The algorithm that we will be using for Artemis Financial is TLS\_AES\_256\_GCM\_SHA384 in TLS 1.3 managed with Keyczar. While Manico et al. (2014) recommend TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA, TLS has come significantly further in security over the years. Forward secrecy is mandatory in TLS 1.3 and so ECDHE is no longer in the naming scheme as it is not negotiable (Naziridis, 2021), and this cipher suite is the most popular of the four on offer in TLS 1.3 (Helme, 2022). Forward secrecy is what keeps our data safe in the event of a private key breach, and financial information must be permanently secure, as humans cannot handle randomly changing bank account information, leaving static personally identifiable information (PII) laying in financial transactions. This is especially true given that Artemis Financial has a fiduciary duty to protect its clients’ information, which legally compels them. So forward secrecy is doing a lot of heavy lifting for us to keep customer data safe.

Breaking down the algorithm into its parts may help explain what each part does and what kinds of attacks they defend against. As mentioned earlier, Elliptic-curve Diffe-Hellman (ECDHE) is an asymmetric key exchange protocol that grants us forward secrecy and defends us against private key leaks, where “asymmetric” here means that we are using a private key to sign the data and a public key to verify the signature of the data (Szpisjak, 2017; Manico et al., 2014, Chapter 6). Advanced Encryption Standard (AES) is a data encrypting method and makes our data otherwise unreadable if read in plaintext, and the number 256 designates the security level of a 256-bit key, where larger values reduce the number of possible collisions and preimage problems as hashing is not one-to-one, only unto. Galois Counter Mode (GCM) maps the authentication tag to an isomorphic finite extension field and back (Gallian, 2016, ch. 21 Algebraic Extensions). This is used as a way for the people involved to “talk” with their encrypted information by offering a form of proof of who they are. Finally, Secure Hashing Algorithm (SHA) 384 is a second-generation secure hashing function that we use for data integrity, essentially it exists to answer the question “Was this data altered?”, and the 384 simply means the number of bits used, where larger values are more difficult to guess (Manico et al., 2014, pp. 156-159).

While this all must sound incredibly secure, it is not. These are all public algorithms that actors are already trying to defeat in any way possible. Much of this encryption rides on randomness generation being secure, and pseudorandom functions are still predictable in certain circumstances as we often have to rely on hardware implementations that are out of date. The first largest vulnerability is to assume that you have none, as that is how someone walks into your office building, puts a corrupted USB into an envelope and writes someone’s name on it and leaves it in an unsecured area of the building, tricking that employee into plugging it in and generating a negative security event. While there are uncountably many different attacks, best practices involve using access control, input validation, error handling, the principle of least privilege, using secure API’s, employing cryptographic functions for all data, high code quality, and encapsulation. The most important form of defense outside of the program, however, is security training.

Now, it will always strike someone to ask if this is the best method of encryption for the business. The best cipher is one that is fast while not being easily reversed or having its keys publicly available that your communication partner has access to. This frustratingly means that there is never a singular answer to “What is the best cipher” without a list of requirements and uses generated by a professional. Sadly, speed is often the largest requirement, as an insistence on all software being as fast as possible even when time is not of the essence has led to users only ever accepting instantaneous outputs that never trigger a false positive security event, meaning that we must trade significant amounts of security for user experience (MIT OpenCourseWare, 2017, 19:42).

To contrast the previous statement, a One-Time Pad (OTP) is the most secure cipher I know how to use. It is also by far and away the most useless cipher I know unless you’re a government organization communicating over a numbers station to a spy network, and it’s incredibly old. It requires true randomness for an input, a key the size of the message (that is to say, it is symmetric), any key must only be used exactly once and destroyed afterward, and that the recipient of the encrypted message already has the key. This makes it slow, impractical, clunky, impossible to scale, but also impossible to crack. You probably also want a Gerät 32620 or equivalent secure transmitter for fake human speech, as well as a shortwave radio frequency you can broadcast globally (Crypto Museum, 2017). There is a good reason why only strange people who are into numbers stations like myself know what half of these things are, or even that they exist. Nothing about an OTP is quick or efficient, you use this cipher when breaking the message would lead to Swan Lake being played on television, or other such messages that endanger millions of lives if broken.

The proposed cipher is both fast and reasonably secure. It is an attempt to get the best of both worlds and offers excellent protection against modern day digital attackers. It will secure Artemis Financials’ data so long as their staff receive security training if they have not already.

References

Crypto Museum. (2017, July 19). *Device 32620 - Speech/Morse Generator*. <https://www.cryptomuseum.com/spy/owvl/32620/index.htm>

Gallian, J. A. (2016). Contemporary abstract algebra (9th ed.). Cengage Learning.

Helme, S. (2022, June 30). *Top 1 million analysis - June 2022*. Scott Helme. <https://scotthelme.co.uk/top-1-million-analysis-june-2022/>

Manico, J., Detlefsen, A., & Kenan, K. (2014). *Iron-Clad Java: Building Secure Web Applications* (M. Smith, Ed.). McGraw-Hill Education.

MIT OpenCourseWare (2017, March 30). *3. Buffer Overflow Exploits and Defenses* [Video]. YouTube. <https://www.youtube.com/watch?v=xSQxaie_h1o&t=3306s>

Naziridis, N. (2021, February 25). Guide to TLS standards compliance. SSL.com. <https://www.ssl.com/guide/tls-standards-compliance/>

Szpisjak, D. (2017, August 25). I am assuming you are talking about these in context of TLS, particularly TLS ciphers. There seems to be some [Comment on the post “TLS-RSA vs TLS-ECDHE-RSA vs static DH”]. *StackExchange*. <https://security.stackexchange.com/questions/166063/tls-rsa-vs-tls-ecdhe-rsa-vs-static-dh>