Module 4: Algorithm Cyphers

Software Security

Version 1.4

Ronny Z. Valtonen

Software Security

As already known, protecting data in transit as well as at rest are equally important. Encryption ciphers provide a way to protect data as well as a way to reverse the protection when it is authorized to be accessed. For example, when you need to purchase something with a credit card, you'd want the credit card number to be protected until the process of making the purchase needs to be done, then encrypted again. The best security practices that need to be taken into account when defending against security attacks are to have strong **key management**, implementing strong **encryption algorithms**, **forward secrecy** and proper **access controls**. Strong key management involves having a secure way of storing encryption keys and regularly rotating these keys as a way to restrict access. As discussed in O'Reilly Iron-Clad Java, stealing these keys is easier than one might think. Which is why it's also important to implement forward secrecy. This means that, even if an attacker got ahold of private keys, they would not be able to decrypt the communication between a server and client. Using a strong encryption algorithm like AES-192 (discussed further below) provides proper encryption for data in transit (Manico & Detlefsen, 2014). Without having good key management, there is no point in having the encryption in the first place if it's easy to gain access to the key to decrypt the data. This is why it is important for each of these security practices to be strong as each of them work on top of each other. Implementing the principle of least privilege would limit who can and has access to the key management systems. With regular audits to permissions and roles, these practices together will ensure a secure system of communication for data in transit that Artemis Financial needs.

To protect Artemis Financial, the recommended encryption cipher is AES with 192 bit. This is recommended because it is a widely accepted standard which is important because you cannot always control all endpoints and any two endpoints need to have compatible encryption or you will end up with a handshake_failure exception. AES is strong against brute-force attacks due to the high bit key size of 192, proving strong security. The **risks** involved for this recommendation is that without strong key management as discussed previously, all data access will be lost if secret keys are lost (Manico & Detlefesn, 2014). Secondly, having higher bit key sizes means higher resource demand. The AES-192 - while strong - can be more resource intensive compared to a key bit size of 128 (although this is nearly impossible to brute force, too), though most modern endpoints can handle this without issue. Not using a key size of 256 will ensure every endpoint can handle the encryption and decryption without any worries. With my experience of working at Intel as a software engineer, we must follow the Federal Information Processing Standards for all data transit as well as General Data Protection Regulation for any data between the USA and EU. Using this knowledge, the AES-192 cipher complies with both of these regulations according to NIST and GDPR Advisor (NIST, 2025). Implementing the AES-192 cipher for artemis financial would be used to encrypt long-term files, key generation and to meet the FIPS and GDPR regulations (Manico & Detlefsen, 2014). It will also be used to mitigate risk from data breaches as then important data would be encrypted if leaked.

There really is no answer to “What is the best cipher?” The reason being is that it all depends on the application it’s being implemented in. Most modern day ciphers are the same “strength” as in, you’d need a super-computer and time longer than the age of the universe to crack even a key size of 128. AES is the most accepted cipher in the world and most likely to remain strong with intuitive implementation, which is why it’s recommended for most applications. The other issue is that you need to know if you want to implement a symmetric or asymmetric encryption for bulk data storage or key exchanges. You wouldn't want to use AES for digital signatures, and instead implement RSA. The main reason as to why you wouldn't want to pick the "most secure cipher" is due to the hardware demand of "stronger" ciphers. It might not be practical or efficient for certain tasks where performance is critical. You must seek a balance between security and performance, further strengthening the reason as to why there is no one single "best cipher."

Justification

The purpose of a cipher's **hash functions** is for integrity verification; this ensures that data has not been modified during transit or corrupted during storage. This provides a level of security that prevents man-in-the-middle attack. An attacker cannot modify a file before it arrives as the decrypt would fail the handshake. Secondly, it allows for password storage with a hash instead of storing passwords in plaintext - an obvious security vulnerability. A cipher's **bit level** is what defines the security strength, referring to the length of a cryptographic key for encryption. A higher bit level provides more resistance to brute-force attempts. The AES-128 being the lowest accepted bit level, has never been brute-forced and is said to take longer than the length of the universe to brute-force with a super computer (Oracle, n.d.). This also means that a higher bit level is slower for encryption and decryption due to each additional bit level doubling the complexity. The use of random numbers in encryption ensures the keys are non-predictable which prevent attackers from pattern based attacks or simply guessing the encryption key. A symmetric key encryption such as AES uses the same key to encrypt and decrypt data. This makes it fast for bulk storage like database files, however it also means that the key needs to be shared to multiple people. This is why key management needs to be secure and the principle of least access is 'key.' An asymmetric key encryption uses a private key and public key to encrypt and decrypt data. This encryption type is great for digital signatures and authentication but is slower due to it being more hardware demanding than symmetric key encryption. Implementing asymmetric encryption to transmit symmetric keys is a typical use case.

In history, the most well known and oldest cipher is the Caesar Cipher which is a simple shift cipher and the most simple encryption techniques. A left shift of 4 means that the letter "D" would become "Z." This is now a popular coding homework for a lot of professors (Smith, James). Another encryption method that is widely known is the Enigma machine which was used in World War 2 in all branches of the German military (EnigmaHistory, n.d.). This was, during its time, the most secure encryption device to encipher top-secret messages. As hardware has improved, the need for stronger ciphers has become important. The emergence of DES in 1977 provided a great improvement for encryption. However, due to a short key length of only 56, it can be hacked in 22 hours and 15 minutes (Wikipedia contributors, 2025). Today we have RSA and AES which are the strongest ciphers to exist and with current supercomputers are impossible to brute-force. With hardware continuously improving, there are even stronger algorithms being produced like post-quantum encryption. The NIST has released 3 post-quantum encryption standards; the purpose of which quantum computers could break current encryption standards easily. Some experts predict that a device with the capability to break AES-256 could appear within a decade (NIST, 2025).

References

"Java Security Standard Algorithm Names," n.d., https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names.

Jim Manico and August Detlefsen, *Iron-Clad Java: Building Secure Web Applications (Oracle Press)*, 2014, http://dl.acm.org/citation.cfm?id=2826076.

Auth, "Signing Algorithms," Auth0 Docs, n.d., https://auth0.com/docs/get-started/applications/signing-algorithms#:~:text=HS256%20(HMAC%20with%20SHA-256,shared%20between%20the%20two%20parties.

James Smith Project Structure, "Writing Secret Messages With a Caesar Cipher | Golang Project Structure," *Golang Project Structure* (blog), June 27, 2022, https://golangprojectstructure.com/caesar-cipher-secret-messages/.

"Enigma History," n.d., https://www.cryptomuseum.com/crypto/enigma/hist.htm.

Wikipedia contributors, "Data Encryption Standard," Wikipedia, March 28, 2025, https://en.wikipedia.org/wiki/Data_Encryption_Standard#:~:text=The%20EFF%27s%20DES%20cracker,DES%20key%20in%2056%20hours.&text=Together%2C%20Deep%20Crack%20and%20distributed,22%20hours%20and%2015%20minutes.

Chad Boutin, "NIST Releases First 3 Finalized Post-Quantum Encryption Standards," NIST, February 4, 2025, https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards.