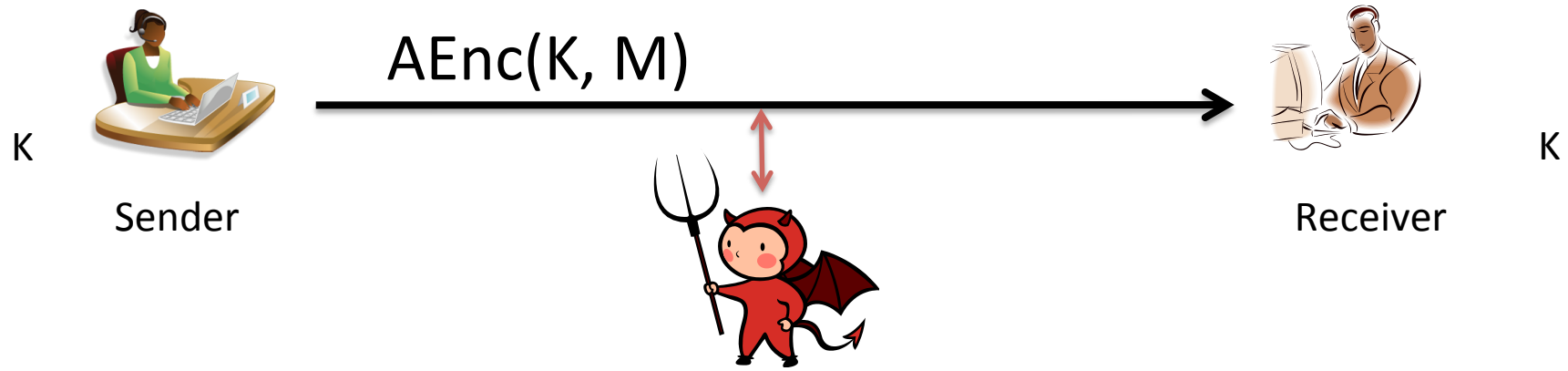
Today in Cryptography (5830)

Authenticated encryption
Using the same key
Authenticated encryption w/ associated data (AEAD)
Hash functions
HMAC
Encrypt-then-HMAC

Authenticated encryption is secure encryption



AEnc(K,M) outputs ciphertext
ADec(K,C) outputs message or error (invalid ciphertext)

Correctness: ADec(K,AEnc(K,M)) = M always

Security goals for encryption

1. **Confidentiality**:

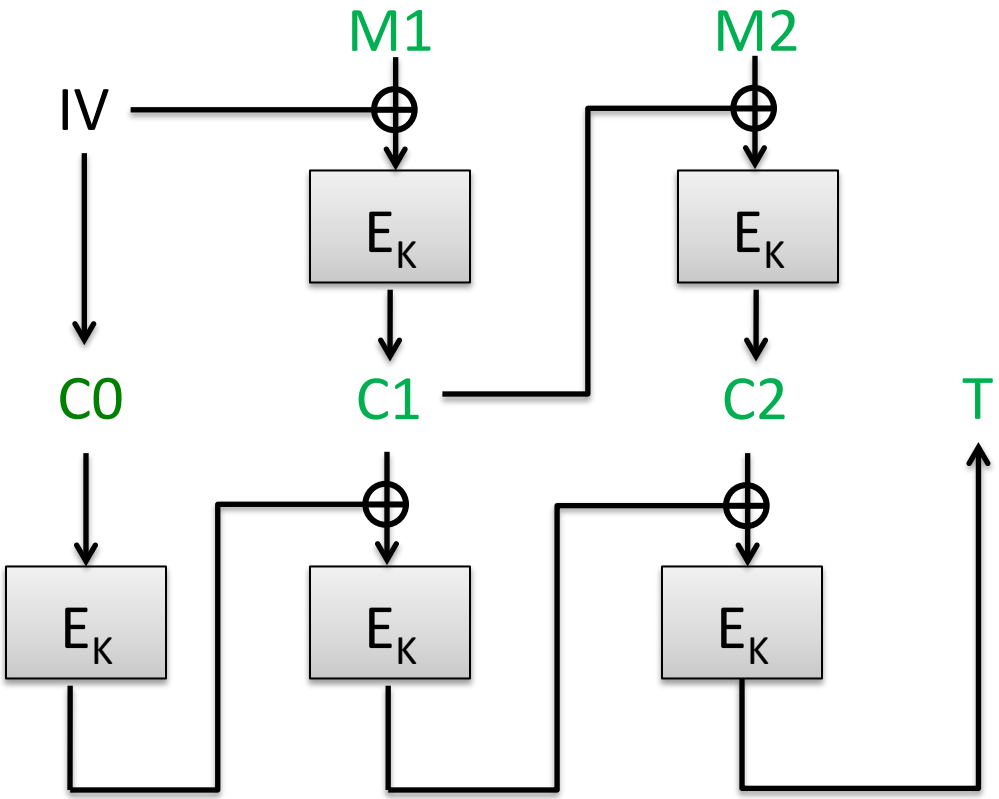
   No information about plaintexts should leak to any computationally-bound adversary

