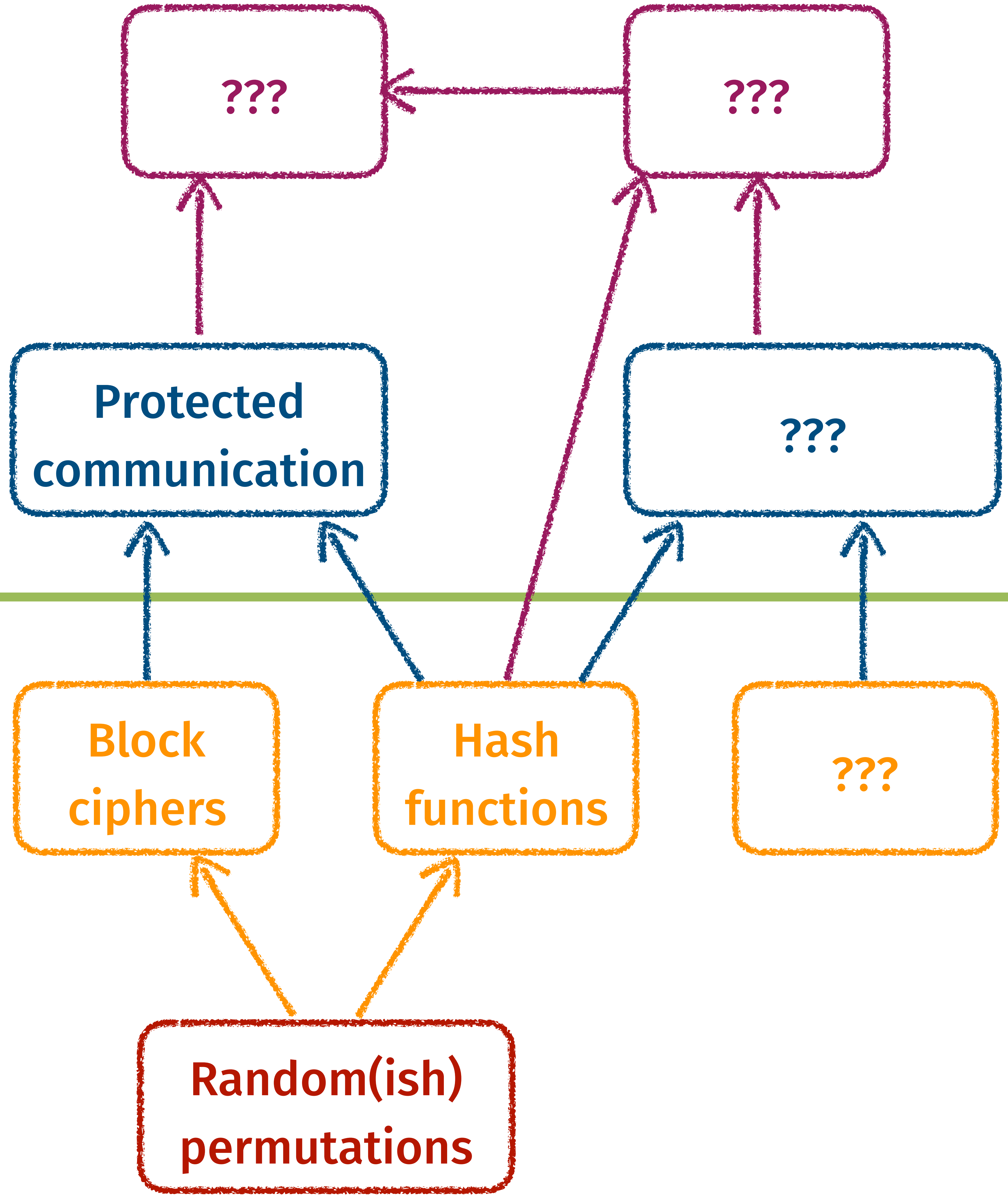
Lecture 11: HMAC and Authenticated Encryption

- Homework 6 will be posted today, due Monday 3/16 (after spring break)
- Weekly reading: The Block Cipher Companion, section 4.5
- (Non-class-related reminder: vote today!)

Crypto in this course

Elegant protocols

Utilitarian tools



Cryptographic Doom Principle

“If you have to perform any cryptographic operation before verifying the MAC on a message you’ve received, it will somehow inevitably lead to doom!”

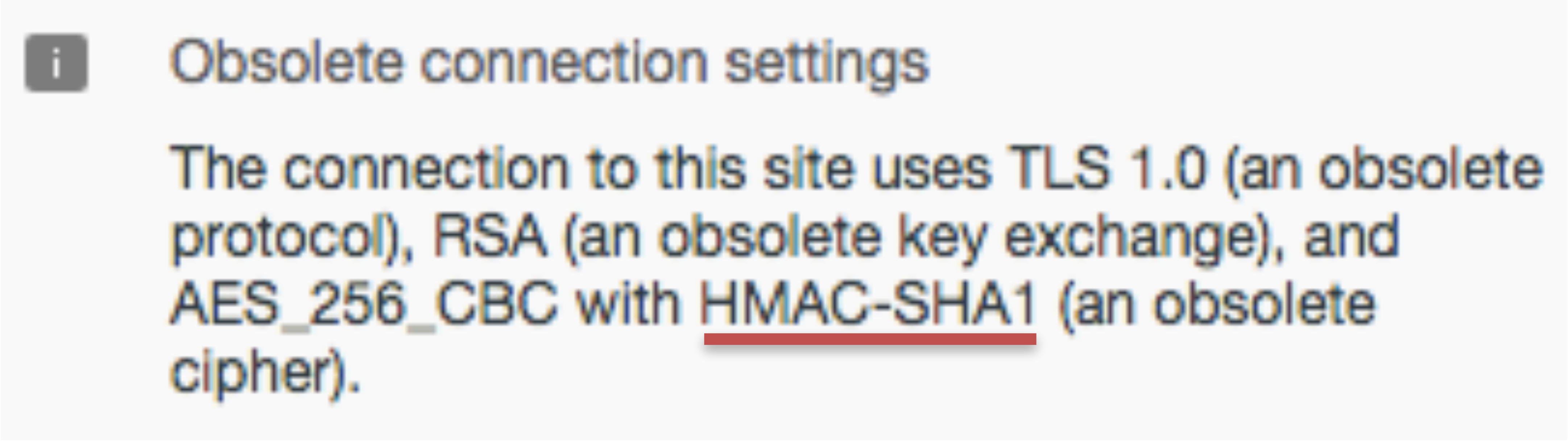
– Moxie Marlinspike

MAC Constructions

- Based on block ciphers

CBC-MAC → FMAC → CMAC → OMAC

- Based on hash functions (screenshot from bu.edu in 2017)



Obsolete connection settings

The connection to this site uses TLS 1.0 (an obsolete protocol), RSA (an obsolete key exchange), and AES_256_CBC with HMAC-SHA1 (an obsolete cipher).

Hash function = 1 public, infinite-size codebook

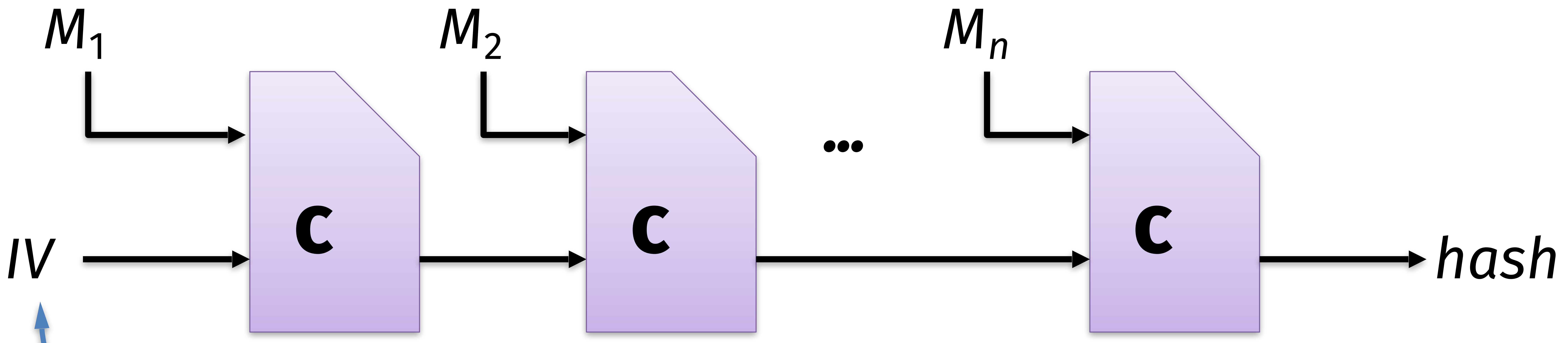
- Hash function **H** : $\{0,1\}^\infty \rightarrow \{0,1\}^\eta$ compresses messages into short digests
- Collision resistant, even if Mallory sees the code of **H**
  - Unlike block ciphers or MACs, whose security depends upon hiding the key
- NIST standards: SHA-2, SHA-3
  - Four output lengths: 224, 256, 384, and 512 bits
  - SHA = Secure Hash Algorithm
- Question: How do we build a hash function?

