****

# 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** | **12/12/2024** | **Taylor Jones** |  |

## Client



## 

## Developer

Taylor Jones

## Algorithm Cipher

Given the sensitive nature of the information Artemis Financial must store and communicate for its operations, I recommend deploying a combination of the SHA-256 hashing algorithm with the AES-256 encryption cipher. Using these techniques in tandem will provide comprehensive security featuring exceptional confidentiality, data integrity checks, and authentication. For example, financial records can be encrypted with AES-256 for confidentiality, while SHA-256 hashes verify that the records arrive as intended from sender to receiver.

The 256-bit version of the Secure Hashing Algorithm (SHA-256) creates hashes that can be thought of as unique fingerprints for data. “[SHA-256] takes an input (message) and produces a fixed-size hash value, typically 256 bits long. Its primary purpose is to verify the integrity of data and create a unique representation of a message” (Das, 2023). In practice, the hash values are compared between the sender and receiver of a message to verify that the data has not been tampered with. SHA-256 is designed to be collision resistant, meaning that it is extremely unlikely that two different inputs would produce the same hash output, it is relatively fast compared to other secure hash functions, and it has been adopted as an industry standard for finance, healthcare, and government applications (White, 2024). According to IBM (2024), SHA-2 supports key sizes of 224, 256, 384, and 512-bit key sizes balancing security with processing efficiency.

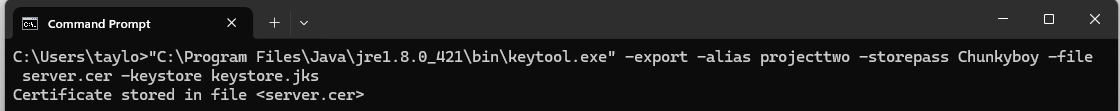
Meanwhile, the Advanced Encryption Standard with 256-bit key length (AES-256) is a symmetric key encryption algorithm meaning that the same key is used for both encryption and decryption and is regarded as the “the most secure encryption algorithm available today and is used extensively in government and military applications, as well as by businesses operating in highly regulated industries” (Kiteworks, 2023). Utilizing an algorithm with symmetric encryption like AES-256 is also very efficient for encrypting large amounts of data when compared to a non-symmetric encryption algorithm making it ideal for Artemis Financial’s operations. Symmetric encryption algorithms are optimized for handling large volumes of data because they use simpler mathematical operations which reduces computational complexity and overall processing time. Additionally, AES utilizes secure pseudo-random number generation to create cryptographic keys to prevent attackers from recognizing patterns or trends that could be used to exploit systems.

The result of deploying SHA-256 and AES-256 for Artemis Financial is secure communications where both the sender and the receiver need to have matching “fingerprints” (hashes) to authenticate data as well as the correct “vault combination” (key) to access the confidential data. AES-256 and SHA-256 are widely considered the gold standard as of the time of writing this and were developed to address vulnerabilities in older algorithms such as DES or MD5. As technology continues to advance and quantum computing becomes more accessible, improved security technologies will be required, but for the time being these algorithms offer the utmost security available to the industry.

## Certificate Generation

A screenshot of a computer program

Description automatically generated



A screenshot of a computer

Description automatically generated

(Chunkyboy is my dog’s nickname.)

## Deploy Cipher

A screenshot of a computer

Description automatically generated

## Secure Communications

A screenshot of a computer

Description automatically generated

## Secondary Testing

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

## Before:

A screen shot of a computer program

Description automatically generated

After:

A screen shot of a computer

Description automatically generated

## Summary

When addressing security concerns for the Artemis Financial project, I focused on ensuring that the cryptography was very secure, the code quality was as high as possible, sensitive information was encapsulated to avoid revealing information attackers could use, and that exceptions were handled in a way to provide minimal information.

Regarding refactoring SslServerApplication.java, I added a secure Rest Controller for the “/hash” endpoint. In doing research to improve the security of my first iteration of the controller, I noticed that my exception handling was exposing the stack trace which is generally not a good idea as it could reveal sensitive information to attackers. To address this vulnerability and minimize exploitable code, I declared a private instance of MessageDigest so that it cannot be accessed or modified externally. I then securely initialized the creation of the Digest with @PostConstruct which is nested in a try/catch block to handle exceptions. The hash computation converts an input string into a byte array and then passes that array to the digest method which computes the hash and returns it. The hash value is then converted to hexadecimal format and finally to a normal string that is returned.

One of the key security enhancements as well as challenges for this project was enabling HTTPS. After some research, I discovered that self-signed certificates need to be configured properly, or most modern browsers will throw warnings that show your communications as insecure. To fix this, I generated a new certificate specifying “localhost” as a subject alternative name (SAN) using the -ext option. I also installed the certificate to the Trusted Root Certification Authorities which was the last step in ensuring HTTPS was configured properly enabling secure communications. After working on this project, I can see the appeal of using a known certificate authority to handle these things because it is very time consuming and meticulous to set up manually. Additionally, the pom.xml file was refactored to ensure that all dependencies were checked (updated to version 11.1.1) for vulnerabilities and a suppression.xml file was added to suppress false positives. The program could be further secured by updating the dependencies to patch known vulnerabilities. For production level software, the hashing function would also need to avoid being hard coded in as this is a major vulnerability done for the sake of the project.

## Industry Standard Best Practices

When this product is moving towards production, the code should be refactored so that the hashed data is not hard coded in. Input validation should be included to prevent harmful strings from being passed into the system to prevent SQL injection or cross-site scripting attacks from occurring. Dependencies should be updated routinely to ensure that known vulnerabilities are patched. Error handling should provide the minimum amount of information necessary to diagnose issues while not providing potential attackers clues on how to exploit the system. Users are often one of the most vulnerable points of a system, so utilizing the most secure authentication methods, such as 2FA, and password management systems can help to prevent unauthorized access. Finally, utilizing protocols like HTTPS ensure that data is communicated securely and to the correct recipient so that confidentiality and data privacy is maintained. These secure coding practices are crucial for maintaining customer trust, staying within industry regulations, and safeguarding the company from attacks.

References

Algorithms and key sizes. (n.d.). Www.ibm.com. <https://www.ibm.com/docs/en/zos/2.4.0?topic=2-algorithms-key-sizes>

Das, D. (2023, October 10). *Difference Between AES and SHA256*. Medium. https://diptendud.medium.com/difference-between-aes-and-sha256-706d6b2eb2ef‌

Kiteworks. (2023). Everything You Need to Know About AES-256 Encryption. Kiteworks. https://www.kiteworks.com/risk-compliance-glossary/aes-256-encryption/‌

White, M. (2024, November 12). [New research] How well does SHA256 protect against modern password cracking? Specops Software. <https://specopssoft.com/blog/sha256-hashing-password-cracking/>

‌