Suppose H is a Merkle-Damgård hash function built from a <u>secure</u> compression function

Several ways to build a keyed function:

1. Prepend key : $F(k, m) := H(k \| m)$
   ↳ Insecure due to structure of Merkle-Damgård: can mount an "extension attack:" given $H(k \| m)$, can compute $H(k \| m \| m')$ by extending Merkle-Damgård chain
2. Append key: $F(k,m) := H(m \| k)$
   ↳ Similar to hash-then-MAC construction and vulnerable to same offline attack: adversary finds a collision in the Merkle-Damgård prefix and uses that to construct a forgery
      ↳ Structure exploited in SHA-1 collision demonstration (can generate <u>arbitrary</u> collisions once prefix matches)
      → for SHA-1, they used PDF files
3. Envelope method: $F(k,m) := H(k \| m \| k)$
4. Two-key nest: $F((k_1, k_2), m) := H(k_2 \| H(k_1 \| m))$

} for reasonable pseudorandomness assumptions on h (e.g., both $F_1(k,m) := h(k,m)$ and $F_2(k,m) := h(m,k)$ is a PRF), both of these constructions are secure PRFs on a variable-size domain

hash-based MAC

HMAC is a PRF/MAC based on the two-key nest (though with correlated keys):

$$\mathrm{HMAC}(k, m) := H(k_1 \| H(k_2, m))$$

where $k_1 \leftarrow k \oplus \mathrm{ipad}$ and $k_2 \leftarrow k \oplus \mathrm{opad}$

and ipad and opad are fixed strings (specified in the HMAC standard)

0x36 repeated   0x5C repeated

<u>Security</u>: Since k_1 and k_2 are correlated, need to make stronger assumption on security (e.g., h remains pseudorandom under a <u>related-key</u> attack)

<u>Instantiations</u>: Typically, denoted HMAC-H where H is the hash function

e.g., HMAC-SHA1

HMAC-SHA256 — one of the most widely-used MAC on the web (used in SSL/TLS, IPsec, SSH, and more)

<u>HMAC for key-derivation</u>: Recall that under reasonable assumptions, HMAC is a secure PRF

In many protocols, we need to derive multiple keys from a single master key (e.g., derived from a password)

↳ To derive multiple independent cryptographic keys, a PRF is a natural primitive:

$$k_{\mathrm{enc}} \leftarrow \mathrm{HMAC}(k_{\mathrm{master}}, \text{"enc"})$$
$$k_{\mathrm{mac}} \leftarrow \mathrm{HMAC}(k_{\mathrm{master}}, \text{"mac"})$$

} PRF security says derived keys are computationally indistinguishable from uniform

derived keys   master key   tag (just has to be unique)

This approach is used in TLS and IPsec to derive session keys durin session setup

↳ General paradigm is the "expand" step in hash-based key-derivation (HKDF — RFC 5869)

↳ Consists of two procedures:

- <u>Extract</u>: derive a master key from entropy source (e.g., a user password)
- <u>Expand</u>: derive sub-keys from the master key

Both steps rely on HMAC

How do we combine confidentiality and integrity?

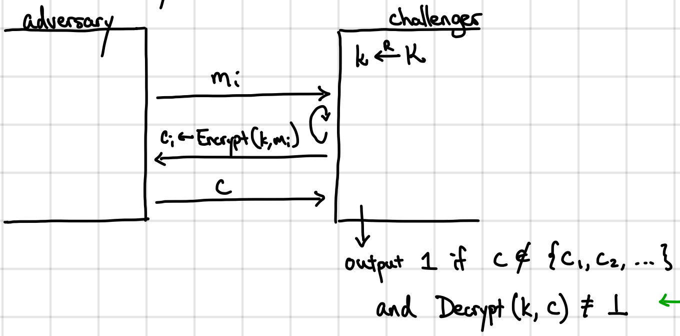
↳ Systems with both guarantees are called authenticated encryption schemes – gold standard for symmetric encryption

Two natural options:

1. Encrypt-then MAC (TLS 1.2+, IPsec) ← guaranteed to be secure if we instantiate using CPA-secure encryption and a secure MAC
2. MAC-then-encrypt (SSL 3.0/TLS 1.0, 802.11i) ← as we will see, not always secure

Definition. An encryption scheme $\Pi_{SE} = (\text{Encrypt}, \text{Decrypt})$ is an authenticated encryption scheme if it satisfies the following two properties:

- CPA security [confidentiality]
- ciphertext integrity [integrity]


adversary
challenger
$k \xleftarrow{R} \mathcal{K}$
m_i
$c_i \leftarrow \text{Encrypt}(k, m_i)$
c
output 1 if $c \notin \{c_1, c_2, \ldots\}$
and $\text{Decrypt}(k, c) \neq \perp$

special symbol $\perp$ to denote invalid ciphertext

Define $\text{CIAdv}[A, \Pi_{SE}]$ to be the probability that output of above experiment is 1. The scheme Π_{SE} satisfies ciphertext integrity if for all efficient adversaries A,

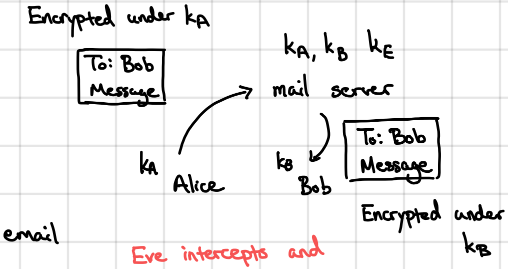
$$\text{CIAdv}[A, \Pi_{SE}] = \text{negl}(\lambda)$$

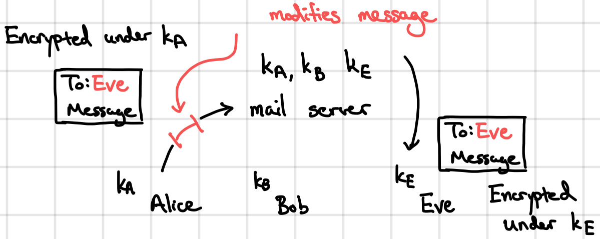
↑ security parameter determines key length

Ciphertext integrity says adversary cannot come up with a new ciphertext: only ciphertexts it can generate are those that are already valid. Why do we want this property?

Consider the following active attack scenario:

- Each user shares a key with a mail server
- To send mail, user encrypts contents and send to mail server
- Mail server decrypts the email, re-encrypts it under recipient's key and delivers email





If Eve is able to tamper with the encrypted message, then she is able to learn the encrypted contents (even if the scheme is CPA-secure)

↳ More broadly, an adversary can tamper and inject ciphertexts into a system and observe the user's behavior to learn information about the decrypted values – against active attackers, we need stronger notion of security