****

# Practices for Secure Software Report

Table of Contents

[Document Revision History 3](#_Toc102040754)

[Client 3](#_Toc102040755)

[Instructions 3](#_Toc102040756)

[Developer 4](#_Toc102040757)

[1. Algorithm Cipher 4](#_Toc102040758)

[2. Certificate Generation 4](#_Toc102040759)

[3. Deploy Cipher 4](#_Toc102040760)

[4. Secure Communications 4](#_Toc102040761)

[5. Secondary Testing 4](#_Toc102040762)

[6. Functional Testing 4](#_Toc102040763)

[7. Summary 4](#_Toc102040764)

[8. Industry Standard Best Practices 4](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **10/19/2022** | **Ken Rodriguez** | **Implemented recommendations and explanations for algorithm cipher,** |

## Client



## Developer

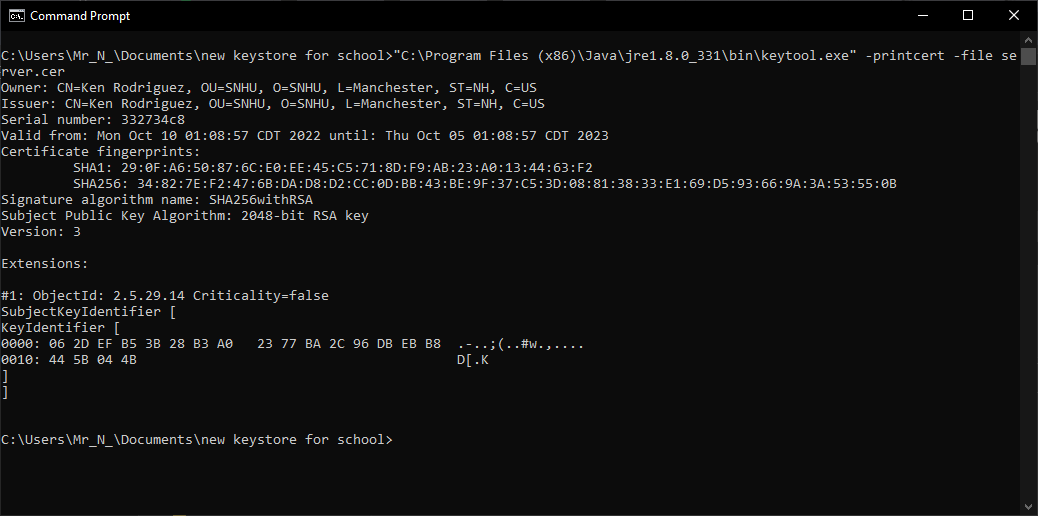
Ken Rodriguez

## Algorithm Cipher

To meet the client’s needs, I would suggest that the organization utilize the SHA-256 algorithm cipher. The SHA family of algorithms was developed in a competition to provide a new cryptography standard for the United States’ National Institute of Standards and Technology; these algorithms are available for private and public use and provide varying levels of security and performance that can be determined by user needs. In this case, the SHA-256 algorithm provides a value that is 256 bits long, providing an encrypted result of a “random” string of characters and numbers to serve as a representative hash value. There are other SHA algorithms that provide shorter results (offering greater performance) or longer results (offering greater security), I believe that utilizing SHA’s 256-bit algorithm will provide a good balance of encryption speed and quality.

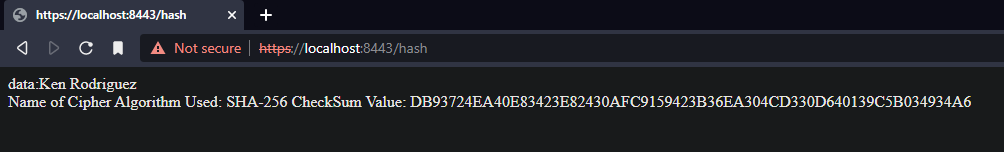
## Certificate Generation

Insert a screenshot below of the CER file.



