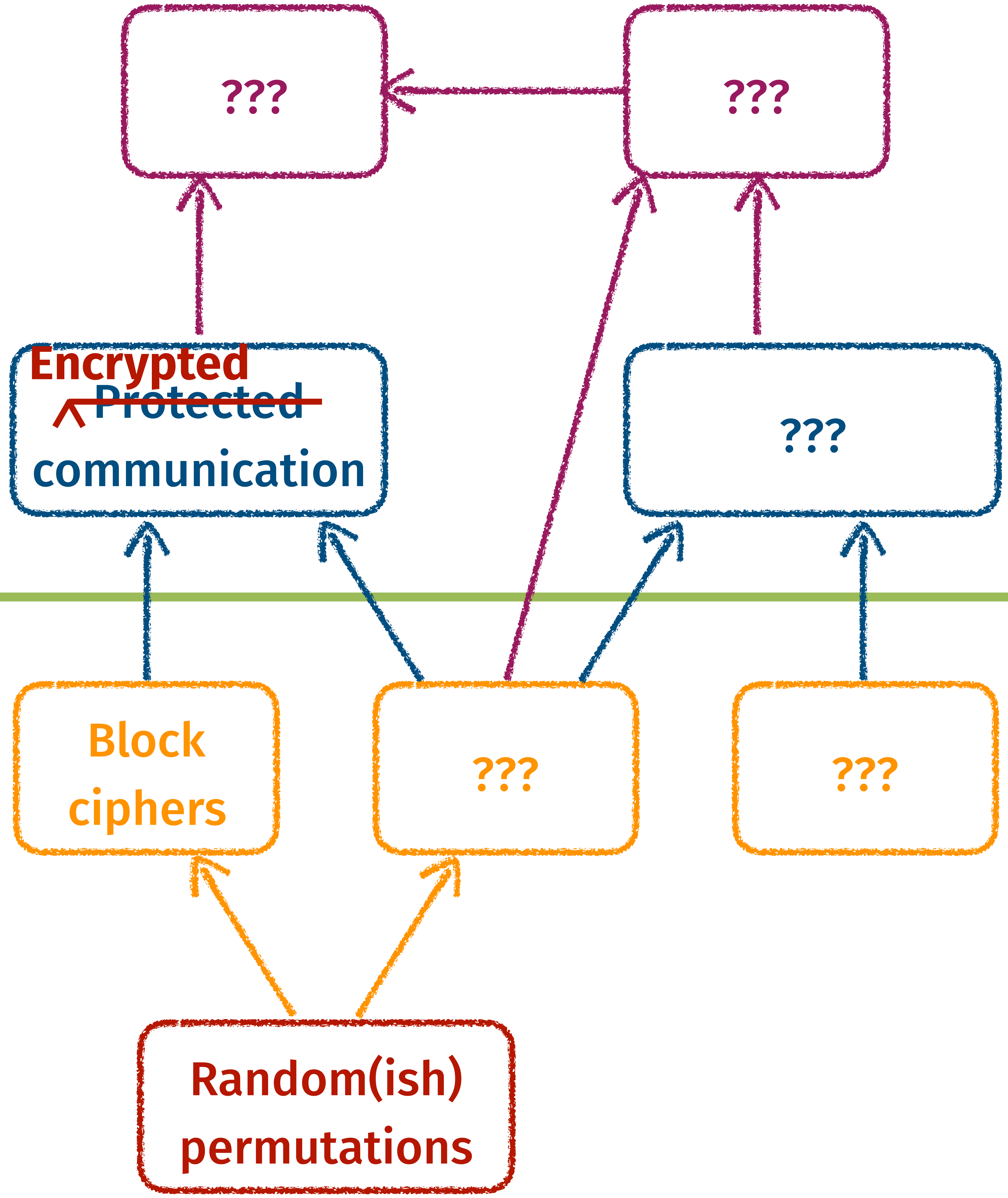
Lecture 9: Message Authentication Codes

- Midterm 1 has been graded, available on Gradescope
  - Nicolas' discussion section on Friday will review the test
- Homework 5 will be posted today
- Required reading: portions of two textbooks
  - The Block Cipher Companion (section 4.4)
  - The Hash Function BLAKE (sections 2.1, 2.2, 2.4)

Crypto in this course so far

Elegant protocols

Utilitarian tools



Which string “looks” random and unpredictable?

