****

# 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** | **6/25/2023** | **Eyoel Tesfu** |  |

## Client



## 

## Developer

Eyoel Tesfu

## Algorithm Cipher

AES (Advanced Encryption Standard) is a symmetric encryption algorithm. This would mean that it would use the same key for encryption and decryption. This would also mean that both the sender and receiver must have access to the same key for effective communication (for communication with better speed and less resources). Although extremely efficient in the 128-bit form, it also can use 192- and 256-bit keys for even more demanding encryption requirements. The highest level of security is offered by AES 256, which is considered nearly impenetrable at this time (Sousi et al., 2020)

If this key is intercepted during transmission, the data could potentially be decrypted by an unauthorized person. For this reason, AES is usually paired with an asymmetric (public-key) encryption like RSA (Rivest–Shamir–Adleman). With RSA, people can share their public keys with anyone who wants to send encrypted data. They will use the public key to send the data, and the receiver will use the private key to decrypt the data. This can be considered as one of the key risks when dealing with AES. So, in practice, both symmetric and asymmetric encryption are needed. RSA would be used to securely share an AES key, which is then used to encrypt the actual data.

In terms of which cipher algorithm is actually best, I would consider choosing AES-256. This is a very secure choice because the only solution there may be to decrypt it is brute force that will probably take more time than the age of the universe.

Governments worldwide widely accept AES-256. The US National Institute of Standards and Technology (NIST) approves AES-256 to protect information which is considered to be top secret. Additionally, it can be used in encrypting large amounts of data, encrypting user data, and securing communications between servers and the like.

Looking at its history, AES was developed by the National Institute of Standards and Technology (NIST) in 2001 in order for it to replace the aging Data Encryption Standard (DES).

The AES-256 algorithm goes through fourteen rounds of encryption through randomization and adding stuff on top of the key. AES uses random numbers during the initialization of the vector in cipher block chaining mode. This enables it to make each encryption unique from another.

AES’s hash functions and bit levels help in maintaining the data’s integrity. Identifying data change or tampering can be done by identifying changes the fixed hash function (which happens when the data is tampered with).

Some of the reasons for not choosing the most secure cipher include prioritizing faster algorithm speeds, requiring lower resource costs, and incompatibility with older systems.

It is imperative to remember that encryption is only a small part of a security strategy. Regular updates, secured passwords, two factor authentication, secure coding, and continuous monitoring are all also critical aspects of maintaining secure systems and networks.

However, on a later requirement, we are asked to deploy a cryptographic hash function. As such, we will be using SHA-256, or Secure Hash Algorithm 256.

SHA-256, crafted by the National Security Agency (NSA), was unveiled to the world in 2001. SHA-256 isn't a typical encryption algorithm. It's not there to transform data into unreadable text or symbols. Instead, it's a cryptographic hash function, taking in any piece of data—be it a paragraph, a digital image, or an entire novel—and outputting a unique, fixed-size hash value.

Give it any input and it will churn out a near-unique 256-bit signature. If you change just a single character in your original input—the resulting hash value will shift dramatically. It's like a fingerprint for data. This is a feature that is especially handy when it comes to maintaining data integrity, be it during storage or transmission.

SHA-256’s defenses have stood the test of time, repelling numerous cryptographic attacks. Governments, industries, and organizations like the National Institute of Standards and Technology (NIST) all vouch for SHA-256's capabilities to preserve data integrity.

Underneath the hood of SHA-256, there is a sophisticated dance of bitwise operations and logical functions. The input data is sliced into blocks, with each block going through a series of transformations to birth the final hash output. No two hash computations are the same, and this adds an extra layer of security to the algorithm.

SHA-256 is also super adaptable. From validating data integrity to signing digital documents and even playing a role in Bitcoin mining, it's a tool with a multitude of uses in the realm of cybersecurity.

In the end, while SHA-256 is a rock-solid choice today, the world of cryptography is one of constant change. Technologies evolve, and what's considered ultra-secure today might be outdated tomorrow. So, it's vital to always be ready to adapt and learn as the landscape of security transforms.

## Certificate Generation

A picture containing text, screenshot, font

Description automatically generated

## Deploy Cipher

A screenshot of a computer program

Description automatically generated with medium confidence

## Secure Communications

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

A screenshot of a computer

Description automatically generated with low confidence

## Secondary Testing

The refactored code:

A screenshot of a computer code

Description automatically generated with medium confidence

The dependency-check report before the removal of vulnerabilities:

A screenshot of a computer

Description automatically generated with medium confidence

## Functional Testing

After refactoring the code, it does not seem to have any syntactical, logical, and security vulnerabilities.

Screenshot of successful execution

A screenshot of a computer

Description automatically generated with medium confidence

Screenshot after the removal of vulnerabilities

A screenshot of a computer

Description automatically generated with low confidence

## Summary

The first thing I did was including a self-signed signature on the server side and pointing the properties towards it in application.properties. I also added a hash algorithm that encrypts the data that is to be sent.

The second part of securing the application was getting rid of its numerous vulnerabilities. I used the Maven Dependency-check plug-in to analyze the vulnerabilities. After identifying the vulnerabilities, I updated the java version, the spring-boot-starter-parent version, and the tomcat.apache version to their latest versions. In the end, there was one vulnerability left; the snakeyaml version 1.33. I overrode this vulnerability by injecting snakeyaml version 2.0.

The parts of the Vulnerability Assessment Process Flow I acted upon were APIs, Cryptography, Code Error, and Code Quality.

## Industry Standard Best Practices

To ensure that my code is functional as well as resistant to security threats, I applied the following best practices in my code:

* I would always validate user input before processing it.
* I would handle errors carefully in order to prevent unhandled leakage.
* I applied regular code review for vulnerabilities and applied static analysis tools to scan for security issues.
* I kept the software and its dependencies up to date.

I approached maintaining the application’s security by:

* Performing regular security checks to find vulnerabilities.
* Regularly patching software to fix any security flaw.
* Staying updated with the latest security practices and vulnerabilities.
* Having a process for handling security breaches

Applying industry standard best practices contributes enormously to a company by:

* Reducing the risk of security breaches
* Increasing the trust of customers and contributing to the reputation of the company
* Saving costs by fixing issues early
* Promoting anti-disruptive smooth operation

References

Sousi, A.-L., Dalia Yehya, & Joudi, M. (2020, November). AES Encryption: Study & Evaluation. Research

Gate.<https://www.researchgate.net/publication/346446212_AES_Encryption_Study_Evaluation>.

Stallings W. (2017). Cryptography and network security: principles and practice seventh edition (Global).

Pearson Education.