2. **Authenticity:**

   No adversary should be able to force recipient to accept a ciphertext not sent by sender

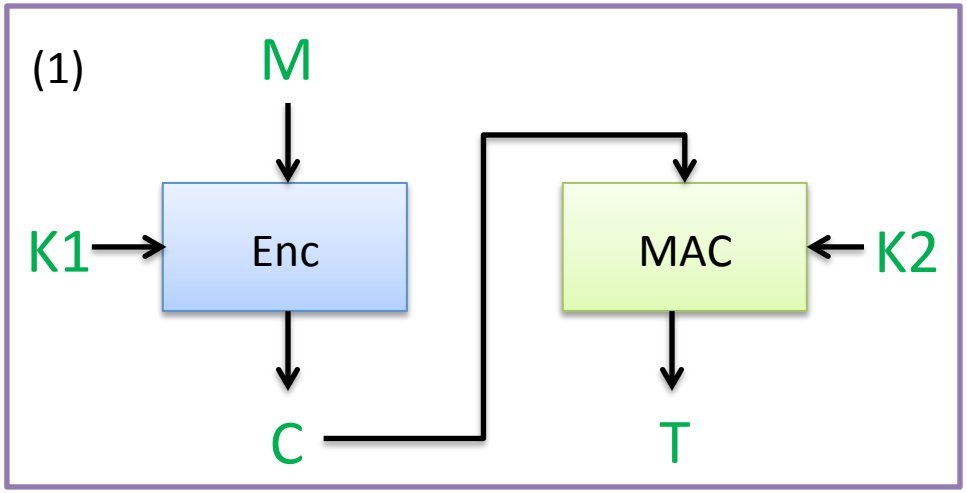
Key separation is critical.

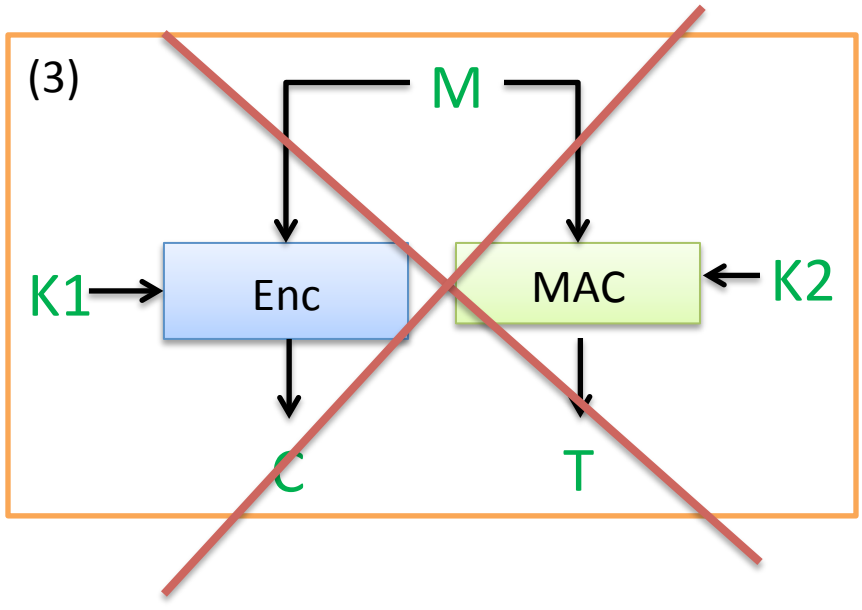
Using same key with CBC-Mode + CBC-MAC:

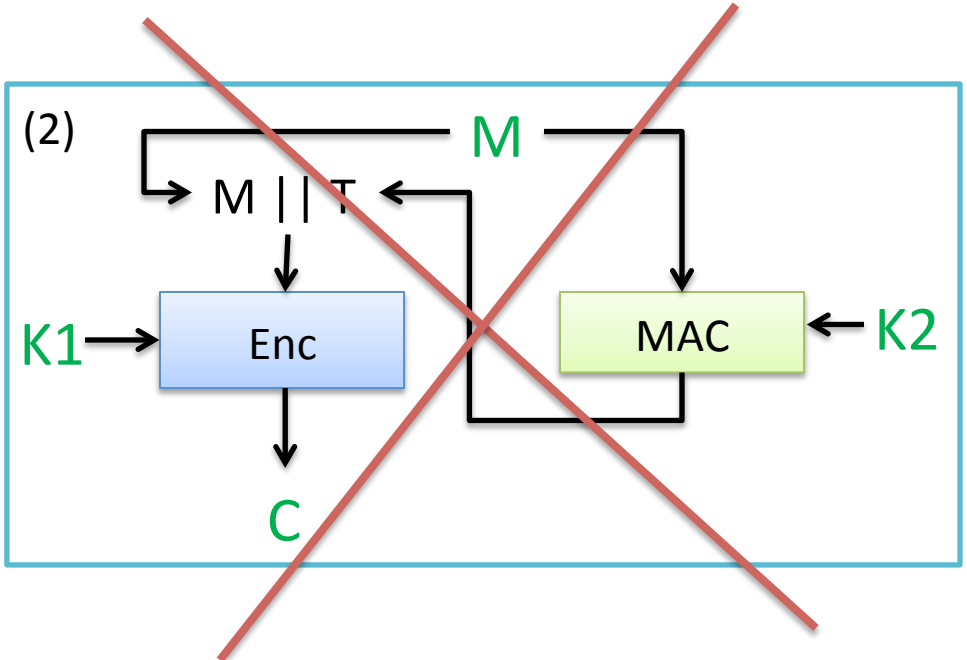


Build a new scheme from encryption mode and MAC
Use K1 for Enc and K2 for MAC

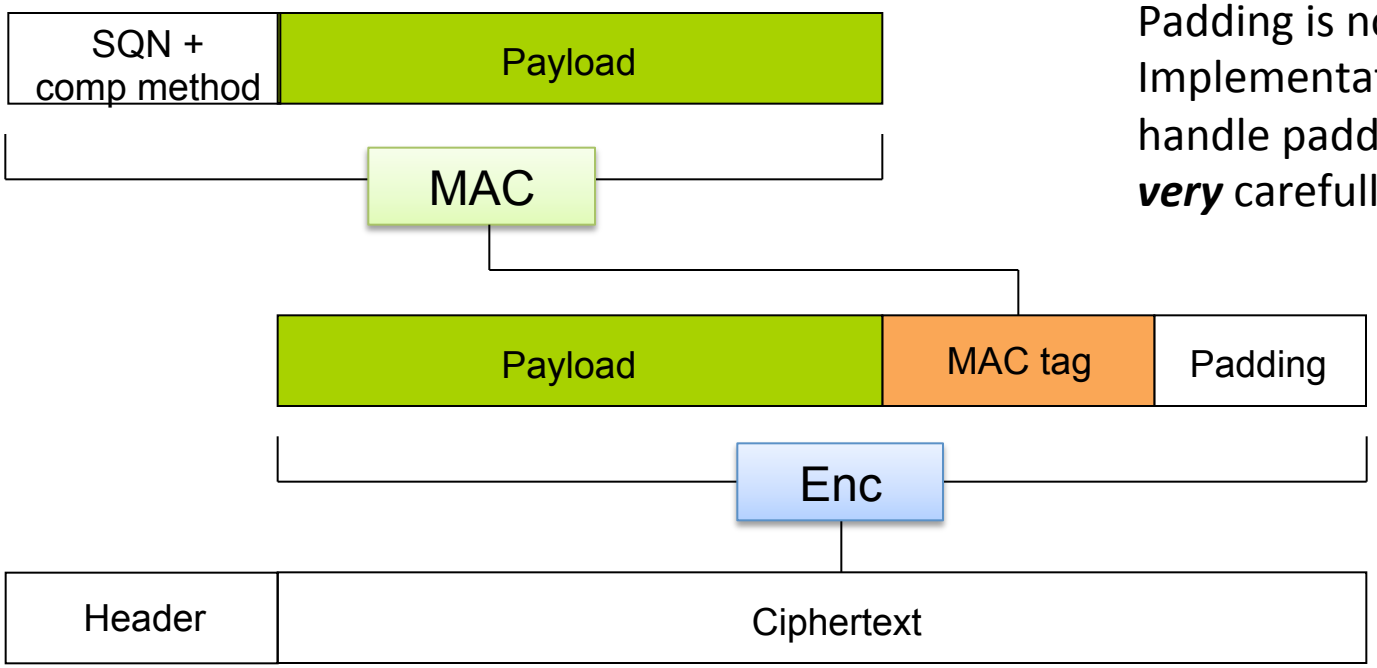
Several ways to combine:
(1) encrypt-then-mac
(2) mac-then-encrypt
(3) encrypt-and-mac