11111111

01010101

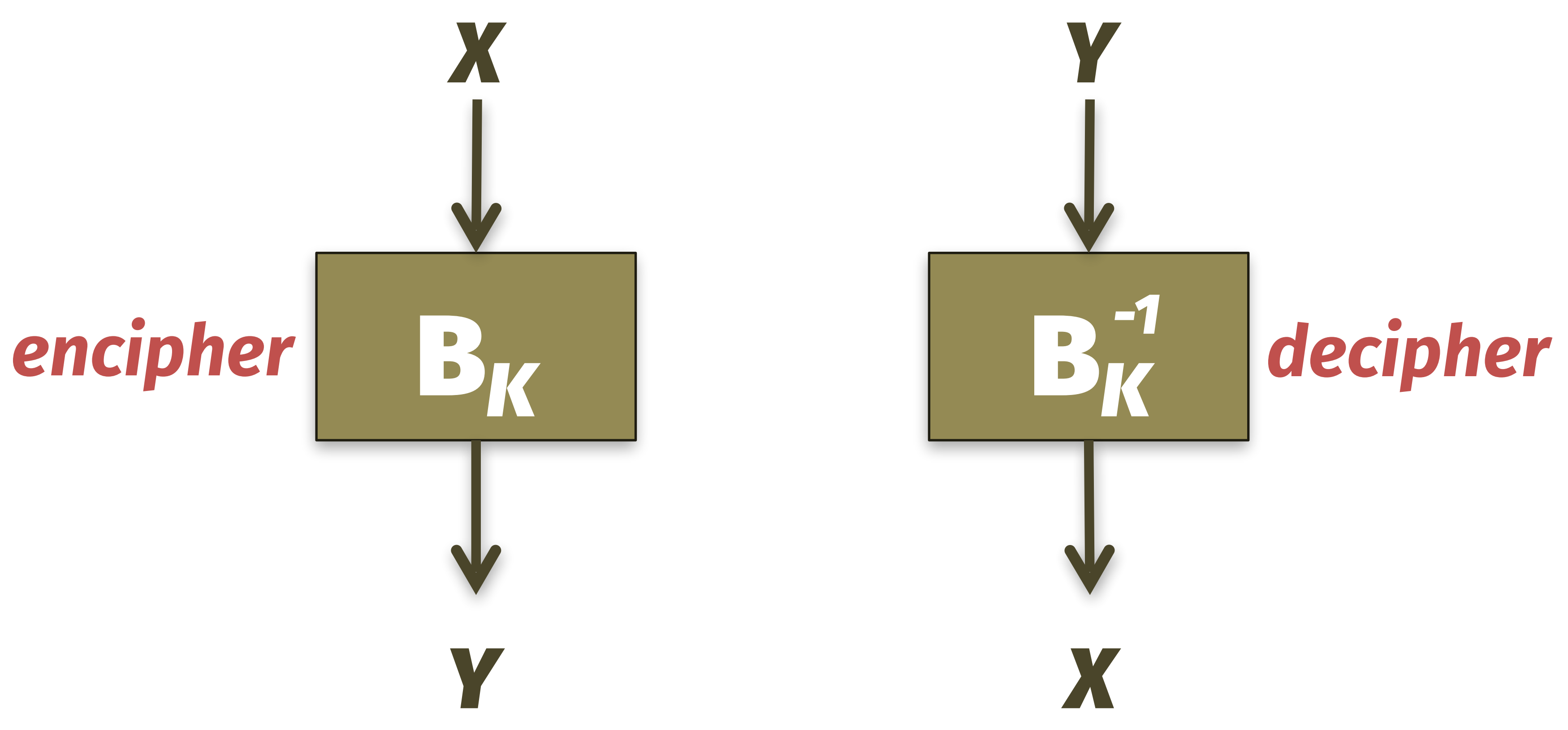
10100011

- Each string is equally likely to occur
- You cannot look at a single output string and determine its (un)predictability
- Same problem occurs when evaluating the unpredictability of a single codebook
- Our pseudorandomness definition instead evaluates the *process* of choosing a codebook

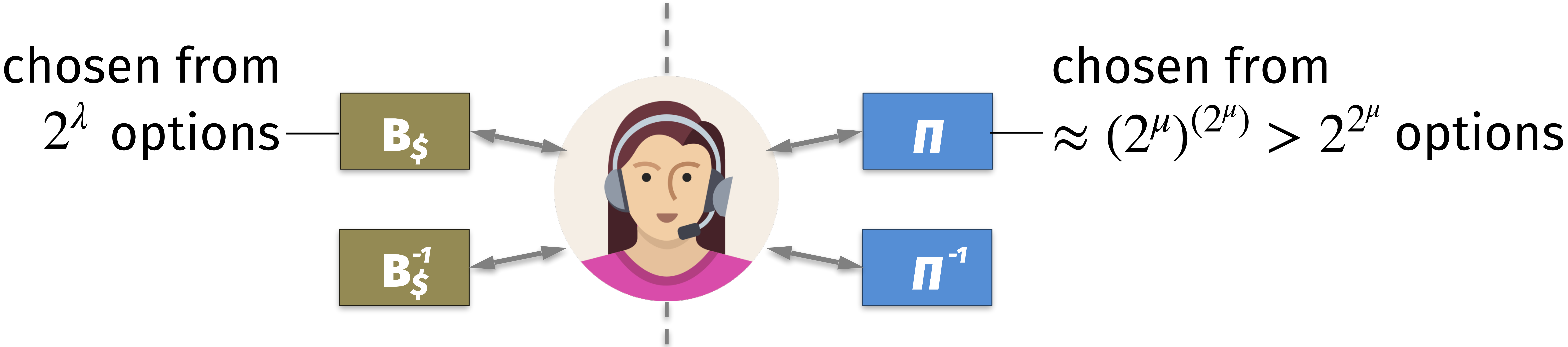
X	*Y*
000	001
001	110
010	000
011	111
100	011
101	010
110	101
111	100

