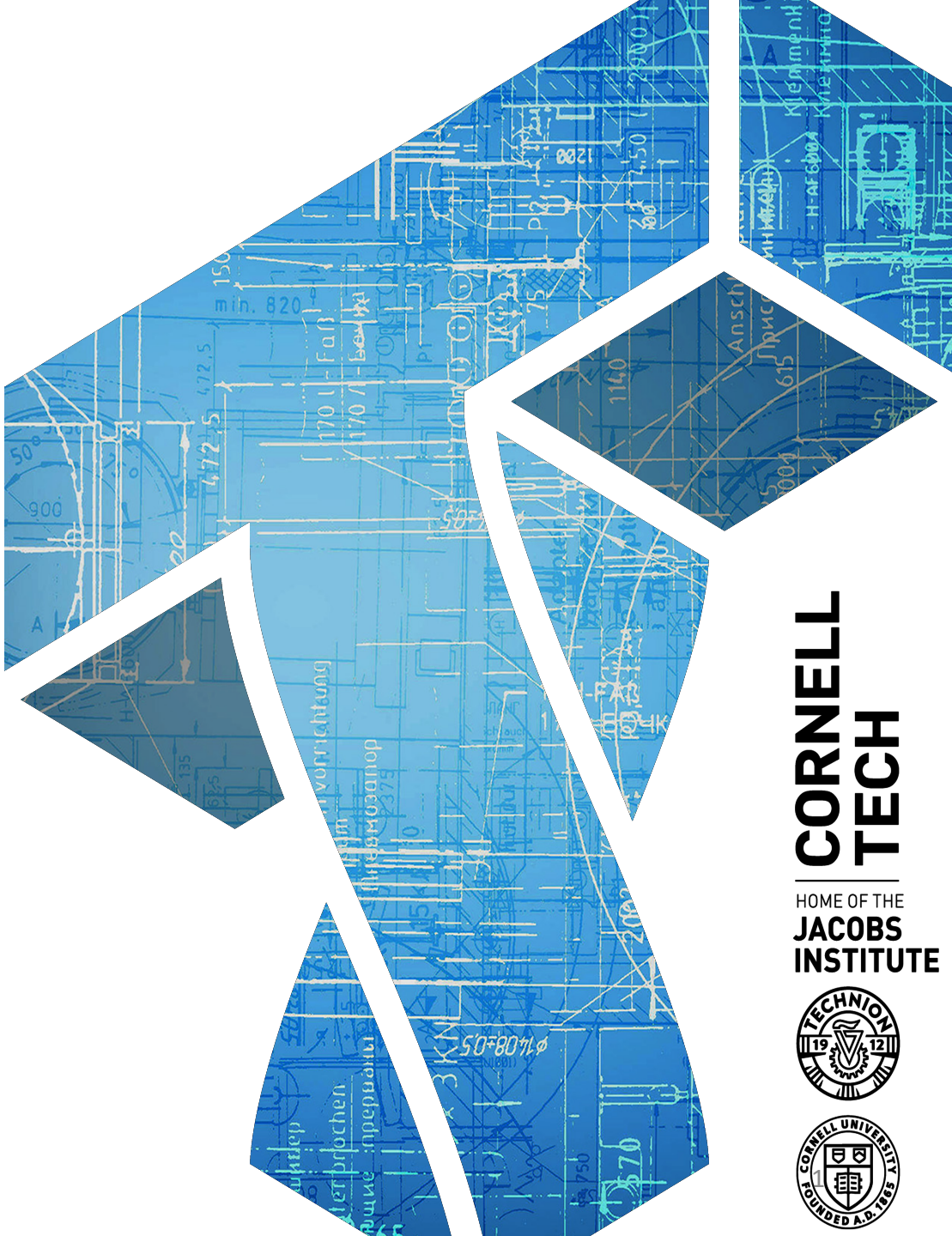
CS 5830
Cryptography

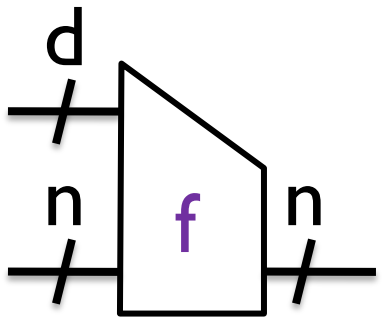
Instructor: Tom Ristenpart
TAs: Yan Ji, Sanketh Menda



