****

# Practices for Secure Software Report

Table of Contents

[Practices for Secure Software Report 1](#_Toc163998326)

[Document Revision History 3](#_Toc163998327)

[Client 3](#_Toc163998328)

[Developer 4](#_Toc163998329)

[1. Algorithm Cipher 4](#_Toc163998330)

[2. Certificate Generation 5](#_Toc163998331)

[3. Deploy Cipher 6](#_Toc163998332)

[4. Secure Communications 7](#_Toc163998333)

[5. Secondary Testing 7](#_Toc163998334)

[6. Functional Testing 8](#_Toc163998335)

[7. Summary 11](#_Toc163998336)

[8. Industry Standard Best Practices 11](#_Toc163998337)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| **1.0** | **04/12/2024** | **Margarita Kiseleva** | **Initial revision and refactoring** |
| **1.1** | **04/12/2024** | **Margarita Kiseleva** | **Certificate generation** |
| **1.2** | **04/13/2024** | **Margarita Kiseleva** | **Secondary revision and dependency check run** |
| **1.3** | **04/14/2024** | **Margarita Kiseleva** | **Algorithm cypher and summary** |
| **1.4** | **04/15/2024** | **Margarita Kiseleva** | **Final revision** |

## Client



## Developer

Margarita Kiseleva

## Algorithm Cipher

***SHA-256***

SHA-256 is a way of hashing information used in the Bitcoin network and many other cryptocurrencies. Its name is an abbreviation for Secure Hashing Algorithm. The creator of this technology is the national security division of the US Department of Defense. The key task of the algorithm is to convert information into a certain value of a specific length, which acts as an identifier.

The SHA-256 hashing algorithm allows you to control the issuance of cryptocurrency, as it is an important part of mining, and ensures maximum network security. In order to mine digital assets, you need to carefully analyze the features of this technology, in particular, to understand why it is needed and what tasks it is capable of solving.

SHA-256 is a hashing algorithm used to convert input information of any size into a fixed-size string. Hence, it ensures that it accepts an input and then produces an output of a well-defined length. This is what a hash is. It does not matter at all whether the user enters a letter, a word, or a whole book because at the output they will get a code of a fixed length.

The algorithm converts the information into a 256-bit code consisting of 64 letters or digits composed randomly. As a result, almost unique combinations are generated, which are extremely difficult to decipher. The reverse conversion is impossible, which makes SHA-256 one of the most secure algorithms.

Determinism is another important feature of SHA that should not be overlooked. The generated code will always be the same if identical input parameters are used. This characteristic makes this hash function ideal for use in the Bitcoin peer-to-peer network.

Hashing is commonly referred to as the process of converting the original information into a digital code. The uniqueness of the generated value makes it easy to identify this data. The encrypted string is created by a special algorithm, which is commonly referred to as a hash function. If we talk about Bitcoin, the valid number in the signature string of the block will always contain a certain number of zeros. It depends on the volume of already mined coins. Therefore, it is impossible to decrypt the information to the original state.

With the SHA-256 algorithm, you can encrypt absolutely everything. For example, it is realistic to turn any literary work into a digital code consisting of 64 symbols. However, the reverse transformation is impossible. You cannot transform a 64-character combination of random numbers and letters into a text to make corrections to it. Everyone can use the hash function by opening the site of automatic encryption.

If we talk about the features of the hash function, we cannot bypass the complex technical data. The protocol under consideration is based on the fragmentation of 256 fractions, hence the corresponding prefix in the name. Each fragment contains 64 bytes. Shuffling of the fractions is carried out by a special cryptographic system. As a result, we get a ready code.

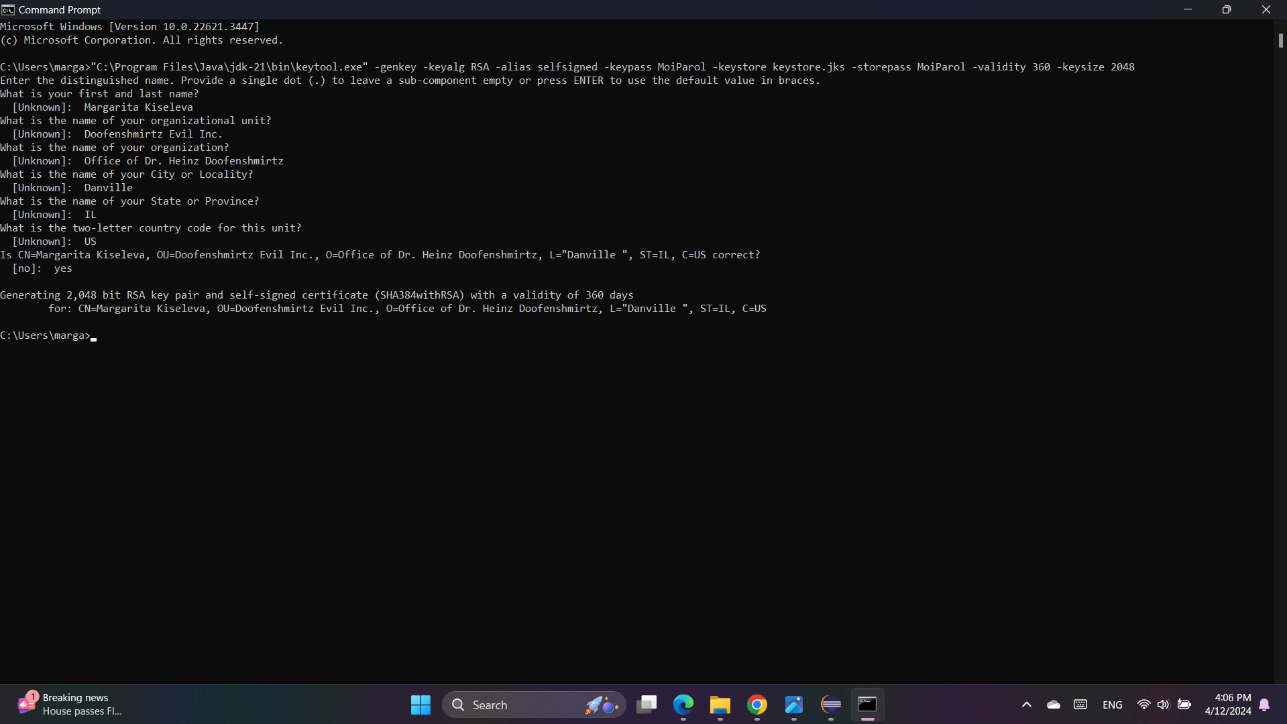
Main parameters:

