****

# Practices for Secure Software Report

**Table of Contents**

Practices for Secure Software Report 1

Document Revision History 3

Client 3

Instructions 3

Developer 4

1. Algorithm Cipher 4
2. Certificate Generation 4
3. Deploy Cipher 4
4. Secure Communications 5
5. Secondary Testing 7
6. Functional Testing 7
7. Summary 9
8. Industry Standard Best Practices 9

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **17JUN2024** | **Gavin Bish** | **Updated page to fill in information requested by Artemis** |

## 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

Gavin Bish

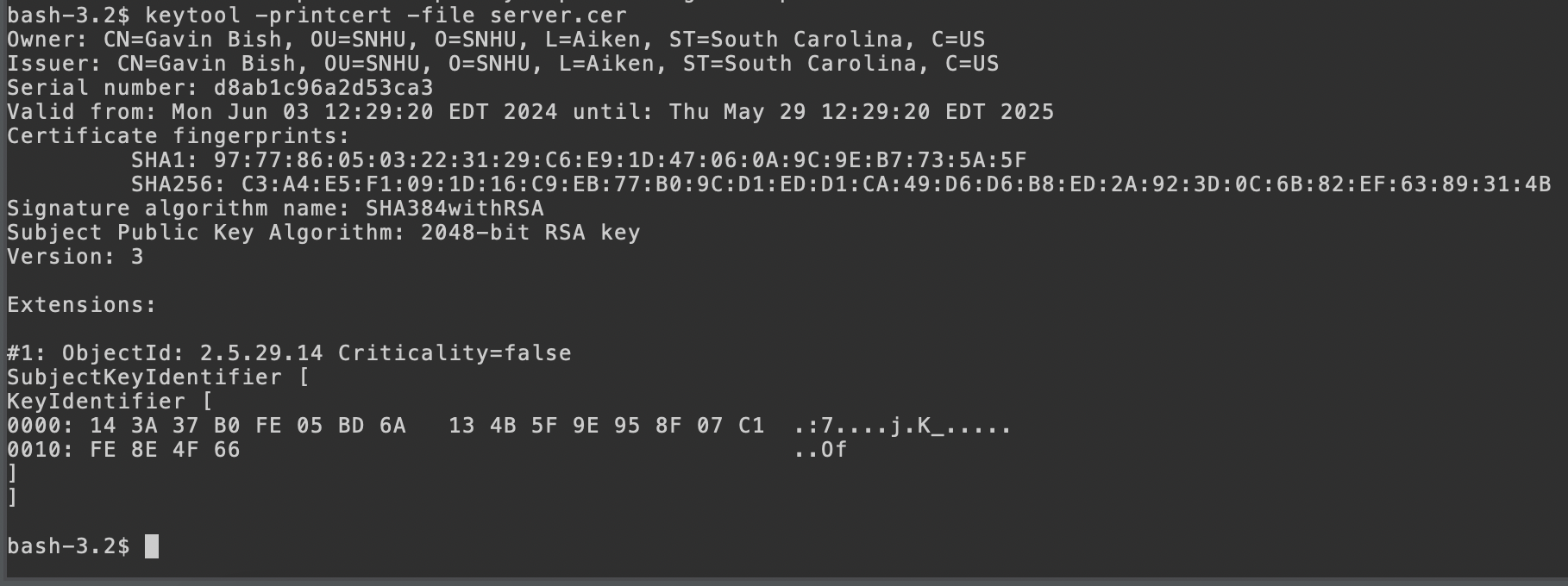
## Algorithm Cipher

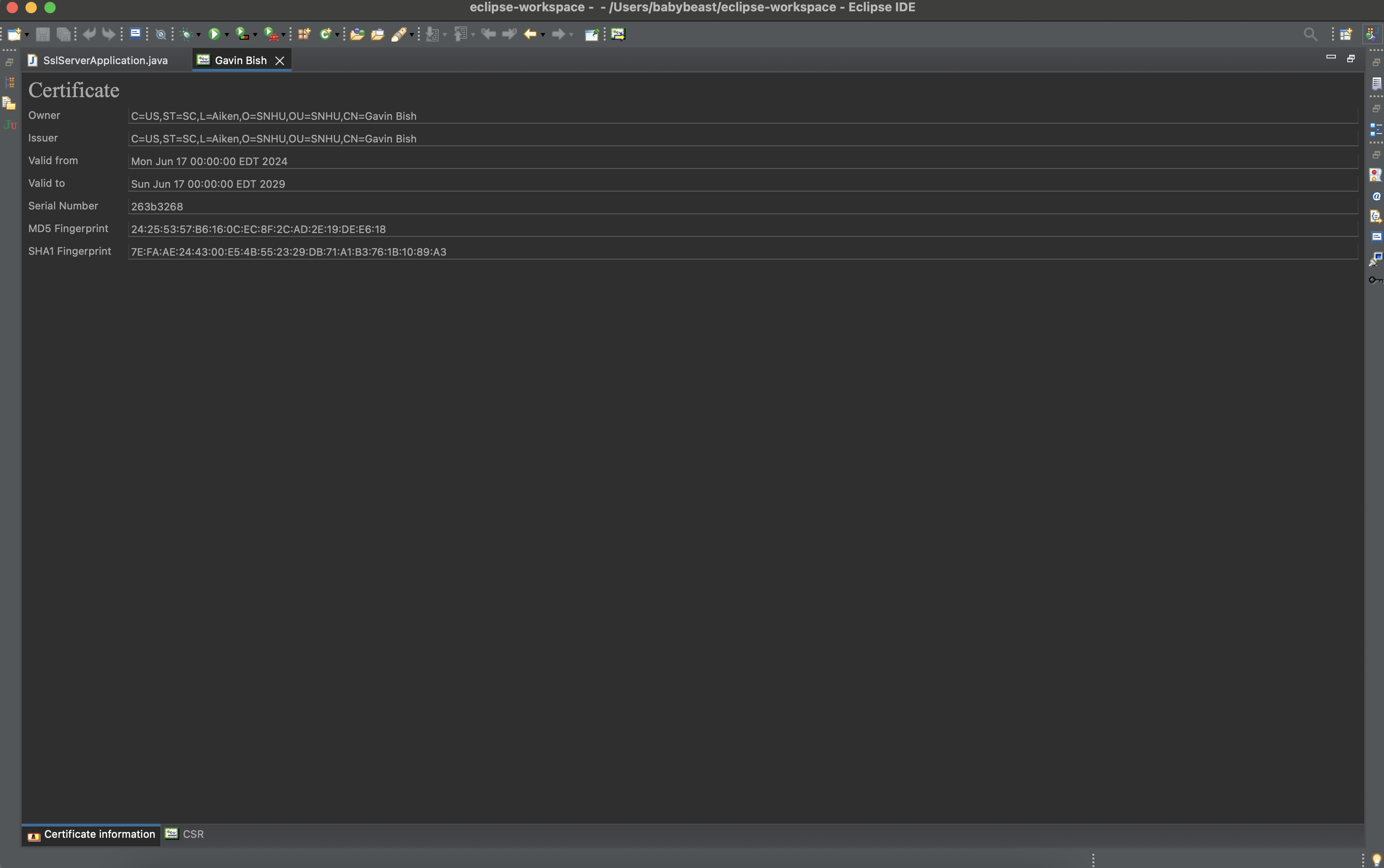
Artemis Financial has requested an encryption algorithm recommendation for securing long-term archive files. Considering that the primary risk is unauthorized access to these files, they should be encrypted in a way that renders them useless if stolen. Since the files will remain in storage and not be transported, asymmetric keys are unnecessary. Additionally, encryption speed is not a concern due to the long-term nature of the archive.

I recommend using the SHA-256 algorithm with 256-bit keys for this purpose. SHA-256 is widely recognized for its security and is included in standard Java installations. The 256-bit key length provides robust protection by offering a vast number of possible key combinations, making it highly resistant to brute-force attacks and collisions.