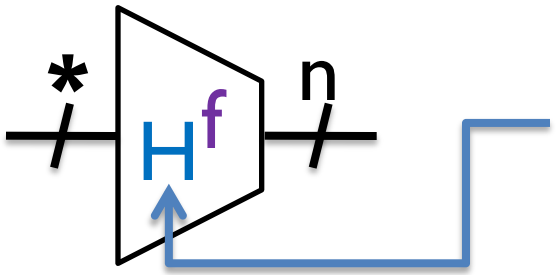
CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Two-step design for hash functions

Compression Function

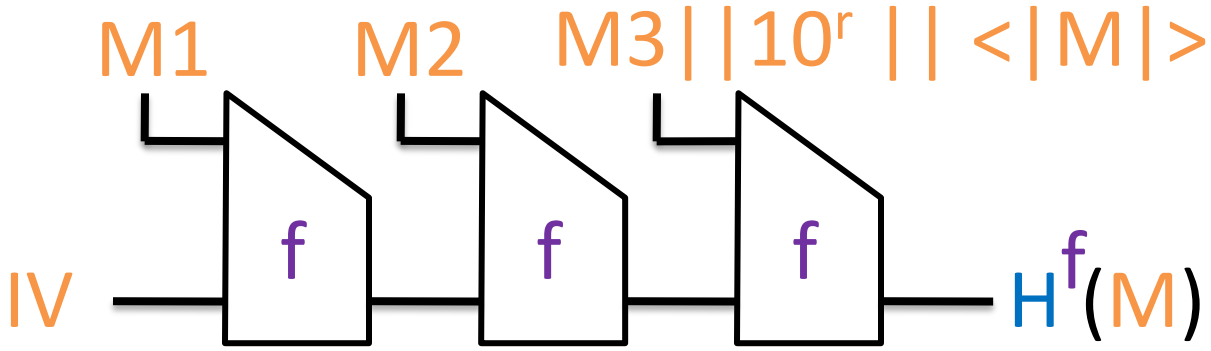


Hash Function



Domain extension transform

Domain extension called “Merkle-Damgard with strengthening”

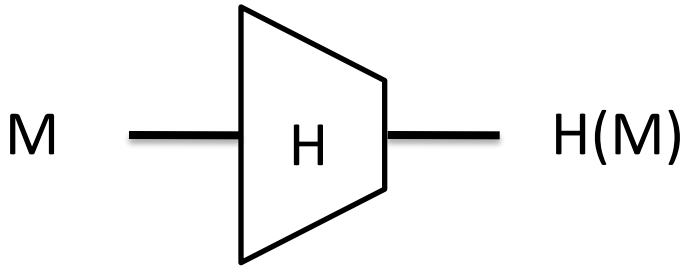


Used in MD-x, SHA-1, SHA-256, …

IV is a fixed constant. Not randomly chosen!

Cryptographic hash functions

A function H that maps arbitrary bit string to
fixed length string of size n



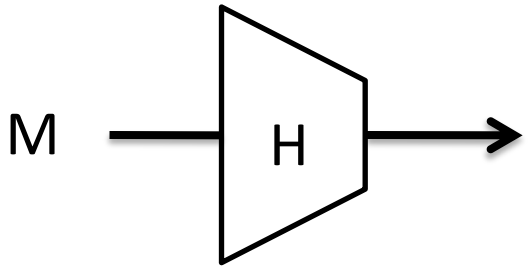
MD5: n = 128 bits
SHA-1: n = 160 bits
SHA-256: n = 256 bits

Collision resistance:
No computationally efficient adversary can find $M \neq M'$ such that $H(M) = H(M')$

Preimage resistance:
Given H(M) hard to recover M, if M has sufficient entropy

PRF when keyed appropriately

How can we build a PRF from a hash function?



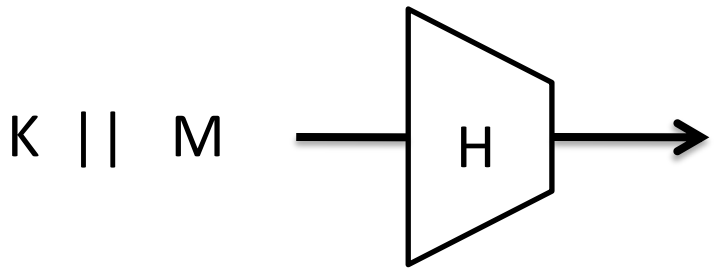
Hash functions are public, no secret key

PRFs / MACs need secret key

How can we add secret key to hash?

What's wrong with this PRF construction?

