When it comes to protecting archived data, utilizing encryption to ensure their security is the standard for almost all industries with sensitive data. Encrypting data-in-transit isn’t enough, developers must also consider utilizing data-at-rest algorithms to provide the best chance at protecting sensitive data. For Artemis Financial, the concern is long term archival of files. The amount of files might vary, and the size of the files could be very large, so we need to utilize an algorithm that can efficiently handle large streams of data.

To this end, I would recommend using AES as the encryption standard for encrypting the files that we are archiving, or at least breaking them up into sections, and encrypting those sections utilizing AES. AES utilizes private key encryption, so only one key is needed for encrypting and decrypting that data. The concern of protecting that key comes into play, so I would then utilize AESWrap to encrypt the AES private key, and store it both digitally and on paper in an approved safe.

Using AES is computationally slow, and depending on how large the datasets we need to protect are, it could take a lot of time to complete the encryption. This is partly caused by its function as block cipher, in which blocks of data are encrypted together at a time, compared to stream ciphers which might act on a per-bit or per-byte basis. This also means that the servers encrypting the data will be vulnerable to attack, and should be protected until the process has been completed, as bad actors could observe the encryption process and determine what the characteristics are of the key or algorithm being used.

Although RSA can offer some advantages with asymmetric encryption, utilizing it at bit lengths of up to 4096 could be incredibly consequential with large file sizes. When it comes to the best cipher, there isn’t a one size fits all solution for all cases. If this was hosting a web server, utilizing public key encryption would be a more attractive option. In addition, if we were utilizing devices that weren’t computationally very powerful, we might want to back down our encryption to an even lower standard so that we have enough performance to encrypt the data on a reasonable timeline.

AES comes in three different key sizes: 128, 192, or 256 bits. These bit levels represent the size of the encryption key in the algorithm for each of the 128 bit chunks that AES works on. These bit sizes also represent the amount of “rounds” of encryption that are used per chunk, with more rounds used the higher the key size. Each round consists of linear algebra matrix operations that manipulate the data utilizing the key that’s generated. This process continues like a factory, grabbing the next 128 bit block, manipulating it, and storing that manipulation as the encrypted data.

As stated before, symmetric keys are known as “private key encryption” in that the key used for both encryption and decryption are the same, and thus it needs to be always protected. Non-symmetric key encryption involves the use of a public key and a private key to encrypt data. A public key is utilized to encrypt data, and then a corresponding private key is used for decrypting that data. To make these processes even more secure, random numbers are generated during the encryption process to help randomize the output of the algorithm. Since encryption uses a two-way algorithm, it’s important that the data is as algorithmically different when encrypted as possible from its input.

Cryptography as a field has been around for a long time – the most notable uses of which were during World War 2, in which Alan Turing essentially founded Computer Science by developing a machine to decipher German encrypted transmissions. Over time, the field changed drastically, as encryption algorithms developed, and subsequently were cracked. In 2024, AES has proved to be a foundational standard for encrypting data-in-transit and data-at-rest.