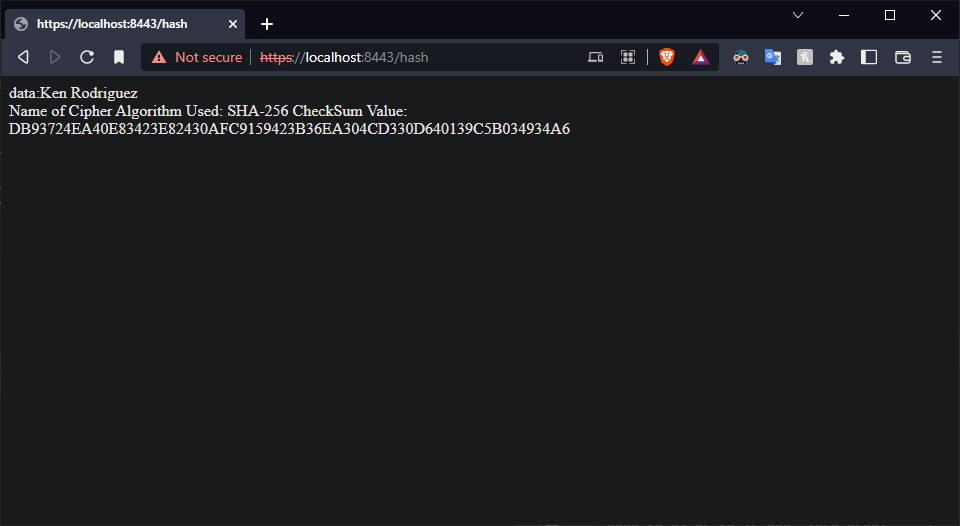
*Example of a generated certificate utilizing the Java Keytool. This certificate has been issued by Ken Rodriguez for the organization SNHU in Manchester, New Hampshire.*

## Deploy Cipher



*Example of the SHA-256 algorithm being used to encrypt the string “Ken Rodriguez” and the hash value associated with it.*

