****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[CS 305 Project Two 1](#_Toc64819510)

[Document Revision History 3](#_Toc64819511)

[Client 3](#_Toc64819512)

[Instructions 3](#_Toc64819513)

[Developer 4](#_Toc64819514)

[1. Algorithm Cipher 4](#_Toc64819515)

[2. Certificate Generation 5](#_Toc64819516)

[3. Deploy Cipher 5](#_Toc64819517)

[4. Secure Communications 5](#_Toc64819518)

[5. Secondary Testing 5](#_Toc64819519)

[6. Functional Testing 6](#_Toc64819520)

[7. Summary 7](#_Toc64819521)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **02-20-2021** | **John Ledbetter** |  |

## 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

John Ledbetter

## 1. Algorithm Cipher

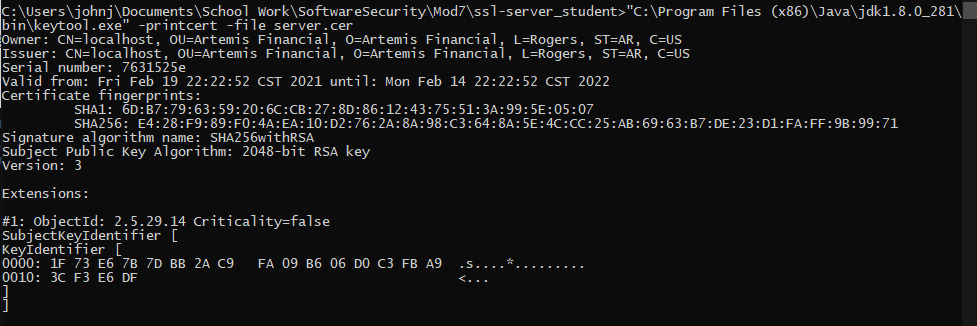
Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

Advanced Encryption Standard (AES) will encrypt Artemis Financials archive files. AES is the U.S. government standard for data encryption, including SECRET and TOP SECRET data levels. AES is a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits.1 AES begins with a cryptographic key used in a key expansion routine to generate a set of Round Keys (a double array of bytes of 4 rows and Nk columns). The input is placed into block ciphers that will go through various transformations or rounds. At the end of each round, a Round key is combined with the cipher block. This means that even if one knows the process of AES encryption or decryption, without the original cryptographic key, the ciphertext cannot be decrypted. Hash functions are mathematical algorithms that take an arbitrary length of data and output a fixed-length value. This Hash value can be used to validate the authenticity of data being received between two sources that know the hash function algorithm being used. Data input will obtain the same Hash Value each time. This means if the data sent from source A does not have the same hash value when sent to source B. There has been a change in the data. Useful hash functions are also collision-resistant. Meaning a hash value from data input X does not equal the same hash value from data input Y.2 This is where bit-level comes into play. "Encryption strength is often described in terms of the size of the keys used to perform the encryption: in general, longer keys provide stronger encryption. Key length is measured in bits." 3 An RSA encryption key of 128 bits is more likely to be cracked than an RSA key of 512 bits. Two different algorithm ciphers can therefore have the same encryption strength depending on the length of their keys. Symmetric encryption means only one key is required for both the encryption and decryption of data. In contrast, asymmetric encryption is based on multiple keys, with one for encryption and another for decryption of data. Cipher keys can be generated through the use of random numbers. Recommendations for implementing a random number generator can be found in NIST Special Publication 800-90A.3 The randomness of a random number is essential for the strength of a cipher key. Another way to say this, if the process used to create the random number for the cipher key can be determined, then the chances of breaking the cipher key are increased. This will make your application less secure. Encryption of information has been apart of information security since as far back as 600 BC where ancient Spartans wrote secret messages that could only be read by wrapping the message around a specifically sized wooden rod. As time moved on, encryption techniques changed. Information required knowledge of the specific pattern used to encrypt to be able to decode it. Computers today can now create randomized encryption keys that are applied to information and, through the data's permutations with the key, encrypt data more securely than before. Better encryption algorithms are put to the test to become the next standard for encryption. As computers advanced and the ability to process more information in shorter, more efficient amounts of time. The need for newer, more robust encryption increases.

## 2. Certificate Generation

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

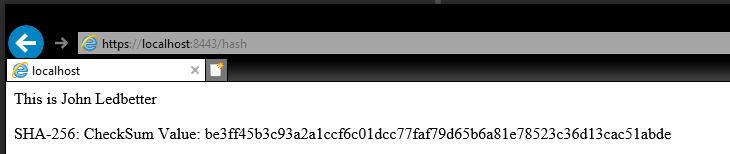
* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.