* size of a block containing information - 64 bytes;
* maximum length of one message - 33 bytes;
* block signature size - 32 bytes;
* 64 shuffles are performed during a round;
* information is transmitted within the network at a speed of 140 MiB/s.

The hash function is based on the Merkle-Damgor principle. The data array is divided into independent blocks. They are divided into 16 parts. After 64 shuffles, the system generates a specific result, which is the basis for processing the next block. It is a continuous process. It is interconnected, because it is impossible to get the code of a particular block without having data about its predecessors.

This algorithm has some superiority over other hashing methods. First of all, it has a high level of popularity. The hash function is used in the Bitcoin network, so absolutely everyone, including manufacturers of ASIC miners, will adjust to this protocol. Secondly, SHA-256 provides a high level of security. Transactions are hard to falsify or forge.

## Certificate Generation



A screenshot of a computer screen

Description automatically generated

A computer screen with a message box

Description automatically generated

## Deploy Cipher

A screenshot of a computer

Description automatically generated

## Secure Communications

A computer screen with a message box

Description automatically generated

A screenshot of a computer

Description automatically generated

## Secondary Testing

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Functional Testing

A computer screen shot of a program

Description automatically generated

A screen shot of a computer

Description automatically generated

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Summary

To summarize the above outlined work conducted on the project, I will begin with the hash cipher. The cipher I chose for this project is SHA-256. This cipher mechanism has been continuously proving itself to be secure and relatively simple to implement. Additionally, because of the way SHA-256 works, it has a very low chance of collisions, which is important to our client.

When refactoring the code base, I noticed a couple things that needed to be changed. I decided to switch the access modifier of the methos called *checksumCalculator* from public to private to make sure the code corresponds to the best secure coding practices. I also added a secure *RestController*, which now acts as a secure controller of the program’s hash RESTful stop. Additionally, I made sure that *ServerController* class performs the function of addressing the issues identified by the vulnerability assessment diagram.

In order to protect Artemis Financial and the privacy of their clients, my recommendation is to run Maven dependency check every two weeks. This will allow our program to respond to and manage potential security threats accordingly and in a timely manner. In addition, I suggest updating all the plugins and used libraries regularly. This will help us manage all the vulnerabilities connected to the outdated software.

## Industry Standard Best Practices

Secure coding is a set of practices that ensure the security of software applications. Secure coding involves identifying potential security vulnerabilities and addressing them during the application development process. It also includes implementing security controls to protect against malicious attacks.

Best practices for secure coding:

*Shift-left Security.* The concept behind Shift-left Security is to prioritize security throughout the software development life cycle. This helps detect and address flaws early on, minimizing the likelihood of security issues being ignored due to tight task deadlines, or crunches.

*Embedding security* as an integral part of the development process allows teams to assess the risks associated with dependencies and open-source components before incorporating them into the project. It also helps ensure version consistency throughout the project.

Teams can use information from sources such as the *OWASP Top 10* list to get professional guidance on vulnerability management and use tools to speed up and make the process more efficient. The following are common tools for implementing Shift-left Security:

*Static Application Security Analysis (SAST)*. Programmers can use SAST tools to scan source code for known security vulnerabilities. SAST tools scan code during the development phase, usually before the code is compiled. The goal is to identify problems and help teams fix them during the code writing phase.

*Dynamic Application Security Analysis (DAST)*. Unlike SAST tools, which scan code at rest, DAST tools analyze source code at runtime or after compilation. Teams can use DAST tools to detect security issues missed by static scans. They can help identify security vulnerabilities such as runtime bugs and SQL injection (SQLi).

*Software composition analysis (SCA)* are automated tools that scan source code to identify open-source components and third-party libraries. SCA tools can detect known vulnerabilities and notify teams of available updates or patches.

*Reduced patching costs*. Finding and fixing vulnerabilities early in development reduces the time and resources required to fix problems in the future. It also reduces the likelihood that critical vulnerabilities will be discovered after the product is deployed.

*Higher product quality*. Integrating security into the development process increases the quality of the software, making it more robust and resistant to attack.

*Reduced development time*. Applying Shift-left Security can speed up the development process because it takes less time to fix bugs and vulnerabilities in the final product.

*Compliance with standards and regulatory requirements*. Shift-left Security enables developers to better comply with security standards and regulatory requirements such as GDPR, HIPAA, and PCI DSS.

Secure coding is important because it helps prevent cyber-attacks and data breaches. Cyberattacks can cause significant damage to an organization, including financial loss, reputational damage, and legal liabilities. Secure coding can help prevent these types of attacks by creating software that is resistant to vulnerabilities and malicious attacks.