Review: Block ciphers like AES

- Family of permutations, each of the form $B : \{0,1\}^\mu \rightarrow \{0,1\}^\mu$
- Key $K \in \{0,1\}^\lambda$ determines which permutation to use

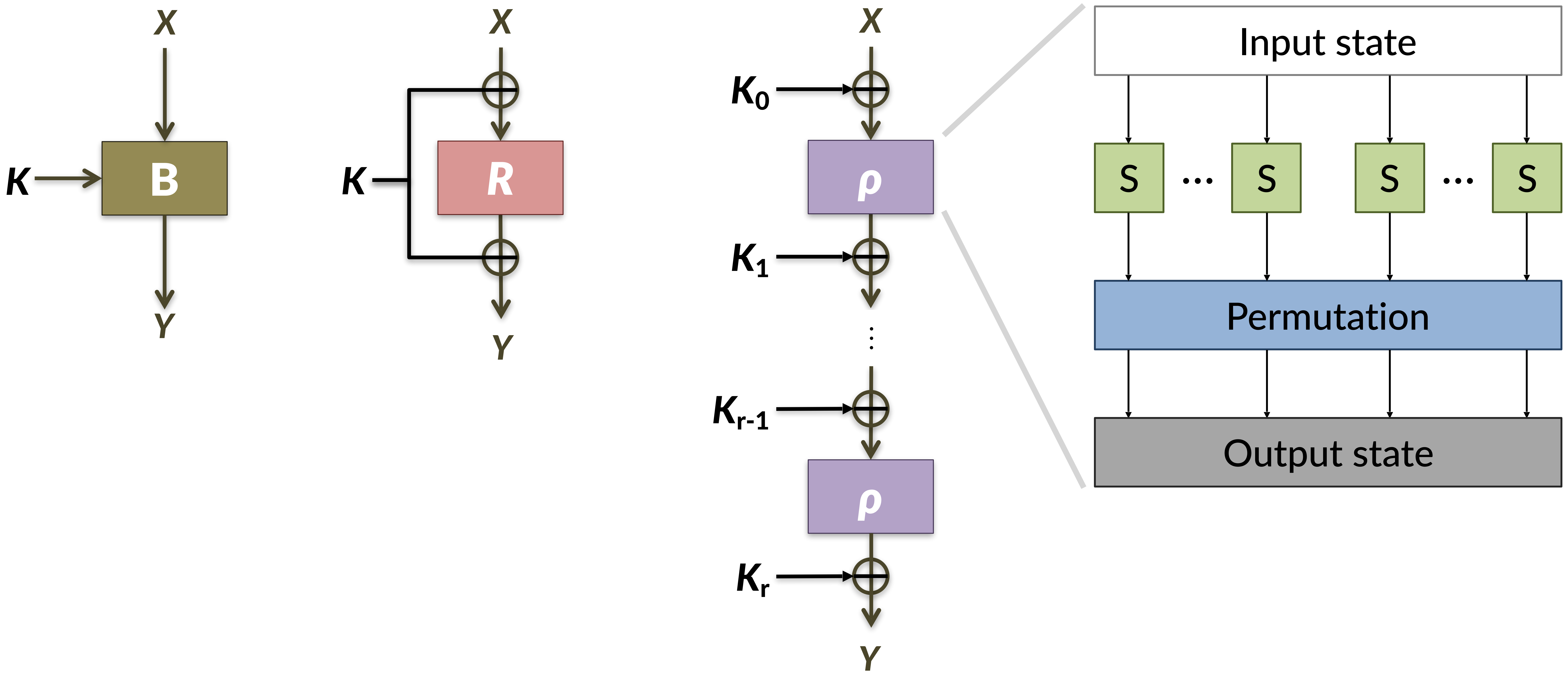


- B_K is *strongly pseudorandom* if every adversary running in time ≤ t and making ≤ q queries cannot tell it apart from a secret, truly random Π

