****

# 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** | **4/19/2025** | **Rich Knoll** |  |

## Client



## Developer

Rich Knoll

## Algorithm Cipher

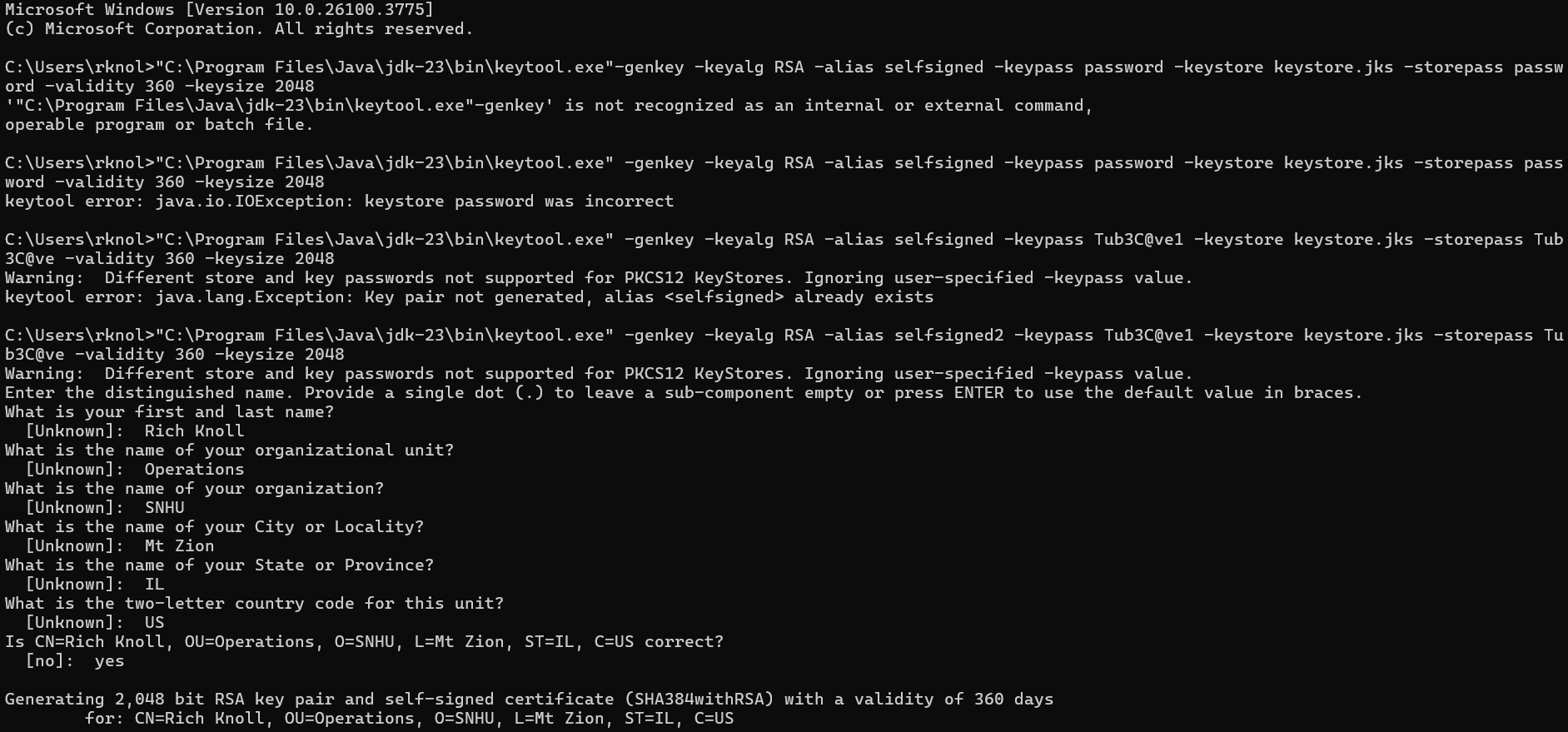
We will be implementing the SHA-256 hash algorithm to ensure the integrity of transferred data. SHA-256 will take the data transferred as an input and run it through the cryptographic hashing algorithm. This algorithm will produce a 256-bit output as a hexadecimal number of 64 digits. This output will be the fingerprint of the transferred data.

