****

# 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 5](#_Toc102040760)

[4. Secure Communications 5](#_Toc102040761)

[5. Secondary Testing 6](#_Toc102040762)

[6. Functional Testing 7](#_Toc102040763)

[7. Summary 7](#_Toc102040764)

[8. Industry Standard Best Practices 7](#_Toc102040765)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **2/23/2025** | **Tristan Maloy** | **Updated initial comments for sections 1-8** |

## 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

Tristan Maloy

## Algorithm Cipher

As a financial institute, Artemis Financial needs strong security, confidentiality, and encryption practices to appropriately protect its customer’s data. I would suggest AES128 encryption utilizing the GCM mode and the hash function SHA256 for the best protection. AES is considered symmetrical as it uses the same key to encrypt and decrypt which means that if the key were to be found, then all data will be able to be accessed. If key management is secure, AES is efficient and ultra secure. To contrast, RSA is another form of encryption that uses asymmetrical keys which means that there is one key for decrypting and one key for encrypting.

AES128 offers efficiency and speed while being resistant to attacks. It uses a 128-bit secret key to encrypt and decrypt data. The key generates random numbers in the length of 128 bits which helps to ensure only authorized access is granted. AES256 would protect even more from cyber-attacks, however it does decrease speed and increase system resource consumption. AES128 is considered highly secure and almost impossible to break, with AES256 containing double the key length.

A secure hash algorithm takes a message and inputs it into a hash algorithm and outputs a message digest that varies in length depending on the bytes. SHA256 takes a message and outputs a 256-byte message that cannot be decrypted. SHA256 is reliable for preventing collisions as it has an extremely small chance of collision, however, SHA512 would be even better. The use of SHA512 would slow down speeds so SHA256 should be more than enough to handle Artemis financials’ needs.

Encryption dates as early as 600BC when the Spartans used a scytale device to communicate secret messages during battle. Modern day encryption practices started in 1917 when Edward Hebern invented an electro-mechanical machine that had the key embedded in a rotating disk. This was then improved in 1918 when Arthur Scherbius invented the Enigma machine which included several rotors with keys that were used to send coded transmissions. Computer based encryption did not start developing until the early 1970s when IBM formed a crypt group that designed a block cypher to protect its customers data. Currently, AES128 and AES256 have yet to be cracked so they ensure security when utilized, however with advancements in quantum computing this might start to change.

## Certificate Generation

Insert a screenshot below of the CER file.

A screenshot of a certificate

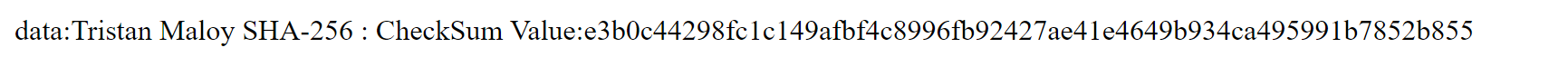
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

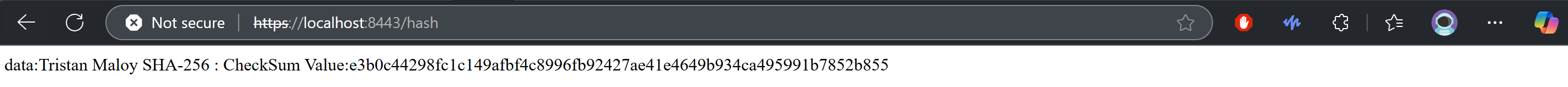
## Deploy Cipher

Insert a screenshot below of the checksum verification.



## Secure Communications

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



I have the certificate done and loaded it into the resources folder, and the application.properties file fixed but I can’t seem to get the page to be secure. I even tried to force the page to load from http to https but that wouldn’t work either. I am not sure what I am doing wrong.

## Secondary Testing

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

A screenshot of a computer error

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

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

Cryptography is essential when it comes to protecting customers and company data. I utilized a SHA-256 hash algorithm so that way the chance of collision was low, but the efficiency of the application wasn’t hindered. SHA512 would provide more protection and encryption for the application but would end up slowing down the system.

Client/server was the area of security that was addressed when refactoring the code. The ServerController was coded to help with communicating and converting the message digest from the server to the client. Annotations such as @RestContoller and @RequestMapping were used to help SpringBoot communicate from the server. Unfortunately, I was not able to get the server to become secure, even though I had a certificate and then tried to force the website to load using HTTPS. With that being said, APIs were also used during the project in order to make a secure connection. A self-signed certificate was generated which was supposed to allow a secure website. If the site was secured, information would have been safe from leaks and unauthorized access.

Adding layers of protection started with generating a certificate to allow a secure environment for the server to communicate with. Once the certificate was generated, the secure hash algorithm took the data and encrypted it and output the encrypted string to the secure website where the client has access to the information.

## Industry Standard Best Practices

Striving for HTTPS was the one industry standard best practice that I was hoping to achieve with this project. HTTPS would have helped to ensure protection of user data and company data. Performing a security test and suppressing the necessary vulnerabilities is also essential for industry standard best practices. This helps to narrow down false positives and lets the program appropriately identify high risk vulnerabilities. Certain vulnerabilities that were suppressed were those based on having an updated spring boot and tomcat version which also helped to improve the security of the system.

Updating the application.properties file with the appropriate server key information was also essential for industry standard best practices as that was supposed to allow the server to be secure when loaded by the client. Being a financial institute, proper key management is essential for Artemis financial.