Merkle-Damgård paradigm (used in SHA-2)

Can build a variable-length input hash function from two primitives:

1. A fixed-length, *compressing* random-looking function

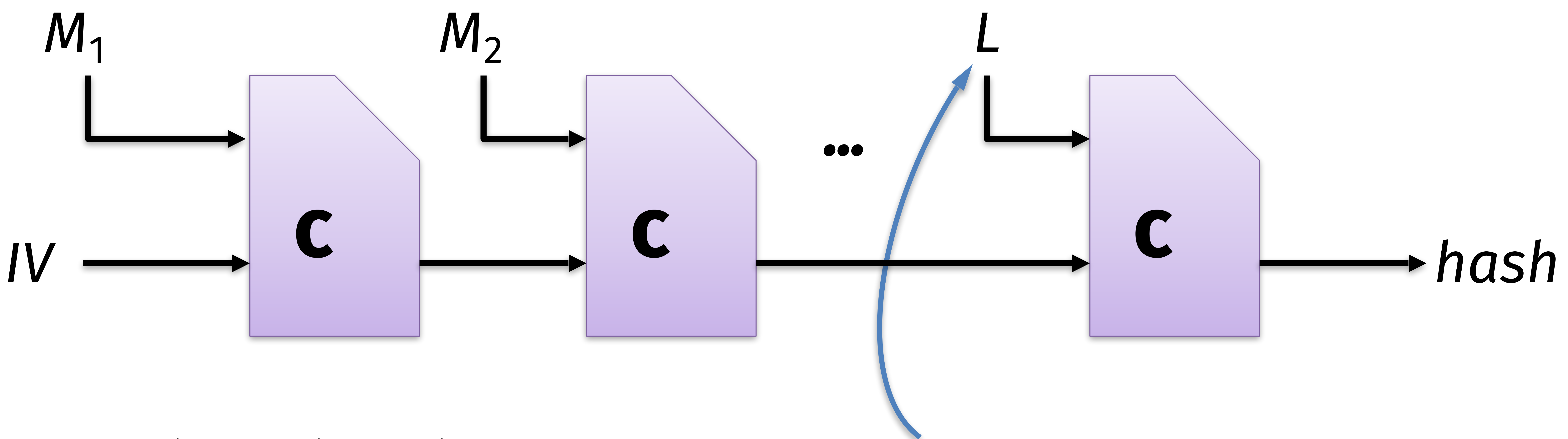
2. A mode of operation that iterates this function multiple times in a smart manner



IV for hash function is typically fixed in spec (unlike CBC, this value never changes)

Mihir Bellare's constraints for collision-free padding

1. Message M is always a prefix of pad(M)
2. If |M1| = |M2|, then |pad(M1)| = |pad(M2)|
3. If |M1| ≠ |M2|, then last block of pad(M1) ≠ last block of pad(M2)



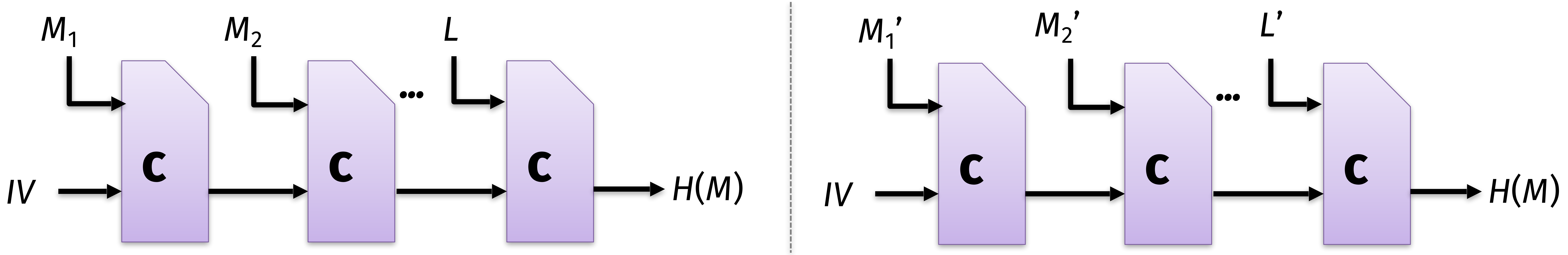
One convention: write string length as the sole contents of final block

Why Bellare's padding suffices

Theorem. If ***C***: $\{0,1\}^{2n} \rightarrow \{0,1\}^n$ is collision-resistant, then ***H***: $\{0,1\}^* \rightarrow \{0,1\}^n$ following the Merkle-Damgard construction is also collision-resistant

Proof sketch. If Mallory finds a collision in ***H*** ⇒ Mallory finds collision in ***C***

- If $|M| \neq |M'|$: Collision in the final steps: ***C***(something || L) and ***C***(something || L')
- If $|M| = |M'|$ but $M \neq M'$: State after each ***C*** start out different, become identical somewhere!



Hash function → MAC (first attempt)

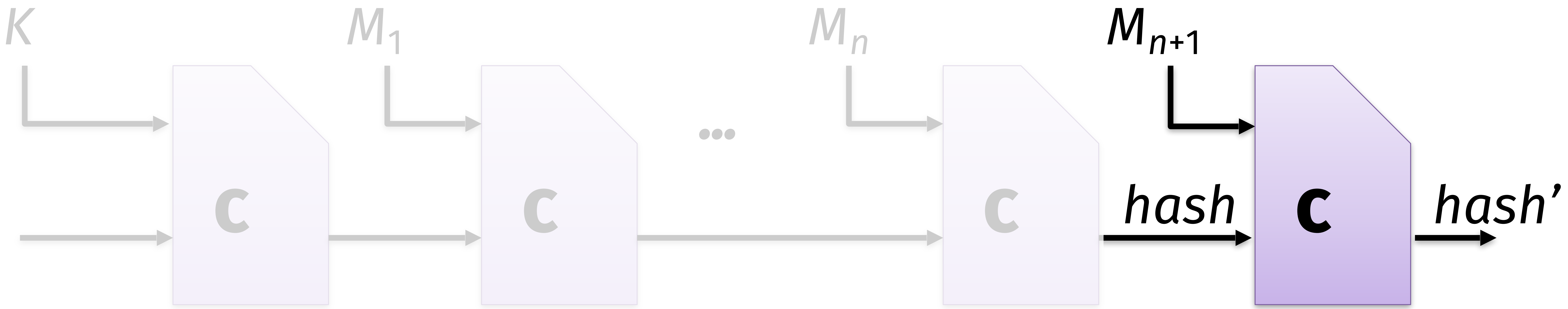
Idea: take a hash of the key and message

- MAC.KeyGen samples a key $K \leftarrow \{0,1\}^\lambda$ uniformly at random
- $\text{MAC.Tag}_K(A) = \mathbf{H}(K \,||\, A)$, note that this tag has fixed length $\tau = \eta$
- MAC.Verify operates in the usual way: re-compute the tag and compare

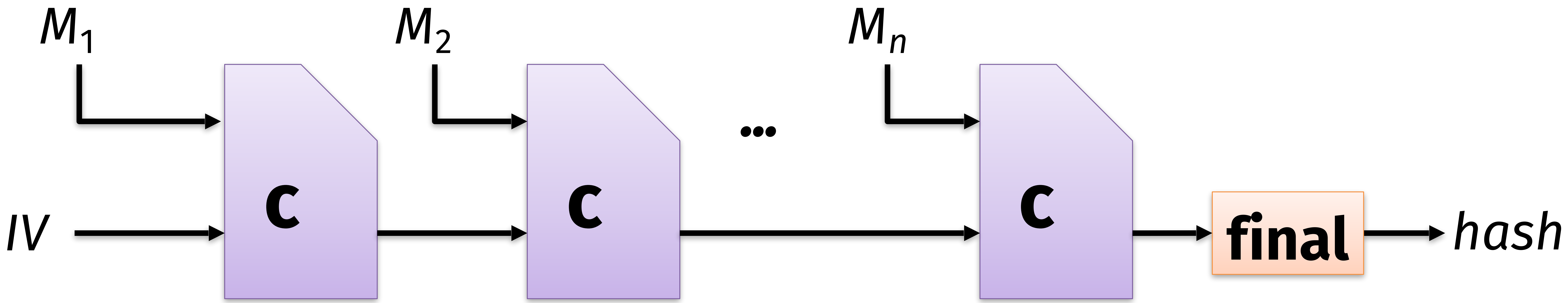
Notation: || denotes the concatenation of two bytestrings

Problem with Merkle-Damgård: Length extension

- Length extension breaks our MAC, but doesn't break collision resistance
- Bellare's padding does *not* save us from length extension (why?)

