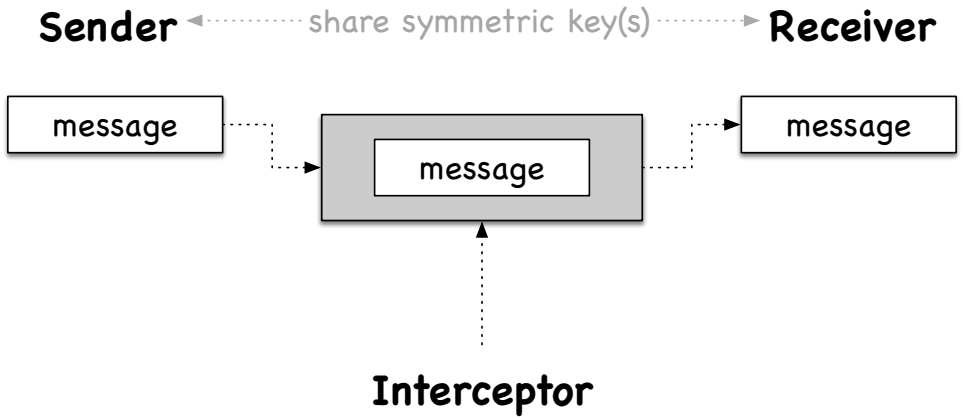
Message Authentication Code (MAC)

- MACs — what are they and how do they work?

Message Authentication Code (MAC)

Problem:

- MITM attacks possible on network communication
- MITM can **intercept** and **modify** data
- Receiver needs to verify integrity of data