Assume H is a Merkle-Damgard iterated hash function, define F(K,M) = H(K||M)

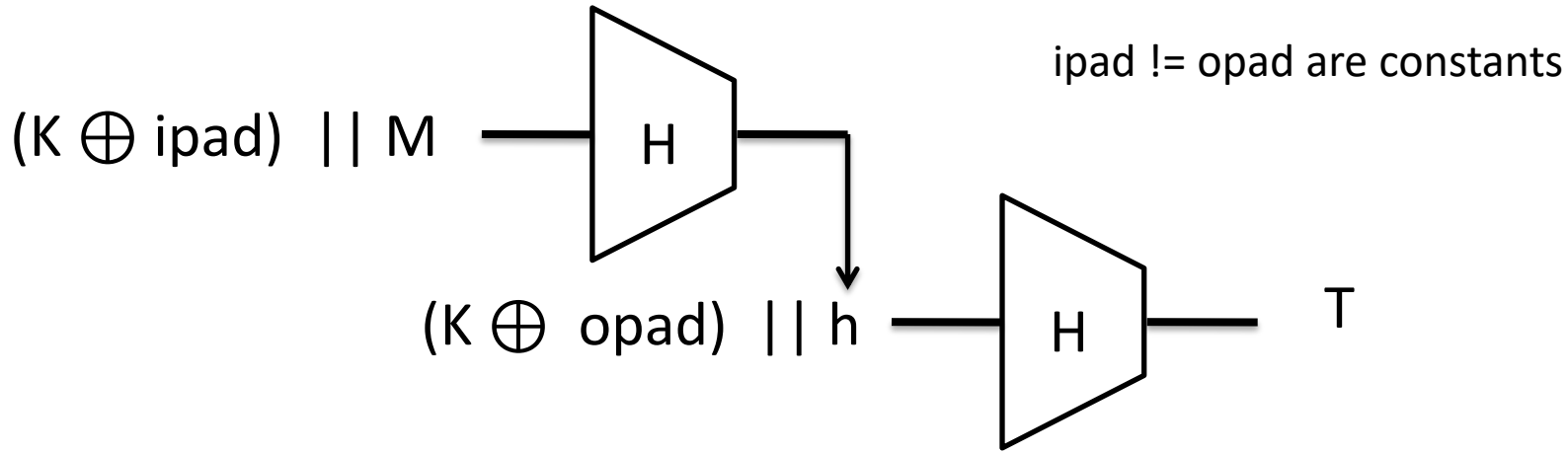


Length-extension attacks:

Given F(K,M) can compute F(K,M||S) for attacker-chosen suffix S

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| less than block length of H

Otherwise, hash key first using H

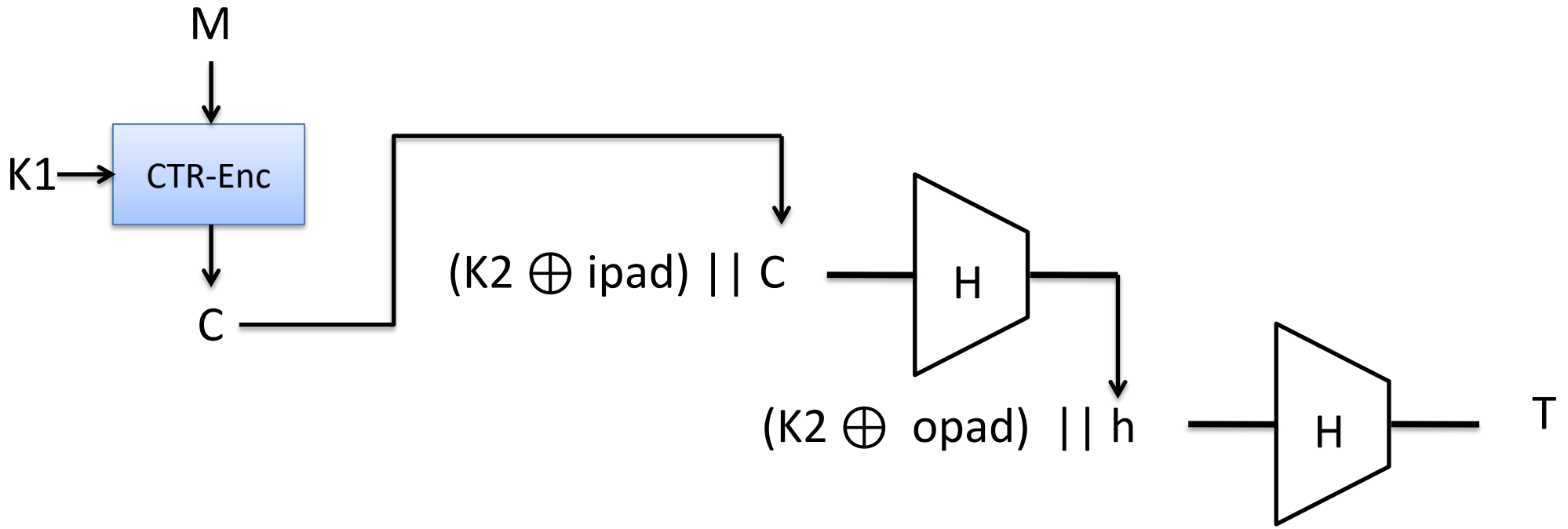
Either way, append zeros to K in order to get one block of data

HMAC-SHA-1, HMAC-SHA-256, etc.

