****

# 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** | **08/14/2022** | **Mitch Sfakianos** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Mitch Sfakianos

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The appropriate encryption algorithm cipher I would recommend for Artemis Financial to deploy is the advanced encryption standard. Most banks use AES and it was set as a standard by the US National Institute of Standards and Technology for electronic encryption (Smirnoff). AES uses key sizes ranging from one to two hundred characters, making brute force guessing very improbable. The large key size can be good for security, but this also means it takes more time to encrypt and decrypt data, making the performance slow. The biggest security risk with using AES is an attacker getting ahold of the key somehow. So, when companies need to share the key to a collaborator, the key is also encrypted with asymmetric encryption algorithms.

In the U.S, companies are legally required to encrypt data and use reasonable security procedures; if they don't, they can be sued by users who had their data compromised. Protecting the integrity and security of users' data is the law.

The advanced encryption standard is used by arranging the data into matrices and substituting each 16 byte block with its encrypted version based on the encryption key.

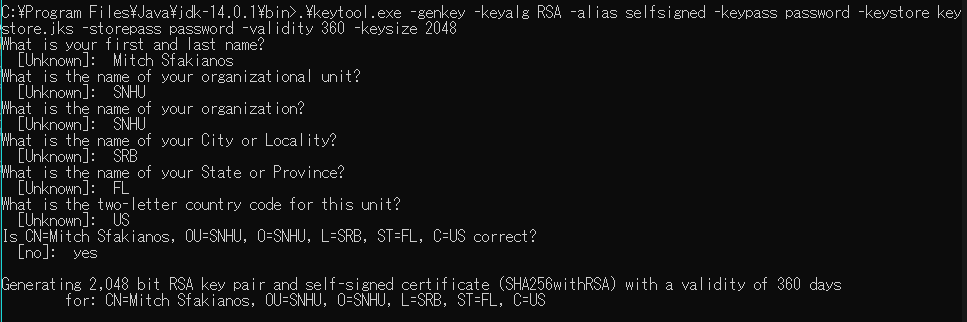
The AES is widely used and not easy to manually break, making it one of the best ciphers. Using the most secure cipher ever may take more time on the processing of the site, making it slow to run. Triple encryption even with a smaller key could potentially take more time to function than one AES encryption.

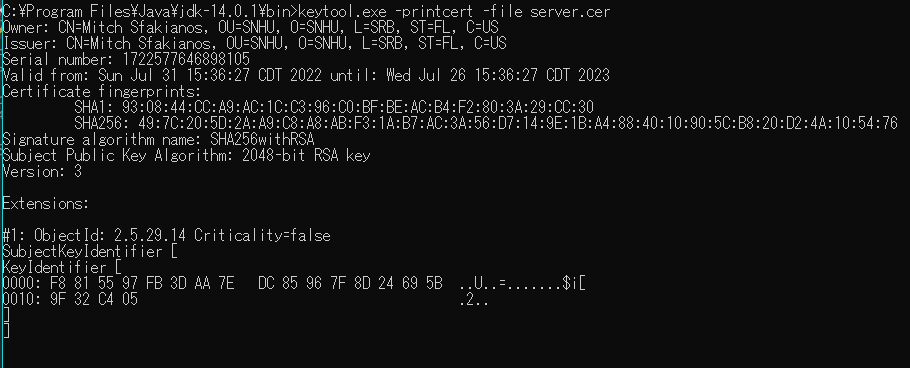
In this cipher, the hash function's purpose is to take data as an input and garble it in the output so it is unreadable. The data is hashed using the key, which is where the random numbers come in, since generally keys are long strings of characters. A symmetric key can be used to encrypt and decrypt the data, whereas asymmetric key systems need one key to encrypt and another to decrypt. The AES is a symmetric key system. While the DES is similar and included in the Java Security Standard Algorithms list, it was cracked in 1997. The AES replaced it in 2000.

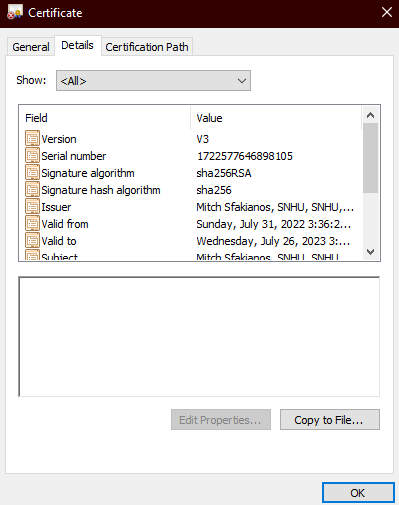
## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



