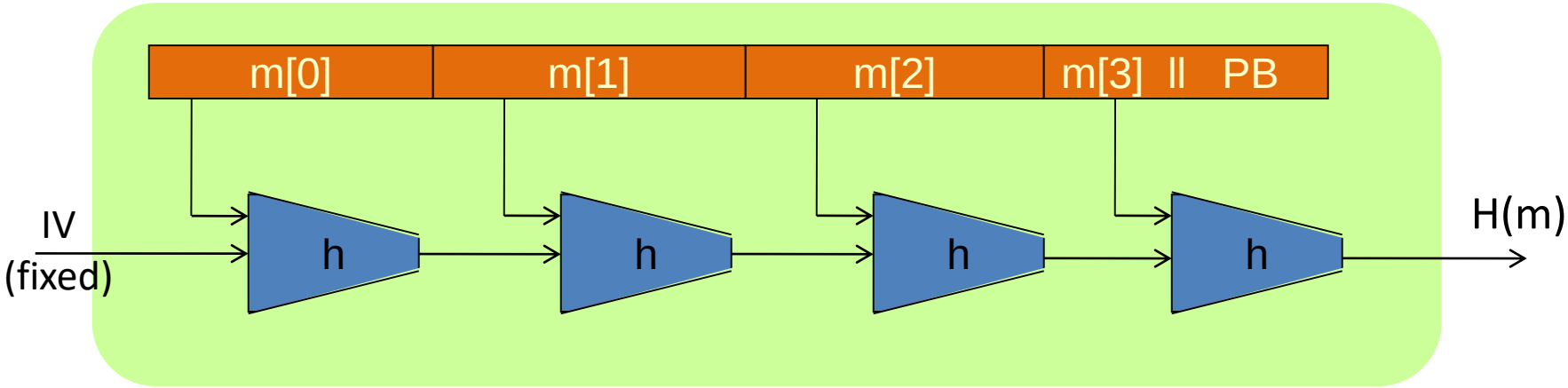
Collision resistance

Collision resistance

HMAC:
a MAC from SHA-256

The Merkle-Damgard iterated construction



Thm: h collision resistant ⇒ H collision resistant

Can we use H(.) to directly build a MAC?

MAC from a Merkle-Damgard Hash Function

H: $X^{\leq L} \longrightarrow T$ a C.R. Merkle-Damgard Hash Function

<u>Attempt #1</u>: **S(k, m) = H(k || m)**

This MAC is insecure because:

- Given H(k || m) can compute H(w || k || m || PB) for any w.
- Given H(k || m) can compute H(k || m || w) for any w.
- Given H(k || m) can compute H(k || m || PB || w) for any w.
- Anyone can compute H(k || m) for any m.

Standardized method: HMAC (Hash-MAC)

Most widely used MAC on the Internet.

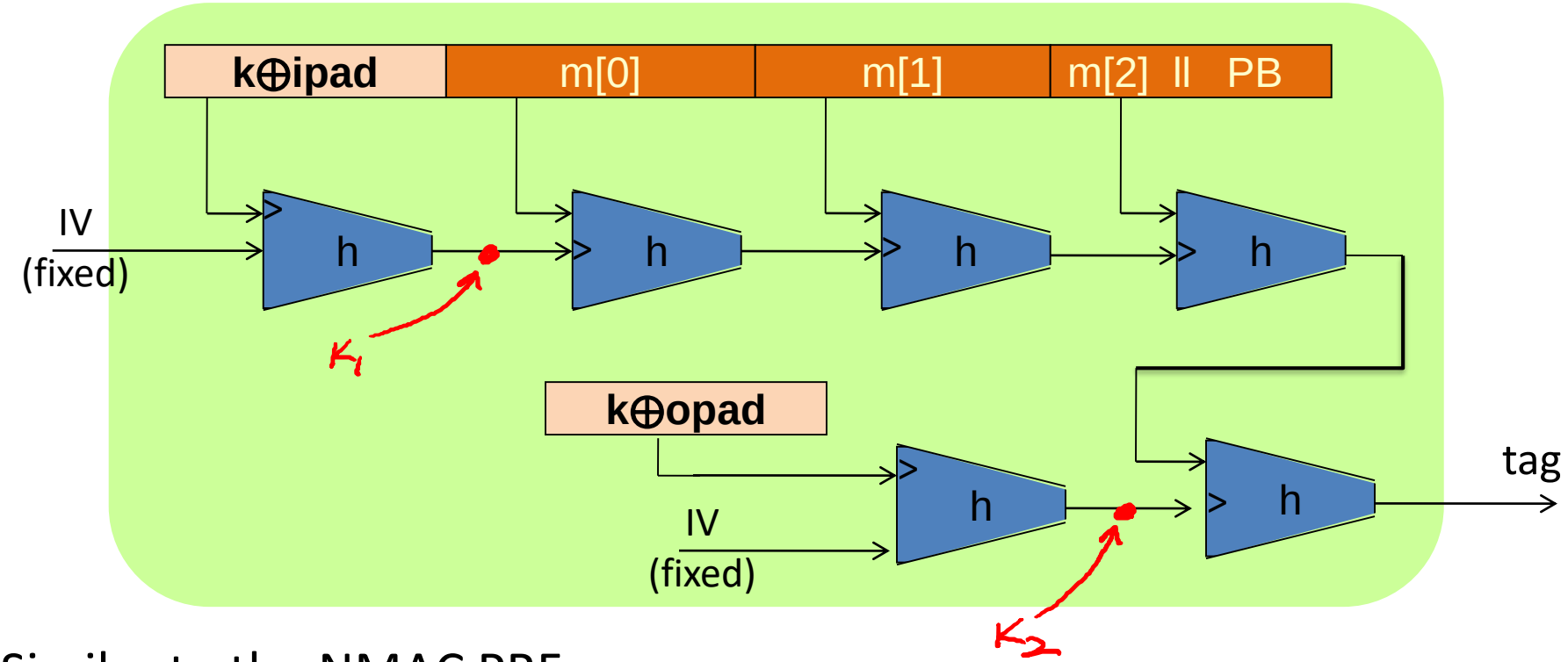
H: hash function.

