Team members

1. Jana Adel 2205105
2. Menna Shady 2205138
3. Nadeen Ehab 2205081

Data Integrity and Authentication

Assignment: Demonstrating and Mitigating a Message Integrity Attack (MAC Forgery)

1. What is a Message Authentication Code (MAC)?

A Message Authentication Code (MAC) is a short piece of information used to authenticate a message and to provide integrity and authenticity assurances. It is generated by a cryptographic algorithm that takes a secret key as input and the message. The MAC value accompanies the message and allows the receiver, who also knows the secret key, to verify that the message has not been altered and that it originates from the legitimate sender.

The primary purpose of a MAC is twofold:

- Integrity: Detect any changes or tampering of the message.
- Authentication: Confirm the message is from the authorized sender who holds the secret key.

MACs are widely used in network protocols, secure communications, and data storage to ensure messages remain trustworthy.

2. How Does a Length Extension Attack Work in Hash Functions like MD5/SHA1?

Certain cryptographic hash functions such as MD5 and SHA-1 process data iteratively using a Merkle–Damgård construction. They absorb input data in fixed-size blocks and update an internal state accordingly.

A **length extension attack** exploits this property by allowing an attacker who knows the hash output of an unknown message (and its length) to compute the hash of that message concatenated with additional data — *without knowing the original message or secret*.

This works because hash functions like MD5 and SHA-1:

- Maintain an internal state after hashing the original message.
- Use padding that depends only on the total length of the message.

Thus, an attacker can:

- Take the known hash (internal state after the original message).
- Append padding and new data.
- Continue the hash computation from the known state.
- Produce a valid hash for the extended message.

This vulnerability is critical when a MAC is constructed as `hash(secret || message)` because the attacker can append to `message` and forge a valid MAC without knowing the secret key.

3. Why is MAC = hash(secret || message) Insecure?

Constructing a MAC as `hash(secret || message)` is **insecure** because it directly inherits the vulnerability of length extension attacks from the underlying hash function.

In this construction:

- The secret key is prepended to the message.
- The hash is computed over `secret || message`.

Due to the length extension property, an attacker who knows:

- The `message`,
- The MAC (hash output of `secret || message`),

Can:

- Perform a length extension attack to append extra data to `message`,
- Compute a valid MAC for the extended message,
- Without ever knowing the secret key.

This breaks the integrity and authentication guarantees of the MAC, allowing attackers to forge valid messages.