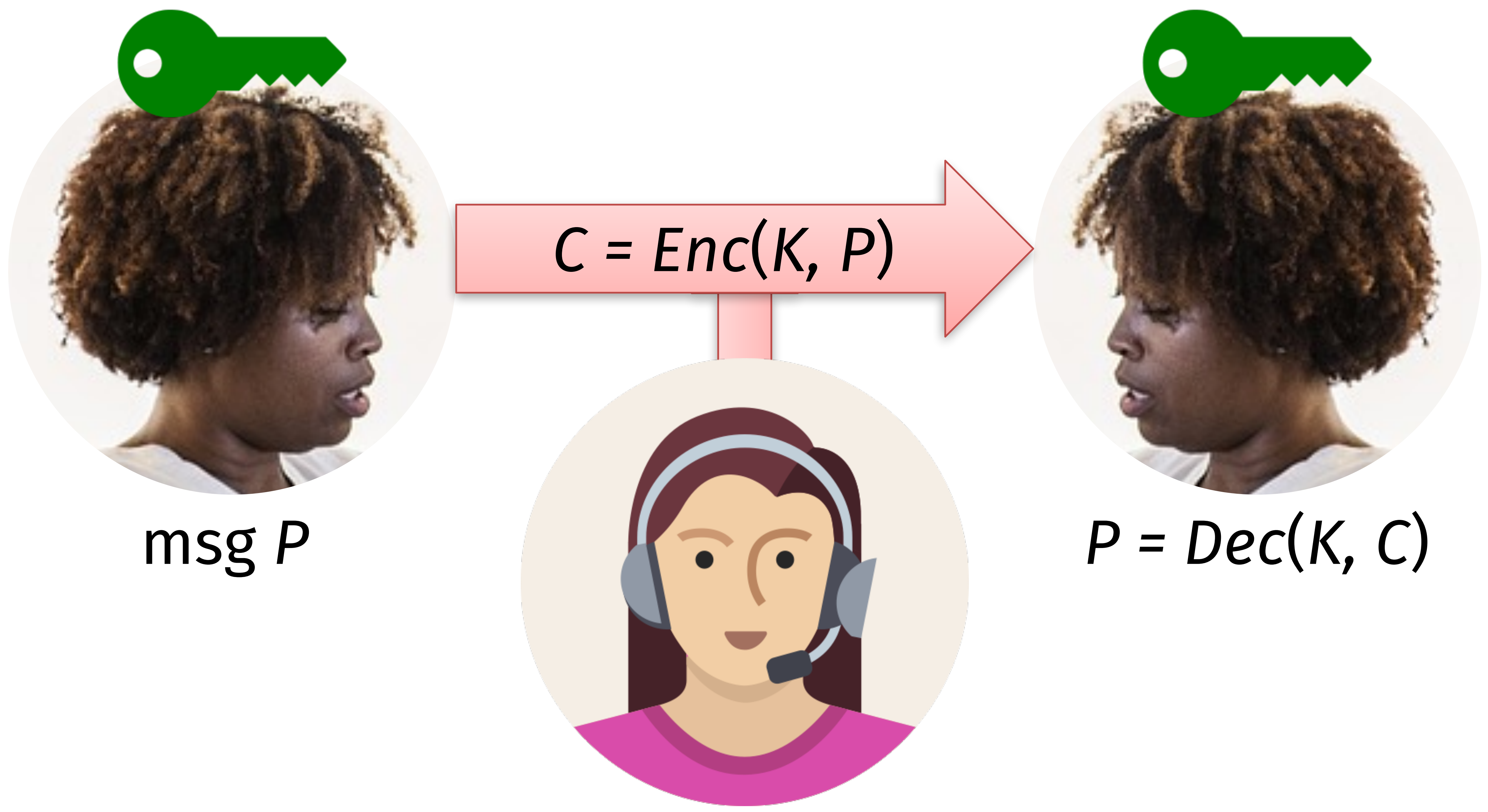


Review: Block cipher design

Block cipher ← Key alternation ← Iterated rounds ← Substitution-Permutation



Review: Alice's confidentiality + integrity goals



- *Data privacy*: Eve cannot learn P
- *Data authenticity*: if Eve tampers with C, then Alice can detect the change
- *Entity authenticity*: future Alice knows that she previously created C

