****

# 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** | **10.7.2022** | **Jiwon Lee** |  |

## Client



## Developer

Jiwon Lee

## Algorithm Cipher

To find out which encryption algorithm cipher is suitable for a given situation, we first need to know how the data is protected. SSL/TLS focuses on three factors in data security: confidentiality, integrity, and authentication.

* Confidentiality:

We use encryption to prevent data between the client and server from being exposed to third parties. There are two types of encryptions: symmetric encryption and asymmetric encryption. In the first symmetric encryption, the same key is used for data encryption and decryption between the sender and receiver. Symmetric encryption is less secure than asymmetric encryption because same key is shared for both encryption and decryption. However, since it uses the same key, it can be encrypted at a high speed, so it is suitable for encrypting a large amount of data. Conversely, asymmetric encryption is slower but more secure than symmetric encryption because it encrypts data using two different keys.

* Integrity:

We use a hashing algorithm to make sure that nobody modified the data between the client and server while data is being transferred. The hash function converts dada of arbitrary size to fixed size value. It is impossible to encrypted or decrypted once it is converted to a fixed size value. In the hash function, the bit represents the size of the output digest. When two pieces of data in a hash table share the same hash value, a hash collision occurs. Therefore, the larger the number of the bits, the lower the probability that two pieces of data are the same digest. Key sizes of 128 bits or larger is recommended for bulk ciphers.

* Authentication:

We verify who the data is coming from through PKI. PKI is formed with three entities: Client, Server, and CA PKI authentication allow users of public networks that are not secure, such as the internet, to exchange data securely by using a certification granted by a trusted authority.

A typical example of this is the padlock shape we see when connecting to a web browser. When we connect to the web address of the CA-certified organization, we can see the padlock shaped icon displayed next to the web address, and we can transmit sensitive data to the web that is certified as secure.

Artemis puts security first, saying that security is everyone’s responsibility in their part of the company’s mission. Furthermore, for the client company that deals directly with customer’s assets, security must be treated above all else. So, to recommend a cryptographic algorithm suitable this situation, I focused on the keywords of high security level and file verification.

AES is famous for its strong security that, once properly installed, cannot be breached. The only downside is the tricky installation, the strongest security, AES, is the answer for this company’s needs. It was developed in 1997 by NIST as an alternative to DES. AES is widely used around the world and is chosen by the US government to protect confidential information. The table below is part of my research and explains why the rest of algorithms is not suitable for this company.

|  |  |
| --- | --- |
| Algorithm Name | Reasons not suitable for Artemis Financial |
| AESWrap | The AES key wrap algorithm is designed to wrap or encrypt key data.  The Key Wrap algorithms are intended for applications such as protecting keys while in untrusted storage or transmitting keys over untrusted communications networks. |
| ARCFOUR | RC4 is extremely insecure, so very few applications use it now. |
| Blowfish | Its successor Twofish is recommend to use. |
| DES | The NIST had to replace the DES algorithm because its 56-bit key lengths were too small, considering the increased processing power of newer computers. Encryption strength is related to the key size, and DES found itself a victim of the ongoing technological advances in computing. It reached a point where 56-bit was no longer good enough to handle the new challenges to encryption. |
| DESede | Triple DES Encryption (also known as DES-EDE, 3DES, or Triple-DES). Data is encrypted using the DES algorithm. |
| DESedeWrap | The DESede key wrapping algorithm |
| ECIES | Elliptic Curve Integrated Encryption Scheme is an asymmetric key encryption algorithm. Not suitable for large data encryption. |
| PBEWith<digest>And<encryption> PBEWith<prf>And<encryption> | The password-based encryption algorithm  (couldn’t get access to the resource)  Forbidden Error |
| RC2 | Related to RSA |
| RC4 | Related to RSA |
| RC5 | Related to RSA |
| RSA | The problem with RSA is that as these keys get longer, the increase in security isn't commensurate to the increase in computational power it takes to use them. It's just not sustainable. |

## Certificate Generation

## A screenshot of generating the self-signed certificate

Text

Description automatically generated

## A screenshot of the self-signed certificate

Graphical user interface, application

Description automatically generated

## Deploy Cipher

## A screenshot of the checksum verification

Graphical user interface, text, application, email

Description automatically generated

## Secure Communications

## A screenshot of the web browser that shows a secure webpage

Graphical user interface, text, application, email

Description automatically generated

## Secondary Testing

The OWASP Dependency-Check Maven version 7.2.1 was done, and the refactored code executed without any failures or errors.

