****

# 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** | **[Date]** | **Lance Cain** |  |

## Client



## Instructions

Submit this completed practices for secure software report. Replace the bracketed text with the relevant information. You must document your process for writing secure communications and refactoring code that complies with software security testing protocols.

* Respond to the steps outlined below and include your findings.
* Respond using your own words. You may also choose to include images or supporting materials. If you include them, make certain to insert them in all the relevant locations in the document.
* Refer to the Project Two Guidelines and Rubric for more detailed instructions about each section of the template.

## Developer

Lance Cain

## Algorithm Cipher

For data verification the SHA-256 algorithm is the industry standard as it provides an efficient and fast algorithm with low chances of collision. SHA-256 operates on a 256 bit hash and goes through multiple iterations of bitwise operations, logical functions, and modular arithmetic to derive the hash value from whatever input it receives. SHA-256 being a hash function and not an encryption algorithm, it is only one way and cannot be decrypted except for by brute force measures, which are not feasible on a hash such as SHA-256 in a reasonable amount of time with current technology. This make SHA-256 excellent for verification and signing purposes as it cannot be easily decrypted and re-used. If we were to need the ability to decrypt the payload, then we could use AES instead as it is a symmetrical algorithm.

## Certificate Generation

Insert a screenshot below of the CER file.

A picture containing text, electronics, software, screenshot

Description automatically generated

## Deploy Cipher

Insert a screenshot below of the checksum verification.

A picture containing text, electronics, monitor, display device

Description automatically generated

## Secure Communications

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

This is included in step 3 as I did both steps before taking the picture.

## Secondary Testing

Insert screenshots below of the refactored code executed without errors and the dependency-check report.

A picture containing text, electronics, computer, display

Description automatically generatedA computer screen with text on it

Description automatically generated with low confidence

## Functional Testing

Insert a screenshot below of the refactored code executed without errors.

A picture containing text, electronics, computer, monitor

Description automatically generated

## Summary

Using SHA-256 for the hashing function we have created a secure and industry standard verification method for files or other data we need to have verification abilities for. This combined with RSA signed https certificates give us end to end encryption with our clients as well as secure data verification. When adding layers of security you need to be cognizant of how those layers interact and any new vulnerabilities you may be introducing. The idea is reduce vulnerabilities, not shift them around.

## Industry Standard Best Practices

Using SHA-256 and RSA signed certificates are the industry standard because they work, not simply because they are commonly available. Other more secure algorithms are available at the cost of storage space and processing, but with virtually no effective change in likely outcomes as brute forcing a key that is updated frequently, is nearly impossible. Due to using standards they are well tested and widely understood. This allows for a more secure client server architecture that provides data privacy and security regardless of the applications goals.