****

# 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** | **March 3, 2025** | **Joshua Ladue** |  |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Joshua Ladue

## Algorithm Cipher

For the algorithm cipher I would choose SHA-256. It is a commonly used and very powerful algorithm that has no known cases of being cracked. It would technically be able to be brute forced, but to do that it would take many years to do. The hash function for the algorithm converts whatever the data is that you are trying to encrypt into a string of bits. The bit number of the cipher corresponds to the number of bits that the data is going to turn into. In general, the larger the number, the harder it is to crack as it takes harder mathematical equations to solve.

Random number algorithms can be used to generate pseudorandom numbers while also encrypting them to reduce any blocking in your code that relies on random numbers in any way. Symmetrical ciphers use the same key for encrypting and decrypting, while asymmetrical ciphers use a different one for encrypting and decrypting. This makes asymmetrical ciphers take more time to crack than symmetric ciphers a lot of the time but do offer more security concerns such as distributing keys to everyone that would need one. The asymmetric ciphers would in general be more difficult to manage.

Modern encryption ciphers started with the RES symmetrical encryption algorithm, but this method was quickly cracked and is no longer used today as it is insecure. This brought about the asymmetrical encryption RSA where there were private and public encryption keys which were much more secure than the previous RES cipher. Later in time, due to the rapid advancement of computer processing, ciphers moved back to the symmetrical technique with the AES encryption algorithm. This offered a much longer bit length which made it much harder for a computer to crack and encrypt. This method is widely used today and is one of the most secure methods available, although asymmetrical encryption is still used today to great effect.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a computer program

AI-generated content may be incorrect.

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A white background with black text

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.

## Secondary Testing

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

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Functional Testing

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

A screenshot of a computer

AI-generated content may be incorrect.

## Summary

By refactoring the code, I addressed some of the areas of security to further increase the security of the code. I used cryptography by encrypting the data that is being read through with the SHA-256 cipher. This makes the data much more secure as it would be harder to read from any unwanted eyes. Client and server interactions are also an area that I improved upon by changing the server to be accessible with HTTPS instead of HTTP. It also requires a certificate to access the server which makes it more difficult to be accessed by unwanted users. With API interactions, I updated the dependencies and fixed any issues until there were no vulnerabilities that were being shown in the dependency check, improving the security of the API interactions. Encapsulation is something that I also used to further security. The functions for hashing the data and converting the data into hex are separate functions, along with the server controller and the spring boot being separate classes.

## Industry Standard Best Practices

I developed my code with industry standard best practices in mind. I tried to make my code as readable and easy to understand as possible. Along with this idea, I also commented on the server application code to commentate and explain roughly what is going on with the code as it is read. I used the practices of having proper naming conventions of the variables that were easy to understand for anyone reading the code. The message digest will also throw an exception if it is unsuccessful.