Alexandria National University

Faculty of Computers and Data Science

Data Integrity & Authentication Course

Background write-up Report

Team Participants

Omar Magdy Abdulla 2205221

Mohamed Hany Khatab 2205053

Youssef Saeed Thabet 2205064

Message Authentication Code (MAC) and Its Purpose

A Message Authentication Code (MAC) is a short piece of information used to authenticate a message and ensure its integrity. In the context of cryptography, a MAC is generated using a secret key and a cryptographic function (often a hash function or a block cipher).

The resulting MAC value is then sent along with the message to the recipient. The recipient, who also possesses the secret key, recomputes the MAC using the same cryptographic function and verifies that it matches the received MAC. If the values match, the message is considered authentic and unaltered.

Purpose of MAC:

Data Integrity: Ensures that the message has not been altered during transmission.

Authentication: Confirms that the message was sent by someone who knows the secret key.

MACs are widely used in secure communication protocols such as SSL/TLS, IPSec and in applications like secure file transfers, digital signatures, and payment systems.

How Length Extension Attacks Work in Hash Functions like MD5/SHA1

Hash functions such as MD5 and SHA1 use a design known as the Merkle–Damgård construction. This construction processes messages in fixed-size blocks and maintains an internal state (the hash state) that is updated for each block. Crucially, the length of the original message is included in the final padded block.

A Length Extension Attack exploits this structure. If an attacker knows the hash value of an unknown message (e.g., hash(secret || message)), and the length of the original message, they can append additional data to it and compute a new valid hash without knowing the secret. This is possible because the attacker can simulate the continuation of the hash function using the known hash state.

Example Scenario:

Attacker intercepts MAC = hash(secret || message) and message

Attacker appends data: message || padding || extra_data

Using the original hash value and estimated message length, attacker computes a new valid MAC

Server sees new_message and new MAC and believes it's authentic.

This attack works only for poorly designed MAC schemes that use plain hashes like hash(secret || message).

Why MAC = hash(secret || message) is Insecure

The naive MAC construction **MAC = hash(secret || message)** is vulnerable to length extension attacks because of how the hash function processes input and finalizes its state. The attacker, knowing the hash of a message and its length, can deduce how the internal state was after the original message and continue the hash computation with new data.

Key Issues:

- **The attacker does not need to know the secret to perform the extension.**
- **The hash function's internal state leaks too much information.**
- **The padding and message length information used by the hash function can be guessed or inferred.**

This design flaw enables attackers to forge a valid MAC for a modified message, thus breaking both integrity and authentication guarantees.

Secure Alternative: The secure way to construct a MAC is to use HMAC (Hash-based Message Authentication Code), which is specifically designed to resist length extension attacks by incorporating the secret key in a more robust and secure manner.

HMAC uses a double hashing approach: HMAC(key, message) = hash((key XOR opad) || hash((key XOR ipad) || message)), effectively hiding the internal hash state from attackers.
