

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History3](#_Toc33111302)

[Client3](#_Toc33111303)

[Instructions3](#_Toc33111304)

[Developer4](#_Toc33111305)

[1. Algorithm Cipher4](#_Toc33111306)

[2. Certificate Generation4](#_Toc33111307)

[3. Deploy Cipher5](#_Toc33111308)

[4. Secure Communications5](#_Toc33111309)

[5. Secondary Testing7](#_Toc33111310)

[6. Functional Testing8](#_Toc33111311)

[7. Summary9](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **2-14-2022** | **Dipesh Patel** |  |

## 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

Dipesh Patel

## 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.

For the needs of Artemis Financial, I would recommend using an AES encryption. Advanced Encryption Standard (AES) is a fairly new, strong encryption algorithm that has become the industry standard for security. The specification for the encryption were established by the U.S. National Institute of Standards and Technology (NIST) in 2001 and has become a U.S. federal government standard in 2002.

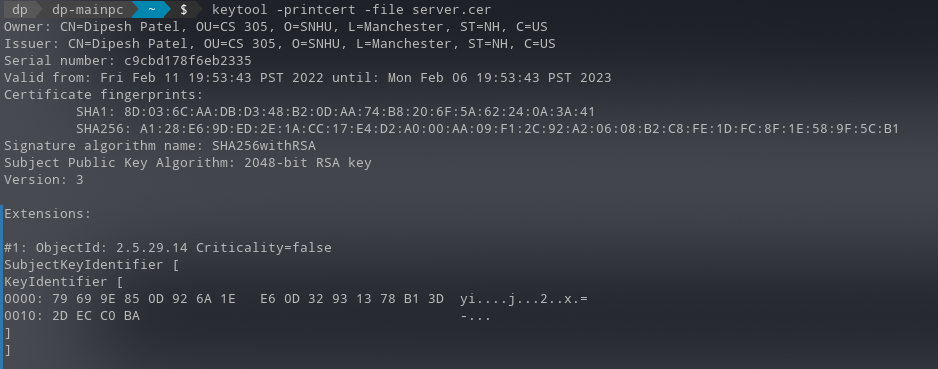
Let’s assume that we want to send data that is readable to us, such as a plain text file. But this plain text can be intercepted and be easily read by anyone. This is where AES comes in, the plain text becomes encrypted, resulting in the encrypted data looking like randomized text. This obscure the original contents of the data from anyone who isn’t authorized to view it without the decryption key. The sender encrypts the message with a secret key and sends the message off, the receiver has the same key, so they can decrypt and view the message in its original form. A system like this would be greatly beneficial to a financial institution when sending and received sensitive data because only the sender and receive will be able to access it.

In order to encrypt something with AES you have to go through four steps, Byte Substitution, Shiftrows, MixColumns, Addroundkey. Having these four steps makes AES very complex to recover without the key. Each key level has a different number of rounds for the steps that it goes through. 128-bit keys go through 10 rounds, 192-bit keys go through 12 rounds, and 256-bit go through 14 rounds. Each round adds another level of complexity to the encryption. AES is a symmetric encryption, which means converting plaintext to ciphertext uses the same key to encrypt and decrypt it. Asymmetric means that there are two related keys to encryption, which are a public key and a private key. Asymmetric is typically used when there are two separate endpoints. Security strength was to be considered the most important factor, which is why AES is ideal for Artemis Financial.

## 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.

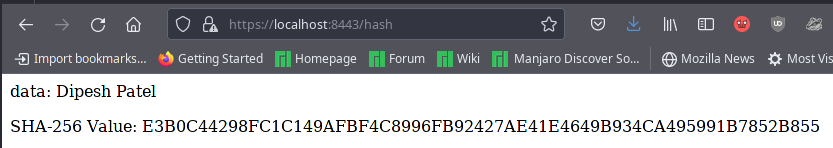
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.

