Southern New Hampshire University

4-2 Written Assignment: Algorithm Ciphers

David Vega

Dr. Vivian Lyon

CS-305 – Software Security

01/29/2023

**Algorithm Cipher**

There is more to security than implementing ciphers. Security should begin at the development phase. Developers should be collaborating with IT security to design code with security as the basis. This will only strengthen the integrity of your code as it progresses through various environments for testing before eventually deploying to production.

To illustrate the importance of security, let’s discuss the remote work paradigm that the world was forced into during the COVID-19 pandemic. There were not many organizations that were prepared to deal with the shift from secure environments to unsecured environments. Remote access to systems and applications were available for some employees. However, company issued devices (i.e. – phone, laptop) were reserved only for employees whose roles required them. Employees without a company issued device were reliant on personal devices to access these systems. This, of course, caused an increase in cybersecurity attacks as the shift from an on-site workforce to a remote workforce allowed attackers to exploit personal devices to extract otherwise sensitive company data (Soni et al., 2020).

Additionally, the pandemic essentially gave birth to a new way for consumers to shop without having a significant impact on the customer experience. This led companies to develop many applications quickly to mitigate the impact to production and revenue stream. This was evident in the fast-food industry where food delivery apps were being developed and deployed to accommodate social distancing protocols with little consideration for security (Soni et al., 2020).

The consequential price for flexibility and the bottom line is usually personal identifiable information. These applications with a customer-centric approach may have birthed new exploits that have yet to be discovered. Now that the threat of the pandemic has somewhat subsided, the focus should be on securing these applications and refactoring as needed. Nothing can be more impactful to the bottom line and to customer loyalty then a data breach.

Some essential defense mechanisms to employ in the Artemis Financial application consist of input validation via whitelisting, and a strong cryptographic protocol. Whitelisting is an approach to validating user input by defining the valid characteristics of the input field while all other input scenarios are rejected. This contrasts with blacklisting which defines the invalid characteristics of the input field. Maintaining a blacklist can be difficult as not all potential threats can be accounted for. The standard cryptographic protocol became TLS (“Transport Layer Security”) back in 1999 (Manico et al, 2014). The purpose of TLS is to mitigate the potential for a hacker to eavesdrop on a data transmission sent over the internet by encrypting the data. The secure connection is only established when the client trusts the certificate sent from the server. However, this protocol only protects the data while in transmission. It does not protect the data that is stored (Internet Society, 2023). Unfortunately, there is no standard for securing data stored at rest. Developers must implement their own mechanisms for encrypting stored data. Fortunately, there are some open-source toolkits such as Keyczar that eliminates the guess work out of key management. Keyczar was originally developed by Google’s Security Team and uses a key versioning system making it easier for developers to rotate and retire keys (Manico et al, 2014).

Based on the type of data that Artemis Financial is storing, my recommendation for encrypting archival data would be to use AES-256 encryption (Advanced Encryption Standard) for this project. I would also employ the open-source tool Keyczar to assist in key management.

The benefit to implementing the AES-256 algorithm cipher is that it is virtually impossible to break, thus mitigating most risk. The only way an attacker can decrypt the data is to have the private key. AES-256 is the strongest of the AES distinctions which also have 128-bit and 192-bit versions. Aside from the obvious length, the difference between the three versions is the number of rounds of encryption. The rounds are basically a set of processes that the plaintext undergoes before it is mutated into its final cipher form. With 128-bit it goes through ten (10) rounds, while 192-bit goes through twelve (12) rounds of encryption. Finally, 256-bit goes through fourteen (14) rounds of encryption making it the most secure of the variations of AES, as well as its predecessor the Data Encryption Standard (DES) which only used 56-bit encryption (Panda, 2016). One of the drawbacks to implementing this algorithm is that the more complex the encryption the more resources are allocated to it having an impact on performance. This could be a game changer if you value speed over security in your application. There are some other potential risks associated with this algorithm cipher that come from side-channel attacks. Hackers can leverage these side-channel attacks by listening during transmission for any data leaks that can help them to identify differences by monitoring execution of time or power consumption (N-able Solutions, 2019).

As for current government regulations, there is very little restrictions on the export of encryption. Telecommunication service providers must ensure that any equipment provided, or services rendered have capabilities to intercept communications and deliver those intercepted communications to the government provided the appropriate lawful authorizations (Global Partners Digital, N.D.).

**Justification**

The AES-256 algorithm would be best suited to fulfill the needs of Artemis Financial. This is because when it comes to internal sensitive information such as security configurations as well as customer sensitive information such as banking account numbers, social security numbers, and other personal identifiable information AES-256 offers the most protection.

Hash functions are an essential part of encryption algorithms because this is what transforms the data. The SHA-256 hash algorithm is the most secure hashing algorithm to date and was developed by the National Security Agency (NSA) of the United States and published in 2001. The purpose of the SHA-256 hash algorithm is to perform a series of mathematical computations that takes in the data and scrambles it to an irreversible fixed length (Prasanna et al, 2021).

One of the key components to “randomness” is to generate a random number of which most, if not all, systems can do. This can be accomplished by using a seed to initialize a random number generator (RNG). Although this will indeed generate a series of random numbers, it will almost definitely repeat the sequence when the application is run again. For example, seeding the RNG with some arbitrary integer might result in a sequence of seemingly random numbers but when the application is run again the same sequence of numbers would be output. There is no randomness in replication. To create true randomness, the RNG would need to be reseeded frequently as it runs (Manico et al, 2014).

AES-256 is a symmetric algorithm, which is when the same key is used for encrypting and decrypting data. Alternatively, asymmetric algorithms use a two key system. The private key is kept secret and kept only by the sender to use to sign the data. A public key is generated to be distributed to anyone for validating the data.

Encryption methods have certainly made tremendous strides since the early 20th century especially after World War I. Electromechanical ciphers were the prominent means of encryption and were marketed predominantly to the U.S. Army and Navy. It wasn’t until the computer era where encryption was taken into the world of mathematics and statistics. This shift allowed computer scientists to create new algorithms thus resulting in stronger encryption algorithms such as AES (Dooley, 2013).

# Works Cited

Dooley, J. F. (2013). *A brief history of cryptology and cryptographic algorithms*.

Springer International Publishing AG.

Internet Society. (2023, 01 29). *TLS Basics*. Retrieved from Internet Society:

https://www.internetsociety.org/deploy360/tls/basics/

M. Panda, "Performance analysis of encryption algorithms for security," 2016

International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, India, 2016, pp. 278-284, doi: 10.1109/SCOPES.2016.7955835.

Manico, J., & Detlefsen, A. (2014). Iron-Clad Java: Building Secure Web Applications.

N-able Solutions. (2019, July 29). *Understanding AES 256 Encryption.* Retrieved 01

2023, from N-ABLE: https://www.n-able.com/blog/aes-256-encryption-algorithm

S. R. Prasanna and B. S. Premananda, "Performance Analysis of MD5 and SHA-256

Algorithms to Maintain Data Integrity," *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, Bangalore, India, 2021, pp. 246-250, doi: 10.1109/RTEICT52294.2021.9573660.

V. Soni, D. Kukreja and D. K. Sharma, "Security vs. Flexibility: Striking a Balance in the

Pandemic Era," *2020 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, New Delhi, India, 2020, pp. 1-5, doi: 10.1109/ANTS50601.2020.9342779.

*World map of encryption laws and policies*. Global Partners Digital. (n.d.). Retrieved July

21, 2022, from https://www.gp-digital.org/world-map-of-encryption/