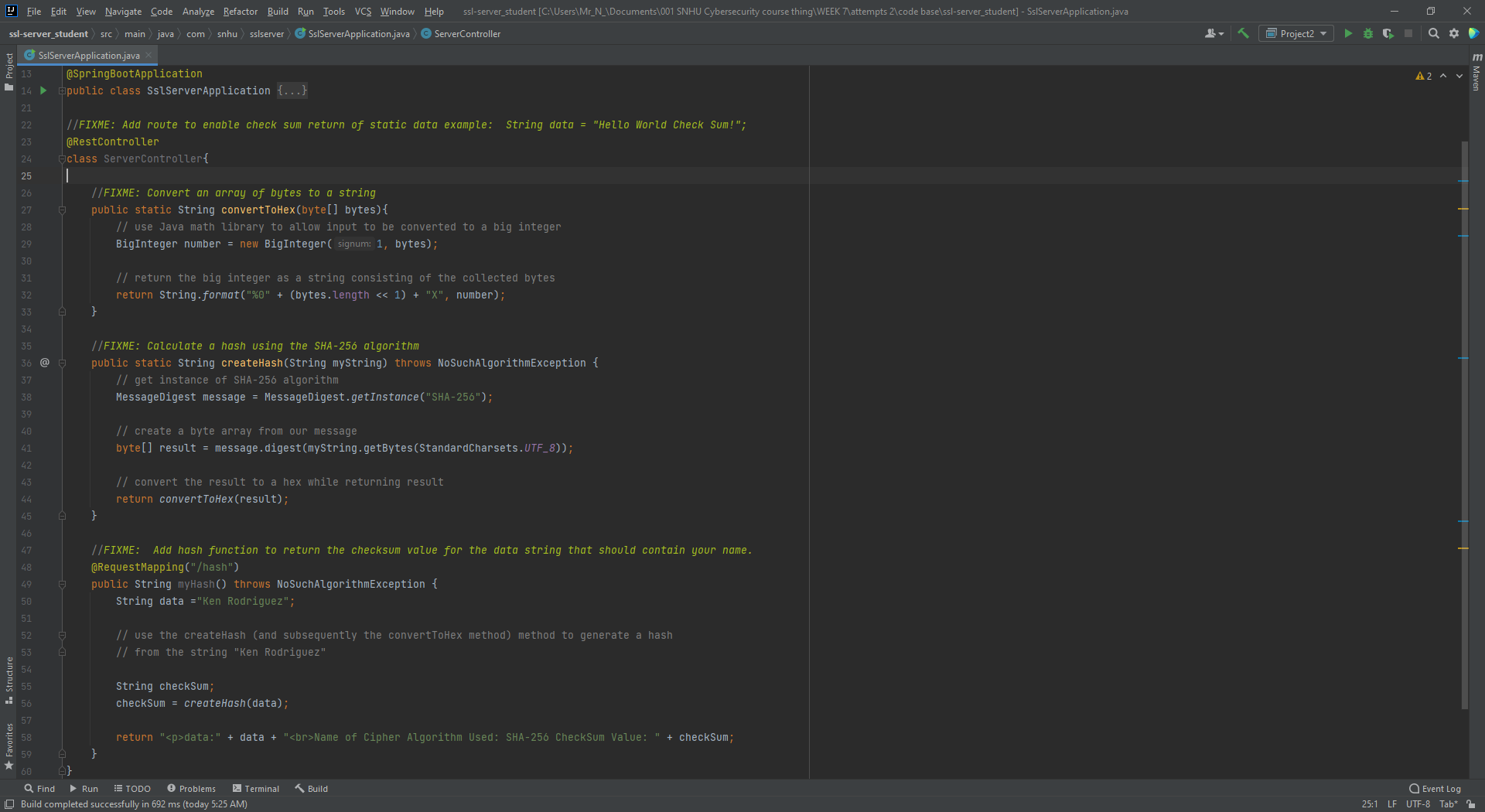
## Secure Communications



*Webpage displaying the connection to localhost using the https qualifier. Due to the browser flagging a common name error, the self-signed certificate was deemed insecure.*

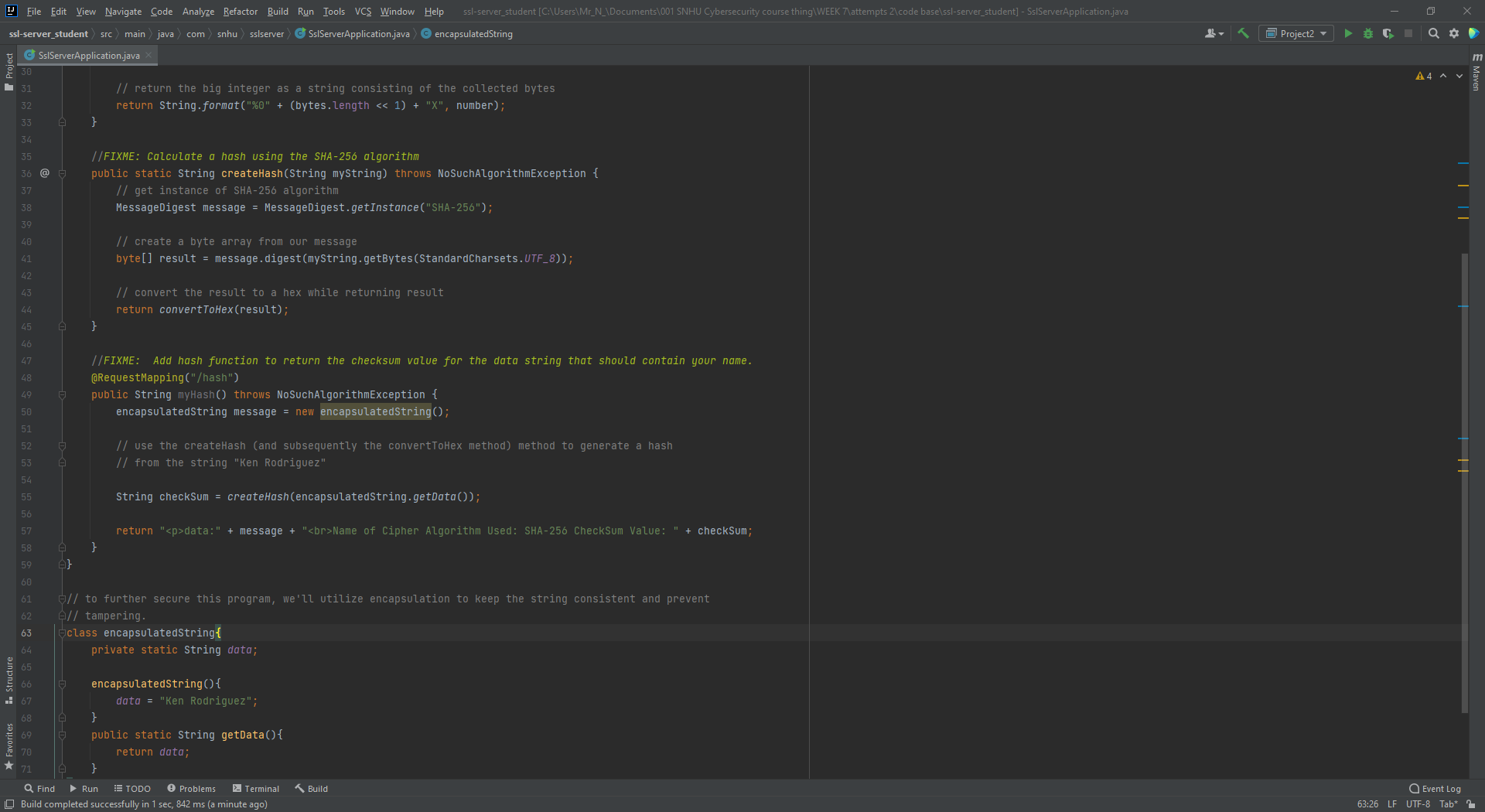
## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.



