash algorithms. Although, most of them turn data into random ciphertext that will be extremely hard to decrypt or read without confusion.

Secure Hash Algorithm 256 (SHA-256) is a secure cipher hash algorithm that uses a cryptographic hash (digest) function to verify the integrity of data. The hash function is designed to produce a unique collision-free value from various data types. SHA-256 is among the most secure hashing algorithms in production. It has about a 0.01% probability of having collisions. Collision-free means that the hash algorithm will not produce the same hash value for two different data. SHA-256 returns characters of either lowercase or numerals, starting from zero through nine.

## 2. Certificate Generation- print out of local-cert.crt

Text

Description automatically generated

## 3. Deploy Cipher

@RestController held a source code for cryptographic hash function SHA-256, which will in return generate a hashed value for the provided data string “Kennedy Uzoho” and a return message “Welcome to a safe browsing environment”

Text

Description automatically generated with low confidence

**Deploy Cipher**

Trusted HTTPS WEB Server connection with Hashed string data type (“Kennedy Uzoho”) displayed

Graphical user interface, text, application

Description automatically generated

## 4. Secure Communications

HTTP to the HTTPS protocol **https://localhost:8443/hash**

I installed my own self-signed cert generated from CA into my computer and I trusted the key, that is why it is showing a secure connection.

Graphical user interface, text, application

Description automatically generated

## 5. Secondary Testing

## Code base was analyzed, refactored, and executed as maven verify, maven install and as a stand-alone java web application listening on tomcat server port 8443.

Run type: Maven verify and install {Dependency report and a jar file was generated after this successful build}

## Text Description automatically generated

## Run type: Java web app listening on tomcat server, port 8443. https://localhost:8443/hash

## Graphical user interface, text Description automatically generated

**Dependency check report**

**Report without suppressing false positives**

Graphical user interface, application

Description automatically generated

**First iteration to suppress known false positives**

Graphical user interface, text, application, email

Description automatically generated

**Second iteration to suppress known false positives**

Graphical user interface, application

Description automatically generated

## 6. Functional Testing

## Reviewed and refactored code base with all parts completed as required for this milestone.

## Key = local\_ssl.p12, cert = local-cert.crt, and the application controller modules.

Text

Description automatically generated

## 7. Summary

## Summary and process for adding layers of security to the software application.

After initial review of the code base, a controller java class was created, and hash algorithm function was added to the controller class. In-code comments was provided for easy readability, analysis and to comply with the industry standard best practices. Code was debugged with no know errors. The pom.xml file was reviewed and refactored by updating the associated APIs and nested apps linked in the application. The maven dependency check was updated for best result. There was a key “local.ssl.p12” and keystore generated in the code resource directory. These key and its cert as published from CA allowed the web application to have a trusted connection after installing and trusting the self-signed certificate from CA into my computer.

**The areas of Vulnerability Assessment Process that was addressed after the code was refactored included:**

API interaction > pom.xml file

Cryptography > Hash functions/ AES-256

Code Error > Debugging

Code quality > in-code comments

Encapsulation > nested APIs and secure data structure

Code review (controllers) and

Secure coding practice pattern

References

*Compliance with cybersecurity and privacy laws and regulations*. NIST. (2021, August 4).

Retrieved July 23, 2022, from <https://www.nist.gov/mep/cybersecurity-resources-manufacturers/compliance-cybersecurity-and-privacy-laws-and-regulations>

<https://docs.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html>

https://www.baeldung.com/spring-boot-https-self-signed-certificate

*Symmetric vs asymmetric encryption: A guide for non-techies*. HackerNoon. (2021). Retrieved August 10, 2022, from https://hackernoon.com/symmetric-vs-asymmetric-encryption-a-guide-for-non-techies-p03c316t