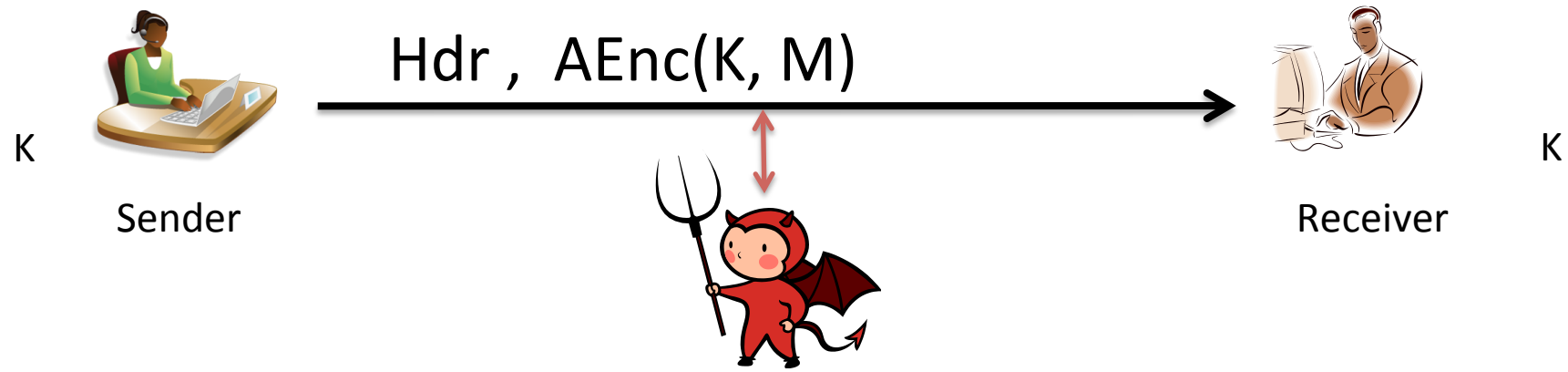
TLS record protocol: MAC-Encode-Encrypt (MEE)



Padding is not MAC'd. Implementations must handle padding checks ***very*** carefully.

MAC	HMAC-MD5, HMAC-SHA1, HMAC-SHA256
Encrypt	CBC-AES128, CBC-AES256, CBC-3DES, RC4-128

AEAD: Authenticated Encryption with Associated Data

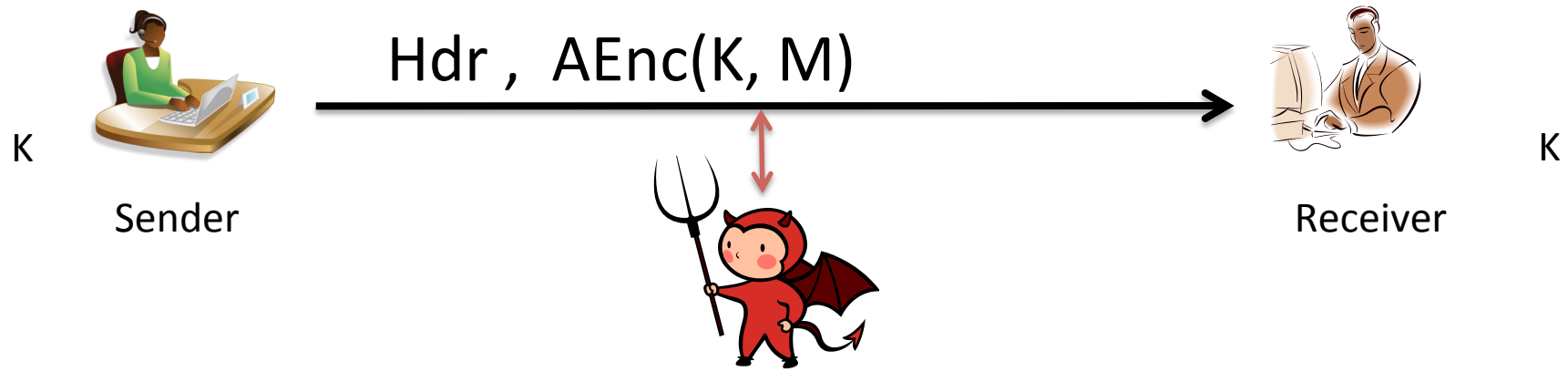


AEAD-Enc(K,AD,M) outputs ciphertext
AEAD-Dec(K,AD,C) outputs message or error (invalid C)

Correctness: AEAD-Dec(K,AD,AEAD-Enc(K,AD,M)) = M

Security:
(1) Confidentiality for M
(2) Authenticity for AD and M

AEAD: Authenticated Encryption with Associated Data



Extending Encrypt-then-MAC to be AEAD:

<u>AEAD-EtM(K,AD,M):</u>

K_1 , K_2 <- K // Split K into two keys *securely*

C <- Enc(K_1,M)