*In addition to the “main” method for booting Spring framework, three more methods were added to create an encrypted hex value, convert that hex to a string, then print that hex value.*

## Functional Testing



*To further secure the application, encapsulation is utilized to prevent tampering with the data string “Ken Rodriguez”.*

## Summary

To refactor the existing code, four primary methods are implemented to ensure that the information provided is inaccessible to third parties. First, the message to be encrypted is implemented and **encapsulated** in its own class, with a method returning the string “Ken Rodriguez”. This string can only be returned and cannot be modified, ensuring unauthorized third parties may have trouble tampering with it. Next, a method is implemented to create an encrypted SHA hash from an inputted string; this will be used to encrypt the message “Ken Rodriguez” and return a byte result. For the purposes of this application, however, the byte result is immediately converted using an implemented conversion method that converts the string of bytes to a string for printing to the web interface. Finally, a method is used to put all these pieces together and map the plaintext string “Ken Rodriguez” along with information relating to the hash and methods used to the domain “/hash”. To tie everything together, certification is used to ensure a safe connection between the client and server.

Observing the vulnerability assessment diagram, the first point addressed by these changes would be cryptography, as an SHA algorithm is used to encrypt a message for the client’s security. Next, the client/server aspect would be addressed as a self-signed certificate is implemented to ensure that the domain being used is legitimate. Each method implements throw exceptions if an algorithm is being used improperly; code error should handle errors without leaking information to third parties. Code is commented and variables are named relative to what they are, and sensitive data is accessible, but not directly modifiable; code quality and encapsulation are accounted for here. These pieces should work together to ensure general software security and that development can continue to scale to client needs.

## Industry Standard Best Practices

Here, best practices are used to ensure that developers can continue to work on this application as the organization’s needs grow. Code is commented frequently to explain what lines or groups of lines are doing in enough detail to understand their functions and variables are similarly named accordingly. If development is continued to make the program more secure, it should not be difficult for developers to identify possible security flaws and update the program accordingly. Given the company’s responsibility to protect the data stored in its servers, I believe implementing best practices to be the best way to ensure that this need is met.