COSC2804 ASSIGNMENT 3

TEAM 20

PHI PHI PHAM

S3969516

RENATO MIGUEL ALVAREZ

S3969740

JACOB EISHO

S3965880

Phase 1-1: Four CCA Secure techniques

Hash-then-encrypt: E(K, (M || H(M))

The composition follows that the message and the hash of the message is concatenated. This concatenation is then encrypted with K1.
Out of these four techniques, Hash is the weakest in terms of security. Black (2004) outlines that it is clear the bare minimum for an authenticated encryption is to use a ‘keyed’ Hash (MAC) with a K1 and a secure encryption that uses a different key, K2. With this in mind, Hash then encrypt is not a choice to be implemented in any secure security system.
An advantage to a Hash-then-encrypt method, Hash-then-encrypt uses a lesser computing power because the nature of the first step only uses secret key cryptography (Hash) without going through an algorithm to turn it into a tag (MAC) (Blumenthal, 2007).

MAC-then-encrypt: E(K2, (M || T=MAC(K1, M))

The composition computes the MAC of the plaintext using K1 and then concatenates that to the plaintext. This concatenation is then encrypted with K2.
Since both the MAC and the plaintext are encrypted together, information on the plaintext cannot be extrapolated from the MAC, and on top of that, preserve both authenticity and integrity to the plaintext (Ballare & Nampremre, 2000).
However, the authenticity and integrity of the actual ciphertext being sent is absent due to the fact that the MAC is not appended to the ciphertext, so the receiver is unable to detect if the plaintext has been altered until after the ciphertext has been decrypted (Ballare & Nampremre, 2000).

Encrypt-then-MAC: (C=E(K2, M), T=MAC(K1, C)

The composition follows the encryption of the message using K2 to get a ciphertext. You then run that MAC over that ciphertext using K1.
As researched by Ballare and Nampremre (as cited in Black, 2004), for the purposes of AE, Ballare and Nampremre (2000) claim that Encrypt-then-MAC produces the strongest definition of AE. This is under the assumption that the encryption and MAC is implemented with a secure scheme. This composition allows for the MAC to not provide any information on the plaintext since the attacker will see a random ciphertext. It allows yourself to not check the contents if the MAC does not match, meaning avoidance of entering spoofed contents.
This composition is overall deemed the best approach and can be seen to have no weaknesses, compared to other compositions. **This was our chosen method.**

Encrypt-and-MAC: (C=E(K2, M), T=MAC(K1, M)

The composition follows the encryption of the message using K2 to get a cipher text. This time, the MAC is not over the cipher text, but over the message instead to get a tag.
Despite the fact that this approach allows the authenticity and integrity of the plaintext to be verified, Encrypt-and-MAC fails to be secure because it does not hide the complete contents of the message. It is assumed that we must have both of these overlapping because MAC by itself is not enough to preserve confidentiality (Joux, 2009)

Phase 1-2: Encrypt-then-MAC (AES-256 → SHA-256)

Why Encrypt-then-MAC?