T <- MAC(K_2 , AD || C) // Encoding of || *must be unambiguous*

Return C,T

Some other AEAD schemes

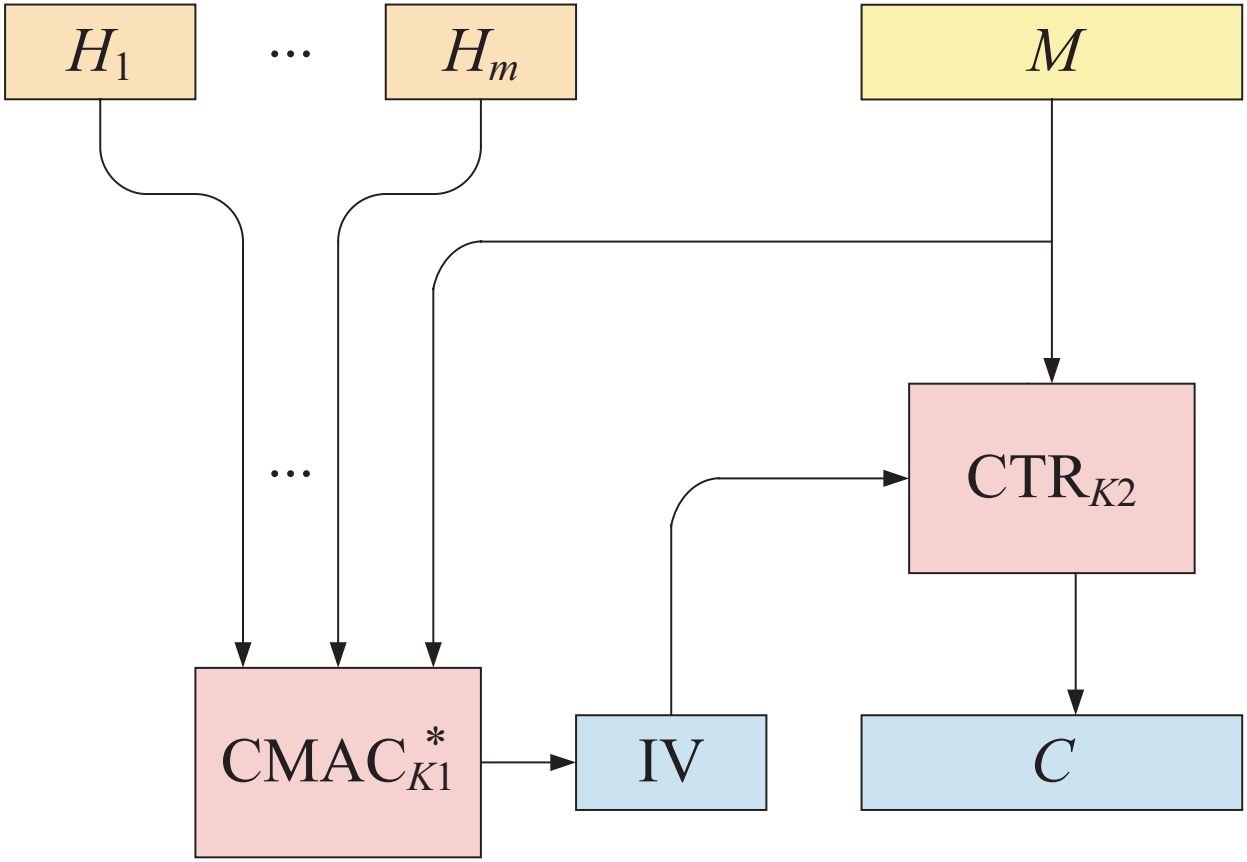
Attack	Inventors	Notes
OCB (Offset Codebook)	Rogaway	One-pass
GCM (Galios Counter Mode)	McGrew, Viega	CTR mode plus specialized MAC
CWC	Kohno, Viega, Whiting	CTR mode plus Carter-Wegman MAC
CCM	Housley, Ferguson, Whiting	CTR mode plus CBC-MAC
EAX	Wagner, Bellare, Rogaway	CTR mode plus OMAC