* A screenshot of the refactored code executed without errors

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* A screenshot of the first report

Graphical user interface, text, application, email

Description automatically generated

* A screenshot of the Second report

Graphical user interface, text, application, email

Description automatically generated

## Functional Testing

## Code review

Text

Description automatically generated

The function myHash() above has the data directly inside of the function and it is declared as public. It was not used in this project, but in reality, I should consider increasing security by getting data using a getter such a s getData().

## Text Description automatically generated

This function does not expose any sensitive data. It is even declared a s private. This also have catch block that can catch the exception error. The security level is high, and the work of converting the digest into a hash value is performed without any problem.

## SslServerApplication.java

Text

Description automatically generated

* Application.properties

Graphical user interface, text, application

Description automatically generated

## Summary

* APIs: RESTful API has been added for secure information exchange between the client and server. @RequestMapping was used with the method myHash() for secure API interactions.
* Cryptography: getSHA256Hash() was used to transform plaintext data to encrypted message for keeping information secure.
* Client/Server: By generating the self-signed certification, a secure network environment was created between the server and the client through https.
* Code Quality: Secure coding practices/ patterns
* Encapsulation: I declared private in my code, so I can improve security by blocking external access.

## Industry Standard Best Practices

First, I created a self-signed certificate to establish a secure network between the client and the server. This step is a cornerstone for creating a secure communication path for Artemis Financial and the client, which will later be used to utilize HTTPS. Next, I converted the data that was the plain text into an encrypted message through a SHA256 hash algorithm. In this way, we can protect the client's sensitive information while it is being transferred. This ensures that our data is not exposed to hackers even if it is hijacked in the middle. Lastly, I used SSL configuration to use HTTPS in the application. Properties file. Requests and responses occur when data transfers between a server and a client on the internet. In this case, if HTTP is used, information may be leaked in the middle when this request and response occur, so we use HTTPS in the context of a client using sensitive information.

In addition, test work was performed along with secure coding. The OWASP Dependency-check Maven version 7.2.1 found vulnerabilities and checked whether there are any security issues related to the outdated versions. This step reduces the risk of attack by ensuring that the application runs without any weaknesses that hackers can exploit. Manual code testing was also conducted. To find errors that could be missed by software testing, the refactored code was checked manually using the vulnerability assessment process flow diagram to confirm that there are no errors. #6 from the functioning test, you can see that the refactored code was running without any errors or failures.

Under the mission that the company’s security is everyone’s responsibility, I applied the three factors of data security: confidentiality, integrity, and authentication throughout the project. As our customer Artemis Financial handles extremely sensitive data that is directly related to their assets, security is more important than ever. Security is not an option you add as needed. We should always design your system with security in mind. These practices protect the assets of clients and have great significance for the well-being of the company.

**How to secure Artemis Financials overall wellbeing**

Reference:

1. Wikipedia. (n.d.). Hash collision. Retrieved October 9, 2022, from <https://en.wikipedia.org/wiki/Hash_collision>
2. Oracle. (n.d.). Java Security Standard Algorithm Names. Retrieved October 9, 2022, from <https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names>
3. Forbes. Kee., Lila. (2021). RSA is Dead-we Just Haven’t Accepted It Yet. Retrieved September 23, 2022 from <https://www.forbes.com/sites/forbestechcouncil/2021/05/06/rsa-is-dead---we-just-haventaccepted-ityet/?sh=6dd345cd5d22>
4. Manico, J., Detlefse, A. (2014). Iron-Clad Java: Building Secure Web Applications. McGraw Hill Computing
5. Wikipedia. (n.d.). Key warp. Retrieved September 23, 2022, from <https://en.wikipedia.org/wiki/Key_wrap>
6. RFC. J., Schaad. (2002). Advanced Encryption Standard (AES) Key Wrap Algorithm. Retrieved September 23, 2022, from <https://www.rfc-editor.org/rfc/rfc3394>
7. Encryption Consulting. (n.d.). What is RSA? How does an RSA work: RSA Vulnerabilities. Retrieved September 23, 2022, from <https://www.encryptionconsulting.com/education-center/what-is-rsa/#vulnerabilities>