Background write-up

Course :

Data Integrity

Dr :

Maged Abdelaty

Team members:

Shadwa ahmed:2205026

Sara ahmed:2205094

Yehia tarek:2205062

1. Background

1.1 What is a MAC?

A Message Authentication Code (MAC) is a cryptographic checksum used to ensure both the integrity and authenticity of a message. MACs are generated using a secret key and the message content. Only parties who share the secret key can generate or verify the MAC. If the MAC is valid, the recipient can be confident the message has not been altered.

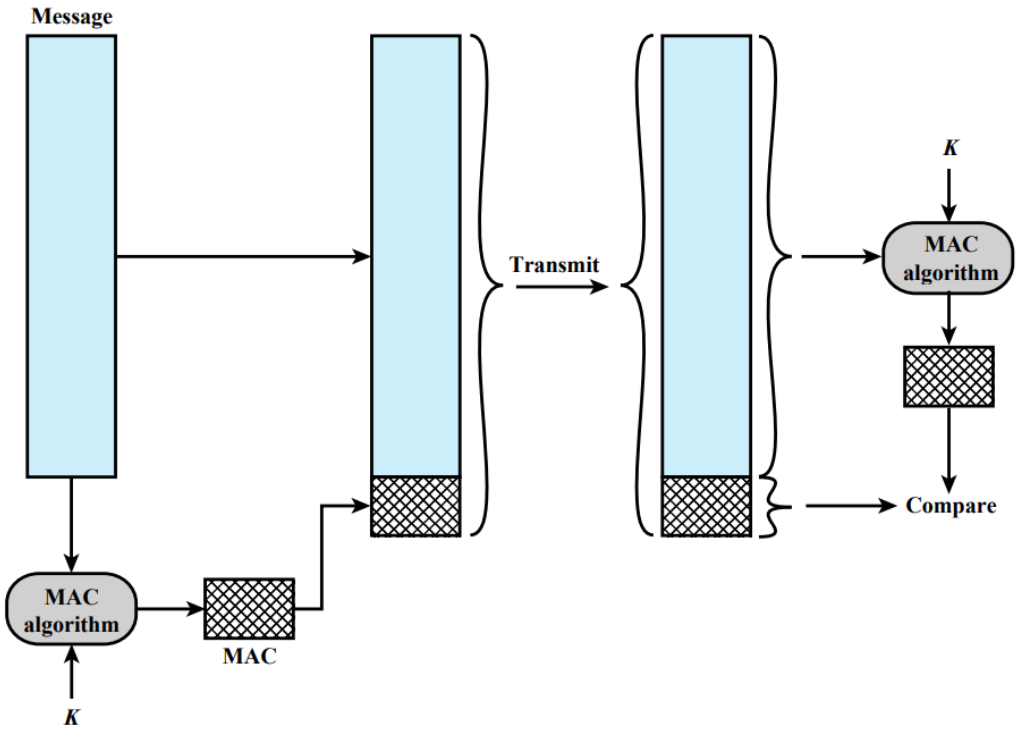

Figure: Message Authentication Code (MAC) generation and verification process. The sender uses a shared secret key K to generate the MAC. The receiver uses the same key to validate the message integrity.

1.2 How does a Length Extension Attack work?

Length Extension Attacks exploit the internal structure of certain hash functions such as MD5 and SHA1. When a MAC is implemented as hash(secret || message), an attacker who knows the hash output for some message can append extra data to the message and compute a valid MAC for the new message — all without knowing the secret key.

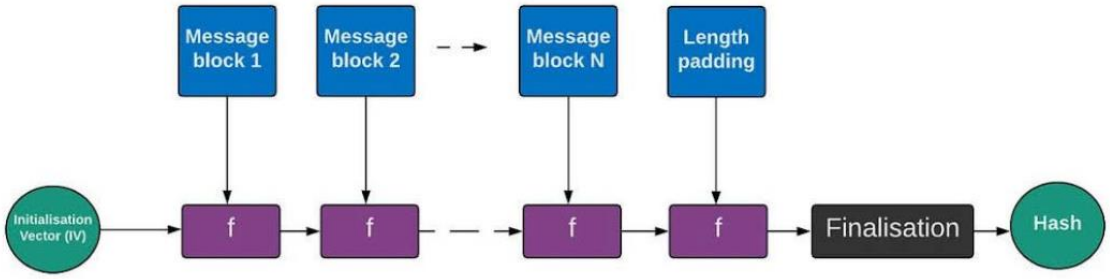

Figure: Hash functions like MD5 and SHA1 process input in a block-wise manner using a compression function. Length extension attacks exploit this structure by resuming the hashing process from an intermediate state using additional data and padding.

1.3 Why is MAC = hash(secret || message) insecure?

This construction is **insecure** because it allows attackers to perform **length extension attacks**. Hash functions such as MD5 and SHA-1 process input data in **fixed-size blocks**, and their internal **intermediate states** can be manipulated.

If an attacker knows the value of `hash(secret || message)` and also has access to the original `message`, they can exploit the structure of the hash function to compute:

```
hash(secret || message || padding || extra_data)
```

This enables the attacker to **forge a valid MAC (Message Authentication Code)** for a modified message without knowing the secret key. The vulnerability arises because the hash function continues from the internal state of the original hash, allowing a malicious extension.

References

1. Krawczyk, H., Bellare, M., & Canetti, R. (1997). *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104. Retrieved from https://datatracker.ietf.org/doc/html/rfc2104
2. Bellare, M., Canetti, R., & Krawczyk, H. (1996). *Keying Hash Functions for Message Authentication*. Advances in Cryptology—CRYPTO'96.
3. Stallings, W. (2016). *Cryptography and Network Security: Principles and Practice* (7th ed.). Pearson.