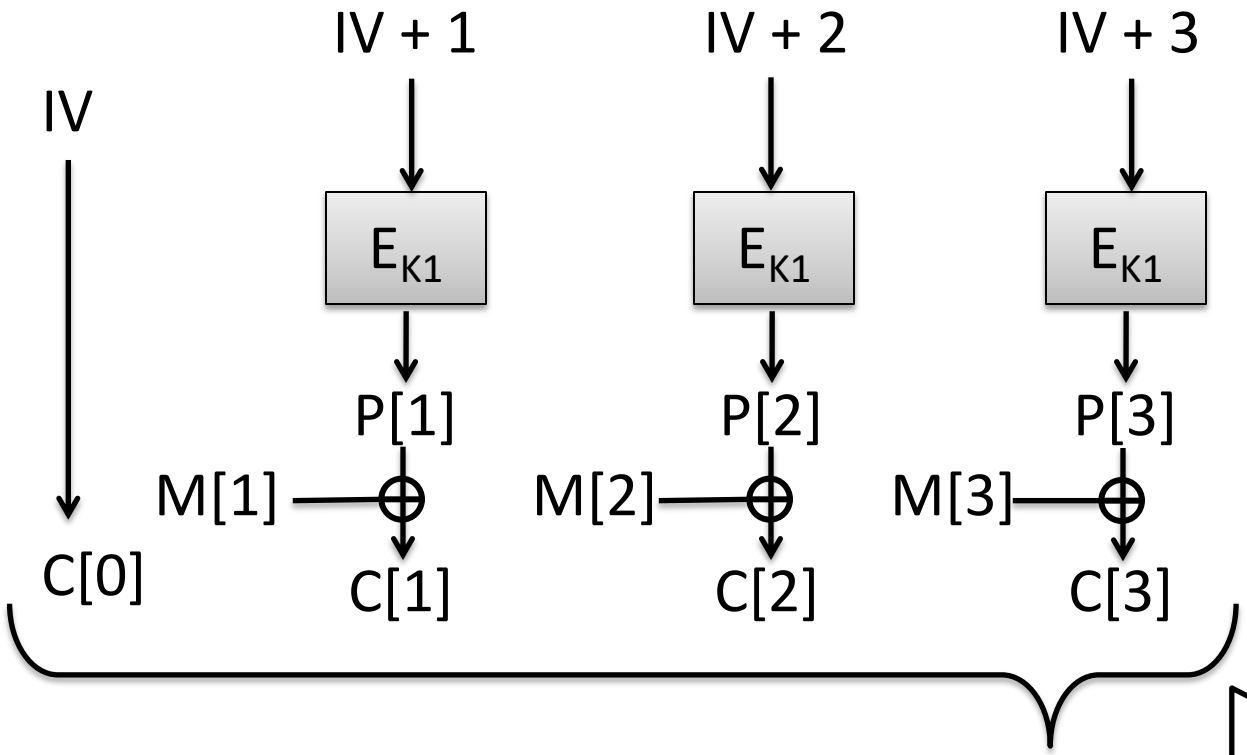
Encrypt-then-HMAC

Combine CTR mode encryption with HMAC to build authenticated encryption

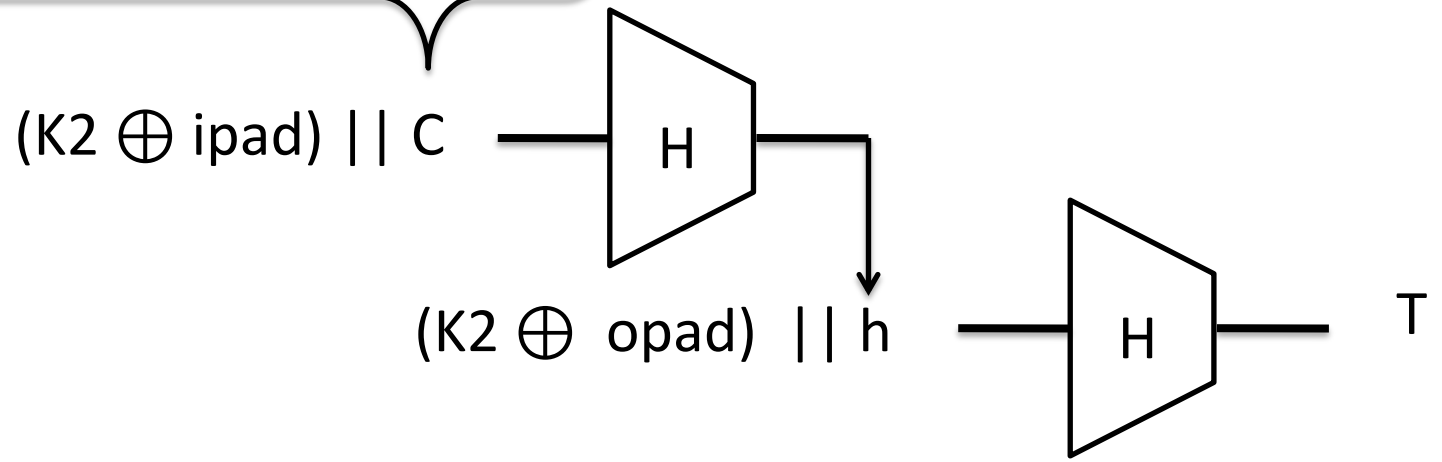
K1, K2 derived from single key K using hash-based KDF (key-derivation function)



Encrypt-then-MAC with CTR & HMAC



<u>Decrypt(K,(C,T))</u>
K1, K2 <- KDF(K)
T' <- HMAC(K2,C)
If T' ≠ T then