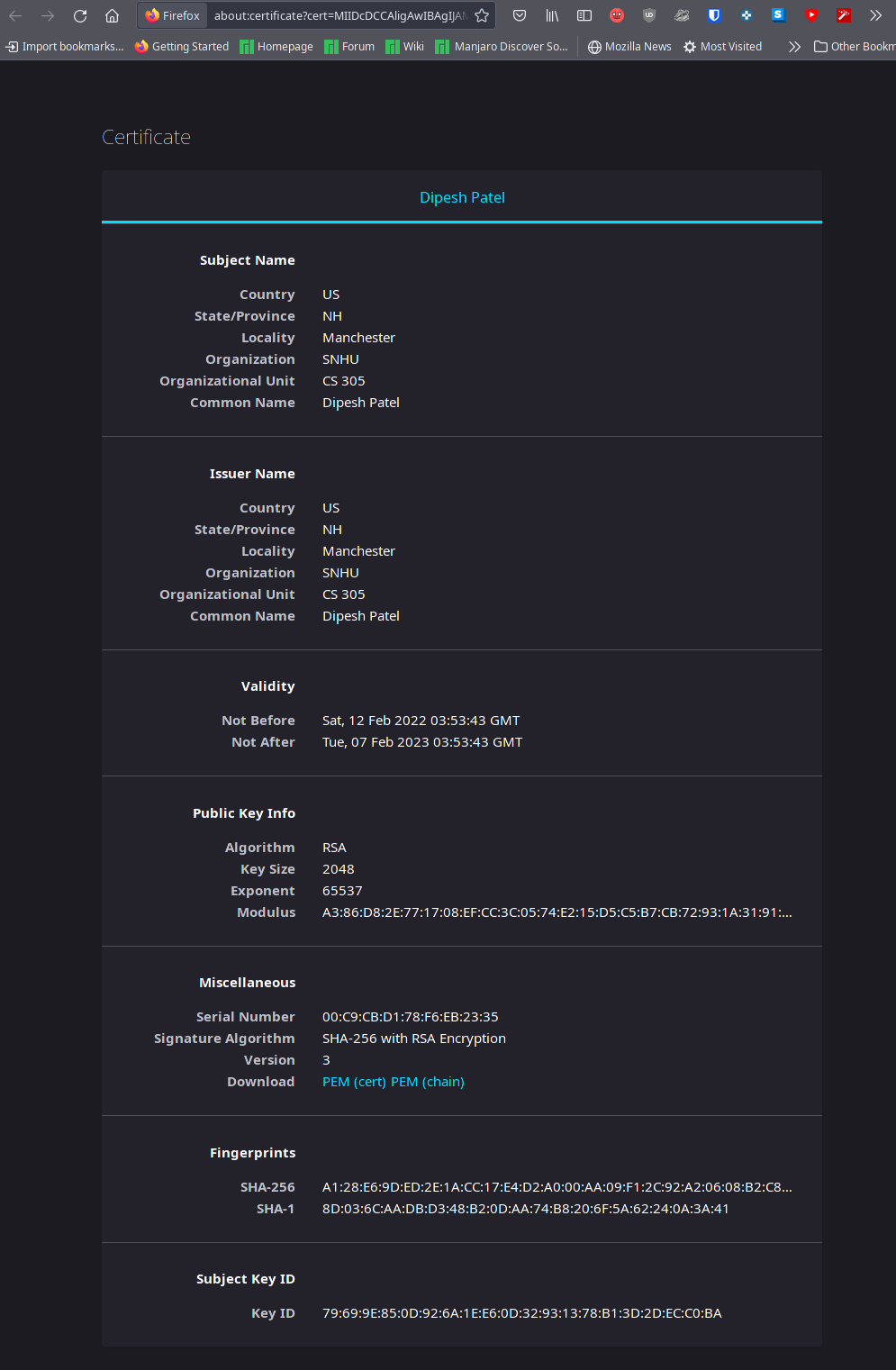
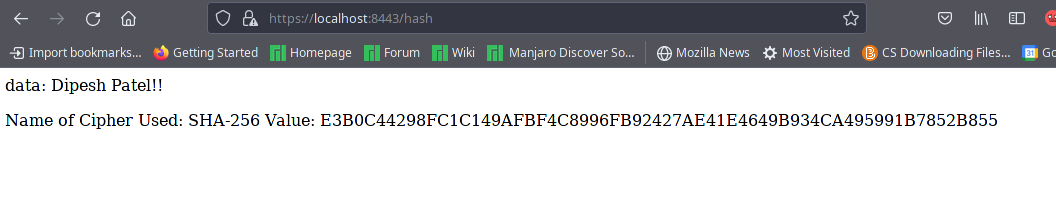


## 4. Secure Communications

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.

