Explanation: Why HMAC Mitigates Length Extension Attacks

Hash functions like MD5 and SHA-1 are vulnerable to length extension attacks when used directly as:

```
MAC = hash(secret || message)
```

because their internal design (Merkle–Damgård construction) allows an attacker who knows `hash(secret || message)` to append data to the message and compute a valid MAC without knowing the secret key.

How HMAC Works

HMAC fixes this by hashing twice with a key mixed in both stages:

$$HMAC(K,M)=H((K'\oplus opad)||H((K'\oplus ipad)||M))$$

- K′ is the key padded to the block size.
- `ipad` and `opad` are fixed padding constants.
- The inner hash processes the key mixed with `ipad` and the message.
- The outer hash processes the key mixed with `opad` and the inner hash output.

Why This Prevents Length Extension

1. **Double hashing breaks the continuity of the hash state.**
   Attackers only see the final HMAC output, which includes an outer hash, so they cannot continue hashing from an intermediate state.
2. **Key mixing in both stages protects the secret.**
   The attacker cannot isolate or manipulate the internal state without the key.
3. **Padding is integral and inseparable from the key and message.**
   This binds the message length and prevents adding extra data without the key.

Visual Analogy

The original MAC scheme is like a single locked box that an attacker can open and add extra items inside (extend message).
HMAC is like two locked boxes, one inside the other, both secured by the key, preventing any