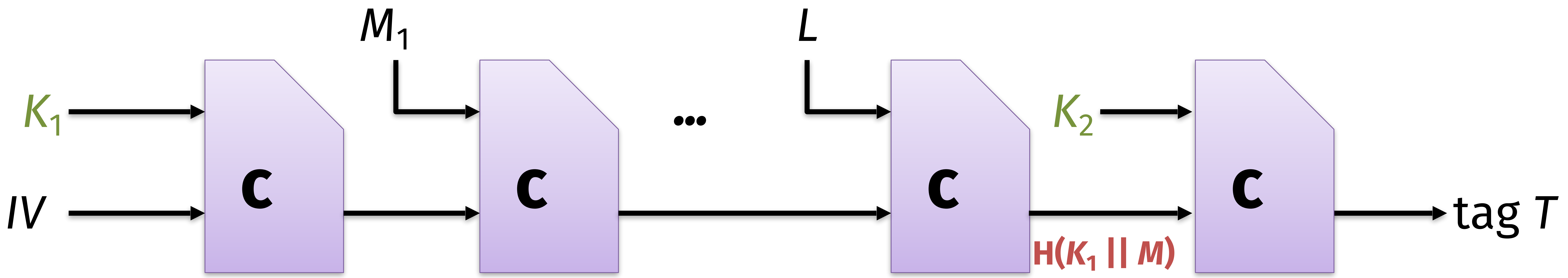


Countermeasure: finalization



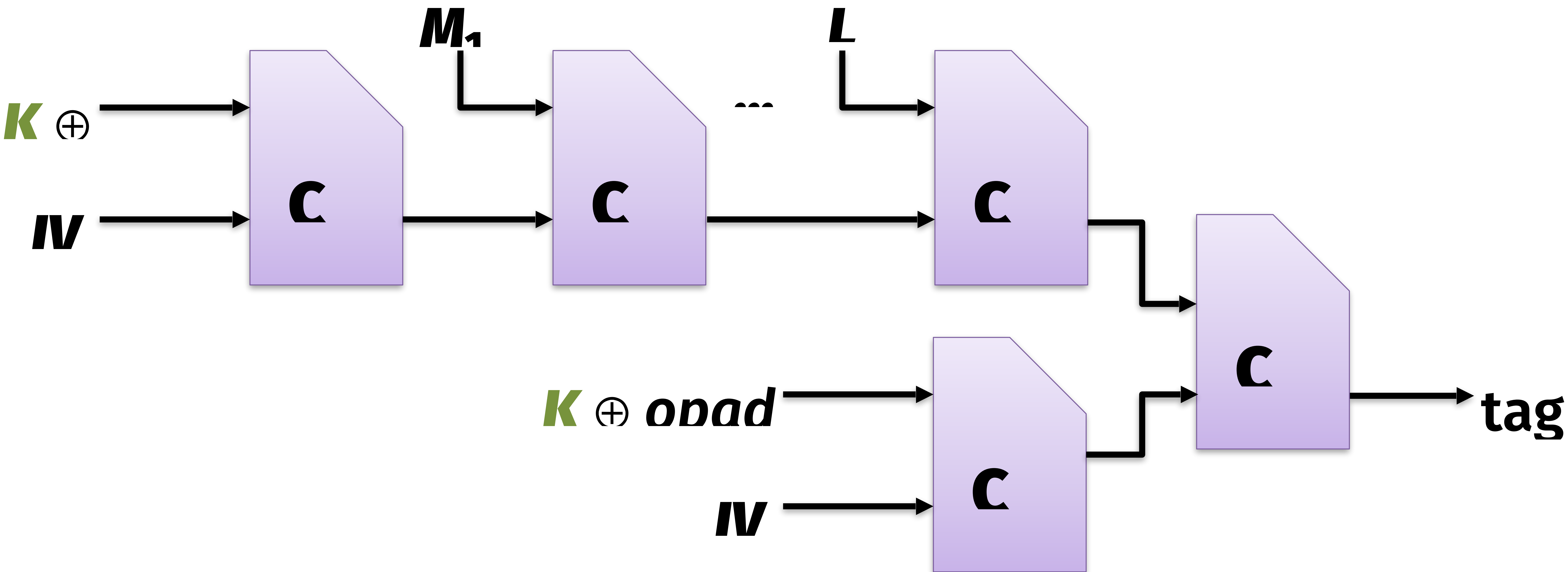
Hash function → MAC (done properly)

- *NMAC*: finalize the hash function by calling C one more time
- There are two keys, and the final step depends on the second key



HMAC [Bellare Canetti Krawczyk 97]

- *HMAC*: Like before, but derive two "independent" keys from one key
  - Recall from CMAC → OMAC story: nobody wants to carry multiple keys
  - Fixed constants ipad = 0x5C, opad = 0x36 repeated to equal the key length



Strength of HMAC

Thm. HMAC is an EU-CMA MAC as long as:

1. The compression function C is pseudorandom
2. The Merkle-Damgard iteration mechanism is collision-resistant

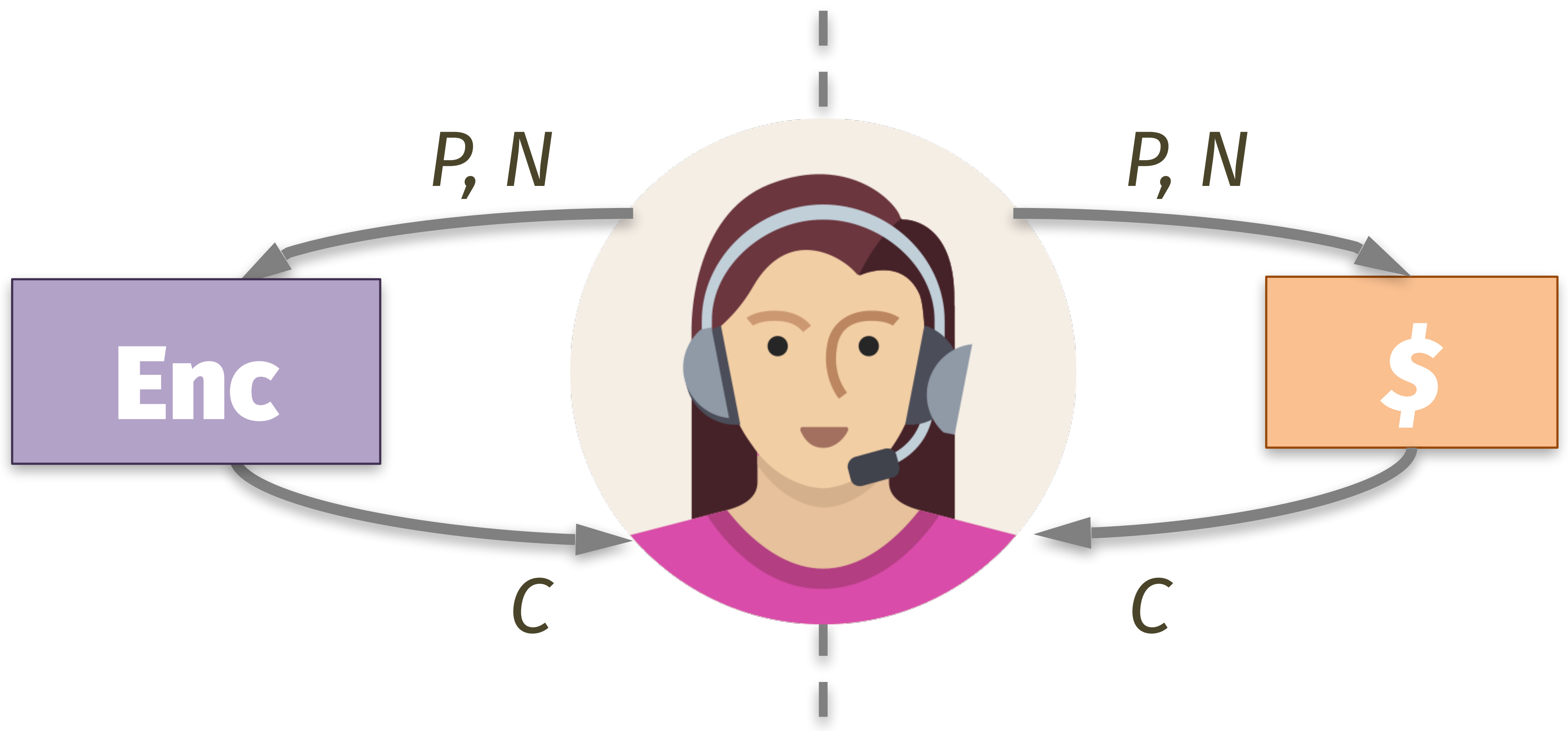
Bellare (2005) removed condition #2, so HMAC is safe with hash functions like MD5 and SHA1 that are preimage resistant but not collision resistant