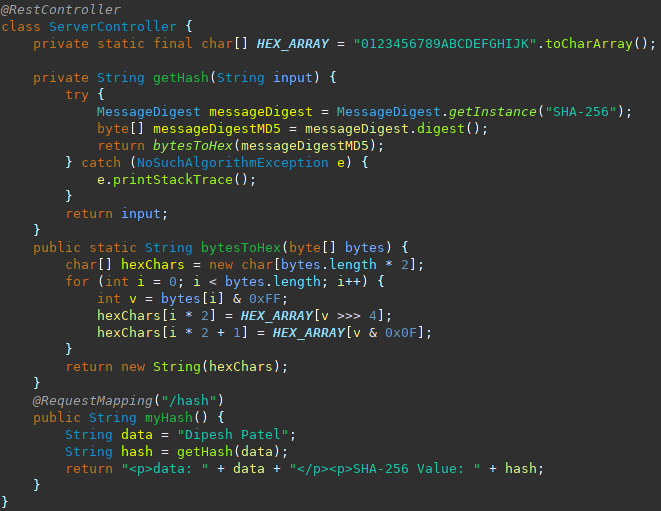




## 5. Secondary Testing

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.

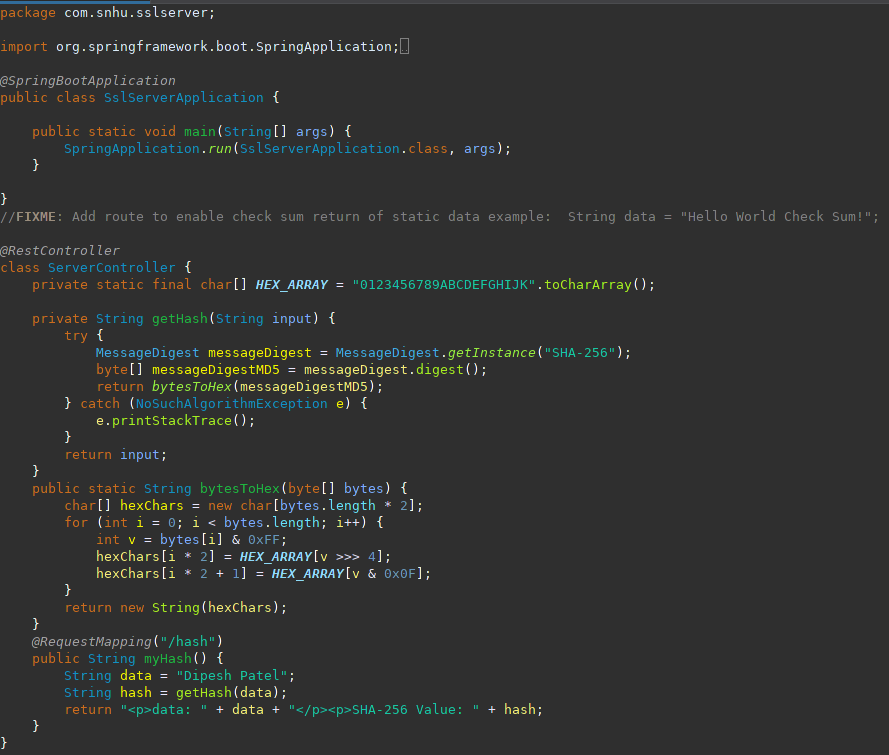
* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report



## 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 Rest Controller was made secure for the hash endpoint. The ServerController class is where the logic is contained. The vulnerability assessment process flow diagram is addressed this class. I chose to use SHA-256 hashing cipher to secure the company’s data and communication. SHA-256 has a strong encryption and has a low chance of collisions. We take the string input and encode it with SHA-256 and output the hash and a string message to the hash page. The best practice maintain the security of the software would be to keep up to date with all the security patches. Having an update interval of 1 to 3 months would help keep all the software up to date without much downtime.