  Return ⊥
M <- CTR-Dec(K1,C)
Return M



HKDF

Hash-based key derivation function (HKDF)

- Derive more key material from one key
- Extract key from non-uniform key material

HKDF(L,K,salt,info):

prk <- HMAC(salt,K)

K_0 <- empty byte string

m <- ceil(L/hashLen)

For i = 1 to m

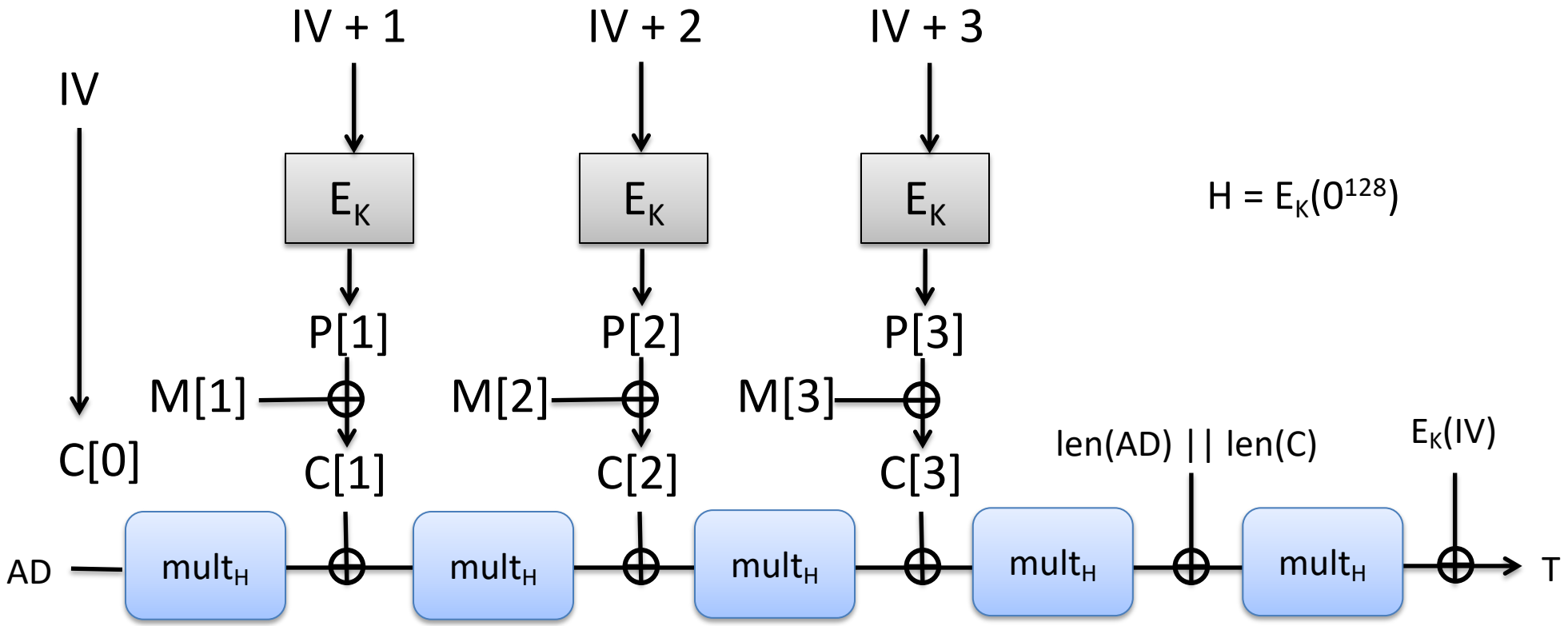
  K_i <- HMAC(prk,K_{i-1} || info || <i +1>)