Recall the bu.edu connection settings in 2017:

> Obsolete connection settings
>
> The connection to this site uses TLS 1.0 (an obsolete protocol), RSA (an obsolete key exchange), and AES_256_CBC with HMAC-SHA1 (an obsolete cipher).

Combining Encryption and Authentication

Encryption: IND$-CPA



Restriction: Mallory can't re-use nonce

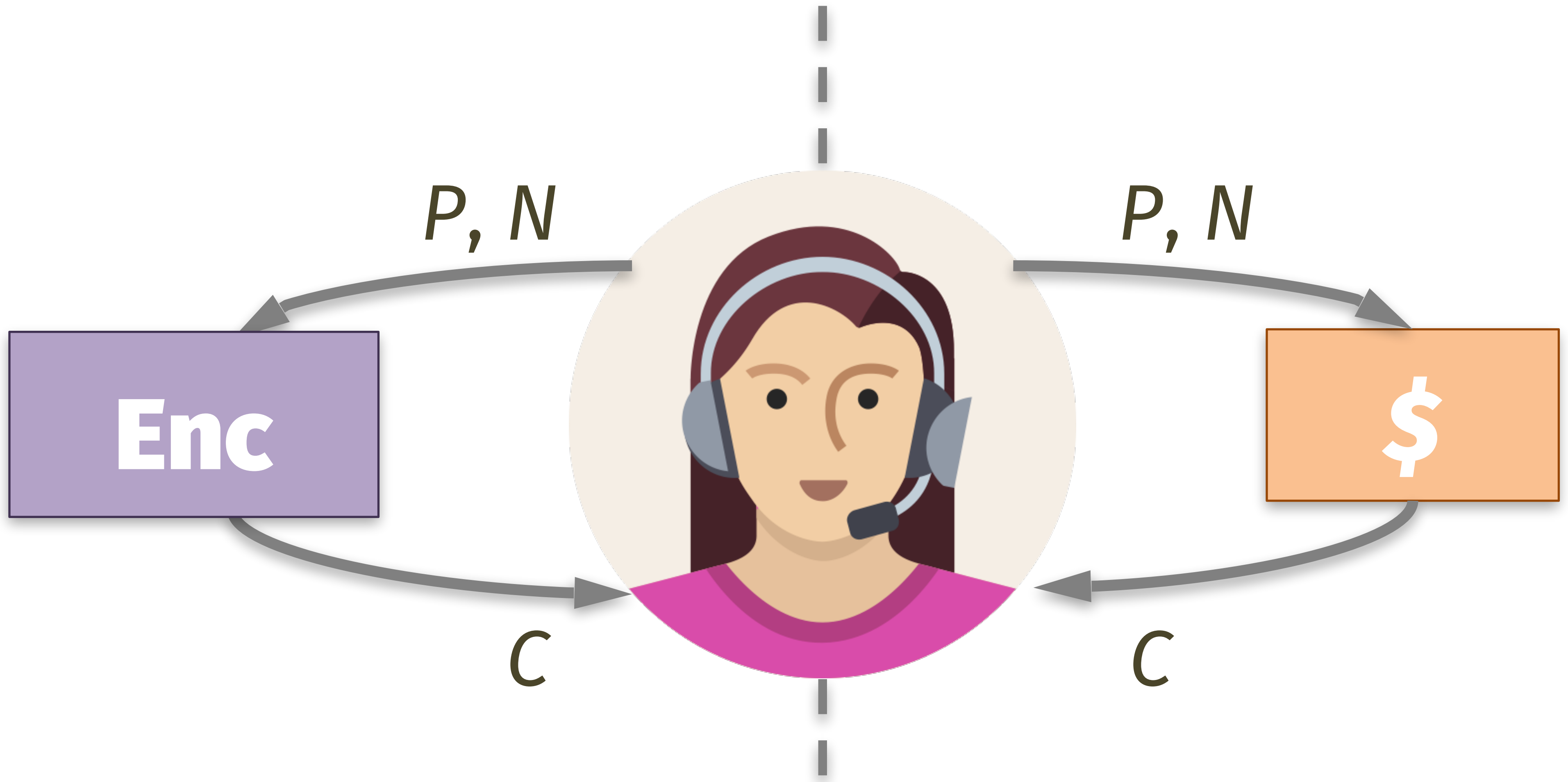
(in the stronger variant, which will be our focus from now onward)

MAC: EU-CMA

Restriction: Mallory cannot verify a MAC tag that Alice produced

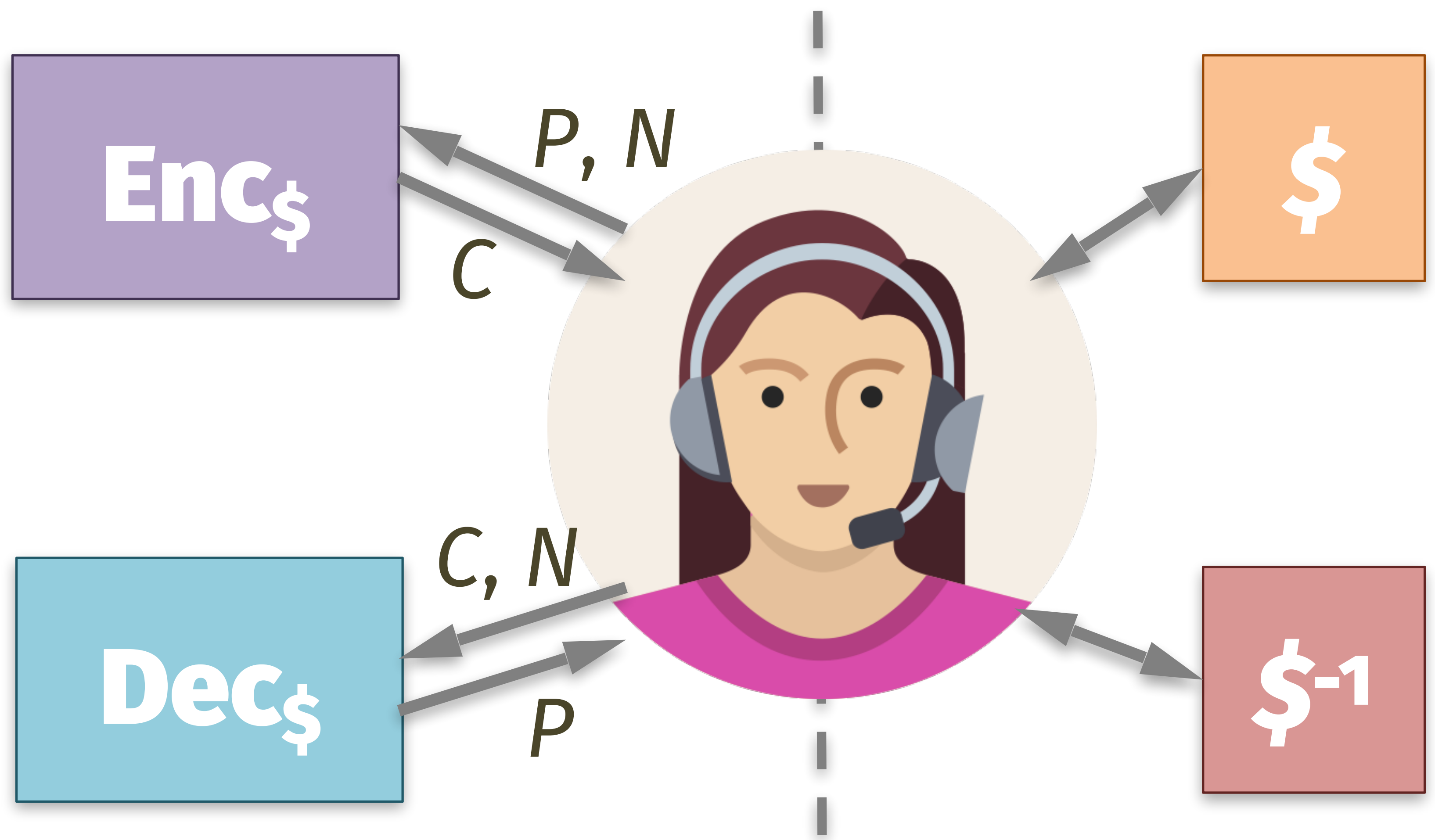
Let’s strengthen our definition of privacy