Other AEAD concepts

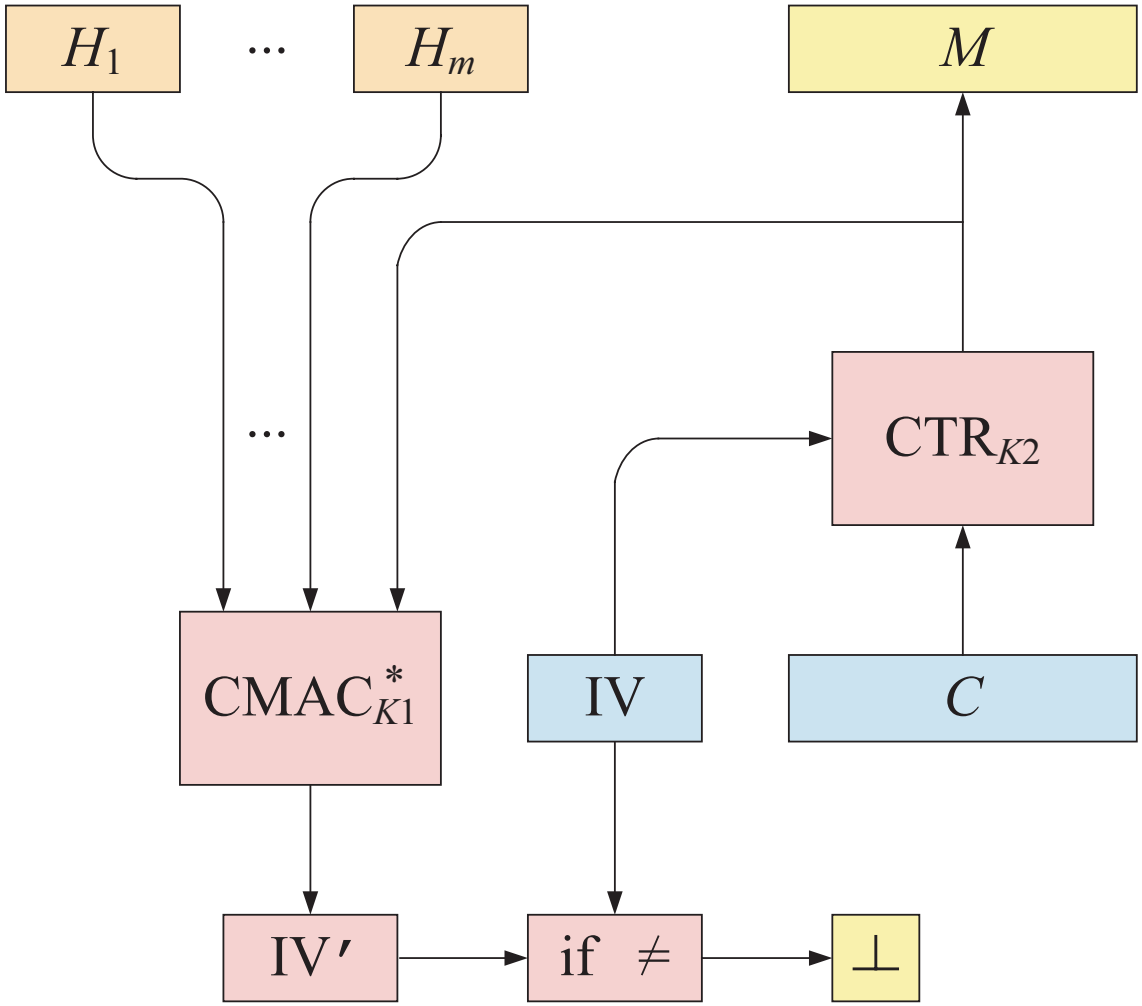
1. **Stateful versus randomized**
   - Stateful uses counter instead of IV. Must be careful. Recipient maintains state.

2. **Robust AEAD**
   - Not all security lost when IV (or counter) repeats
     - Bugs may arise which cause repeats
   - Give up on hiding plaintext repetition. All other security goals are the same

Synthetic IV (SIV) mode encryption

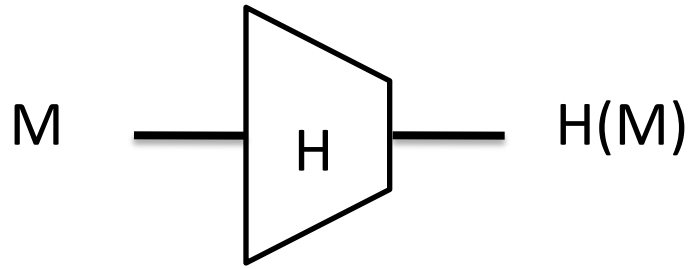


Synthetic IV (SIV) mode decryption



Cryptographic hash functions

A cryptographich hash function H maps arbitrary bit string to fixed length string of size m



MD5: m = 128 bits
SHA-1: m = 160 bits
SHA-256: m = 256 bits

Some security goals:

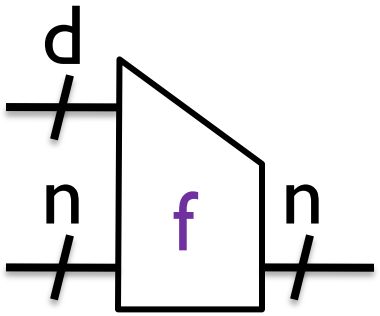
- collision resistance: can't find M != M' such that H(M) = H(M')
- preimage resistance: given H(M), can't find M
- second-preimage resistance: given H(M), can't find M' s.t. H(M') = H(M)
- Behave like a *public*, random function. Sometimes called random oracle model (ROM)