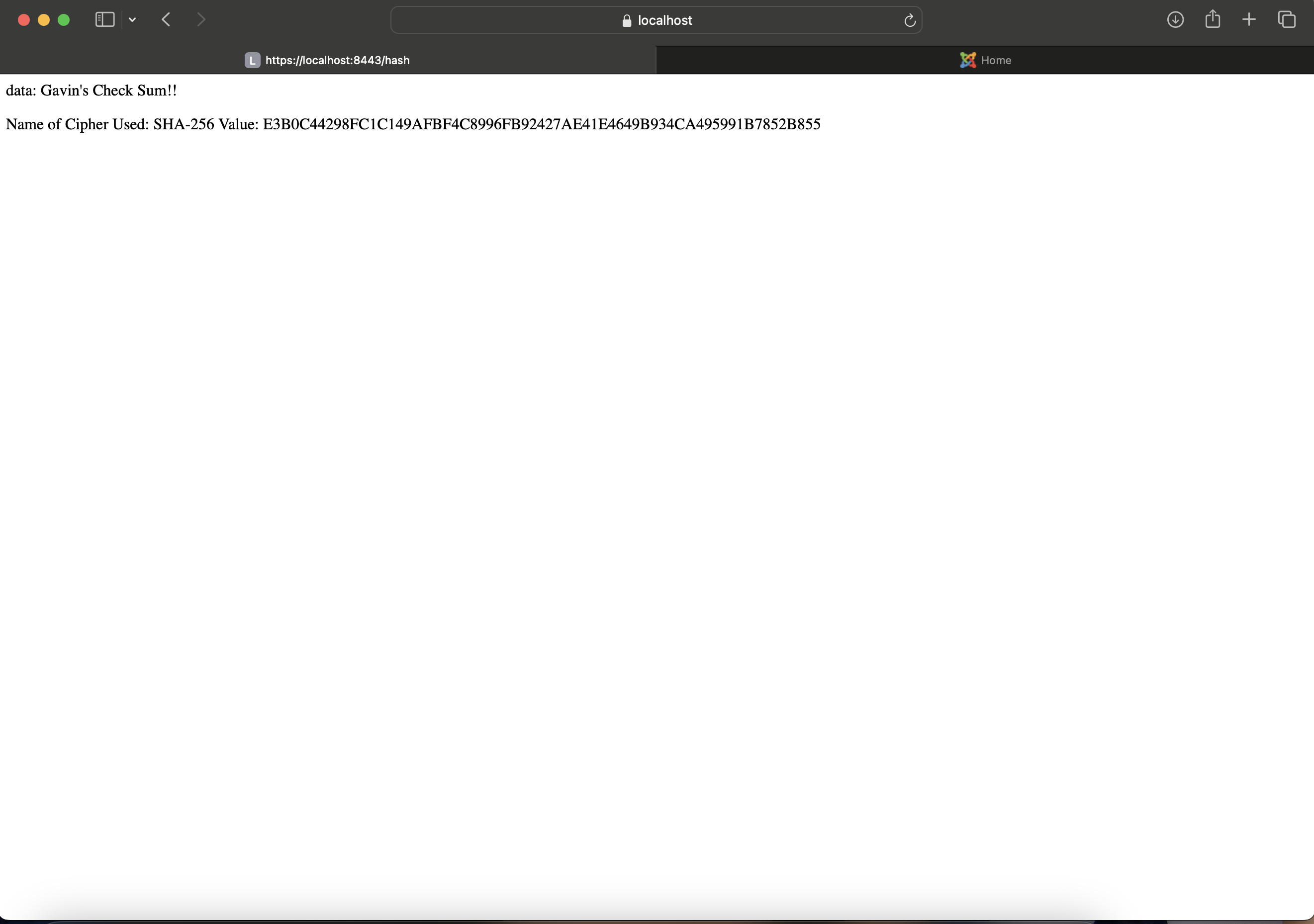
SHA-256 uses symmetrical encryption keys, which is suitable since only Artemis Financial will access the encrypted files. The algorithm also leverages Java’s random number generation to ensure each encrypted file is highly secure. This randomization allows the cipher to create a non-reversible checksum, which verifies the file's authenticity while maintaining security.