IND$-CPA



IND$-CCA

Same thing, but now Mallory has access to encryption *and* decryption oracles



Claim: Encryption schemes meeting this stronger definition also provide authenticity

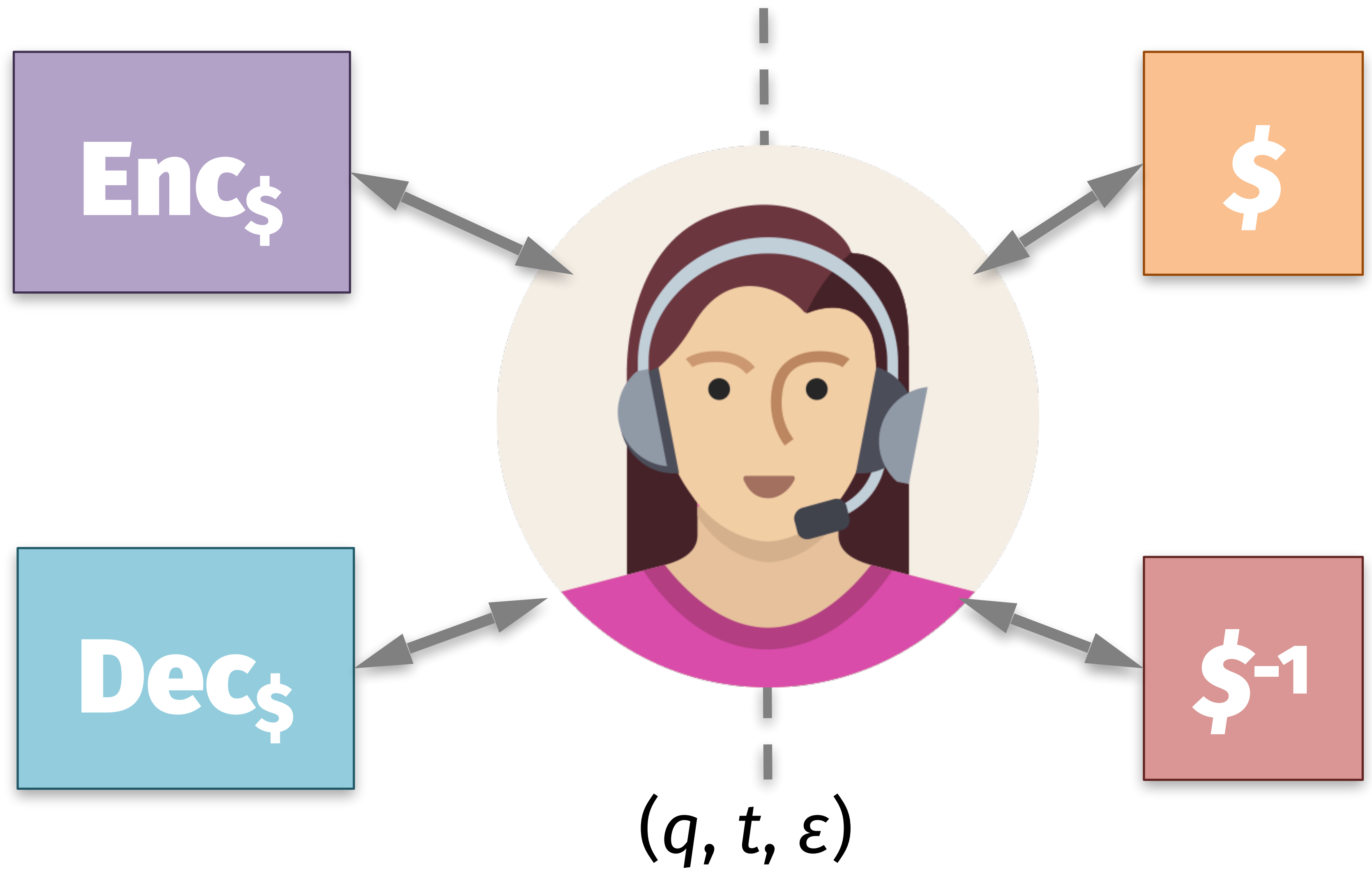
Formalizing IND$-CCA

Comprises 3 algorithms:

- KeyGen(λ) outputs a key $K \leftarrow \{0,1\}^\lambda$
- Encrypt$_K$(message P, nonce N) → C
- Decrypt$_K$(ciphertext C, nonce N) → P

Satisfies 3 constraints

- Performance: all 3 algorithms are efficiently computable
- Correctness: $\mathrm{Dec}_K(\mathrm{Enc}_K(P, N)) = P$ for all $K \in \{0,1\}^\lambda$, $N \in \{0,1\}^\mu$, and $P \in \{0,1\}^*$

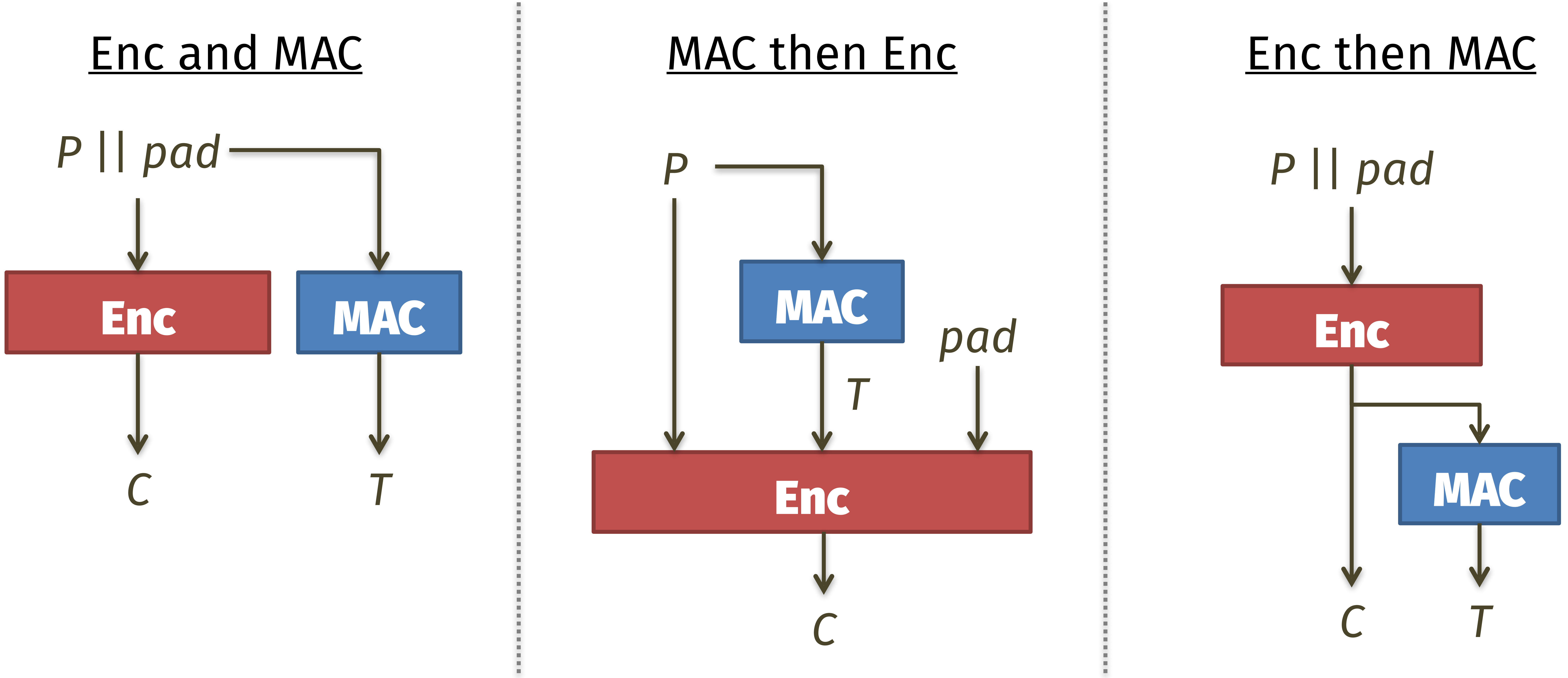


- **(q, t, ε)-IND$-CCA**: for every *nonce-respecting* adversary ***M*** who makes ≤ q queries and runs in time ≤ t,