example: SHA-256 ; output is 256 bits

Building a MAC out of a hash function:

HMAC: $S(k, m) = H(\ k \oplus opad \ \| \ \mathbf{H(\ k \oplus ipad \ \| \ m\ )}\ )$

HMAC in pictures



Similar to the NMAC PRF.

main difference: the two keys k_1, k_2 are dependent

HMAC properties

Built from a black-box implementation of SHA-256.

HMAC is assumed to be a secure PRF

- Can be proven under certain PRF assumptions about h(.,.)
- Security bounds similar to NMAC
  - Need $q^2/|T|$ to be negligible ($q << |T|^{½}$)

In TLS: must support HMAC-SHA1-96

End of Segment

Collision resistance

Timing attacks on MAC verification

Warning: verification timing attacks [L'09]

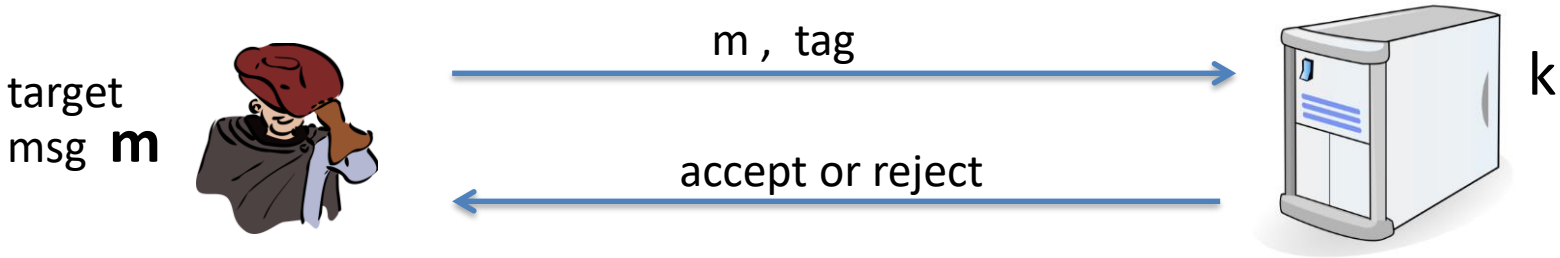
Example: Keyczar crypto library (Python) [simplified]

```
def Verify(key, msg, sig_bytes):
        return HMAC(key, msg) == sig_bytes
```

The problem: '==' implemented as a byte-by-byte comparison

- Comparator returns false when first inequality found

Warning: verification timing attacks [L'09]



Timing attack: to compute tag for target message m do:

Step 1: Query server with random tag

Step 2: Loop over all possible first bytes and query server.

stop when verification takes a little longer than in step 1

Step 3: repeat for all tag bytes until valid tag found

Defense #1

Make string comparator always take same time (Python) :

```
return false if  sig_bytes  has wrong length
result = 0
for x, y in zip( HMAC(key,msg) , sig_bytes):
    result = result | (ord(x) ^ ord(y))
return result == 0
```

Can be difficult to ensure due to optimizing compiler.

Defense #2

Make string comparator always take same time (Python) :

```
def Verify(key, msg, sig_bytes):
    mac = HMAC(key, msg)
    return HMAC(key, mac) == HMAC(key, sig_bytes)
```

Attacker doesn't know values being compared

End of Segment