## Certificate Generation

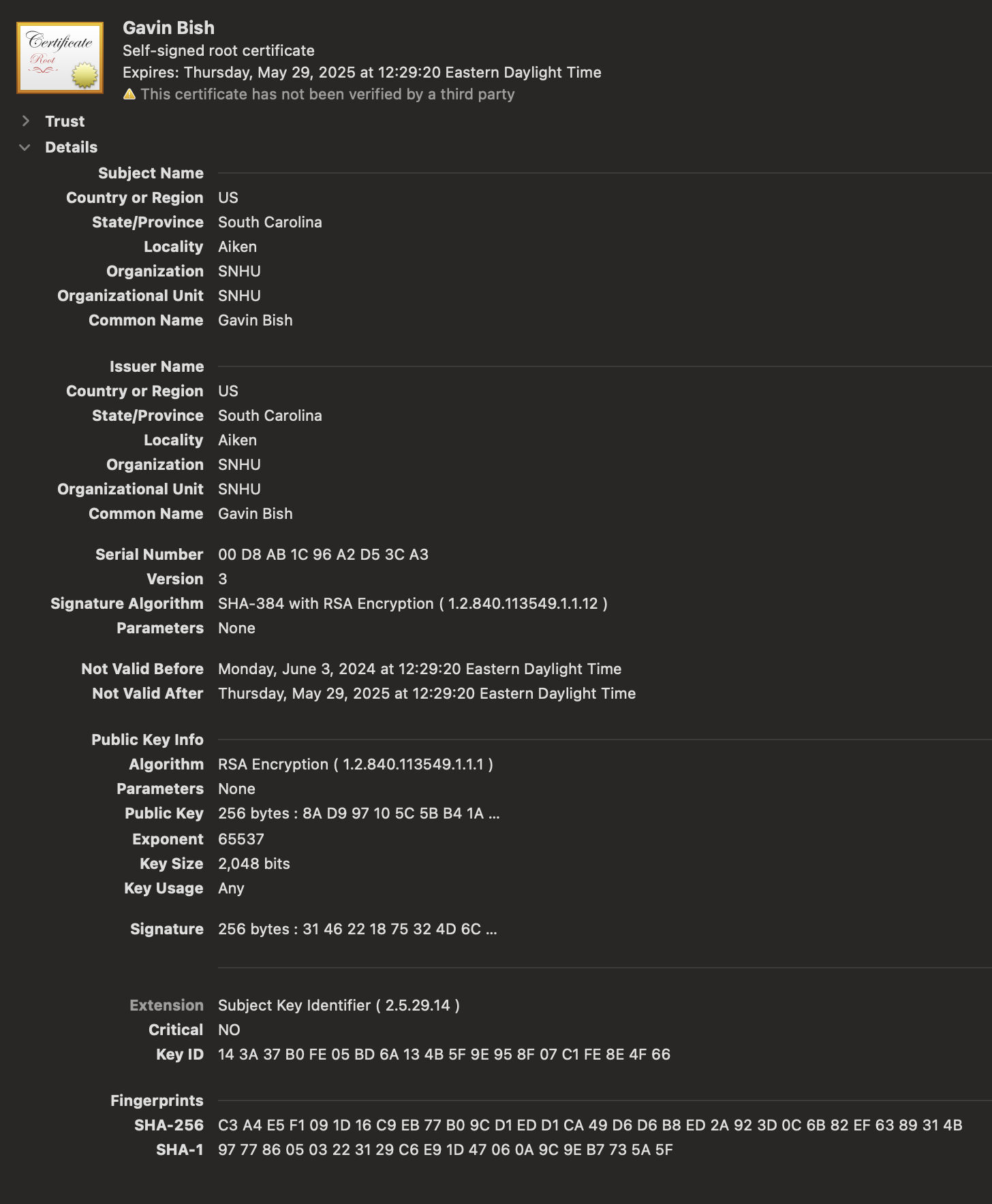
Insert a screenshot below of the CER file.

