Mariam Mostafa Amin 2205084
Hannah Emadeldein 2205123
Nada Mohamed Abdelsatar 2205173

Message Integrity Attack (MAC Forgery)

—

1. What is a Message Authentication Code (MAC) and Its Purpose in Data Integrity and Authentication

A **Message Authentication Code (MAC)** is a cryptographic technique designed to ensure the **integrity** and **authenticity** of a message. It is generated using a secret key and a cryptographic function, producing a fixed-size output known as a MAC tag.

The two primary purposes of a MAC are:

- **Data Integrity**: Verifies that the message has not been altered during transmission or storage. Any modification to the message will result in a mismatch between the computed and received MACs.
- **Authentication**: Confirms that the message was generated by a party who possesses the shared secret key, thereby ensuring the authenticity of the sender.

Key characteristics of MACs include:

- Utilization of a **shared secret key** between communicating parties (symmetric cryptography).
- **Fixed-length output**, regardless of the input message size.
- Resistance to forgery, making it **computationally infeasible** to generate a valid MAC without knowledge of the secret key.

MACs can be implemented using:

- Symmetric encryption techniques (e.g., Cipher Block Chaining).
- Hash functions (e.g., HMAC).
- Dedicated MAC algorithms.

2. Length Extension Attack in Hash Functions

A **length extension attack** exploits a structural vulnerability in hash functions built on the **Merkle–Damgård construction**, such as MD5 and SHA-1. These hash functions process input in fixed-size blocks and maintain an internal state throughout the computation.

The attack works under the following conditions:

- When a MAC is constructed as MAC = hash(secret || message), and the attacker knows the output (MAC) and original message, they can compute hash(secret || message || padding || extension) **without knowing the secret key**.
- This capability allows the attacker to **forge a new valid MAC** for an extended message, effectively bypassing message integrity and authentication checks.

This type of attack demonstrates the critical weakness of naïve hash-based MAC constructions, especially when using vulnerable hash functions.

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

The construction MAC = hash(secret || message) is considered insecure due to several cryptographic weaknesses:

- **Vulnerability to Length Extension Attacks**: As previously explained, this structure allows attackers to extend the message and forge valid MACs without access to the secret key.
- **Lack of Cryptographic Mixing**: The simple concatenation of the secret and message fails to provide robust cryptographic separation, making it easier for attackers to manipulate the input and predict outcomes.
- **Collision Risks**: Legacy hash functions like MD5 and SHA-1 are susceptible to collision attacks, where two different inputs produce the same hash value, further compromising security.

To address these issues, secure MAC implementations should rely on:

- **HMAC (Hash-based Message Authentication Code)**: Applies the hash function in two stages using independently derived keys, making it resistant to length extension and collision attacks.
- **Block cipher-based MACs** or other dedicated cryptographic constructions.

Using HMAC or similarly secure algorithms ensures that both message integrity and authenticity are preserved, even against sophisticated cryptographic attacks.