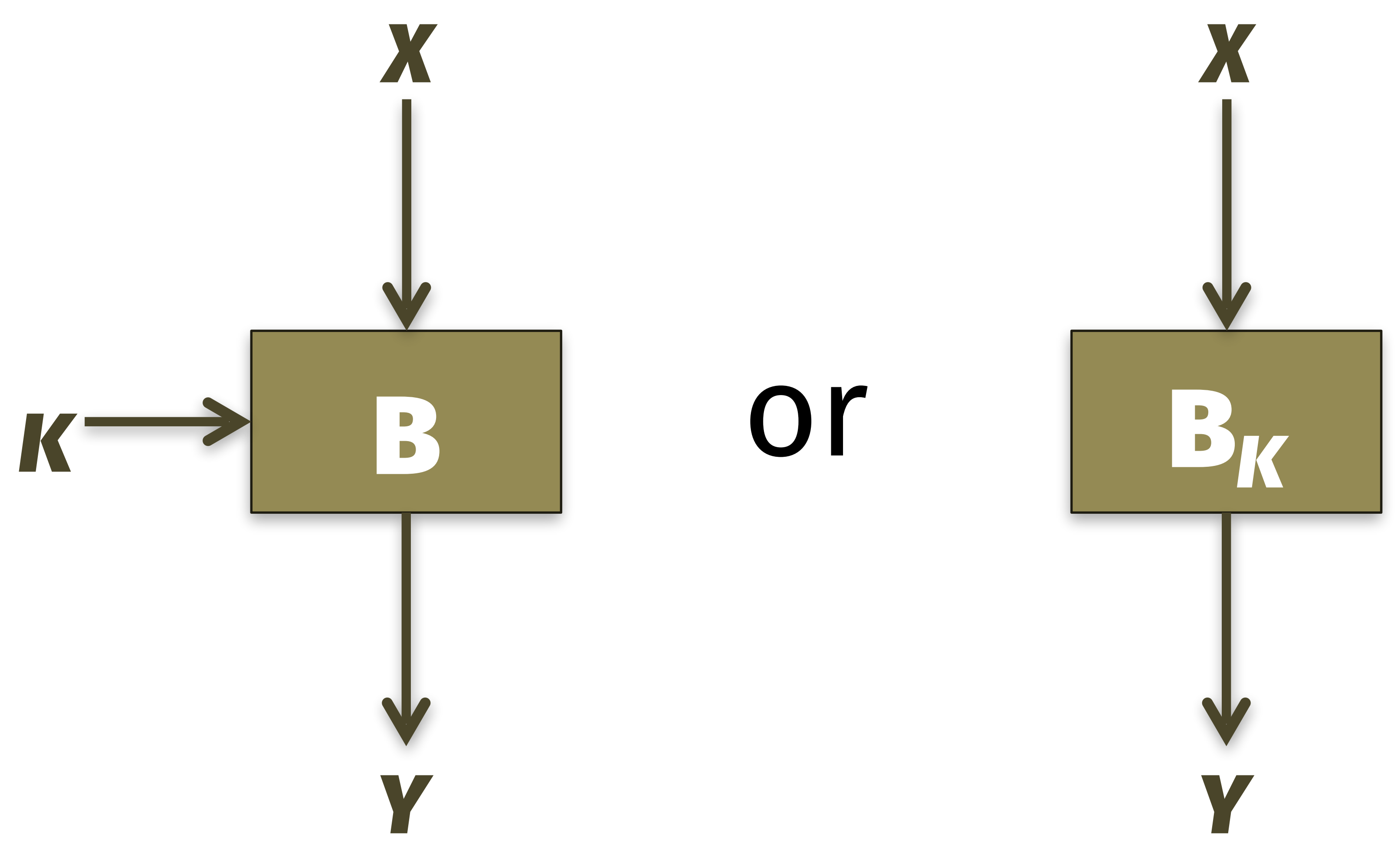
Review: Modes of operation (CBC, CTR)

