****

# 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** | **10/17/2021** | **Timothy McIntyre** |  |

## 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

Timothy McIntyre

## 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.

According to my personal research on which cipher algorithms would be best for Artemis Financial to use in their system. It would be, in my opinion, the JSSE Cipher Suite with the standard name of TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA. I chose this algorithm because “The most common set for transactions (not only financial) is AES, RSA and SHA (for hashing).” Patsakis (2016). This cipher algorithm has RSA, AES, and SHA along with ECDHE which is a cipher suite that protects communications and supports forward secrecy.

Symmetric encryption only uses one key which is used for both encryption and decryption. This single key needs to be shared among the people who need to receive the encrypted information. Now, asymmetrical encryption uses a pair of public keys and a private key to encrypt and decrypt information when communicating. Due to the public keys and the private key, Asymmetric encryption takes more time than the symmetric encryption and Asymmetric encryption is a relatively new encryption type and symmetric encryption is an older encryption type.

## 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.

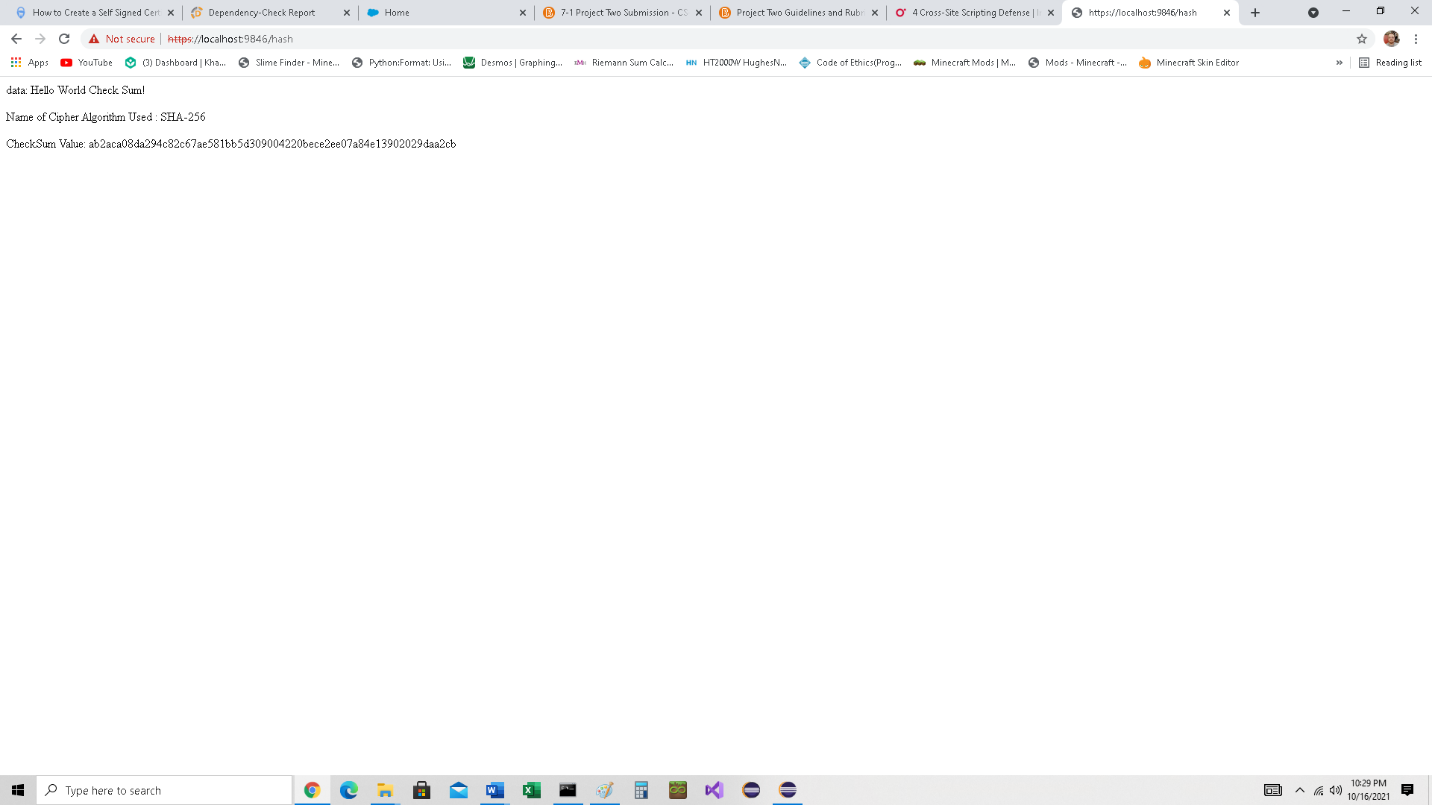
Text

Description automatically generated

## 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.

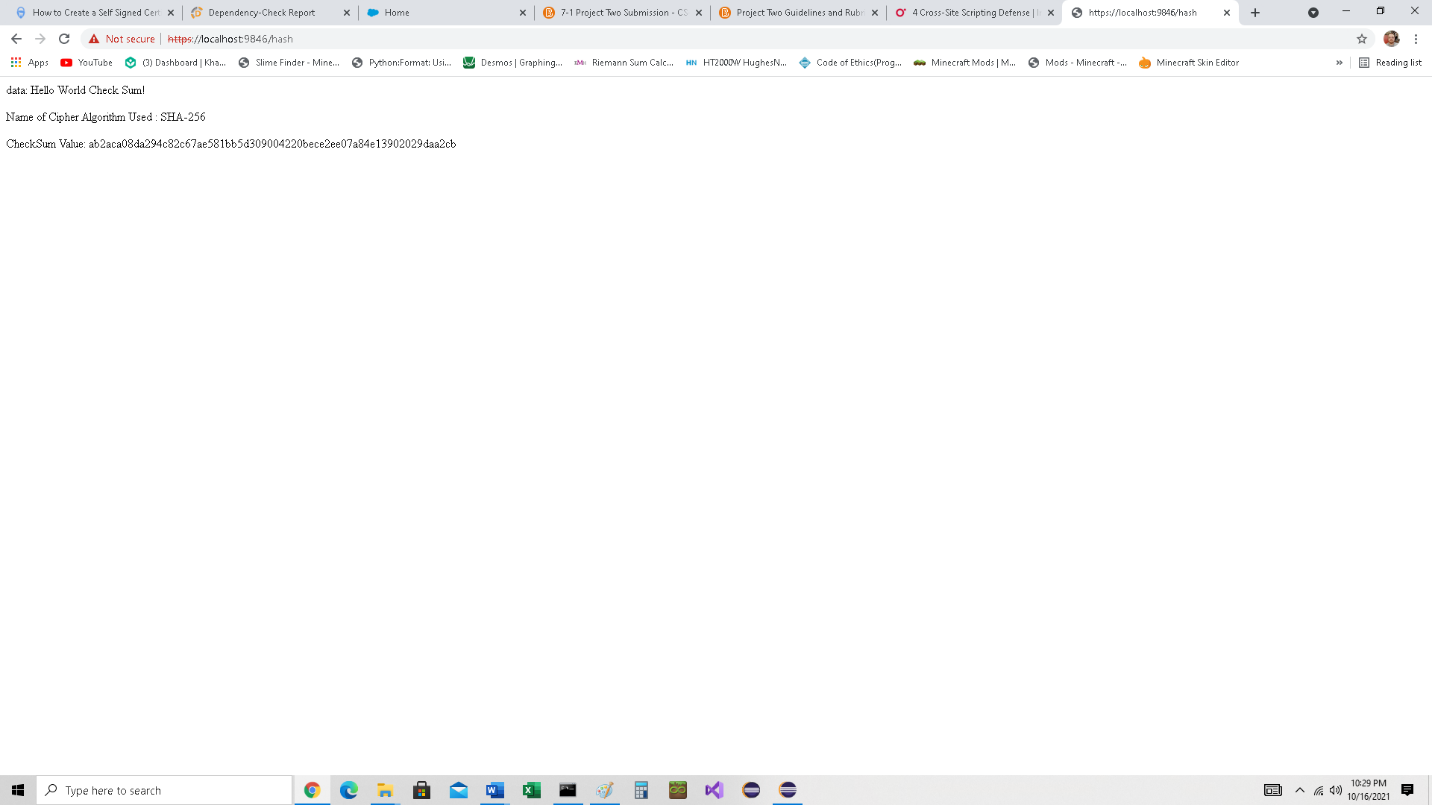
* Insert a screenshot below of the checksum verification. The screenshot must show your name and a unique data string that has been created.



