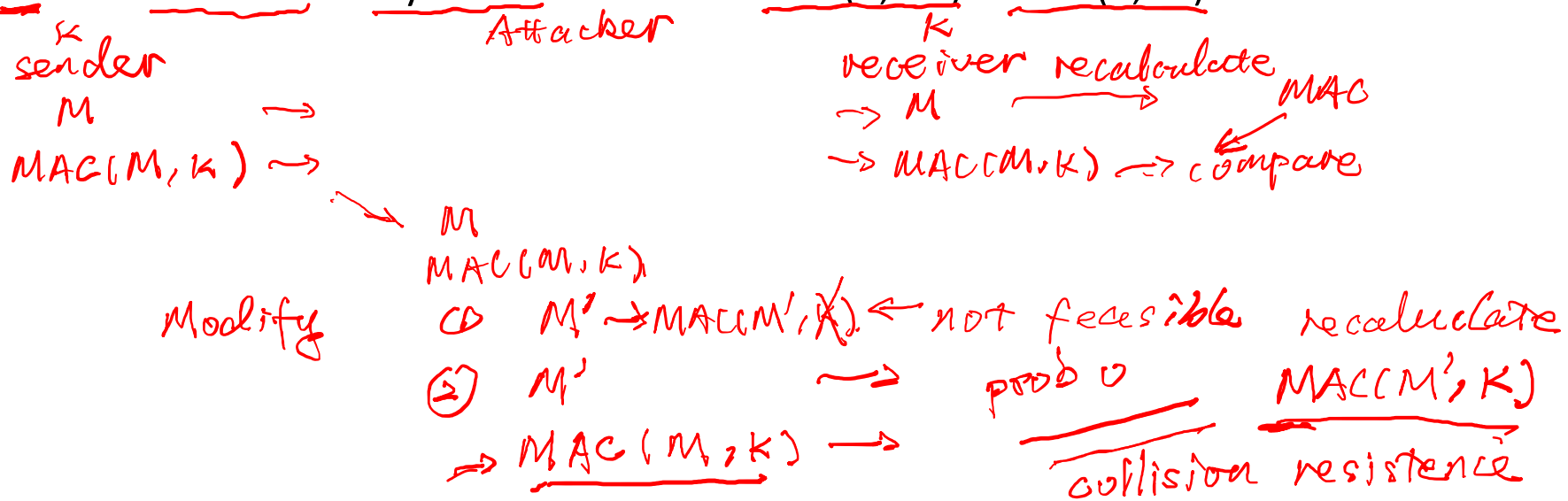
Existentially unforgeable

- A secure MAC is **existentially unforgeable**: without the key, an attacker cannot create a valid tag on a message
  - David cannot generate MAC(*K*, *M*') without *K*
  - David cannot find any *M*' ≠ *M* such that MAC(*K*, *M*') = MAC(*K*, *M*)



Example: HMAC

- issued as RFC 2104 [1]
- has been chosen as the mandatory-to-implement MAC for IP Security
- Used in Transport Layer Security (TLS) and Secure Electronic Transaction (SET)

[1] “HMAC: Keyed-Hashing for Message Authentication”, RFC 2104, https://datatracker.ietf.org/doc/html/rfc2104

HMAC(K, M)

- will produce two keys to increase security
- If key is longer than the desired size, we can hash it first, but be careful with using keys that are too much smaller, they have to have enough randomness in them
- Output $H[(K^+ \oplus opad) \,||\, H[(K^+ \oplus ipad) \,||\, M]]$

Example: HMAC

- HMAC(K, M):
  - Output $H[(K^+ \oplus opad) \,||\, H[(K^+ \oplus ipad) \,||\, M]]$
- Use K to derive two different keys
  - *opad* (outer pad) is the hard-coded byte **0x5c** repeated until it's the same length as K^+
  - *ipad* (inner pad) is the hard-coded byte **0x36** repeated until it's the same length as K^+
  - As long as *opad* and *ipad* are different, you'll get two different keys
  - For paranoia, the designers chose two very different bit patterns, even though they theoretically need only differ in one bit