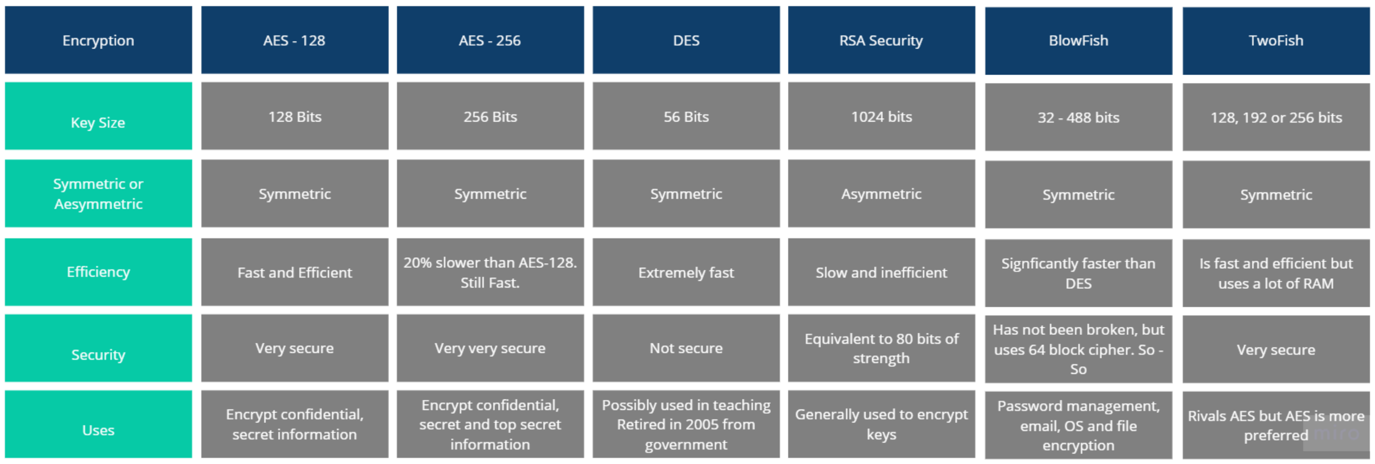
The Authenticated Encryption we decided to use is the Encrypt-then-MAC (EtM) because it was deemed to be the most secure with little to no trade-offs (Ballare and Nampremre, 2000). Comparatively to other compositions, Encrypt-and-Mac (E&M); Hash-then-Encrypt both have security issues which removed them from consideration.

This leaves ‘MAC-then-Encrypt’ (MtE) and EtM. EtM is widely accepted over MtE because MtE faces minor security issues such as the authenticity of the message is unidentifiable until the user decrypts the cipher. Encrypt and Mac avoids this problem.

The trade off in efficiency and security does not lie in the composition itself, but the algorithms that were used to implement the Encryption and MAC.

AES-256

In our assignment, we want to ensure we have the most security whilst still being reasonably efficient. We placed the algorithms in a table and compared one another in terms of key size, symmetric or asymmetric design, efficiency, security and their uses.



Encryption	AES - 128	AES - 256	DES	RSA Security	BlowFish	TwoFish
Key Size	128 Bits	256 Bits	56 Bits	1024 bits	32 - 488 bits	128, 192 or 256 bits
Symmetric or Aesymmetric	Symmetric	Symmetric	Symmetric	Asymmetric	Symmetric	Symmetric
Efficiency	Fast and Efficient	20% slower than AES-128. Still Fast.	Extremely fast	Slow and inefficient	Signficantly faster than DES	Is fast and efficient but uses a lot of RAM
Security	Very secure	Very very secure	Not secure	Equivalent to 80 bits of strength	Has not been broken, but uses 64 block cipher. So - So	Very secure
Uses	Encrypt confidential, secret information	Encrypt confidential, secret and top secret information	Possibly used in teaching Retired in 2005 from government	Generally used to encrypt keys	Password management, email, OS and file encryption	Rivals AES but AES is more preferred

Figure 1: A comparative table of encryptions to compare keysize, encryption type, efficiency, security and its uses.

We prioritize the implementation of a secure system to reach a “conservative security strategy” (McGrew, 2010). Our main objective is to ensure that despite the development of quantum computing and many lifetimes, a middle man will still be unable to decrypt a cipher if we send one. Due to AES allowing the block size and key size to vary, it is one of the most versatile algorithms and most secure. It replaced DES, and the Rijndael algorithm became the new AES we know today, which is now used by the U.S government today (Andriani et al. 2018).

However, according to Andriani et al. (2018), out of AES-128; AES-198; and AES-256, **AES-128** has the fastest processing speed. This is applied to “word documents, images, text, pdf files, presentation and video files. Andriani et al. (2018) also outlines the lowest cpu usage overall would be **AES-198** when encrypting and decrypting “word documents, excel, music/mp3, presentation files, powerpoints, and images.” Also due to the bit size of AES-258, it is slower than AES-128 and AES-196.