## 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.

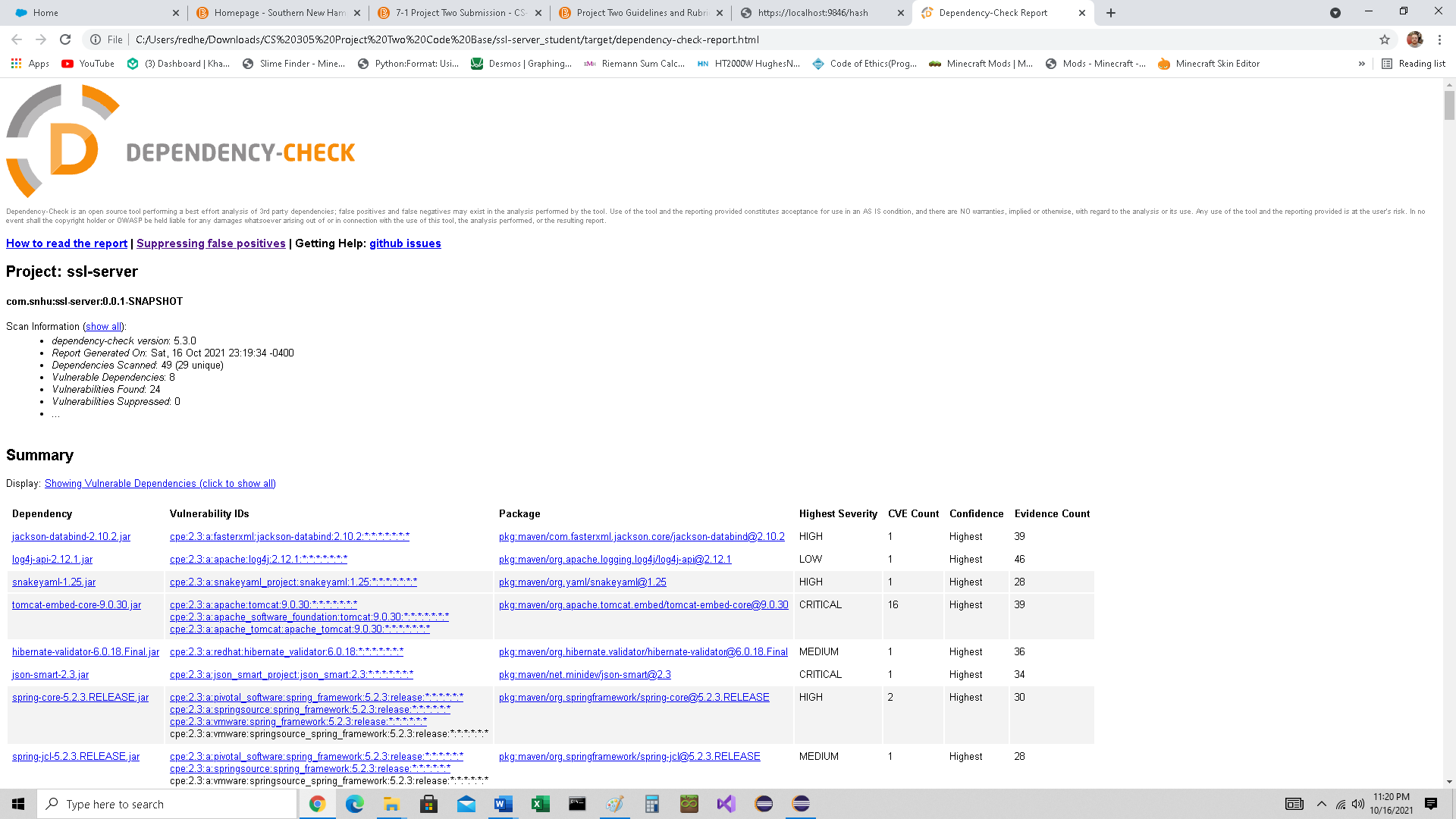
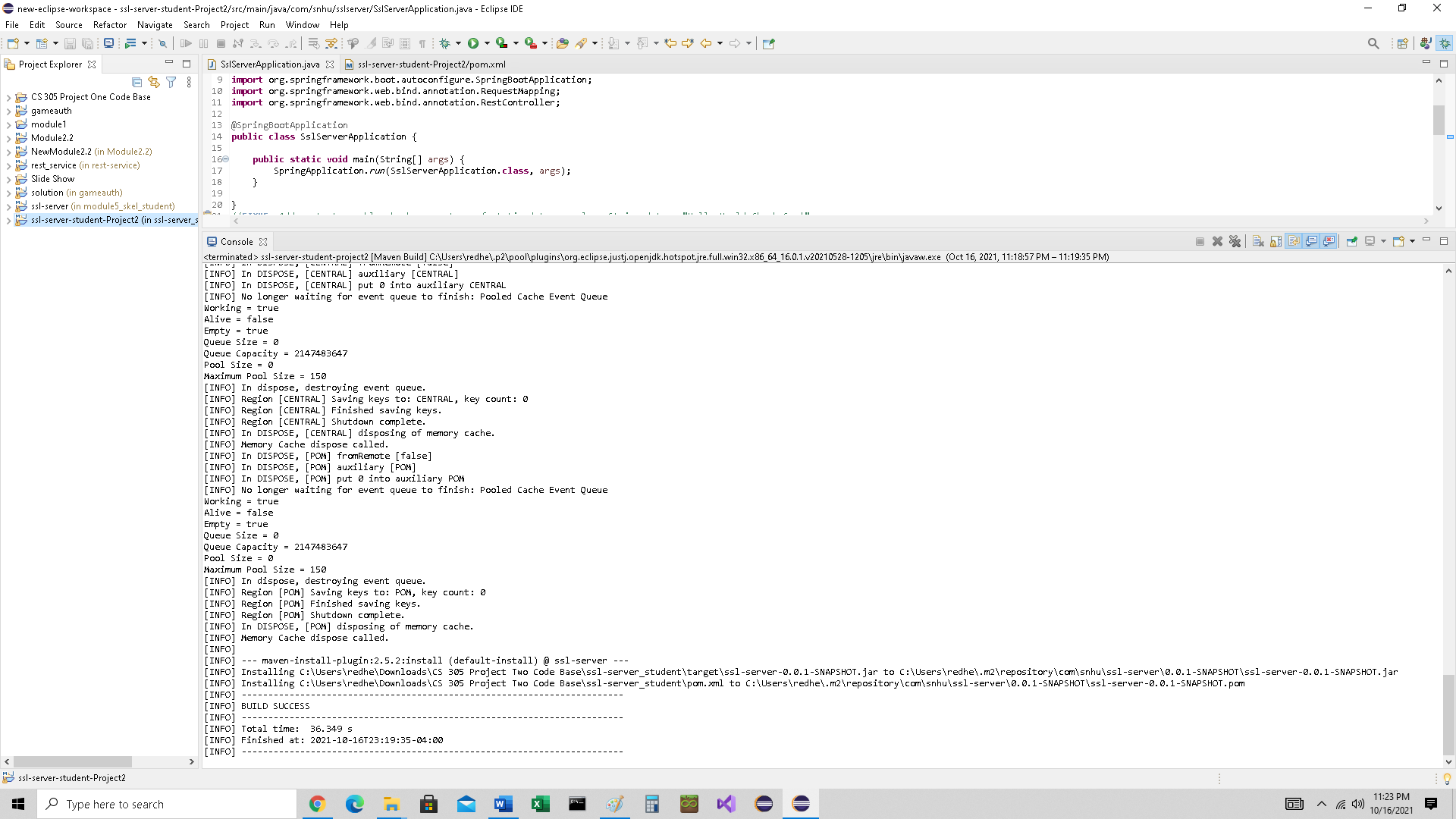
* Insert a screenshot below of the web browser that shows a secure webpage.



## 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



## 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, Word, email

Description automatically generated

Graphical user interface, text, application, email

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.

Hashing is a one-way encryption process and is used in encryptions to secure the information shared between two parties. When it comes to encryption, the latest algorithms may not necessarily be what is best. You should always use the encryption algorithm that is right for the system you are implementing in. It is best that as new algorithms are developed that they are properly vetted to stay ahead of hackers and eavesdroppers. Doing this will assist the company to secure information and enhance confidentiality.

Citations:

Patsakis, Constantinos. (2016). Re: What encryption/ decryption algorithms are used in financial transaction systems?. Retrieved from: https://www.researchgate.net/post/What\_encryption\_decryption\_algorithms\_are\_used\_in\_financial\_transaction\_systems/576e951ef7b67ec6dc49814d/citation/download.