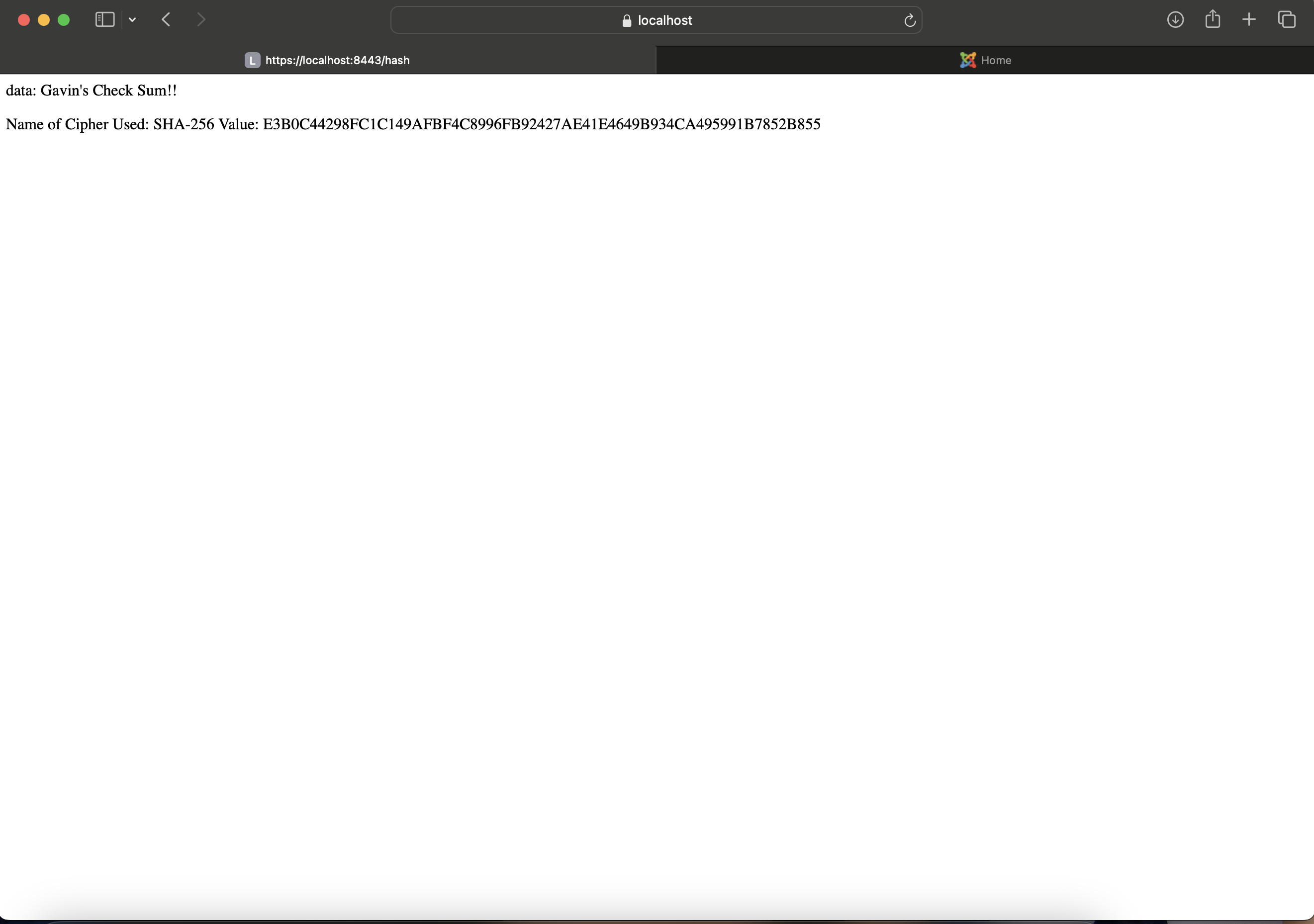


## Deploy Cipher