## 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.

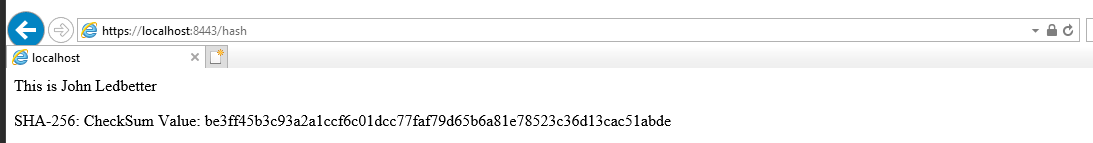
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

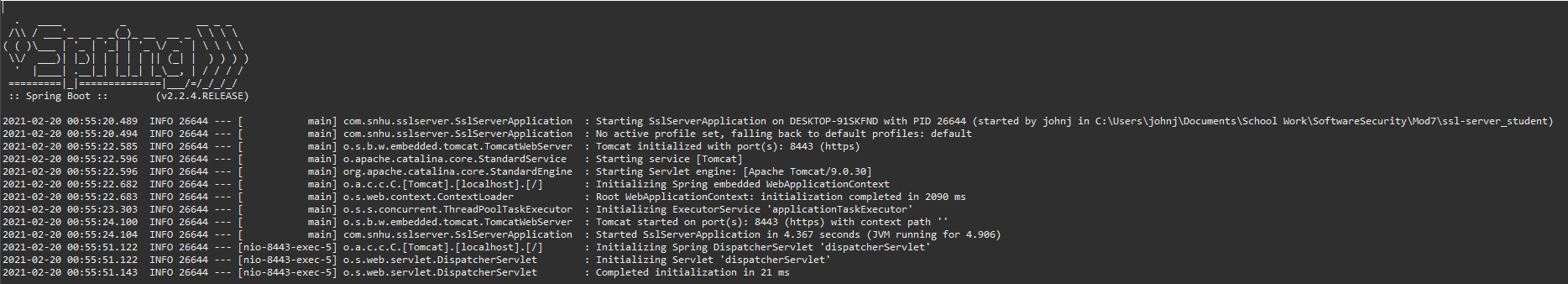
* Insert a screenshot below of the web browser that shows a secure webpage.

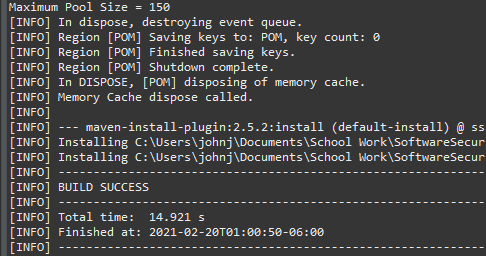


## 5. Secondary Testing

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







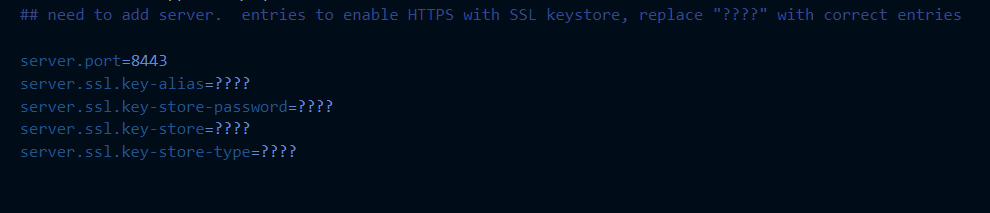
## 6. Functional Testing

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

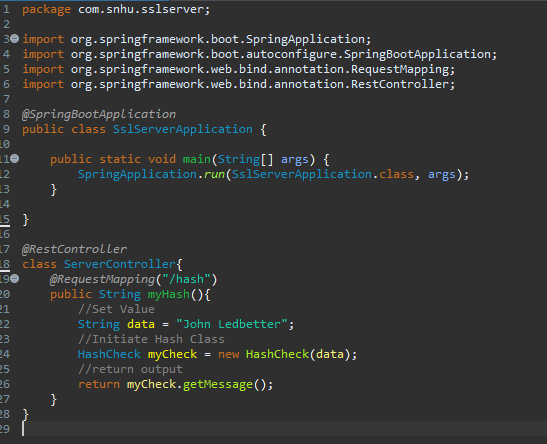
* Had to re-open the original project in VS Code. Eclipse was giving me issues.