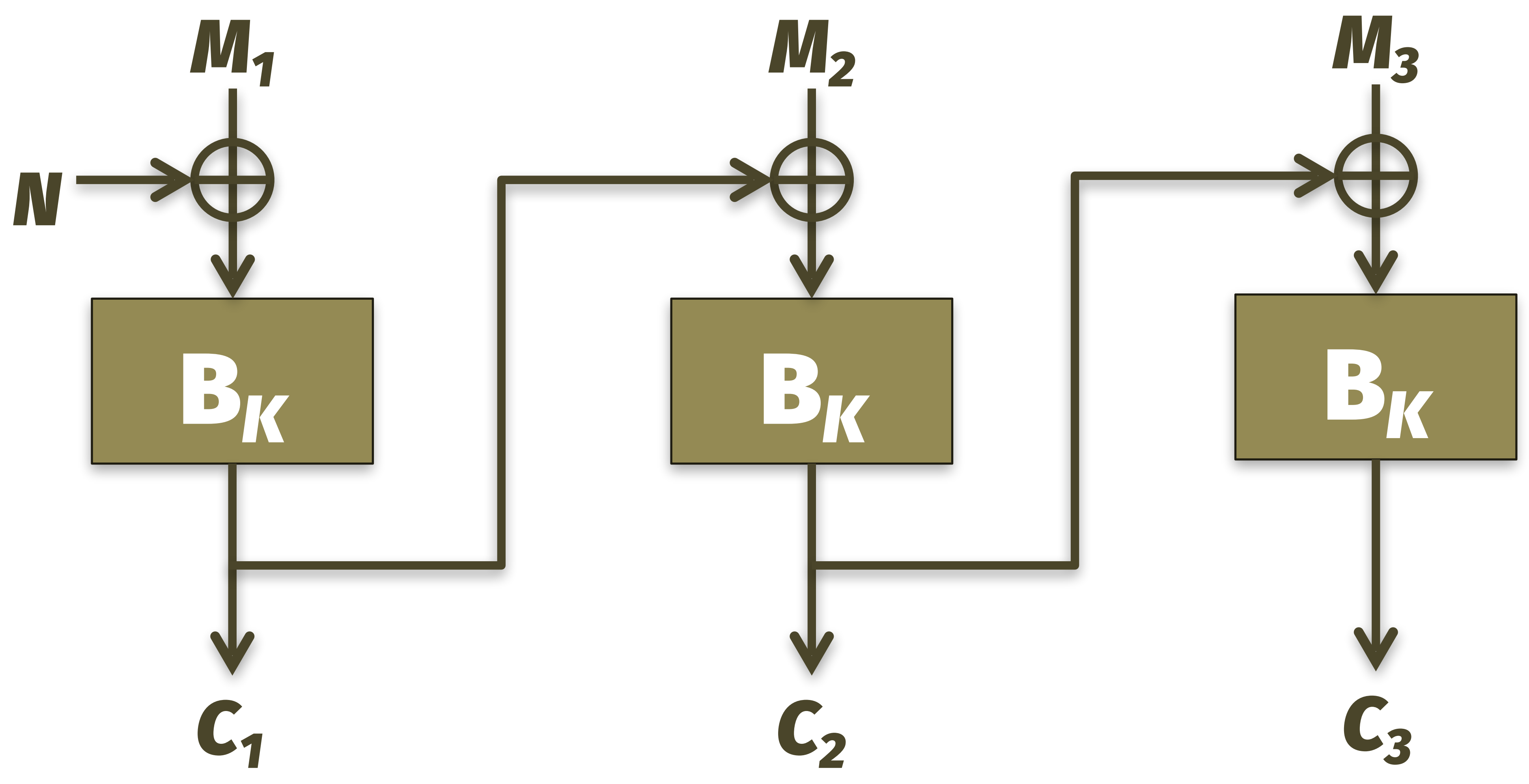
Block cipher = family of codebooks

- Each key K yields different codebook B_K
- Fast to compute: throughput ~3-4 GB/sec