Return $\text{truncate}_L(K_1 \,||\, \ldots \,||\, K_m)$

AES-GCM (Galois Counter Mode)

- Probably most widely used AEAD scheme
  - Encryption takes key, message, associated data (AD)
  - Designed by Viega and McGrew in 2004
  - Uses 96-bit IV (nonce)
- Combines CTR-mode using AES with Carter-Wegman style MAC
  - Uses GHASH, specialized almost-universal hash
    - Let $H = E_K(0^{128})$
    - Define $mult_H$ to be multiplication by H in $GF(2^{128})$ (Galois Field)
    - Very fast

AES-GCM AEAD (Galois Counter Mode)



Tag computation is equivalent to evaluating polynomial in H:

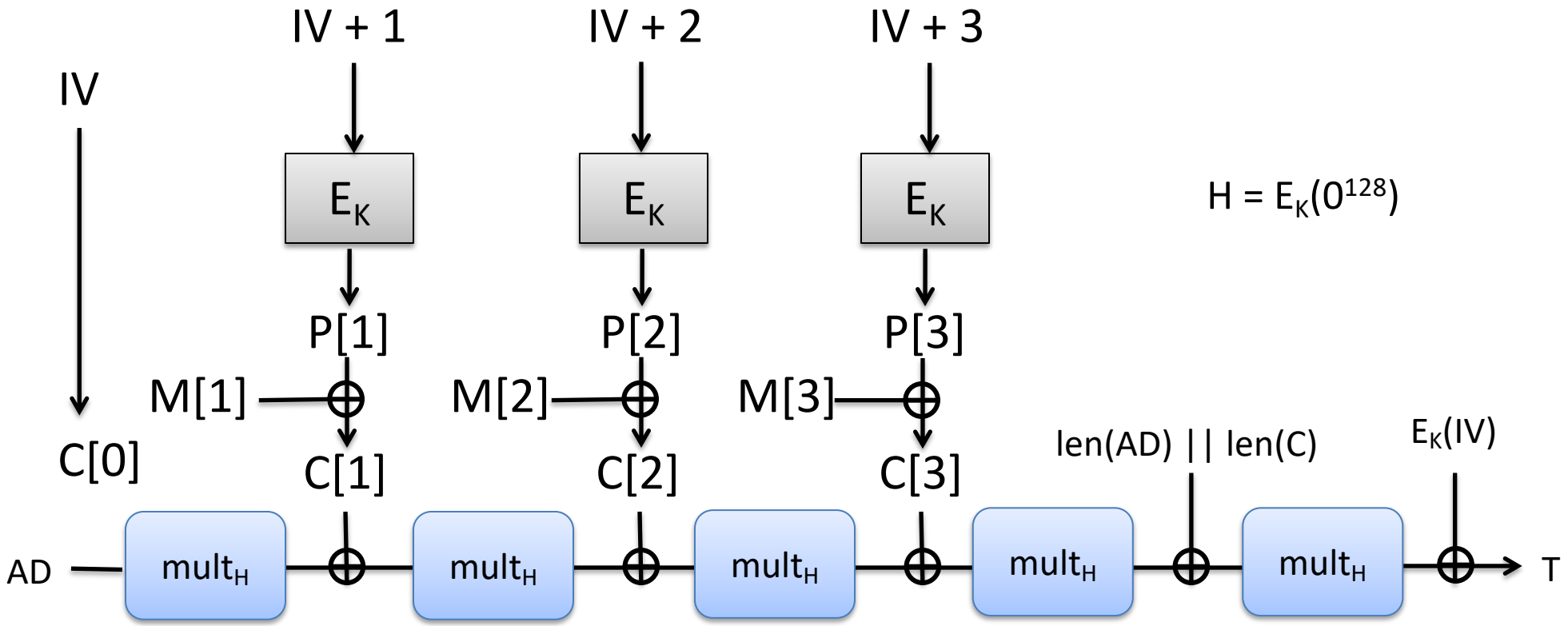
$$T = AD * H^5 + C[1] * H^4 + C[2] * H^3 + C[3] * H^2 + L * H + E_K(IV)$$

