***Bonus Assignment***

***Data Integrity  
Team:   
Yehia Tarek – Sara Ahmed – Shadwa Ahmed***

**a. What is a MAC and its purpose in data integrity/authentication?**

A **Message Authentication Code (MAC)** is a cryptographic checksum used to verify both the integrity and authenticity of a message. It ensures that the message has not been altered in transit (integrity) and that it originates from the expected sender (authentication).

A MAC is generated using a secret key shared between the sender and receiver. The sender computes MAC = F(key, message) and appends it to the message. The receiver recomputes the MAC using the same key and verifies it matches the received MAC.

Common MAC constructions include:

* **HMAC (Hash-based MAC)** (RFC 2104)
* **CMAC (Cipher-based MAC)** (NIST SP 800-38B)

**Purpose:**

* Prevents tampering (e.g., man-in-the-middle attacks).
* Ensures message origin authenticity.

**References:**

* Kaufman, C., Perlman, R., & Speciner, M. (2002). *Network Security: Private Communication in a Public World*. Prentice Hall.
* NIST. (2016). *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication* (SP 800-38B).

**b. How does a length extension attack work in hash functions like MD5/SHA1?**

A **length extension attack** exploits the Merkle-Damgård construction used in hash functions like MD5 and SHA-1. These hashes process messages in fixed-size blocks and maintain an internal state (hash value) between blocks.

**Attack Steps:**

1. The attacker knows H = hash(secret || message) (but not the secret).
2. They append malicious data (padding || new\_data) to the original message.
3. The hash function’s internal state after processing secret || message || padding is H.
4. The attacker computes new\_hash = hash(H, new\_data), effectively generating hash(secret || message || padding || new\_data) without knowing the secret.

**Impact:**

* Allows forging valid hashes for modified messages.
* Renders constructions like MAC = hash(secret || message) insecure.

**Mitigation:** Use HMAC or truncated hashes.

**References:**

* Schneier, B. (1996). *Applied Cryptography*. Wiley.
* Ristenpart, T., et al. (2009). *"How to Strengthen the Security of MACs via Double-Calling."* CRYPTO.

**c. Why is MAC = hash(secret || message) insecure?**

The construction MAC = hash(secret || message) is vulnerable to:

1. **Length Extension Attacks** (as above): An attacker can compute hash(secret || message || extension) without knowing the secret.
2. **Collision Vulnerabilities**: If hash is MD5/SHA-1, collisions can be found, breaking integrity.
3. **Secret Exposure**: If the hash’s internal state leaks, the secret may be recoverable.

**Secure Alternative:** **HMAC** (RFC 2104) uses hash(secret ⊕ opad || hash(secret ⊕ ipad || message)) to prevent length extension.

**References:**

* Bellare, M., et al. (1996). *"Keying Hash Functions for Message Authentication."* CRYPTO.
* NIST. (2012). *"Recommendation for Keyed-Hash Message Authentication Codes (HMAC)."* (FIPS 198-1).