Mode of operation = variability

- Allows long message with short key
- Thwarts frequency analysis





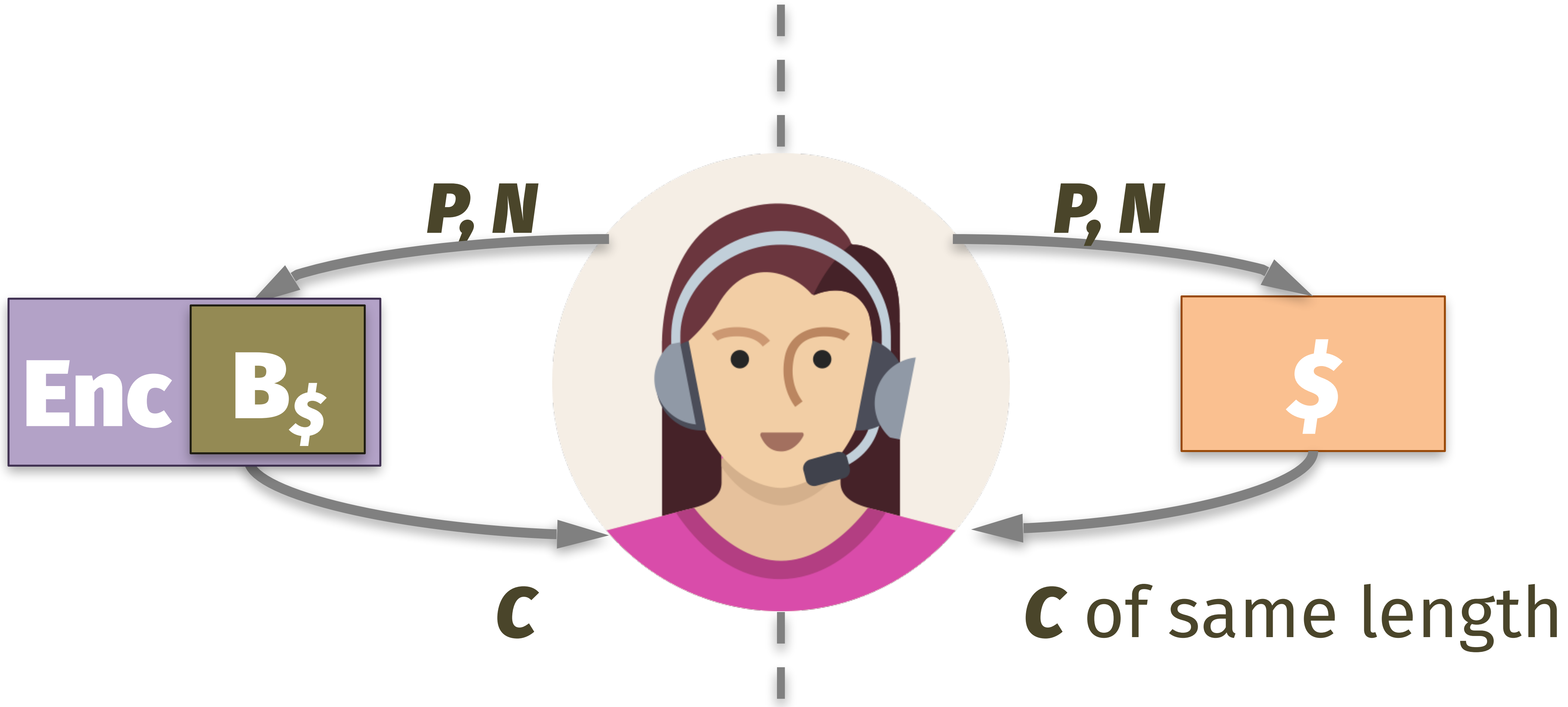
Review: Security definitions

Block cipher

B_K looks like a truly random function, meaning nobody can tell them apart

Encryption scheme

Similar, except even making the same request twice yields different answers



"The length of the encrypted packet clearly leaks which candidate was selected."

–Michael A. Specter, James Koppel, and Daniel Weitzner (MIT)