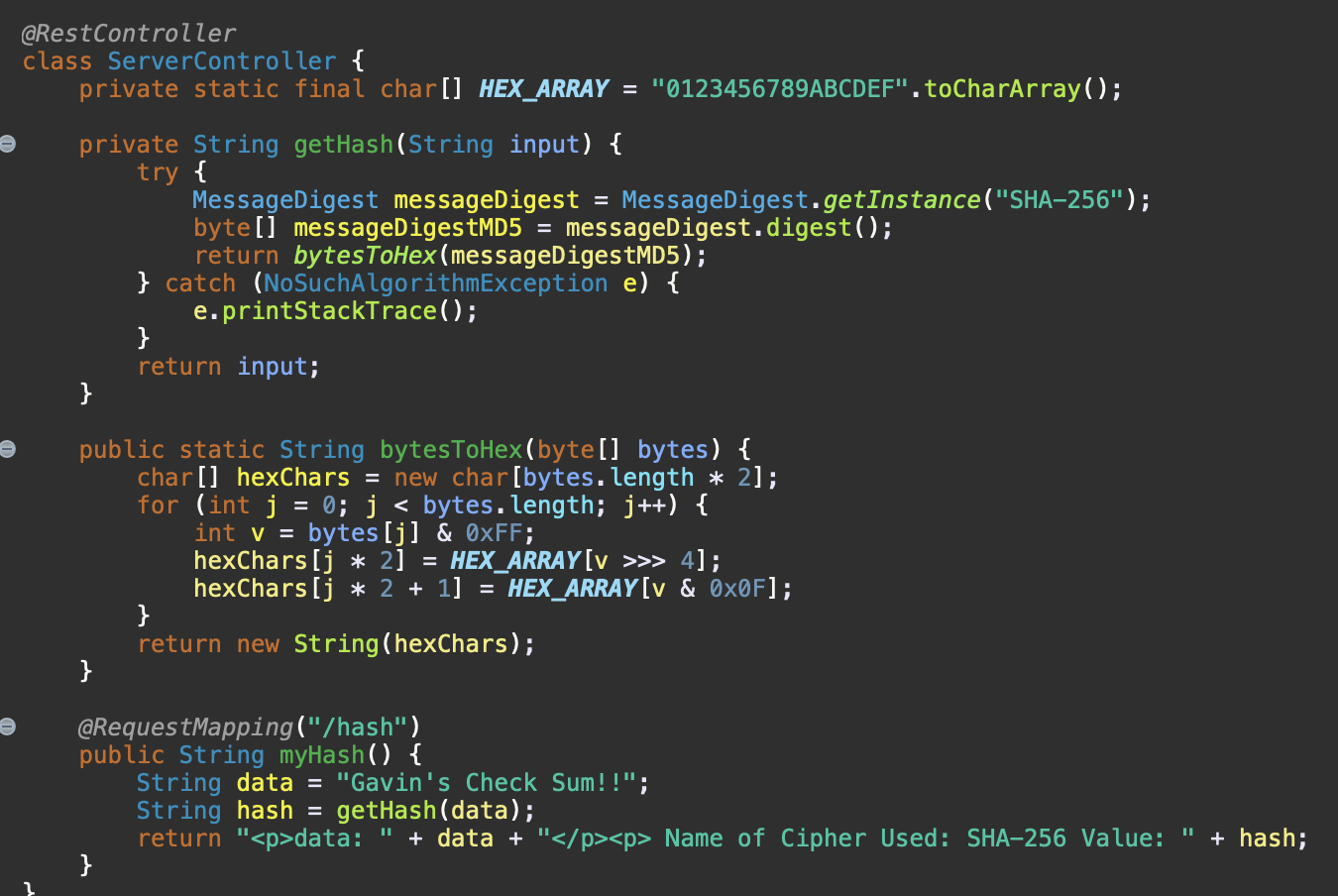
Insert a screenshot below of the checksum verification.

