Technique Overview and Summary

Hash-then-Encrypt (HtE)

Append a hash value to the plaintext. Then, encrypt both together. This is the only technique that does not use a MAC out of all four. It uses a publicly known hash function to protect plaintext integrity. As such, authentic-looking forgeries can be easily made by just breaking the encryption method. The following paragraph demonstrates one way of doing this.

Suppose that a streamcipher is used (i.e. the ciphertext is computed from plaintext $\oplus$ keystream). Assuming known plaintext, an attacker could intercept a ciphertext, XOR it with *plaintext + h(plaintext)* to obtain the keystream. The obtained keystream could then be used to forge an authentic ciphertext among many other malicious possibilities.

Other examples, which are too long to list, exist. This technique is quite weak and other techniques should be used instead whenever possible.

MAC-then-Encrypt (MtE)

MtE can be represented as E(a + A). It means the plaintext (a) will first be appended a generated MAC A, then encrypt them both using a cipher such as RSA or AES. Which will provide two layers of cryptographic operations to the original message. This means that if an attacker sends a tampered ciphertext, after decryption, the MAC and plaintext will not be consistent with each other. However, it will affect property availability.

However, if an Implementation of this technique uses CBC mode (which usually requires padding), a chosen-ciphertext attack is possible. To briefly explain how, if an active attacker could know whether a modified ciphertext results in a decrypted message with valid or invalid padding (i.e. they have access to a decryption oracle), they can then construct guesses that could reveal the plaintext eventually. (Vaudenay, 2002)

Encrypt-then-MAC

Encrypt-then-MAC (EtM) can be illustrated by $\big(C = E(K2, M), T = MAC(K1, C)\big)$. To elucidate, EtM first encrypts the plaintext (M) using the key $(K2)$ which produces the ciphertext $C = E(K2, M)$. After the plaintext has been encrypted, we then produce a MAC (T) over the ciphertext, using another key $(K1)$ which results in $T = MAC(K1, C)$. To rephrase it in simpler way, EtM first encrypts the plaintext and then computes a MAC from the ciphertext produced from encryption

As a result, Encrypt-then-MAC assembles two values to be sent out, the ciphertext and the MAC. In doing so, we can use the MAC on the ciphertext to determine if any attackers tampered with the message, without decrypting our modified ciphertext (Message Authentication Codes (MACs), n.d.). Ultimately preserving the ciphertext integrity, by ensuring that the messages cannot be tampered with.

Ostensibly, EtM is CCA-secure as the attacker cannot tamper with the ciphertext, unless they have access to the MAC key to recompute the MAC.

Encrypt-and-MAC

Encrypt-and-MAC (E&M) can be visualized as $\big(C = E(K2, M), T = MAC(K1, M)\big)$. E&M encrypts the plaintext (M)first, using the key $(K2)$ to guarantee confidentiality. However, whilst EtM appends a MAC over the ciphertext, E&M appends the MAC over the plaintext. The downside to this, is that when E&M appends the MAC over the plaintext, the MAC is not encrypted which allows a potential leak of the message to the attacker if a weak MAC is used. To elaborate, MACs are used for authentication, not for protecting the contents of the plaintext (Ferguson et al., 2010). However, the main argument for E&M is that the encryption and MAC can be done in parallel, making it a faster process.

like EtM, an implementation of E&M in SSH was found to be insecure by Albrecht et al. (2009). The implementation was similarly vulnerable in that the attacker could observe a system behaving differently under different ciphertexts and use that to infer contents of an authentic ciphertext. Thus, becoming CCA-insecure, as it allows for the ciphertext to be tampered with.

Analysis and Conclusion

A general observation from the cons of E&M and MtE is that having an unauthenticated ciphertext could lead to unexpected vulnerabilities in a protocol. HtE, MtE, and E&M don't check for authenticity until after decryption and could allow for decryption oracles to exist. Thus, opening the possibility of a chosen-ciphertext attacks.

It is likely that, with enough care in implementation, protocols derived from MtE, EtM, and E&M could all be made secure. In fact, contrary to our previous observation, MtE has been proven to be CCA-secure under certain configurations (Krawczyk, H., 2001). However, as a precautionary net against incorrect implementations and unexpected vulnerabilities, we will choose to use EtM.

Appendix

REFERENCES

1. Krawczyk, H. (2001). *The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)*. Advances in Cryptology — CRYPTO 2001, 310–331. https://doi.org/10.1007/3-540-44647-8_19
2. Vaudenay, S. (2002). *Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS...*. Advances in Cryptology — EUROCRYPT 2002, 534–545. https://doi.org/10.1007/3-540-46035-7_35
3. Albrecht, M. R., Paterson, K. G., & Watson, G. J. (2009). *Plaintext Recovery Attacks against SSH*. 2009 30th IEEE Symposium on Security and Privacy. https://doi.org/10.1109/sp.2009.5
4. *Message Authentication Codes (MACs)*. (n.d.). Computer Security. Retrieved October 18, 2022, from https://textbook.cs161.org/crypto/macs.html
5. Ferguson, N., Schneier, B., & Kohno, T. (2010, March 15). *Cryptography Engineering: Design Principles and Practical Applications* (1st ed.). Wiley.