Where * is multiplication in $GF(2^{128})$ and + is xor

Further security properties for AEAD

- We've seen two security goals for symmetric encryption:
  - Confidentiality (IND-CPA)
  - Integrity (CTXT)
- These assume:
  - Good randomness (IV is freshly chosen, unlikely to repeat)
  - Symmetric keys are honestly chosen and secret
- In rarer threat models, both assumptions may fail

Misuse-resistant AEAD



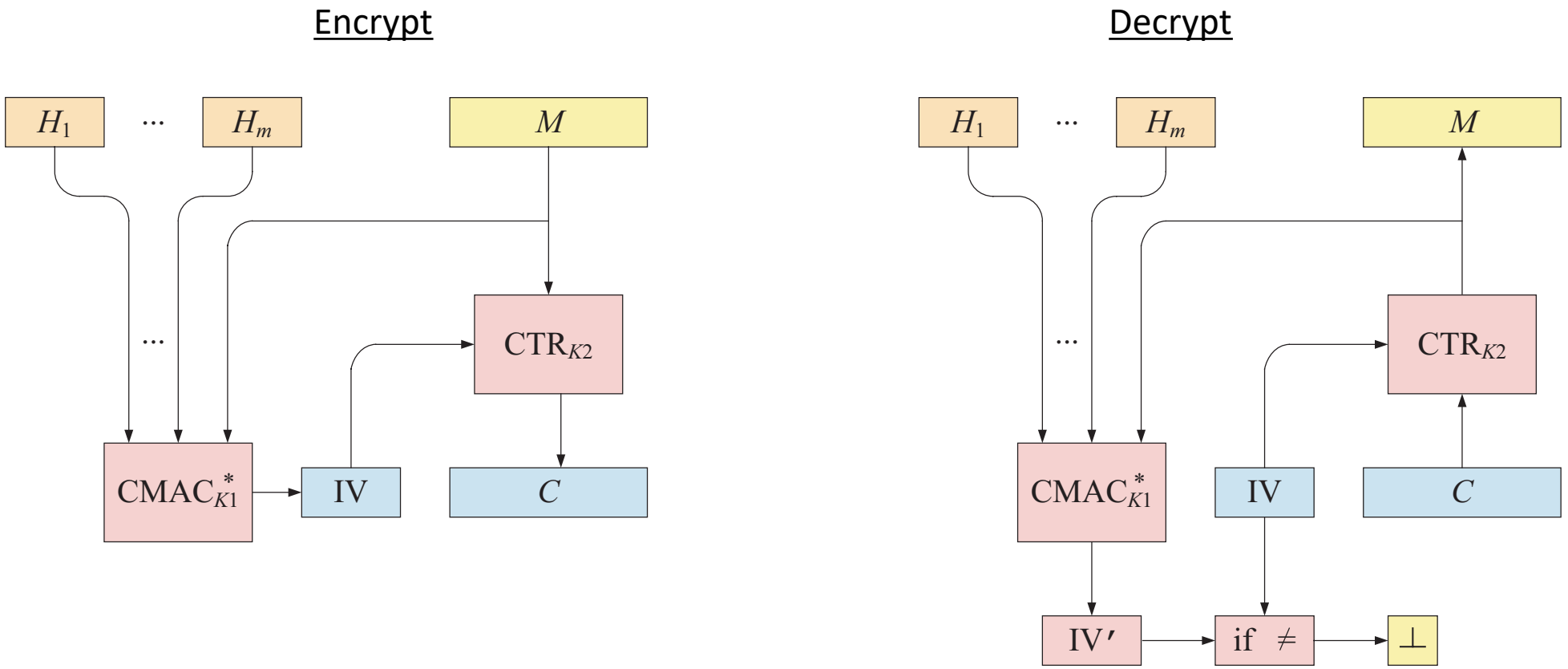
What happens when IV repeats?

Reveal message bits (XOR of two messages)
Can recover H (Joux's "forbidden attack")

Misuse-resistant AEAD

- Misuse-resistant AEAD provides better “fallback” security in case IVs are bad
  - Only plaintext repetition repeats
  - Can’t forge ciphertexts even after seeing ciphertexts with repeat IV
- [Rogaway & Shrimpton 2006] formalized, and suggested a mode Synthetic IV

Synthetic IV mode



AES-GCM-SIV is SIV but using Ghash-based Carter-Wegman MAC

Further security properties for AEAD

- We've seen two security goals for symmetric encryption:
  - Confidentiality (IND-CPA)
  - Integrity (CTXT)
- These assume:
  - Good randomness (IV is freshly chosen, unlikely to repeat)
  - Symmetric keys are honestly chosen and secret
- In rarer threat models, both assumptions may fail