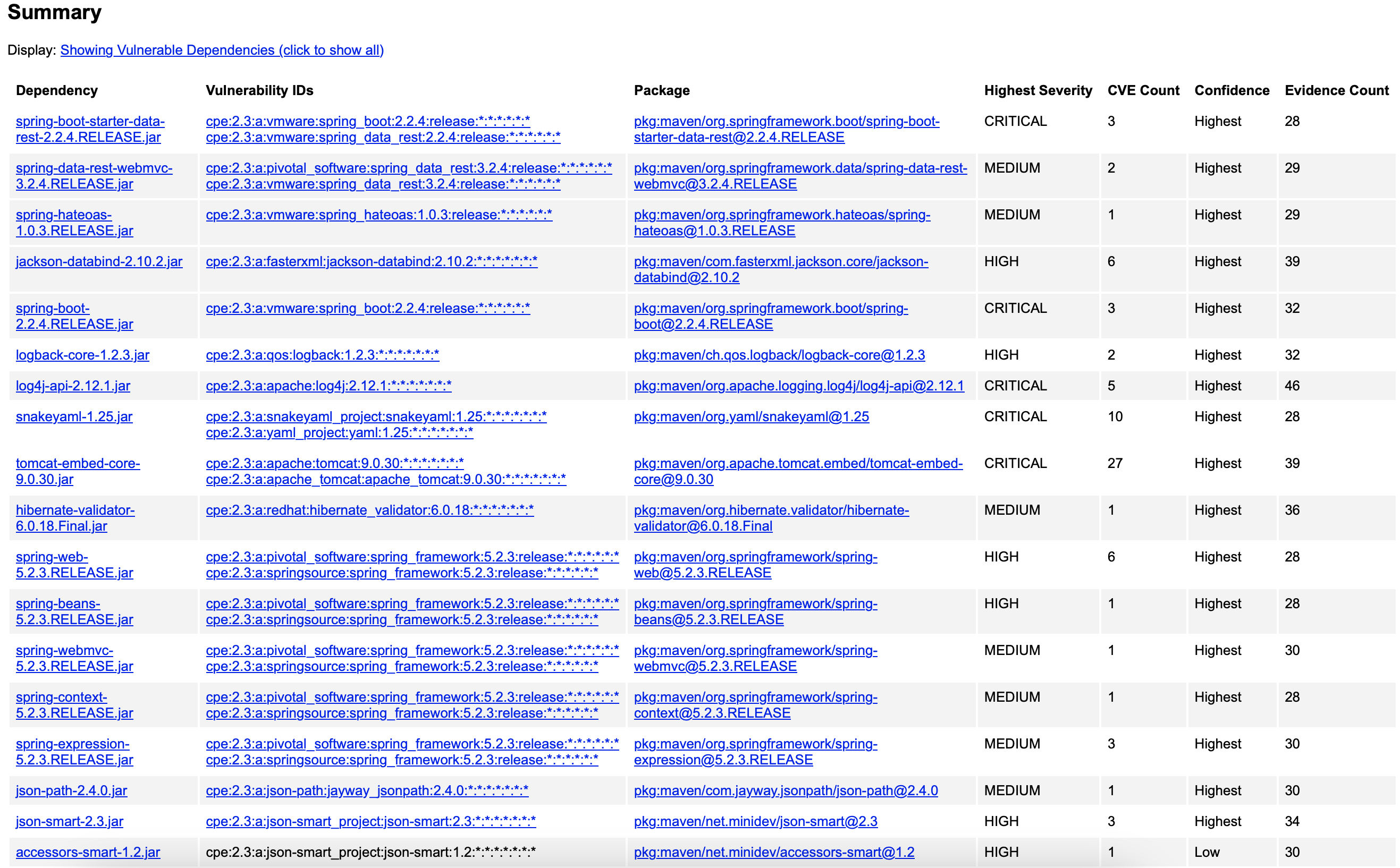
## Secure Communications

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

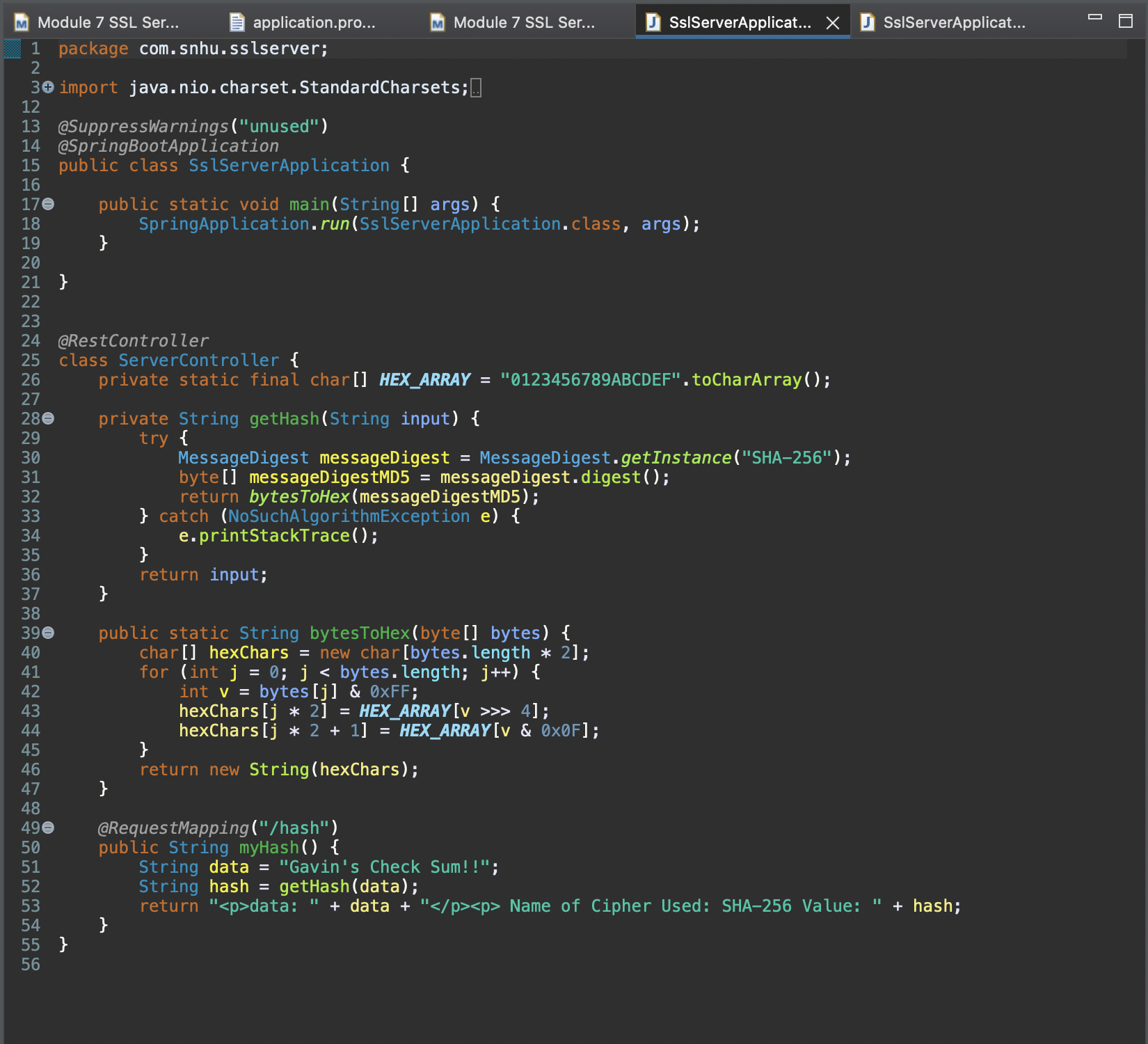


## Secondary Testing

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



## Functional Testing

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

## Summary

In this project, I focused on cryptography, client/server security, and code quality. To enhance code quality, I emphasized secure coding practices and patterns. Specifically, I implemented a secure 256-bit hashing algorithm to protect sensitive data. Although the data used was my name, this method can secure any type of information, such as financial plans, estimates, corporate documents, or client information for Artemis Financial. This data is encrypted before storage to ensure its security.

Client/server security was addressed by integrating a security certificate and enforcing a TLS connection for the web application. TLS protects both the connection and communication between the client and server. Encryption was applied to both the data and the TLS connection to ensure secure communication. Additionally, all code was thoroughly reviewed and inspected for errors or vulnerabilities, whether originating from my code or dependencies.

The first layer of security implemented was an SSL certificate, ensuring the site is secure and indicating to users that their communication is encrypted. This enhances user trust and secures data and communication for Artemis Financial. The second layer was the inclusion of the SHA-256 encryption algorithm. Encrypting the data itself, in addition to using TLS for communication, adds another layer of security. This ensures that even if someone gains access to the server, they cannot decipher the stolen information.

The final layer of security was enforcing HTTPS. This ensures that sessions cannot occur over standard HTTP, requiring an encrypted session for user interactions with the site. This prevents the use of poorly secured connections by outdated browsers, thereby reducing the risk of man-in-the-middle attacks.

## Industry Standard Best Practices

To maintain the security of the software application, best practices include regular dependency checks to identify new vulnerabilities, secure coding for all new modules, input validation for user inputs, building a robust API that clearly defines actions for clients, and conducting regular code reviews to catch errors or unforeseen interactions with existing code.

These practices can be tailored to meet the specific needs of Artemis Financial. For instance, if future code reviews indicate that SHA-256 is no longer secure, it can be upgraded to a more secure standard like RSA-2048 or the latest recommended algorithm. Input validation should focus on allowing only specific types of input, using whitelisting instead of blacklisting to prevent new threats. Regular dependency checks will identify new vulnerabilities reported by organizations like NIST. Any discovered threats can be addressed using recommended fixes or by switching to more secure modules.

Secure coding practices must be consistently applied, and the API should be designed to restrict users to actions within their authorized access levels. By adhering to these best practices, the application will remain secure and functional for Artemis Financial for years to come.