$$M^{\mathrm{Enc}_K, \mathrm{Dec}_K} \approx_{q,t,\epsilon} M^{\$,\$^{-1}}$$

where \$ responds randomly and so does $\$^{-1}$ subject to consistency with \$

Combining Enc and MAC *generically*

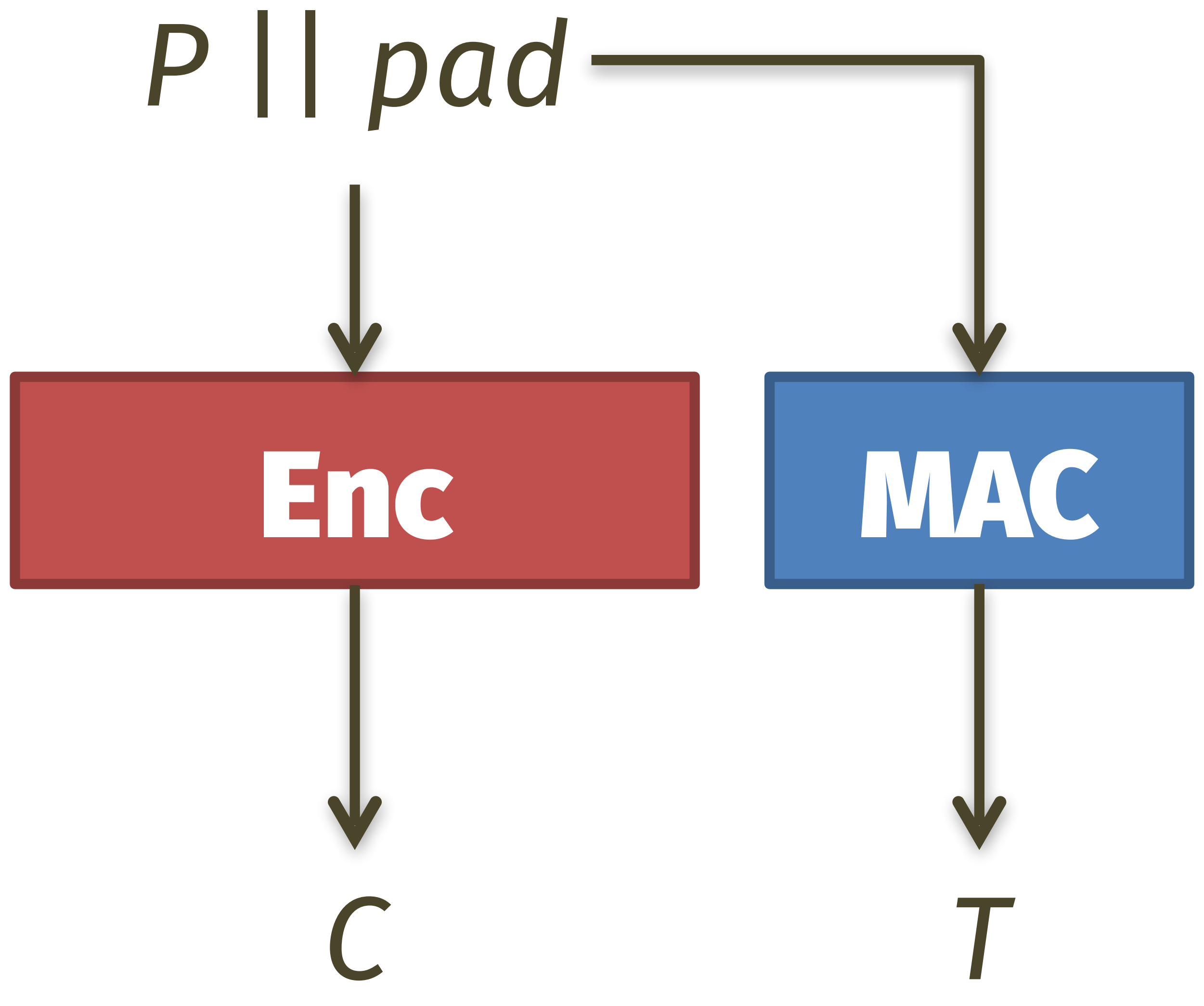


Intuitive concerns with MAC then Enc

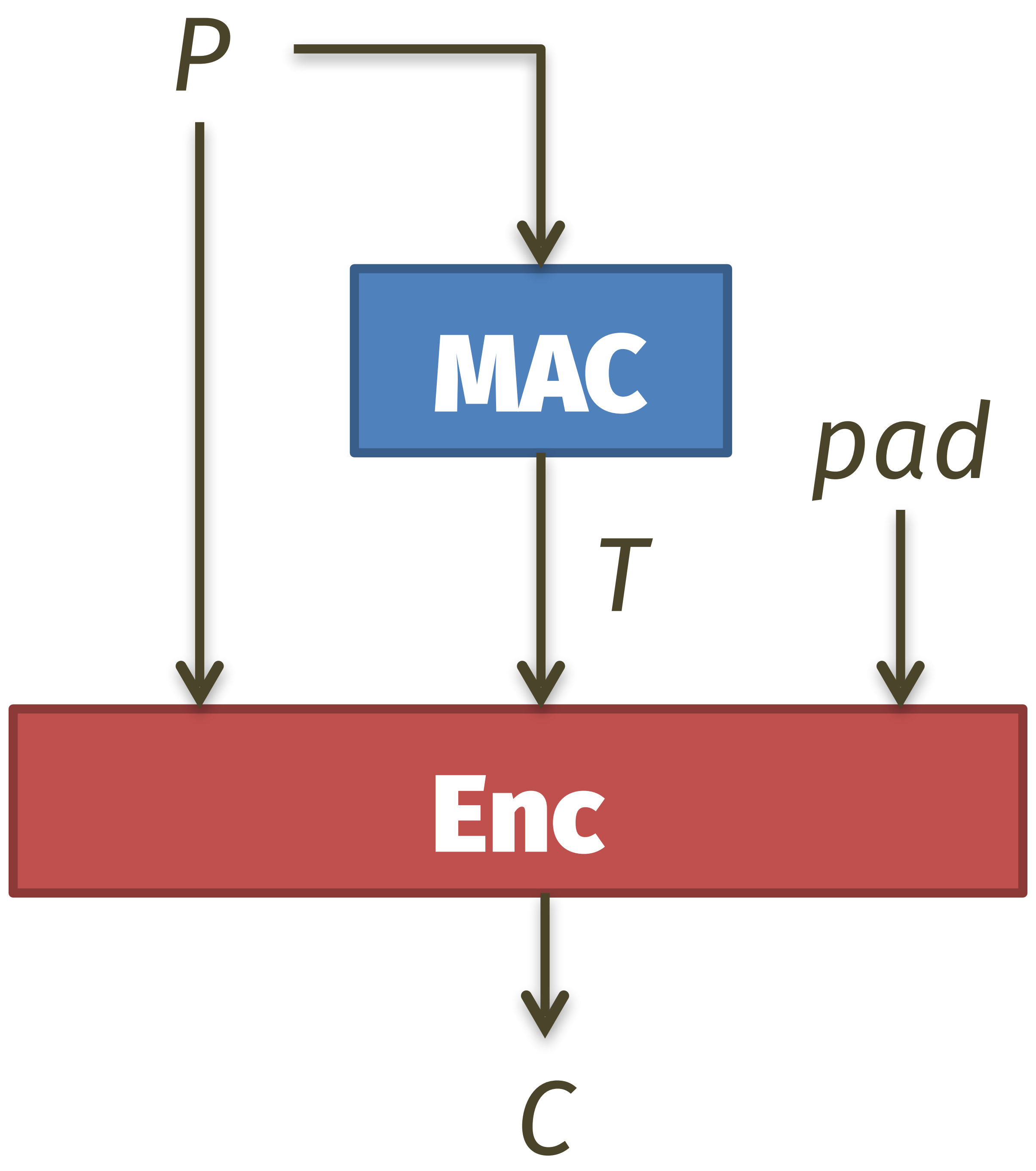
- The private data *P* is authenticated, but *C* is not!
- Recipient must perform decryption before knowing whether the message is authentic