Pseudorandom functions vs. random oracles

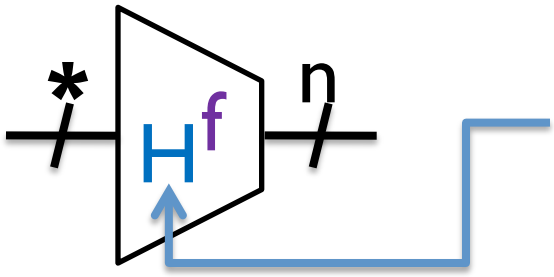
	Inputs	Security	Examples
PRF	Secret key, message	Indistinguishable from random function to any party without key	CBC-MAC HMAC
Random oracle (RO)	Message	Is a random function, but one that everyone can compute	SHA-256 SHA-512 SHA-3

Two-step design for hash functions

Compression Function

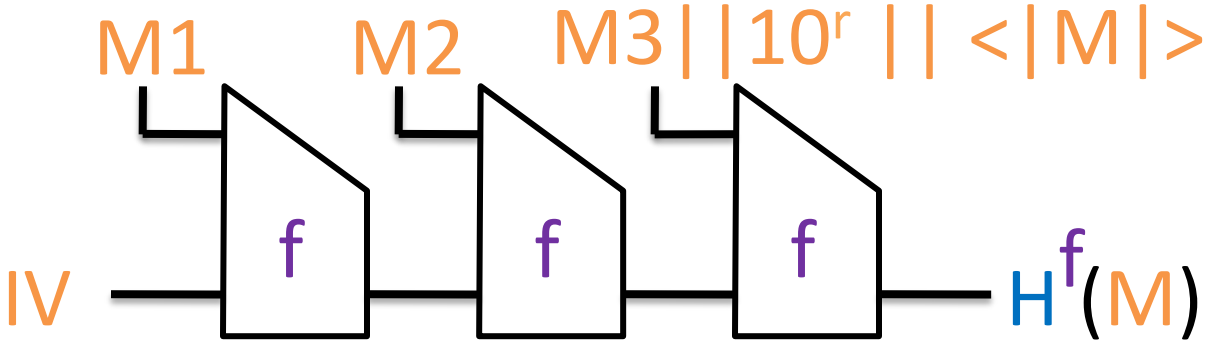


Hash Function



Domain extension transform

E.g., H = "Merkle-Damgard with strengthening"



Used in MD-x, SHA-1, SHA-256, …

Building compression functions

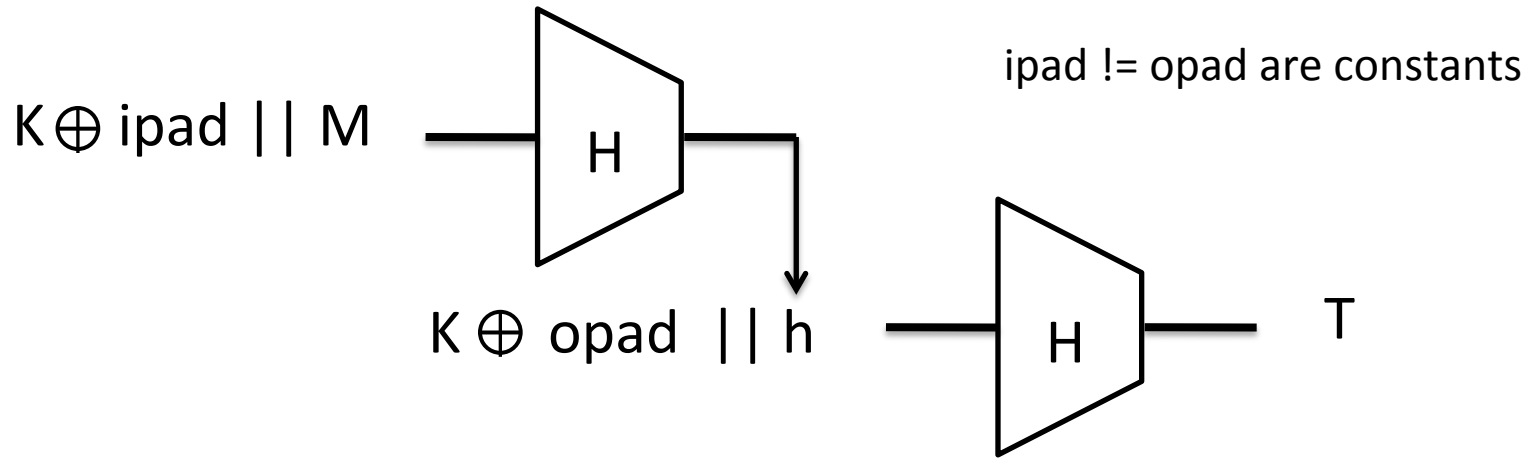
- Can build compression functions from suitable block ciphers

$$f(z,m) = E(m,z) \oplus z$$

- Can use AES, but security too low. Why?