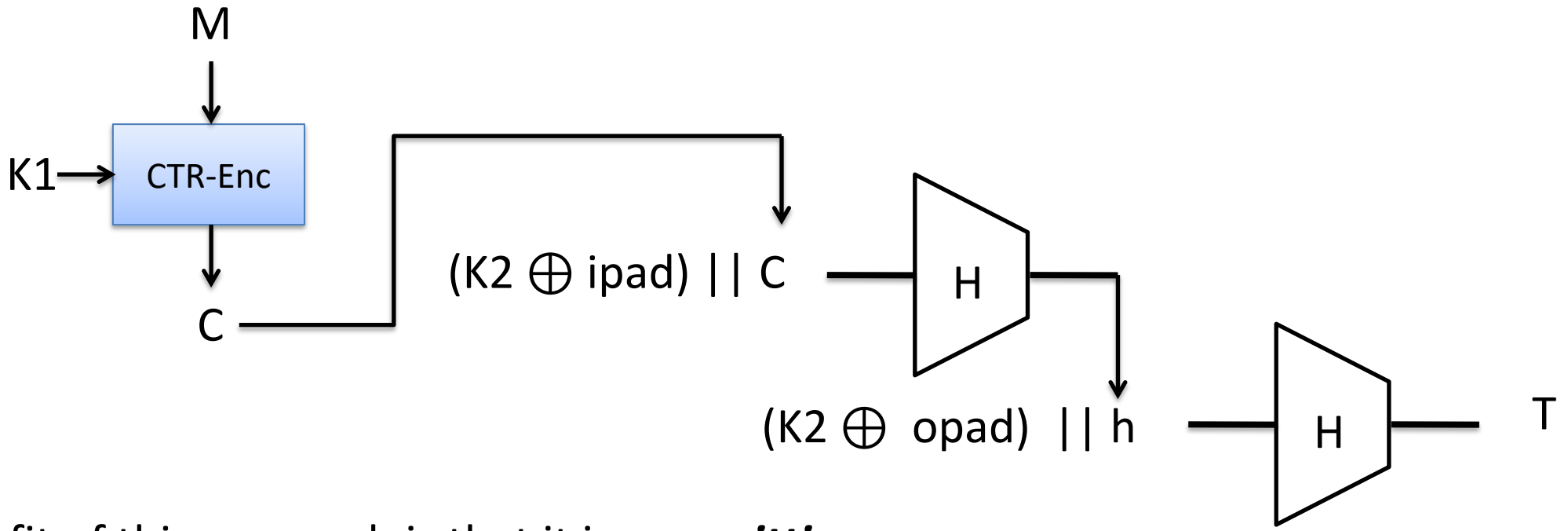
Key commitment

- Can an adversary come up with ciphertext that decrypts under two distinct adversarially-chosen keys?
  - AES-GCM answer is yes: universal-hash based MAC is not collision resistant for adversarially-chosen keys
- Weakness in AES-GCM exploited to break security of Facebook message franking scheme for abuse-reporting
  - [Dodis, Gubbs, Ristenpart, Woodage 2018]
- Recent Google paper shows some problems with not having key-commitment
  - [Albertini et al. 2022]
- Efforts underway to standardize key-committing AEAD

Encrypt-then-HMAC

Combine CTR mode encryption with HMAC to build authenticated encryption

K1, K2 derived from single key K using hash-based KDF (key-derivation function)



Benefit of this approach is that it is ***committing***

CR of H ensures that no efficient adversary can find malicious keys K ≠ K' and C,T such that Dec(K,C||T) and Dec(K',C||T) both succeed

See partitioning oracle paper (https://www.usenix.org/conference/usenixsecurity21/presentation/len)

Summary

- Cryptographic hashing provides one way to build good MACs (HMAC). Widely used
  - Universal hashing only provides useful properties for properly chosen, secret hash keys. Trivial to find collisions if key known
- AEAD schemes provide:
  - Confidentiality
  - Integrity
  - Possibly: IV misuse-resistance, key commitment
- Encrypt-then-MAC generic composition modes as well as custom versions like AES-GCM

Essentials on GF(2^n)

- GF(2^n) is Finite (Galios) Field over n-bit strings
- Represent point *a* equivalently as:
  - Polynomial $a(x) = a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$
  - Integer $a = \sum_{i=0}^{n-1} a_i 2^i$
  - Bit string $a = a_{n-1} \cdots a_1 a_0$

Essentials on GF(2^n)

- GF(2^n) is Finite (Galios) Field over n-bit strings
- Addition = bitwise xor of points (as bit strings)
- Multiplication:
  - fix primitive poly. For n = 128 use

$$x^{128} + x^7 + x^2 + x + 1$$

  - Multiple points *ac* by treating as polynomials, multiplying formal polys, and take remainder mod fixed prim poly