Nonetheless, Rao et al. (2017) shows that AES-128 can be broken by quantum computing attacks. Additionally, he also reveals that AES-256 will not be able to be broken by quantum computing and will be safe in the ‘post quantum computing era’.

Despite knowing this, for our purposes, we want to ensure that security is our biggest priority and we are willing to sacrifice CPU usage and processing speed to ensure that we have confidentiality completely secured. AES-256 is the golden standard that our assignment is looking for.

SHA-256

SHA-256, introduced in 2000 by the NSA (Gilbert & Handschuh, 2004), is a secure hash algorithm that acts as a successor to the first set of secure hash algorithms, namely SHA1, and was subsequently implemented into our security protocol to calculate the HMAC.

The key differences between SHA-256 and other secure hash algorithms has to do with the fact that they encrypt in different key lengths. SHA-256 encrypts data in 256 bit long keys whereas SHA1 encrypts in 160 bit long keys and SHA-512, another commonly used secure hash algorithm, encrypts in 512 bit long keys (RapidSSLOnline, n.d.).

Since SHA-512 does encrypt in 512 bit long keys, it does mean that SHA-512 provides better security compared to SHA-256. SHA-512 is accompanied with limitations however, which include compatibility issues where SHA-512 is only compatible with Windows OS, while SHA-256 is compatible with a wide range of operating systems and web browsers. Additionally, Ahmad and Das (2005) highlight that the implementation and computation of SHA-512 “...suffered due to increased latency.” (Ahmad & Das, 2005, p. 359).

Efficiency and Security Trade-offs

AES-256 and SHA-256 were utilized to produce a secure protocol with very little to no efficiency or security trade-offs. Although, in terms of executing the secure protocol into a Minecraft server, computing these secure algorithms to encrypt Minecraft chat logs and commands may have its drawbacks.

Employing such algorithms may increase latency within the server, especially when connecting and playing with multiple players in the one server, which may possibly make Minecraft slightly unplayable, consequently making players in the server displeased.

REFERENCES

Ahmad, I., & Das, A. S. (2005). Hardware implementation analysis of SHA-256 and SHA-512 algorithms on FPGAs. Computers & Electrical Engineering, 31(6), 345-360. https://doi.org/10.1016/j.compeleceng.2005.07.001

Andriani, R., Wijayanti, S. E., & Wibowo, F. W. (2018, November). Comparison of AES 128, 192 and 256 bit algorithm for encryption and description file. In *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)* (pp. 120-124). IEEE.

Bellare, M., Namprempre, C. (2000). Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (eds) Advances in Cryptology — ASIACRYPT 2000. ASIACRYPT 2000. Lecture Notes in Computer Science, vol 1976. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-44448-3_41

Black, J. (2005). Authenticated Encryption.

CheapSSLWeb. (2023). *SHA 256 vs SHA 512: Key Encryption Algorithms Differences Explained!.* https://cheapsslweb.com/blog/sha-256-vs-sha-512-key-encryption

Gilbert, H., & Handschuh, H. (2004). Security analysis of SHA-256 and sisters. In *Selected Areas in Cryptography: 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003. Revised Papers 10* (pp. 175-193). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-24654-1_13

Joux, A. (2009). *Algorithmic cryptanalysis*. CRC press.

Krawczyk, H. (2001). The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings 21* (pp. 310-331). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-44647-8

McGrew, D. (2011). *The Use of AES-192 and AES-256 in Secure RTP* (No. rfc6188).

Rao, S., Mahto, D., Yadav, D. K., & Khan, D. A. (2017). The AES-256 cryptosystem resists quantum attacks. *Int. J. Adv. Res. Comput. Sci*, *8*(3), 404-408.

RapidSSLOnline. (n.d). *SHA1 vs SHA2 vs SHA256 - What's the difference?.* https://www.rapidsslonline.com/blog/sha1-vs-sha2-vs-sha256-whats-the-difference/#:~:text=The%20basic%20difference%20between%20SHA1,to%20secure%20the%20data%20online.

Selent, D. (2010). Advanced encryption standard. *Rivier Academic Journal*, *6*(2), 1-14.