****

# CS 305 Project Two

**Practices for Secure Software Report**

Table of Contents

[Document Revision History 3](#_Toc33111302)

[Client 3](#_Toc33111303)

[Instructions 3](#_Toc33111304)

[Developer 4](#_Toc33111305)

[1. Algorithm Cipher 4](#_Toc33111306)

[2. Certificate Generation 4](#_Toc33111307)

[3. Deploy Cipher 4](#_Toc33111308)

[4. Secure Communications 4](#_Toc33111309)

[5. Secondary Testing 4](#_Toc33111310)

[6. Functional Testing 5](#_Toc33111311)

[7. Summary 5](#_Toc33111312)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **8/15/2021** | **Jeremia Faust** |  |

## Client



## Instructions

Deliver this completed Practices for Secure Software Report documenting your process for writing secure communications and refactoring code that complies with software security testing protocols.

Respond to the steps outlined below and replace the bracketed text with your findings in your own words. If you choose to include images or supporting materials, be sure to insert them throughout.

## Developer

Jeremia Faust

## 1. Algorithm Cipher

Determine an appropriate encryption algorithm cipher to deploy given the security vulnerabilities, justifying your reasoning. Be sure to address the following:

* Provide a brief, high-level overview of the encryption algorithm cipher.
* Discuss the hash functions and bit levels of the cipher.
* Explain the use of random numbers, symmetric vs non-symmetric keys, and so on.
* Describe the history and current state of encryption algorithms.

The SHA hash algorithm is currently one of the most secure hash algorithms in the market, thus making it one of the most popular hash algorithms. According to freeman Law the SHA family of hash algorithms are considered the most secure on the market because other algorithms have problem with collisions. The current version of the SHA does not allow for collisions. This means that no hash values will produce the same output with two different input values. According to N-Able the US government requires its agencies to use SHA-256 to encrypt sensitive data as it would take attempts to brute force that hash value. What the SHA-256 does it transform the data so it can only be read with a key. Now that the data is protected the data can be posted publicly so users can take the data and decrypt it if they have the key.

Hash functions are critical in protecting the public keys. According to the SSL Store “A hash function is a unique identifier for any given piece of content.” What this does it take the plain data and run it through the hash function, it then spits out a unique cipher text. They are really good at ensuring data integrity in public key cryptography so that the key stays the same while being protected at the same time. Random numbers are used in cryptography to remove any sense to the data thus removing the predictability. Let’s say we encrypt the data, and a hacker comes along to crack the data. The first thing they are going to look for is any patterns because patterns are easy to break. Using random numbers takes away any clues on how to break a cipher. A great example is when Alen Turing cracked the enigma using the encrypted weather reports that were sent back to Germany from the u boats. This lets them find the patterns used and feeds it into their computer “Bombe”.

There are two types of encryptions that can be used to protect data: asymmetric and symmetric encryption. Asymmetric encryption tends to be the most secure of the two. It has a private key and a public key. The public key is used to encrypt and the private key decrypts the data. This is a slower process than symmetric encryption because symmetric encryption uses one key to encrypt and decrypt the data. Due to symmetric using only one key it is less secure than asymmetric encryption. To decide on the best encryption method, you have to balance speed and security. Unfortunately, asymmetric encryption is not really good at scaling with large amounts of data. Asymmetric encryption is generally used to secure communication then switches to symmetric encryption.

Current government law has changed over the years to a point where cyber security laws mostly pertain to how data is protected. In the past there were strict laws on how secure a cipher can be because the government was worried that they would not be able to break open that data with a warrant. This causes a fundamental flaw because leaving a back door to a system provides a way for hackers to get in the system.

## 2. Certificate Generation

Generate appropriate self-signed certificates using the Java Keytool, which is used through the command line.

* To demonstrate that the keys were effectively generated, export your certificates (CER file) and submit a screenshot of the CER file below.

Text

Description automatically generated

Text

Description automatically generated

## 3. Deploy Cipher

Refactor the code and use security libraries to deploy and implement the encryption algorithm cipher to the software application. Verify this additional functionality with a checksum.

* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.

Graphical user interface, text, application

Description automatically generated

## 4. Secure Communications

Refactor the code to convert HTTP to the HTTPS protocol. Compile and run the refactored code to verify secure communication by typing **https://localhost:8443/hash** in a new browser window to demonstrate that the secure communication works successfully.