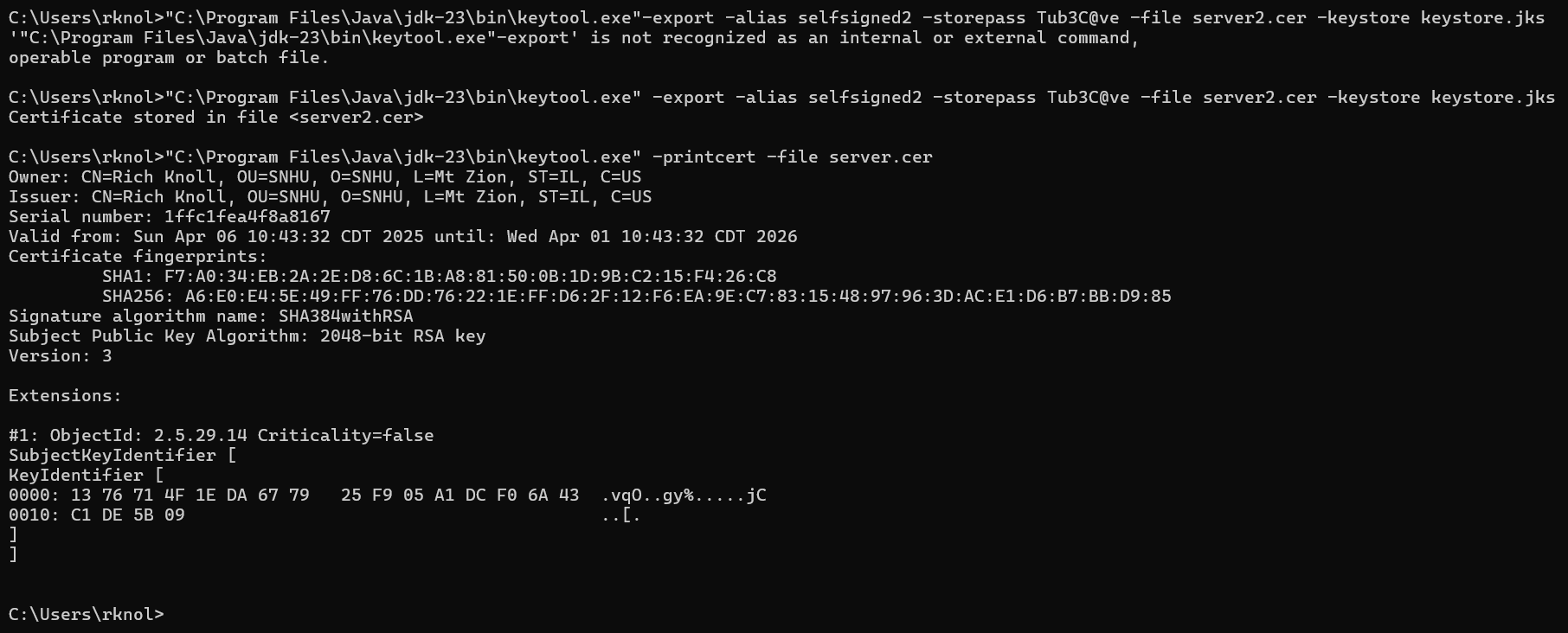
There are a couple key points to understand with this procedure. First, the hashing function will produce a deterministic output for every input. This means that every time the same input is ran through the hashing algorithm, the same output will be produced, which is the fingerprint of that input.

Second, the hashing function will always produce a fixed-length output, regardless of the input length. Third, the hashing function is irreversible. This means that once an input is ran through the hashing function, it cannot be returned to it’s original form.

Knowing these things, we can now understand how this algorithm will ensure the integrity of our transferred data. Every time we transfer data, we will encrypt it, and then run it through our hashing function, producing the fingerprint of that data. Then the data will be transferred with it’s hash. Upon receipt, the recipient will take the transferred data and run it through the hashing function again. If the output matches the hash that was sent with the data, then the data was received in its original form. If the output differs from the hash in any way, we know that the integrity of the data cannot be trusted.

## Certificate Generation





## Deploy Cipher

Insert a screenshot below of the checksum verification.

A screenshot of a computer

AI-generated content may be incorrect.