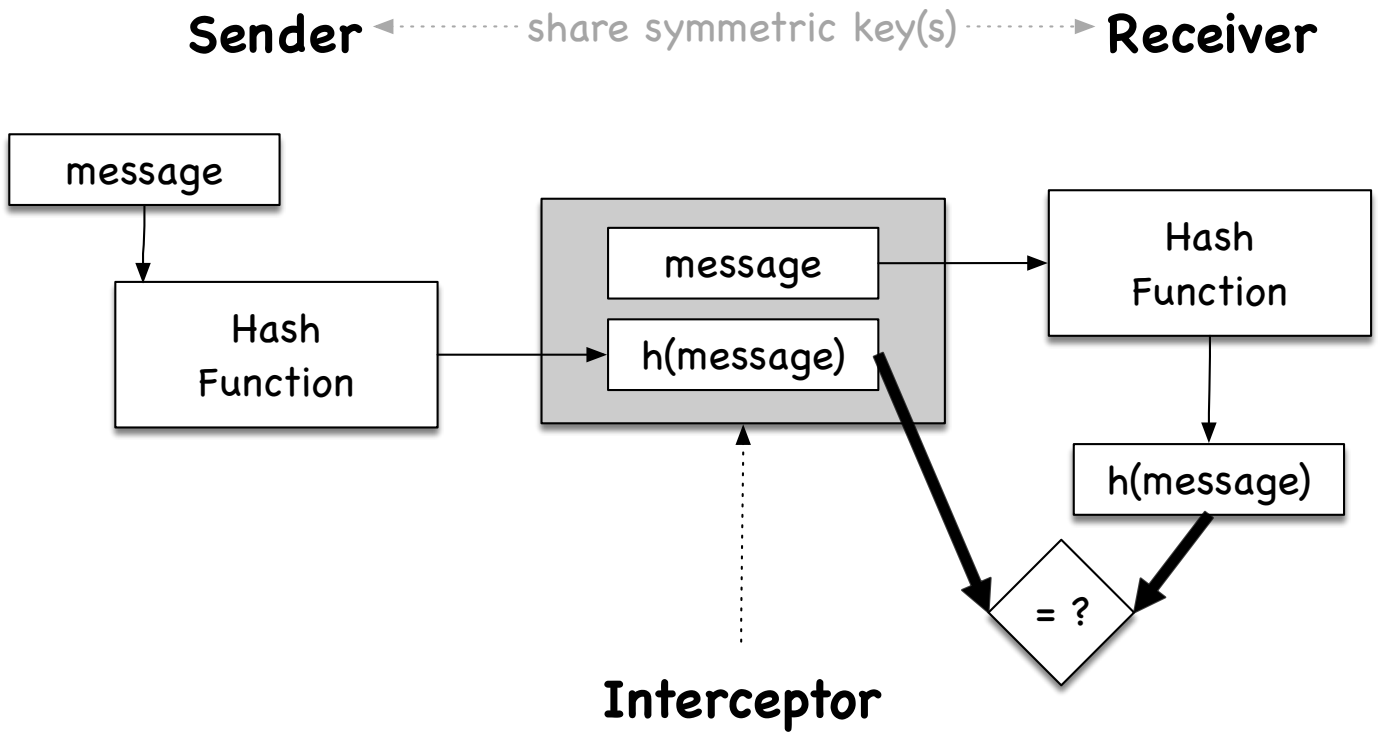
Solution: Attach a tag to data

- **Do not** use (only…) a one-way hash as tag (MITM can recompute hash!)
- **Do** use a shared secret (key) between sender and receiver in the hash
- MITM cannot compute hash without secret key

--> (Keyed) "Hash-based MAC" (HMAC)

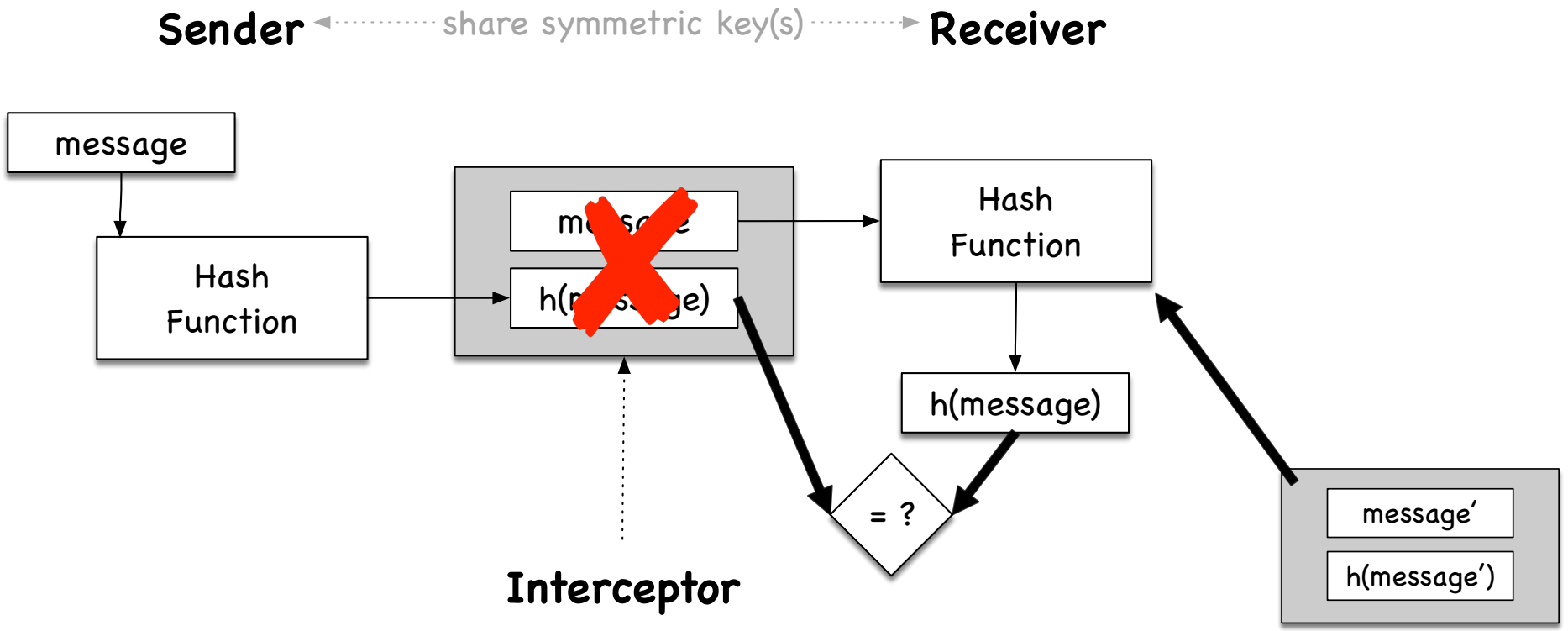
Using Only a One-Way Hash Function…

- Why should we **not** just use a one-way hash function to compute the tag?
  -> MITM can generate a new message (re)compute its hash!