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

A computer screen capture

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

## 5. Secondary Testing

Complete a secondary static testing of the refactored code using the dependency check tool to ensure code complies with software security enhancements. You only need to focus on the code you have added as part of the refactoring. Complete the dependency check and review the output to ensure you did not introduce additional security vulnerabilities.

* Include the following below:
  + A screenshot of the refactored code executed without errors
  + A screenshot of the dependency check report

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## 6. Functional Testing

Identify syntactical, logical, and security vulnerabilities for the software application by manually reviewing code.

* Complete this functional testing and include a screenshot below of the refactored code executed without errors.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

After completing the functional code review, I found that there is security area that could be an issue. In the applications properties file, there is a plain text password that could be an issue. That password should be encrypted for an added layer of security.

## 7. Summary

Discuss how the code has been refactored and how it complies with security testing protocols. Be sure to address the following:

* Refer to the Vulnerability Assessment Process Flow Diagram and highlight the areas of security that you addressed by refactoring the code.
* Discuss your process for adding layers of security to the software application and the value that security adds to the company’s overall wellbeing.
* Point out best practices for maintaining the current security of the software application to your customer.

The highlight areas of security that was addressed by refactoring the code.

* API’s: this is addressed with the implantation of HTTPS for secure interaction with the interfaces
* Server/client: this implemented with the use of SSL certificates
* Code Quality: this is addressed by using best coding practices and by doing a functional review.
* Cryptography: this is addressed by using RSA and SHA-256 hash function.

For this project we added more security in the form of Dependency checks, checksum hash tables, and the addition of an SSL certificate. We also had to update maven to the most current version so we could produce the most current dependency report. Originally the server was HTTP. To covert the server to HTTPS I needed to create and implement an SSL certificate, which allows the secure communication between server and client. We then created a check sum hash table using the SHA-256 hash function. All these things combined produces a secure application that will be difficult to break into.

Security is not something that can be just add at the end of a development. It is something that need to be addressed in the very beginning to the very end of development. Security does not end with the end of development though it needs to be maintained and modified while in use. We hear stories very regularly of companies being hacked and something like customer data is stolen or a denial-of-service attack preventing the use of the application. We don’t want to have to face this thus we need to keep adding or updating security on a regular basis. If we become complacent on security, we could end up getting hacked and, in the end, lose money.

A brief history of encryption. Thales Group. (2016, April 18). https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption

Baeldung. (2021, March 5). *SHA-256 hashing in Java*. Baeldung. https://www.baeldung.com/sha-256-hashing-java.

. Byrne, K. (2019, September 4). How British cryptanalysts solved the 'unbreakable' German Enigma code, shortening World War II. AccuWeather. <https://www.accuweather.com/en/weather-news/how-british-cryptanalysts-solved-the-unbreakable-german-enigma-code-shortening-world-war-ii/329348>.

Freeman, J. B. (2021, July 17). *Hash collisions explained*. Freeman Law. https://freemanlaw.com/hash-collisions-explained/.

Mehta, M. (2021, January 4). What Is Asymmetric Encryption &amp; How Does It Work? InfoSec Insights. https://sectigostore.com/blog/what-is-asymmetric-encryption-how-does-it-work/.

Mukherjee, L. (2020, June 18). Symmetric vs Asymmetric Encryption - 5 Differences Explained by Experts. InfoSec Insights. <https://sectigostore.com/blog/5-differences-between-symmetric-vs-asymmetric-encryption/>.

says:, O. O. (2021, January 25). What Is a Hash Function in Cryptography? A Beginner's Guide. Hashed Out by The SSL Store™. https://www.thesslstore.com/blog/what-is-a-hash-function-in-cryptography-a-beginners-guide/.

*Sha-256 algorithm: N-able*. N. (2021, April 1). https://www.n-able.com/blog/sha-256-encryption.

Stubbs, R. (n.d.). Symmetric Encryption Algorithms - Their Strengths and Weaknesses, and the Need for Crypto-Agility. Cryptomathic. <https://www.cryptomathic.com/news-events/blog/symmetric-encryption-algorithms-their-strengths-and-weaknesses-and-the-need-for-crypto-agility>.

Top 8 Strongest Data Encryption Algorithms in Cryptography. Web Solutions Blog. (2020, April 19). <https://acodez.in/data-encryption-algorithms/>.

User, S. (n.d.). How Safe is AES Encryption? KryptAll. <https://www.kryptall.com/index.php/2015-09-24-06-28-54/how-safe-is-safe-is-aes-encryption-safe>.

What is the AES algorithm? Educative. (n.d.). https://www.educative.io/edpresso/what-is-the-aes-algorithm.