International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Cybersecurity: All-Hash Message Authentication Code (AH-MAC)

Binoy Kurikaparambil Revi

Independent Researcher, USA
binoyrevi@live.com

Abstract

Confidentiality and authenticity of messages are crucial cybersecurity requirements in data exchange. Strong encryption algorithms for message confidentiality are not enough to convey the actual message. A suitable Message Authentication Code (MAC) ensures the authenticity of the message from the sender. The All-Hash Message Authentication Code (AH-MAC) provides a stronger algorithm by enhancing the Hash-Based Message Authentication Code (HMAC) algorithm to stand firm against attacks like Message Extension Attacks.

Introduction:

The Hash-based Message Authentication Code (HMAC) algorithm verifies a message's authenticity. However, this algorithm can be attacked, such as by a message extension attack. All-Hash Message Authentication Code(AH-MAC) provides an enhanced algorithm to overcome such attacks.

Hash-Based Message Authentication Code (HMAC)

The message authentication code generated using the hash-based algorithm starts by breaking the message into multiple blocks. It uses an H0, a start state known by both the sender and recipient, to hash the first message block. The output of this hash function is used to hash the second message block. The process continues until the final message block is reached. The cipher text generated by the last encryption is used as the MAC value attached to the encrypted message to verify the message's authenticity at the recipient's end.

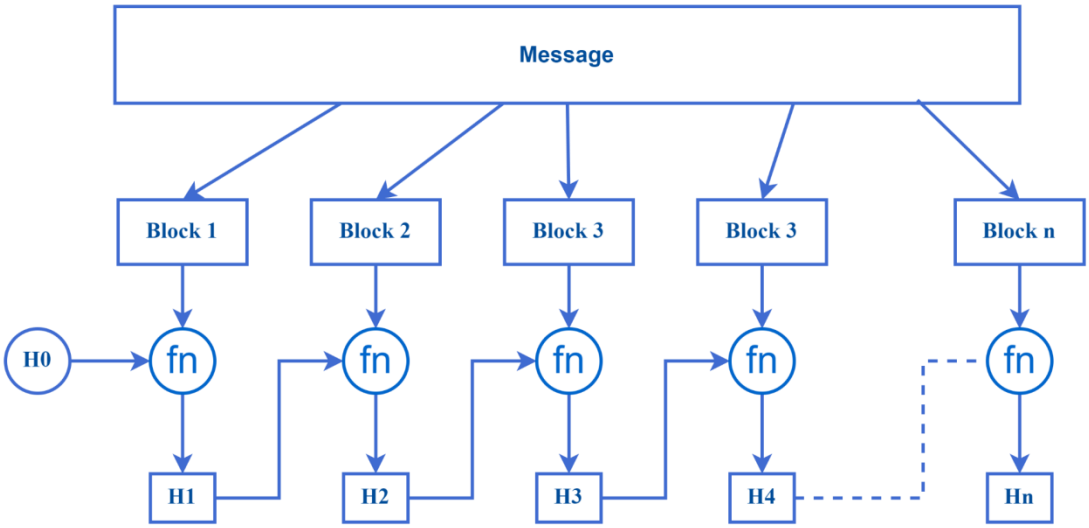

Figure 1: MAC generation using Cipher Block Chaining

Figure 1 demonstrates the HMAC processing. At first glance, this system appears to be very robust. We may think that without all the expected blocks, the final block's output, specifically the MAC (Message Authentication Code) generation, will fail. However, upon further reflection from a different perspective, it's clear that an attacker doesn't need to know the outputs of all the blocks. The attacker only needs the output from the previous block to insert a new data block and generate a malicious MAC. This will lead to a Message extension attack.

Message Extension Attack:

The original message format that needs to be hashed is illustrated in Figure 2. The attacker does not know the key K, nor do they know the message.

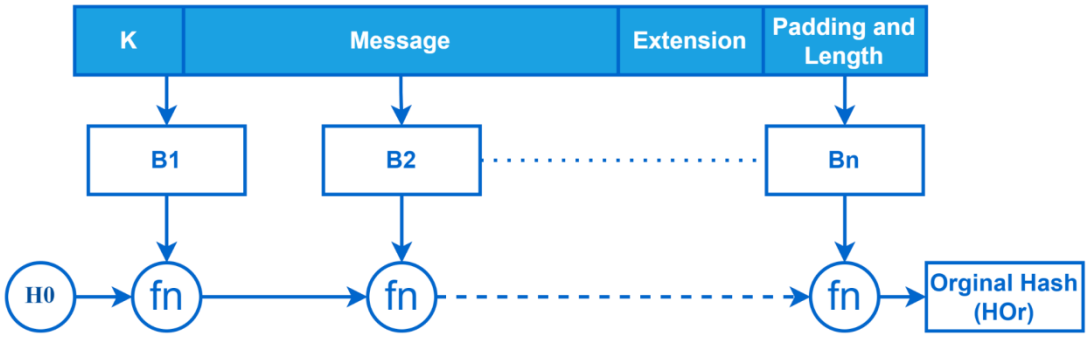

Figure 2: Original message processing to generate hash

The attacker is aware of the hash value derived from the last block. Using this hash value, they can append a message or malicious data to the original message and continue hashing. Using the known hash value, the attacker can produce a new, malicious Message Authentication Code (MAC). Figure 3 illustrates the message extension attack.