Using Only a One-Way Hash Function…

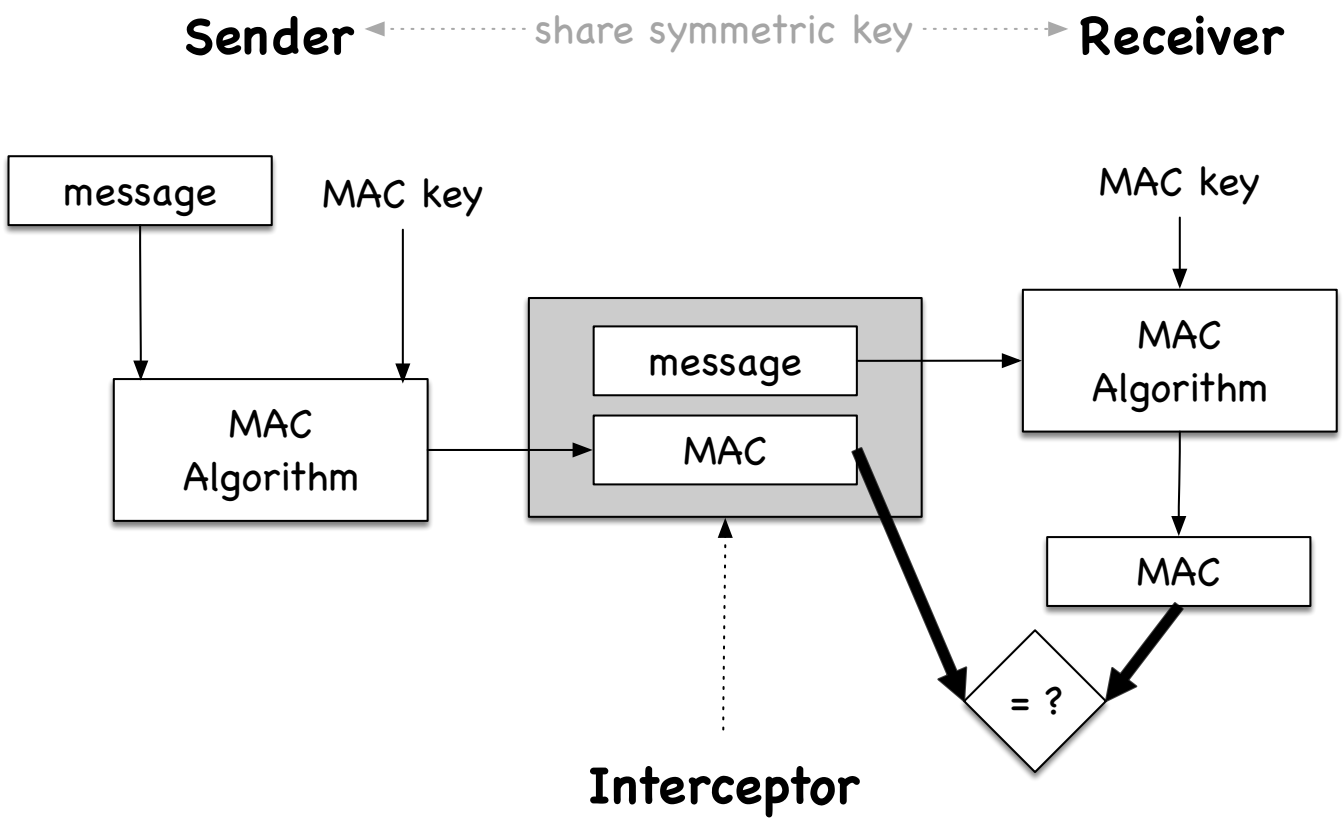
- Why should we **not** just use a one-way hash function to compute the tag?
  -> MITM can generate a new message (re)compute its hash!