Combining Enc and MAC *generically*

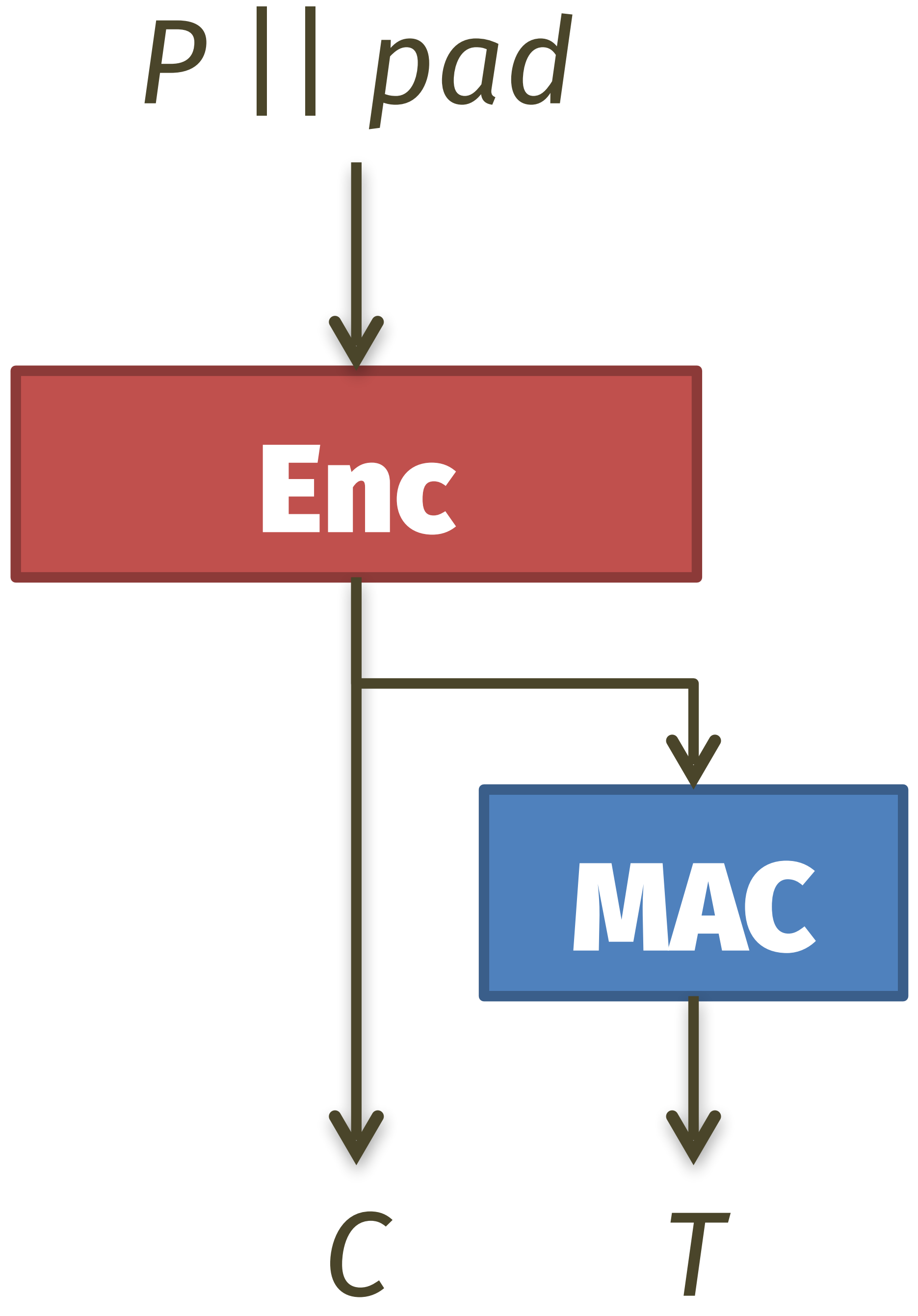
Enc and MAC



MAC then Enc



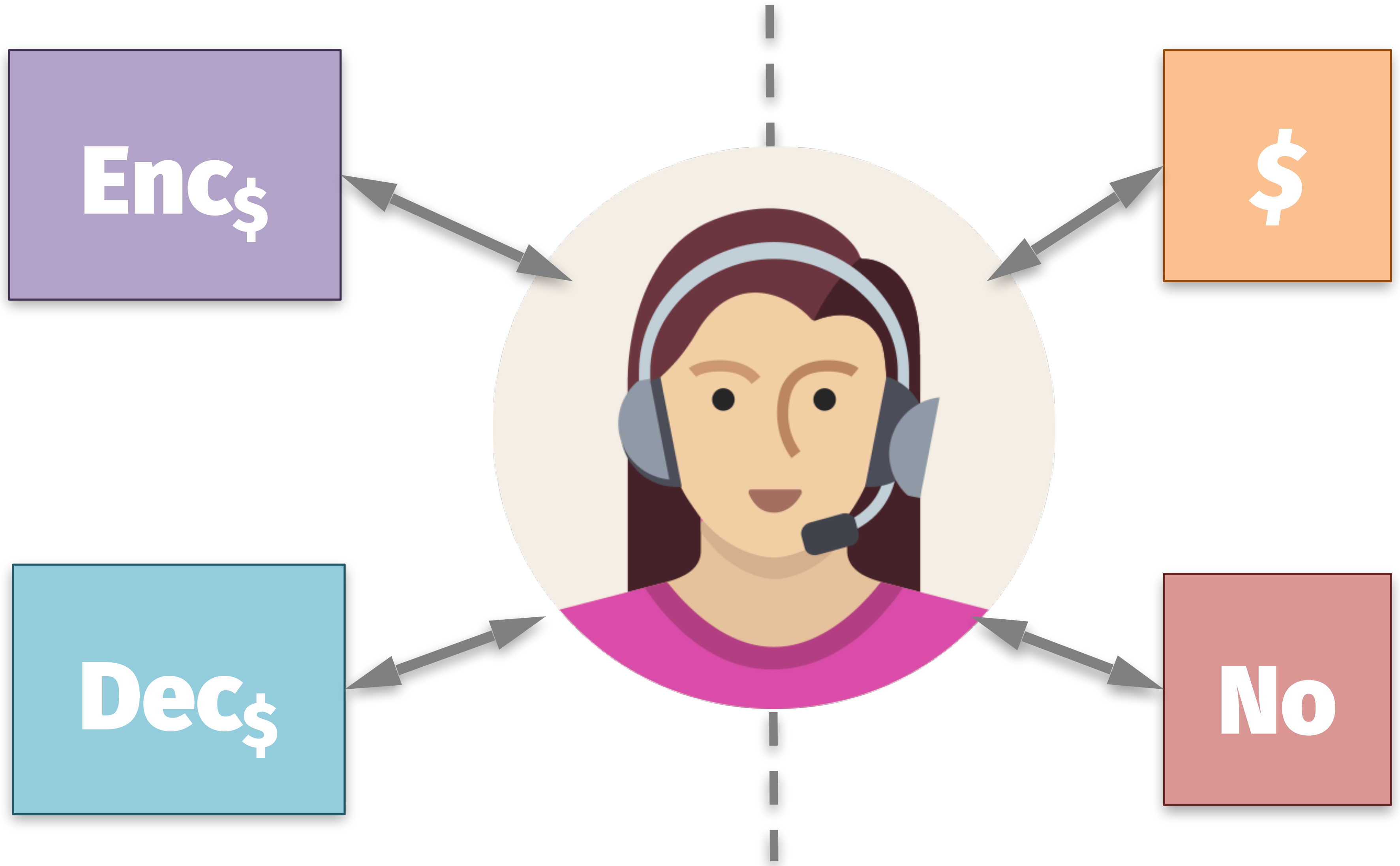
Enc then MAC



Confidentiality	None	CPA	CCA!
Integrity	*Plaintext integrity*: Cannot make CT that decrypts to message that sender never encrypted		*Ciphertext integrity*: Cannot make new valid CTs, only know sender-made ones

Formalizing ciphertext integrity

- Goal: Mallory cannot make a valid CT that wasn't previously made by sender
- Imagine that Mallory is trying to perform a padding oracle attack
- If she spams Bob with malformed CTs, now he simply rejects them all!