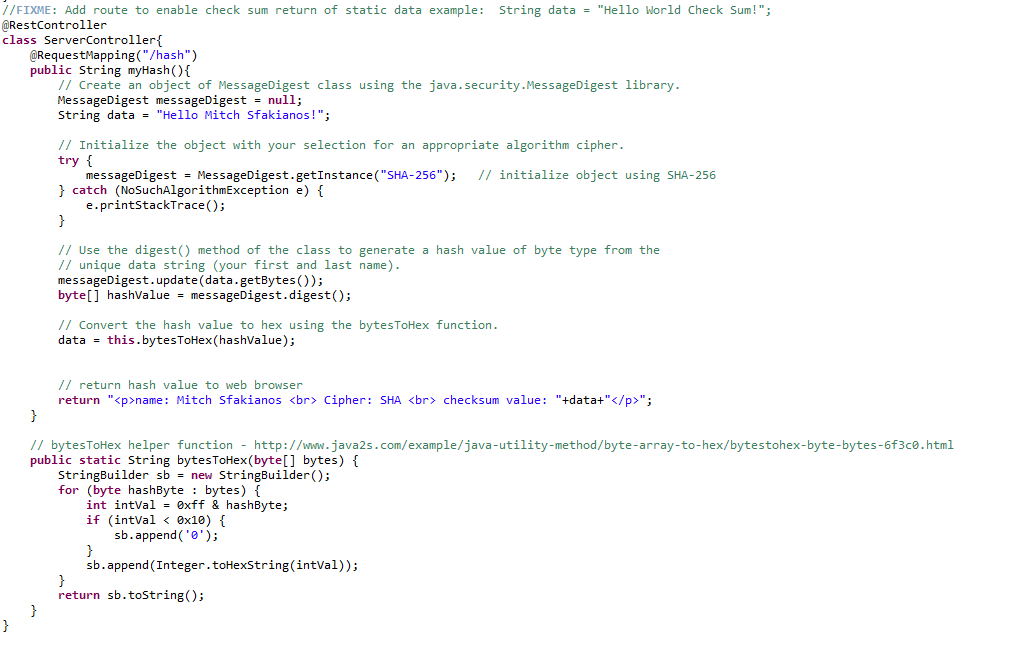




## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

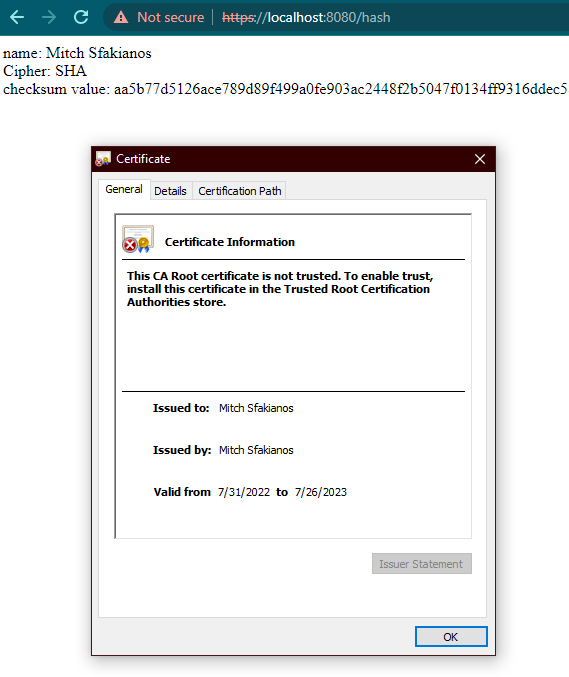
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

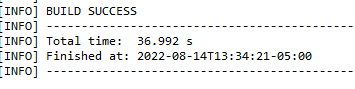
* Insert a screenshot below of the web browser that shows a secure webpage.

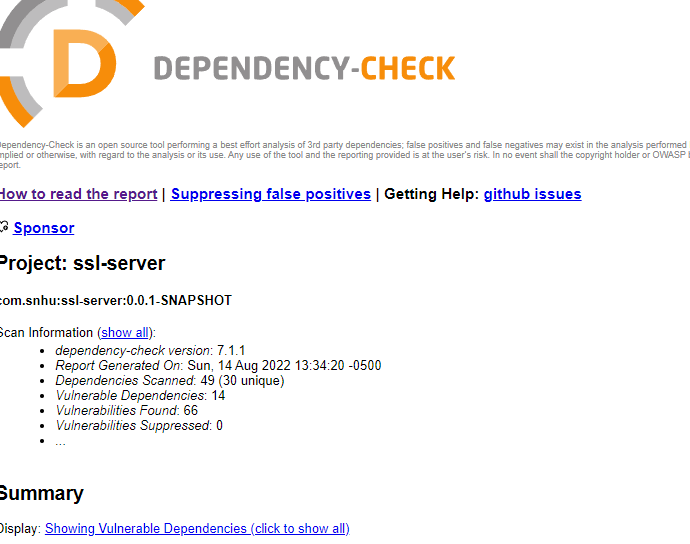


## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

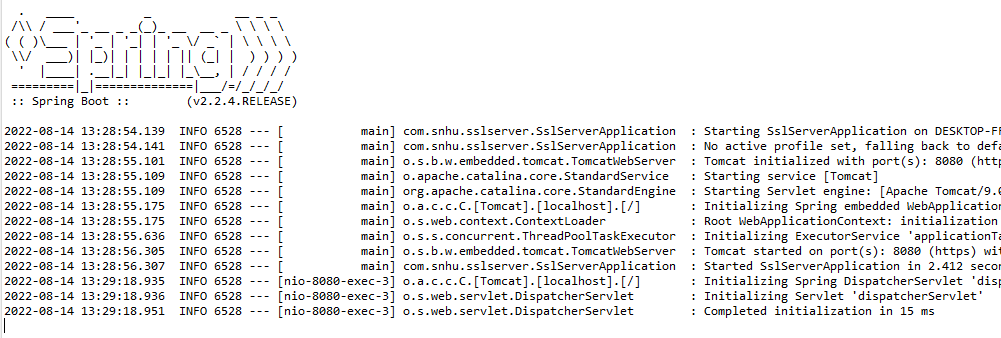




## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.



## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

The areas of security addressed in this refactoring of the Spring application’s code include encryption use and error handling. A security certificate was created using an encryption key and was utilized in the creation of the checksum feature in the hash route of the site. Additionally, security with the encryption usage was solidified through the verified secure communication between the server and the certificate. To properly maintain the current security of this software application to the customer, the certificate should be renewed upon each end of cycle and the algorithm itself could eventually be upgraded in the future if the current one has any new vulnerabilities that are found up later.