## Secure Communications

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

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

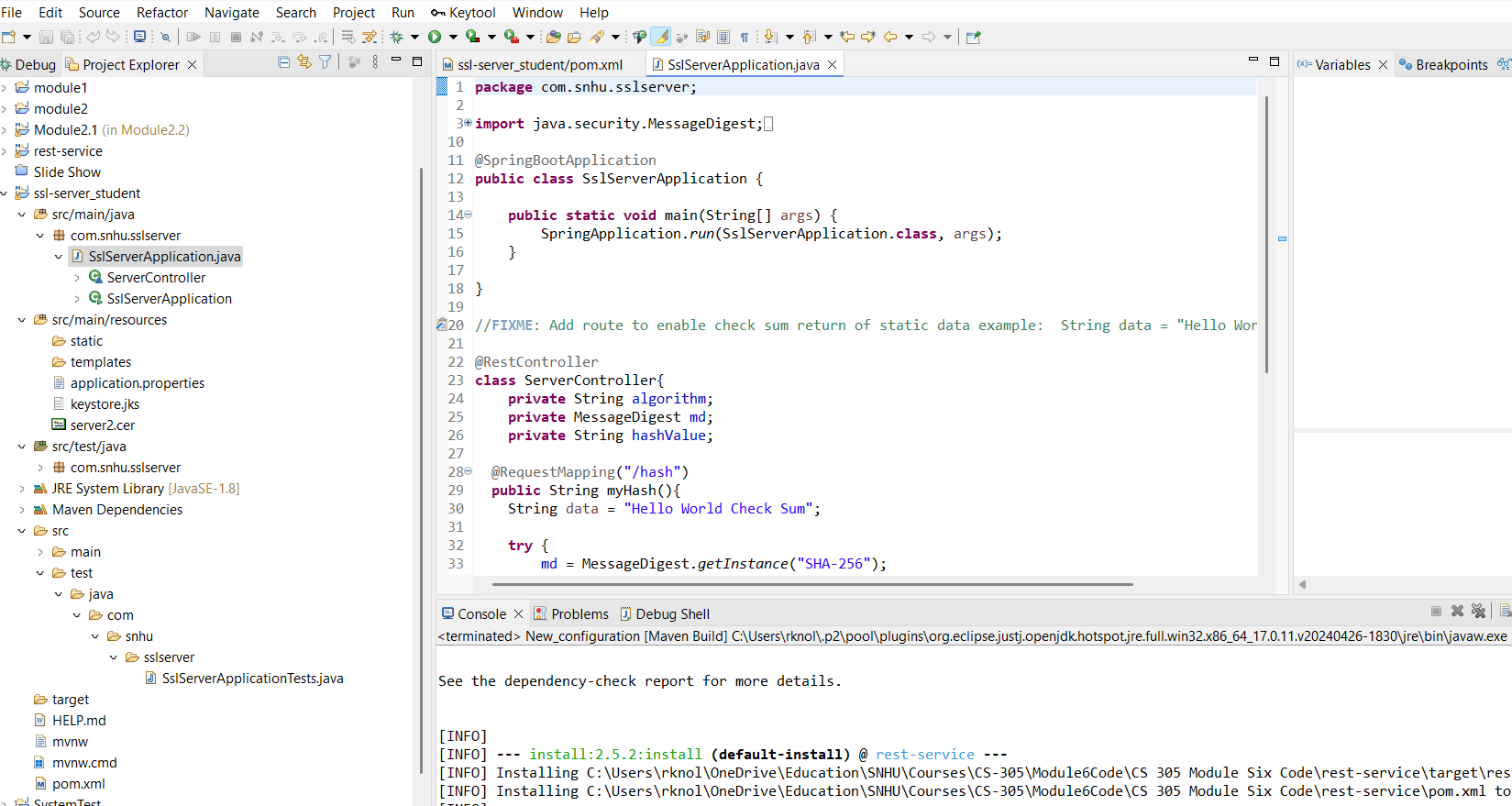
## Secondary Testing

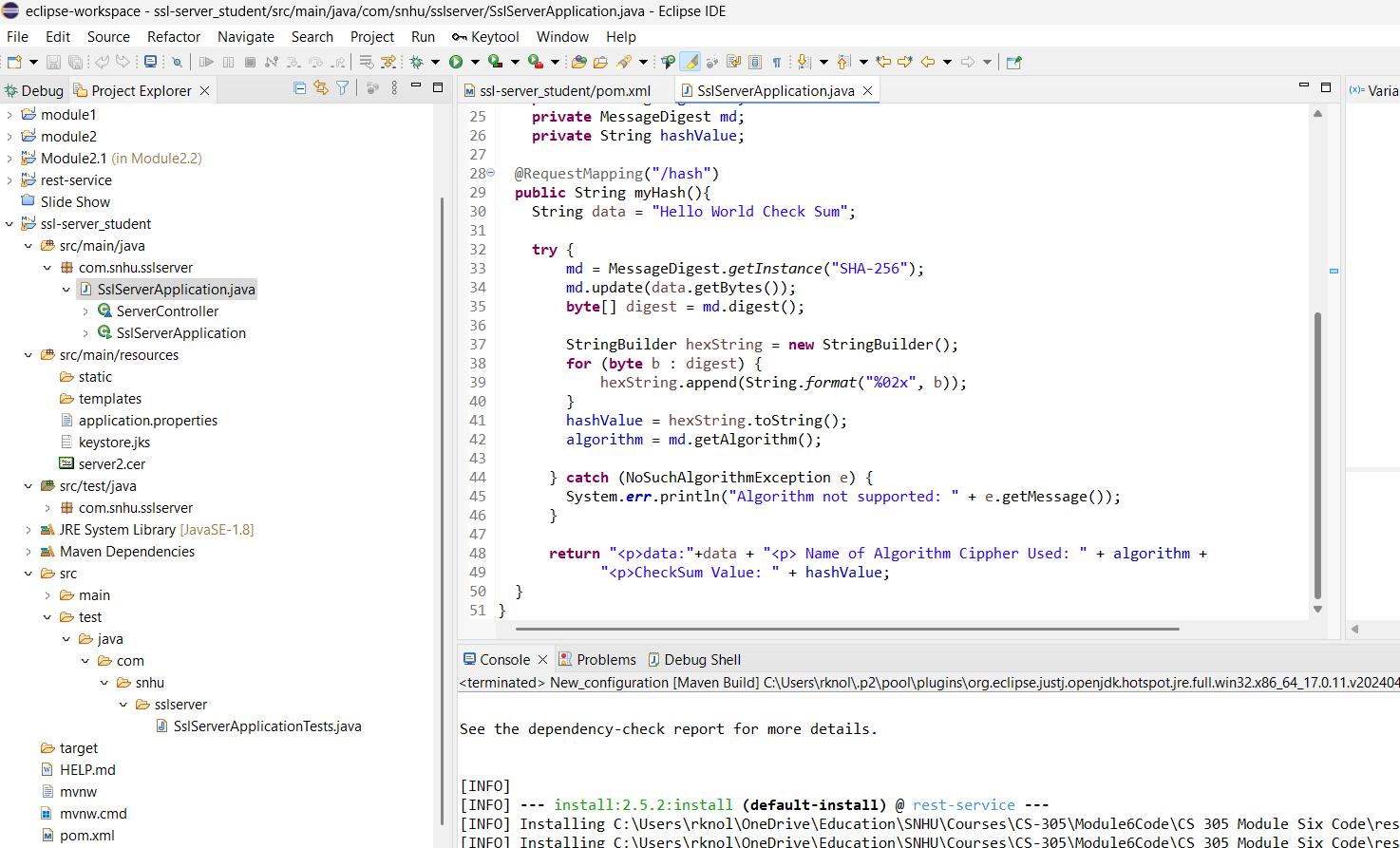
Insert screenshots below of the refactored code executed without errors and the dependency-check report.

Ran Maven build for dependency check and could not get html file to populate in my project even after successful build.

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.





## Summary

We went through the Vulnerability Assessment Process flow and determined that the main areas for vulnerability were cryptography and client/server interaction. The main goal of this project was to add security to our client’s data transmission protocol. As such, the first thing that we did was add a hashing algorithm to ensure that the data shows up in the same condition that it was sent. Basically ensuring that the received data matched the original content.

Next, we are employing certificates to ensure that the client is talking to the server that it wishes to communicate with. We did this by adding trusted certificates to the program’s trust store. This will keep track of trusted servers so that we can ensure we are communicating with who we think we are communicating with.

## Industry Standard Best Practices

Industry standards and best practices are important because they are the current best known practices in the industry. We started by using the SHA-256 hashing algorithm, which is the best known algorithm of its kind today. Next, we are using the TLS communication protocol to securely communicate with servers on a wide network. This protocol employs the use of cryptography, public and private keys, and trusted certificate authorities to ensure secure and trusted communication. It does this by requiring the server to provide a certificate from a trusted authority. Once this certificate has been validated, secure communication can now be established.