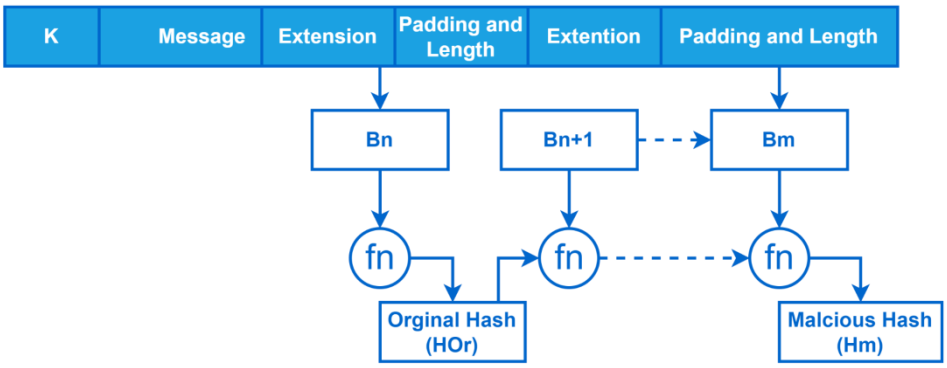

Figure 3: Extension Attack

The extension attack goes undetected in case of large message encryption like the PDF where the extension may not even get to the screen. Message extension attacks can also be used by attackers to trick people who ignore the weird message length.

All-Hash MAC to the Rescue

All-hash MAC uses the intermediate hashes to induce dependency to create the final Hash value. The process starts precisely as the HMAC; however, all the intermediate MACs are temporarily stored. Figure 4 and the algorithm below demonstrate the AH-MAC algorithm.

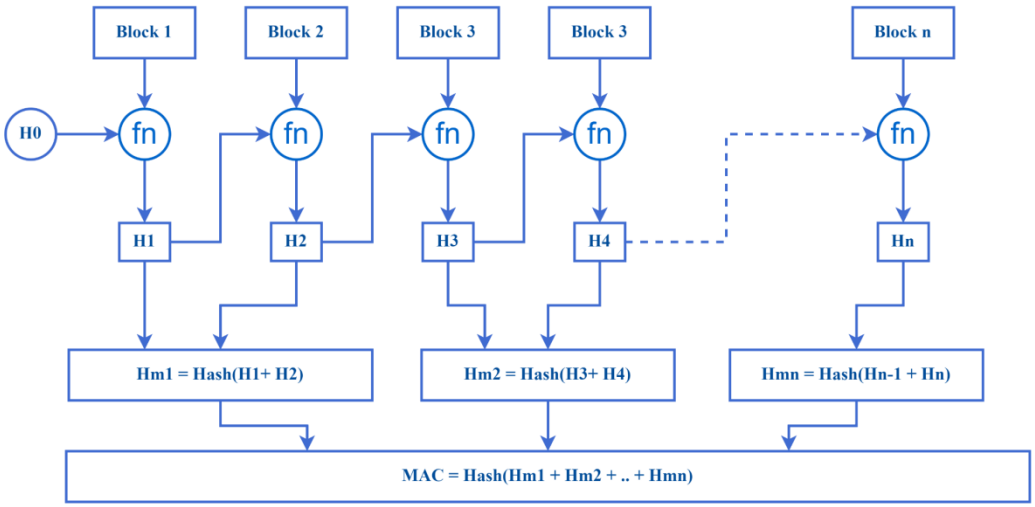

Figure 4: AH- MAC Message Processing

Algorithm:

- Step 1: Follow the HMAC algorithm to generate all the Hash values (H1 to Hn)
- Step 2: Concatenate adjacent hash values starting from (H1, H2) to (Hn-1, Hn)
- Step 3: Create hash values from the concatenated hash values. (Hm1 to Hmn). Now we should have half the number of original hash values.
- Step 4: Concatenate all the hash values from Hm1 to Hmn and generate the new Hash. This is the final MAC which is called AH-MAC.

- Step 5 [Optional]:The final generated MAC can be encrypted using Public key cryptography[1] like, RSA depending on the security level and then used as MAC.

Conclusion:

Technically, the hashing of the hash values does not provide additional security in a typical scenario. Once a message is hashed, it is not irreversible. In other words, we cannot get the original message from the generated hash. However, when generating a MAC, we don't care about getting the original message from the MAC. All we care about is the authenticity of the message. The All-Hash algorithm will fail to calculate the original hash if a bit is changed anywhere in the message or any bit is added. This is what prevents it from the extension attack. The drawback of the AH-MAC is that it may downgrade the performance of calculating the MAC of an extremely large message, like a textbook in PDF format. The HMAC with the baked key in the message is a good option in this case.

References:

1. "IEEE Standard Specifications for Public-Key Cryptography," in IEEE Std 1363-2000 , vol., no., pp.1-228, 29 Aug. 2000, doi: 10.1109/IEEESTD.2000.92292.
2. Peyrin, T., Wang, L. (2014). Generic Universal Forgery Attack on Iterative Hash-Based MACs. In: Nguyen, P.Q., Oswald, E. (eds) Advances in Cryptology – EUROCRYPT 2014. EUROCRYPT 2014. Lecture Notes in Computer Science, vol 8441. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-55220-5_9
3. C. E. Castellon, S. Roy, O. P. Kreidl, A. Dutta and L. Bölöni, "Towards an Energy-Efficient Hash-based Message Authentication Code (HMAC)," 2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC), Pittsburgh, PA, USA, 2022, pp. 1-7, doi: 10.1109/IGSC55832.2022.9969377.