Cipher for Long-Term Archive Files for Artemis Financial

*Recommended Cipher:*

* Use AES-256-GCM in Java as 'AES/GCM/No Padding' for encrypting archive files. This mode provides both confidentiality and integrity. Oracle’s standard algorithm list documents this cipher, and Iron-Clad Java recommends using modern, vetted algorithms instead of outdated ones.

*Why AES-256-GCM:*

* Provides confidentiality and authentication in a single step. - Widely supported and hardware-accelerated. - 256-bit keys ensure strong, long-term security. - Adheres to best practices in Iron-Clad Java.

*Best Practices:*

* Generate a new 256-bit key for each file (Data Encryption Key, DEK).   
  Use Secure Random to generate a unique 96-bit IV each time.   
  Store the ciphertext with the IV and a 128-bit tag.   
  Protect keys in a keystore or HSM.   
  Use SHA-256 for digests; avoid MD5 and SHA-1.

*Risks and Mitigation:*

* Reusing an IV in GCM is dangerous; always use a Secure Random Number Generator.  
  Key compromise: store KEKs in secure modules and rotate them regularly.  
  Implementation mistakes: follow Oracle’s standard names and parameters.

*Justification:*

* AES-256 offers strong resistance against brute-force attacks.  
  SHA-256 provides collision resistance for checksums.  
  Symmetric keys (such as AES) are ideal for encrypting large files.  
  Asymmetric keys (such as RSA or EC) are primarily used for sharing keys or creating digital signatures.  
  Iron-Clad emphasizes the shift from weak ciphers to AES + SHA-2 as the current standard.

*References (APA):*

Oracle. (n.d.). Java Cryptography Architecture (JCA) Standard Algorithm Name Documentation. https://docs.oracle.com/javase/9/docs/specs/security/standard-names.html#cipher-algorithm-names  
Viega, J., & Kohno, T. (2014). Iron-Clad Java: Building Secure Web Applications (Chs. 6 & 8). McGraw-Hill.