Without a KEY, attacker can generate their own message' and valid hash — h(message')

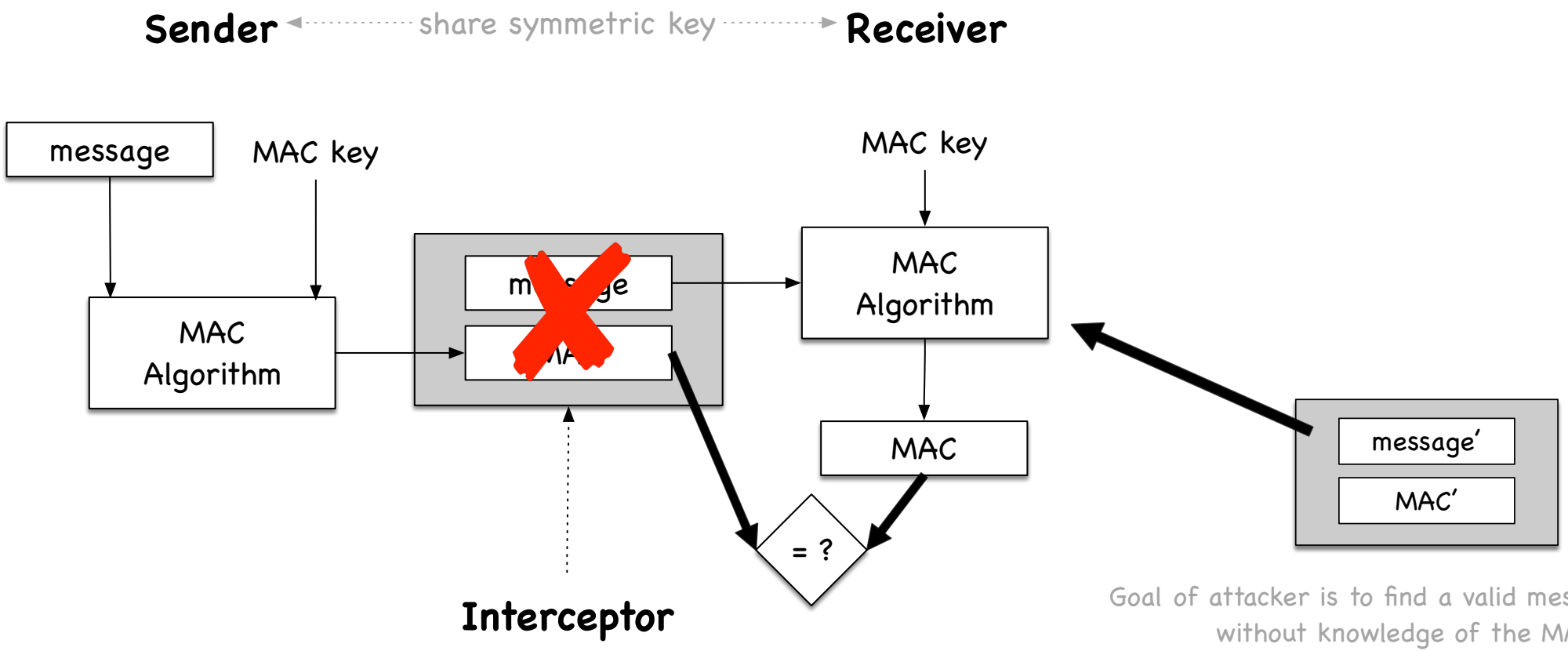
Message Authentication Code (MAC)

- **Why** use a MAC algorithm to generate the MAC?
  -> MITM cannot generate a new message/MAC pair without knowledge of the key!



Message Authentication Code (MAC)

- **Why** use a MAC algorithm to generate the MAC?
  -> MITM cannot generate a new message/MAC pair without knowledge of the key!



Goal of attacker is to find a valid message/MAC pair without knowledge of the MAC key ("MAC Forgery Attack")

Hash-based Message Authentication Code (HMAC)

Different approaches for building MAC algorithm

- Based on block cipher (e.g., CBC-MAC)
- Based on cryptographic hash function (e.g., HMAC-md5, HMAC-sha256) -> $h(K_1 \,||\, h(K_2 \,||\, M))$