****

# 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** | **[2/28/2021]** | **[Eric Engman]** |  |

## 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

Eric Engman

## 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.

I’ll be recommending Twofish as the algorithm cipher to use due to the easy implementation, speed, and the fact that there is no successful cryptanalysis of the block cipher. The earlier iteration, Blowfish that was created in 1993 had been used in a considerable number of secure applications, and over time the create Bruce Schneier and his colleagues have updated to the new and refined “Twofish” as higher end CPUs became available to be more efficient at those levels.

This algorithm cipher follows and exceeds the latest government regulations in its encryption methods. The cipher will be used to efficiently encrypt archived files from Artemis Financials’ database. In an article discussing the many available encryption methods, it was stated that, “Many of today’s file and folder encryption software solutions use this [Twofish] method.”

The best cipher is likely the AES (Advanced Encryption Standard). This encryption method is utilized and trusted by the United States government along with many other organizations. Currently said to be uncrackable by all methods other than brute force (which pretty much every algorithm can be cracked through brute force given enough time and computing power), this is the gold standard for encryption. However, it could possibly be overkill on the encryption necessary for our application in encrypting files. Additionally, a Twofish’s speed and efficiency when encrypting files gives it that extra leg up even if it isn’t the gold standard in terms of encryption.

Hashing is hardly an encryption method, more of a method to verify the data. Because any data that is encrypted utilizing the hashing method cannot be deciphered. Symmetric and asymmetric keys are utilized for ciphering and deciphering data. The symmetric method utilizes just one private key that is used to cipher and decipher the data. The asymmetric utilizes two different but related keys known as the public and the private key. This way you have an additional level of security given the two keys.

A very interesting part of history relating to encryption algorithms dates all the way back to world war 2. The German’s used this machine called the enigma machine which took in a key that was used to decrypt morse code messages. This way they could openly use morse code to send messages very easily, but not be concerned with other’s listening since they didn’t have the key and it would be a bunch of gibberish. Currently, things have become much more complex involving hundreds of iterations of ciphering small amounts of data. These algorithms have been able to take advantage of the increase in processing power resulting in more and more complex algorithms for ciphering data.

References

(n.d.). Retrieved February 01, 2021, from https://www.schneier.com/academic/twofish/

Simplilearn. (2020, December 21). The Most Effective Data Encryption Techniques You Must Know in 2021 [Updated]. Retrieved February 01, 2021, from https://www.simplilearn.com/data-encryption-methods-article

## 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.

[Insert screenshot(s) here.]

## 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.

Text

Description automatically generated

I didn’t do this in the terminal because I spent hours trying to work through that, so I used that video recommended to do it directly in Eclipse IDE.

## 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.



## 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, text

Description automatically generatedGraphical 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, text

Description automatically generated

## 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.

In order to get this code up and running, the resources file needed to be supplemented with the keystore and keystore.p12 file along with addressing all of the “????”’s in the application.properties file. This allowed for the certificates to be certified when connecting to the web server.

Best practices in an application like this will be to simply keep this application up to date with version updates while simultaneously checking the dependency reports to keep the application secure.