HMAC

$H[(K^+ \oplus opad) \,||\, H[(K^+ \oplus ipad) \,||\, M]]$

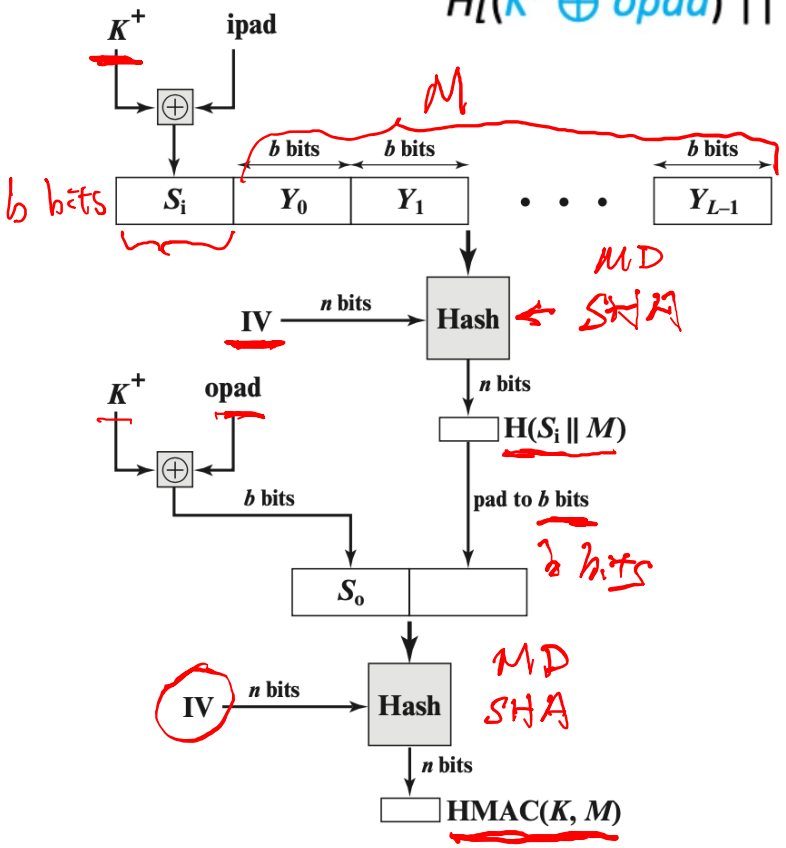


Figure 3.6 HMAC Structure

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

$$K^+ = \begin{cases} H(K) & K \text{ is larger than block size} \\ K & otherwise \end{cases}$$

ipad = 00110110 , repeat b/8 times

opad = 01011100, repeat b/8 times

HMAC procedure

$$H[(K^+ \oplus opad) \,||\, H[(K^+ \oplus ipad) \,||\, M]]$$

- Step 1: Append zeros to the left end of K to create a b-bit string K^+ (e.g., if K is of length 160 bits and b = 512, then K will be appended with 44 zero bytes);
- Step 2: XOR (bitwise exclusive-OR) K^+ with ipad to produce the b-bit block S_i;
- Step 3: Append M to S_i;
- Step 4: Apply H to the stream generated in step 3;
- Step 5: XOR K^+ with opad to produce the b-bit block S_o;
- Step 6: Append the hash result from step 4 to S_o;
- Step 7: Apply H to the stream generated in step 6 and output the result.

HMAC Properties

- HMAC(K, M) = $H[(K^+ \oplus opad) \,||\, H((K^+ \oplus ipad) \,||\, M]]$
- HMAC is a hash function, so it has the properties of the underlying hash too
  - It is collision resistant
  - Given HMAC(K, M), an attacker can't learn M – *one way*
  - If the underlying hash is secure, HMAC doesn't reveal M, but it is still deterministic
- You can't verify a tag T if you don't have K
- This means that an attacker can't brute-force the message M without knowing K

MACs: Summary

- Inputs: a secret key and a message
- Output: a tag on the message
- A secure MAC is unforgeable: Even if David can trick Alice into creating MACs for messages that David chooses, David cannot create a valid MAC on a message that she hasn't seen before
  - Example: HMAC(K, M) = $H((K^+ \oplus opad) \,||\, H((K^+ \oplus ipad) \,||\, M))$
- MACs do not provide confidentiality

Do MACs provide integrity?

- Do MACs provide integrity?
  - Yes. An attacker cannot tamper with the message without being detected
- Do MACs provide authenticity?
  - It depends on your threat model
  - If only two people have the secret key, MACs provide authenticity: it has a valid MAC, and it's not from me, so it must be from the other person
  - More than one secret key, If a message has a valid MAC, you can be sure it came from *someone with the secret key*, but you can't narrow it down to one person
- Do MACs provide confidentiality?

Authenticated Encryption

Authenticated Encryption: Definition

- **Authenticated encryption** (**AE**): A scheme that simultaneously guarantees confidentiality and integrity (and authenticity, depending on your threat model) on a message
- Two ways of achieving authenticated encryption:
  - Combine schemes that provide confidentiality with schemes that provide integrity
  - Use a scheme that is designed to provide confidentiality and integrity