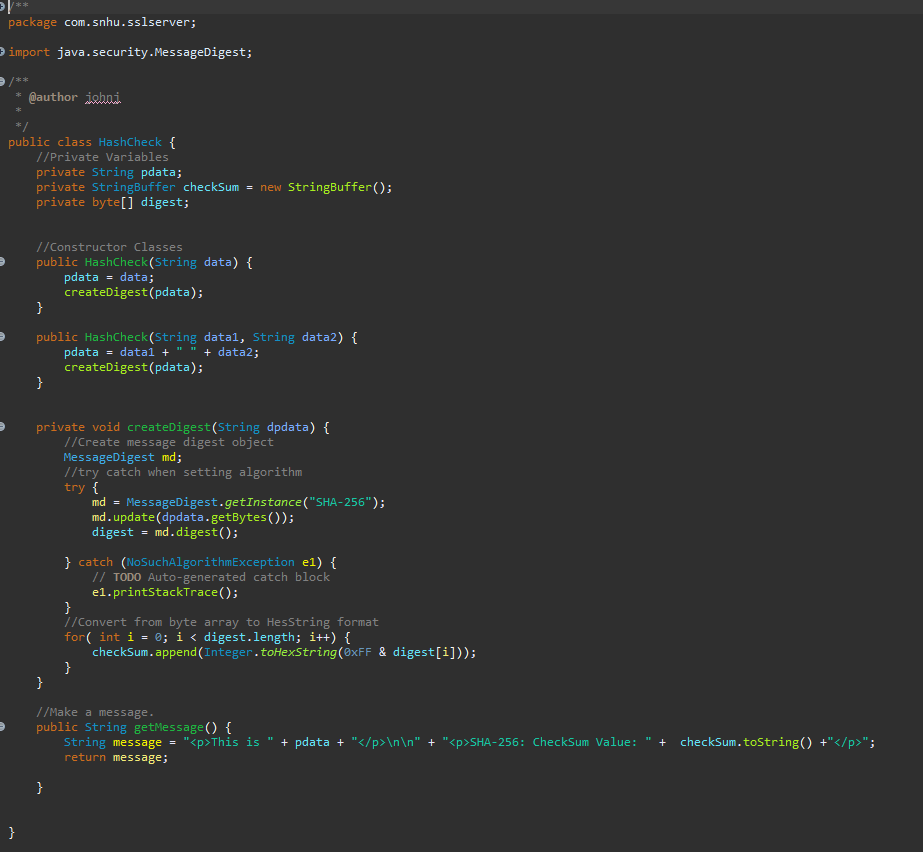
Initial Code:





Refactored Code:





## 7. Summary

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

Specific areas the refactored code tackled cryptography, client/ server, code error, code quality, and encapsulation. Each area is another layer of security added to the application. Secure distributed composing, client/server is achieved through the Spring Frameworks built-in Spring MVC, Model View Controller, and implemented with the @RestContoller and @RequestMapping interact with our applications hashing algorithm. The hash algorithm itself is a security level by validating the contents of data received has not been changed during transit and affected by a man in the middle attack. Our hash algorithm follows best practices by exhibiting encapsulation using a public constructor and a public getMessage() method while keeping the different data points and hash function private. Finally, we enable secure communication with the server with HTTPS and using AES-based Certificates to work with our future Certificate Authority. To maintain current and on-going security, the keys for our certificates should be stored on a server separate from our application along with rotated at regular intervals.

**8. References:**

1 National Institute of Standards and Technology. (2001, November 26). *FIPS 197, Advanced Encryption Standard | CSRC*. <https://csrc.nist.gov/publications/detail/fips/197/final>

2 *Cryptography Hash functions - Tutorialspoint*. (n.d.). Tutorialspoint. https://www.tutorialspoint.com/cryptography/cryptography\_hash\_functions.htm

3 *Key Length and Encryption Strength (Sun Directory Server Enterprise Edition 7.0 Reference)*. (n.d.). Oracle. https://docs.oracle.com/cd/E19424-01/820-4811/aakfw/index.html

**3** Barker, E. (2015, June 24). *SP 800-90A Rev. 1, Random Number Generation Using Deterministic RBGs | CSRC*. National Institute of Standards and Technology. <https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final>

**4** *A BRIEF HISTORY OF ENCRYPTION*. (2021, January 19). Thales. https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption#:~:text=In%20the%20early%201970s%3A%20IBM,papers%20on%20encryption%20were%20classified.