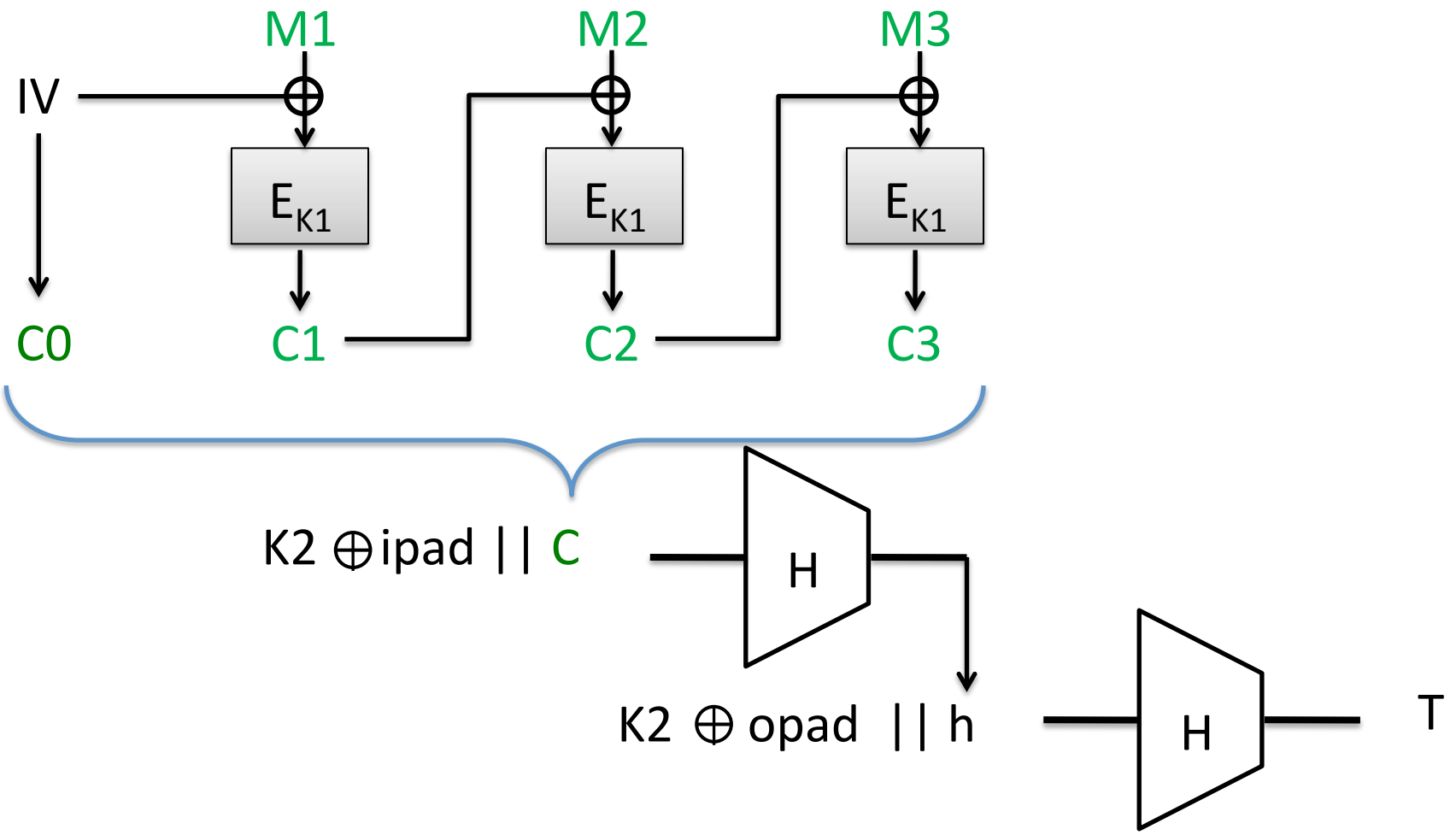
Building PRFs with hash functions: HMAC

Use a hash function H to build a MAC. K is a secret key



This is slight simplification, assuming |K| < d
(recall d is underlying message block length)

Encrypt-then-MAC with CBC and HMAC



Ciphertext is C,T