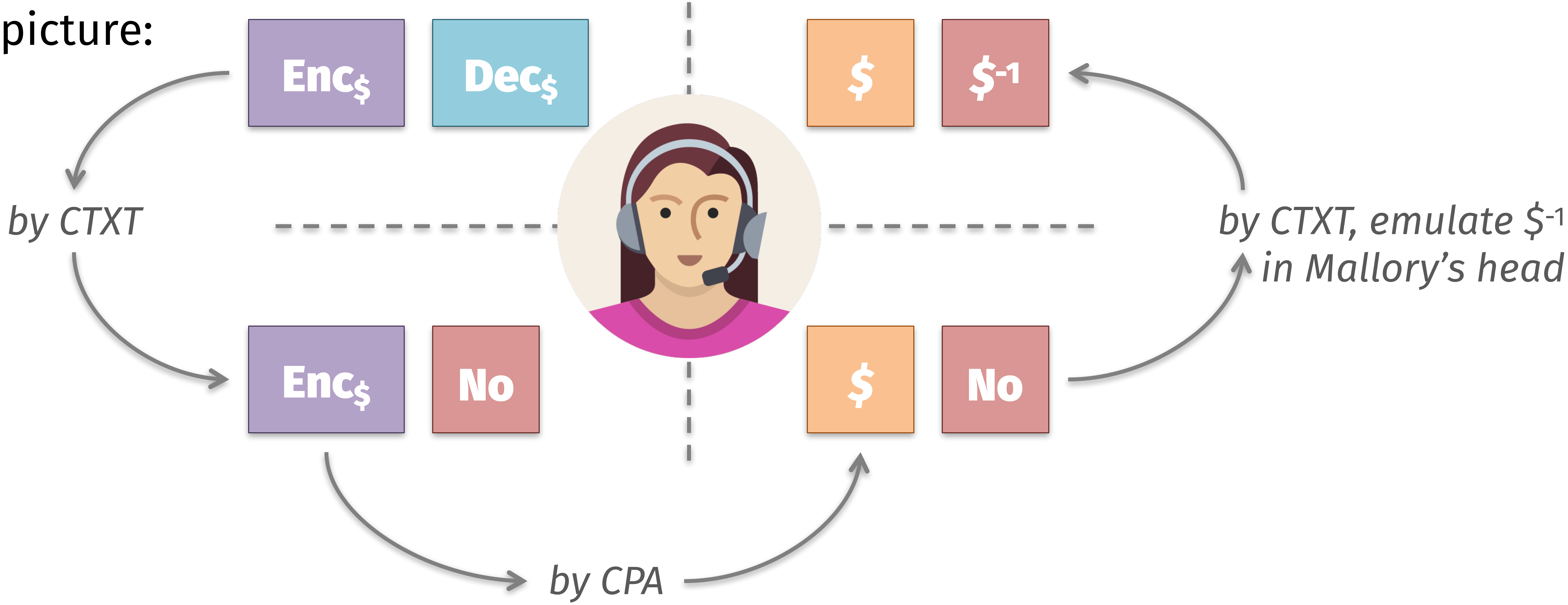


Operation: This box returns a single "integrity failure" error message no matter what Mallory submits!

Restriction: Mallory cannot attempt to decrypt ciphertexts that are the result of prior encryptions.

Relating ciphertext integrity to privacy

- **Theorem.** Suppose that an encryption scheme provides (q, t, ε_1)-CPA privacy and (q, t, ε_2)-ciphertext integrity. Then, it also provides (q, t, $\varepsilon_1+2\varepsilon_2$)-CCA privacy.
- Intuition: If Mallory can't forge new messages, then Dec oracle is useless to her
- Proof by picture:

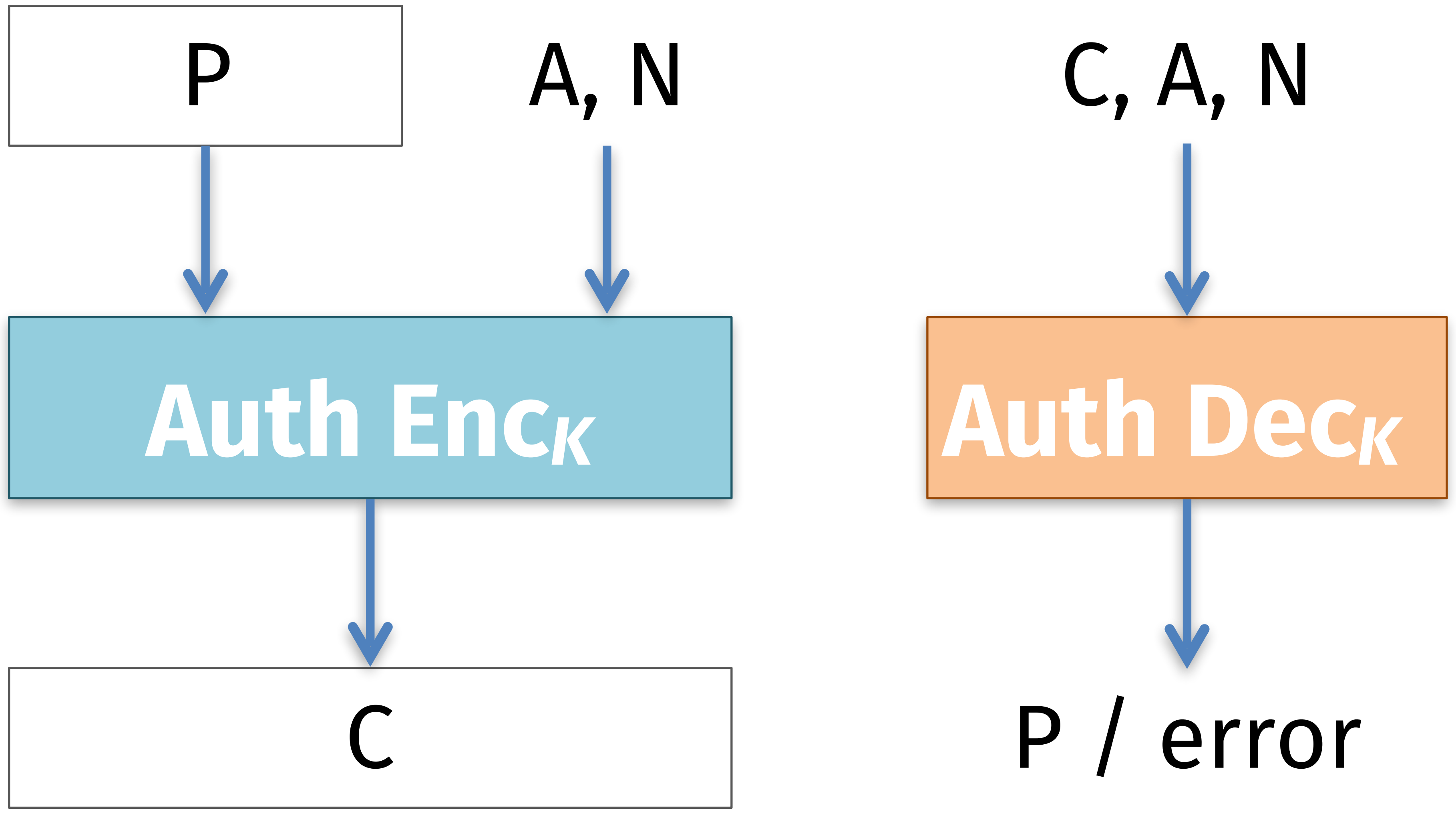


Relating ciphertext integrity to privacy

- **Theorem.** Suppose that an encryption scheme provides (q, t, ε_1)-CPA privacy and (q, t, ε_2)-ciphertext integrity. Then, it also provides $(q, t, \varepsilon_1+2\varepsilon_2)$-CCA privacy.
- Lesson: to build strong encryption, it suffices to satisfy CPA and ciphertext integrity. Let's make one final security definition that combines these concepts...

Def. Authenticated Encryption with Associated Data (AEAD)

- KeyGen: randomly choose K, as usual
- Enc_K(authenticated data A, private + auth data P, nonce N) → ciphertext C of length |C| ≥ |P|
- Dec_K(C, A, N) → P or error



Why combine authentication and encryption?

- Better security: satisfies Moxie's doom principle, resists some side channel attacks
- Simplicity: developers have fewer decisions (i.e., opportunities for mistakes)
- Performance: save in time + space costs, also often only need 1 key

AEAD security definition

Restrictions

- Can't reuse nonce
- Can't ask to decrypt CT made by Alice