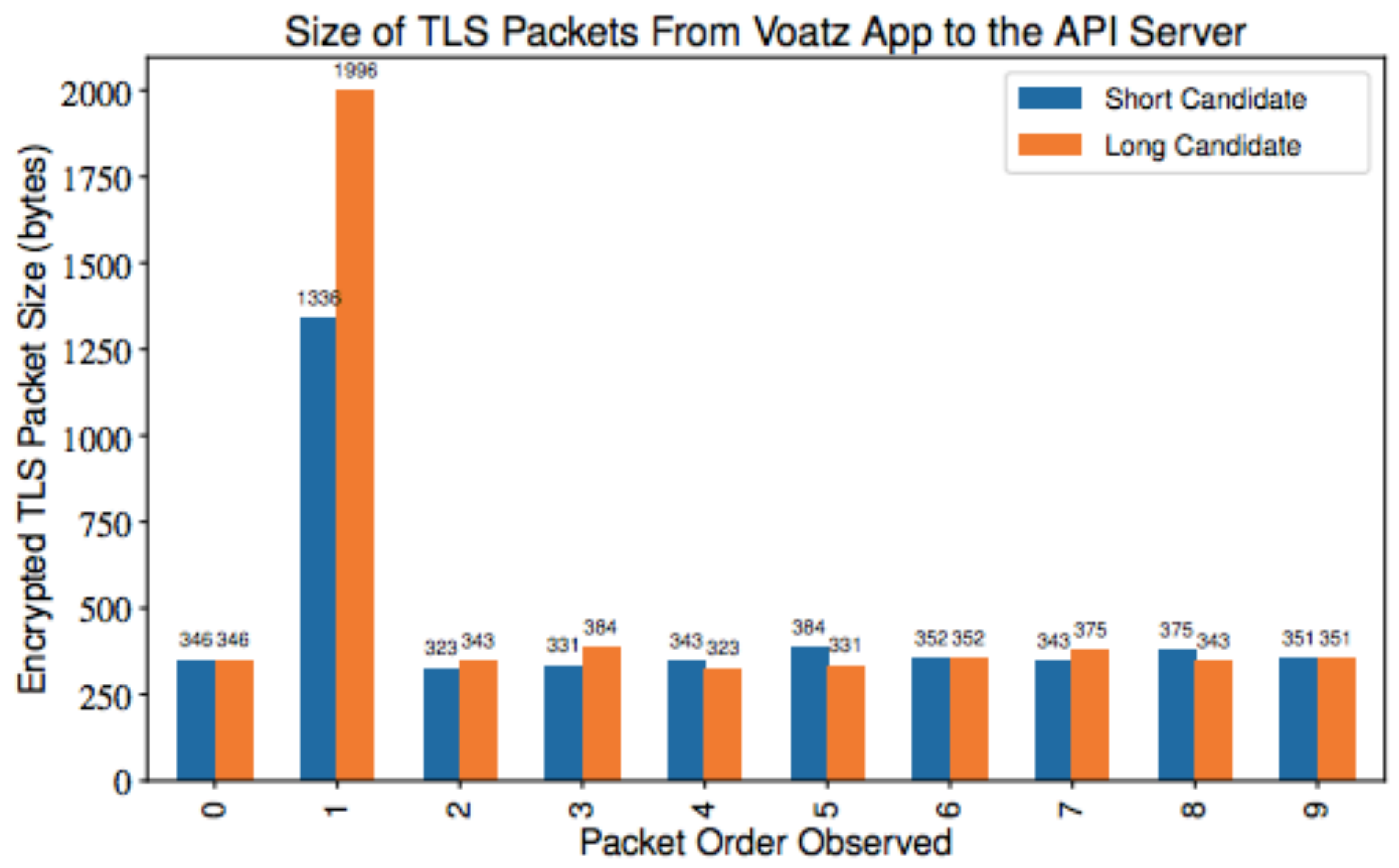
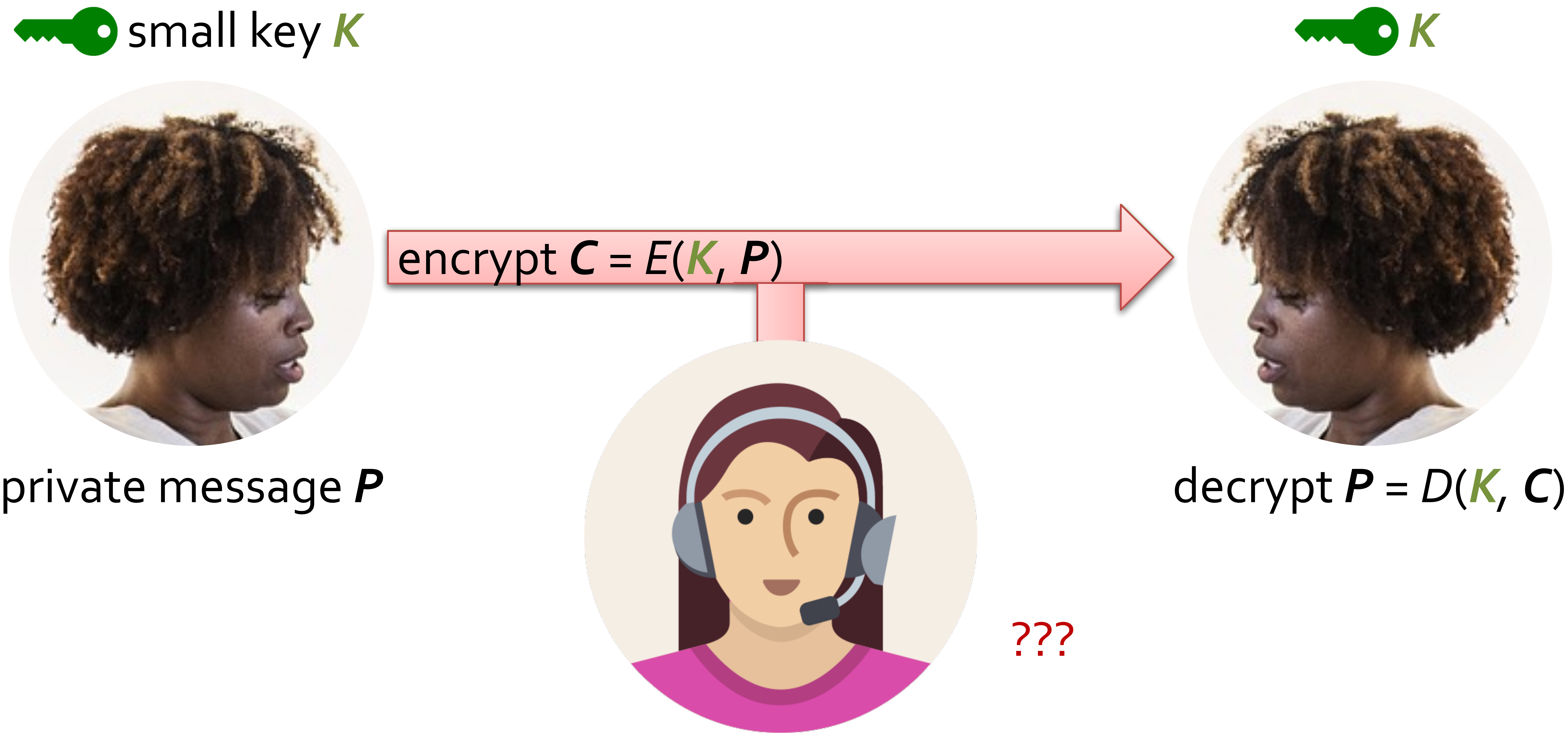


Figure 10: HTTPS encrypted packet lengths immediately after a user submits a vote, in order sent. Note the size of the "short" and "long" candidate in packet 1.

Source: internetpolicy.mit.edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public.pdf

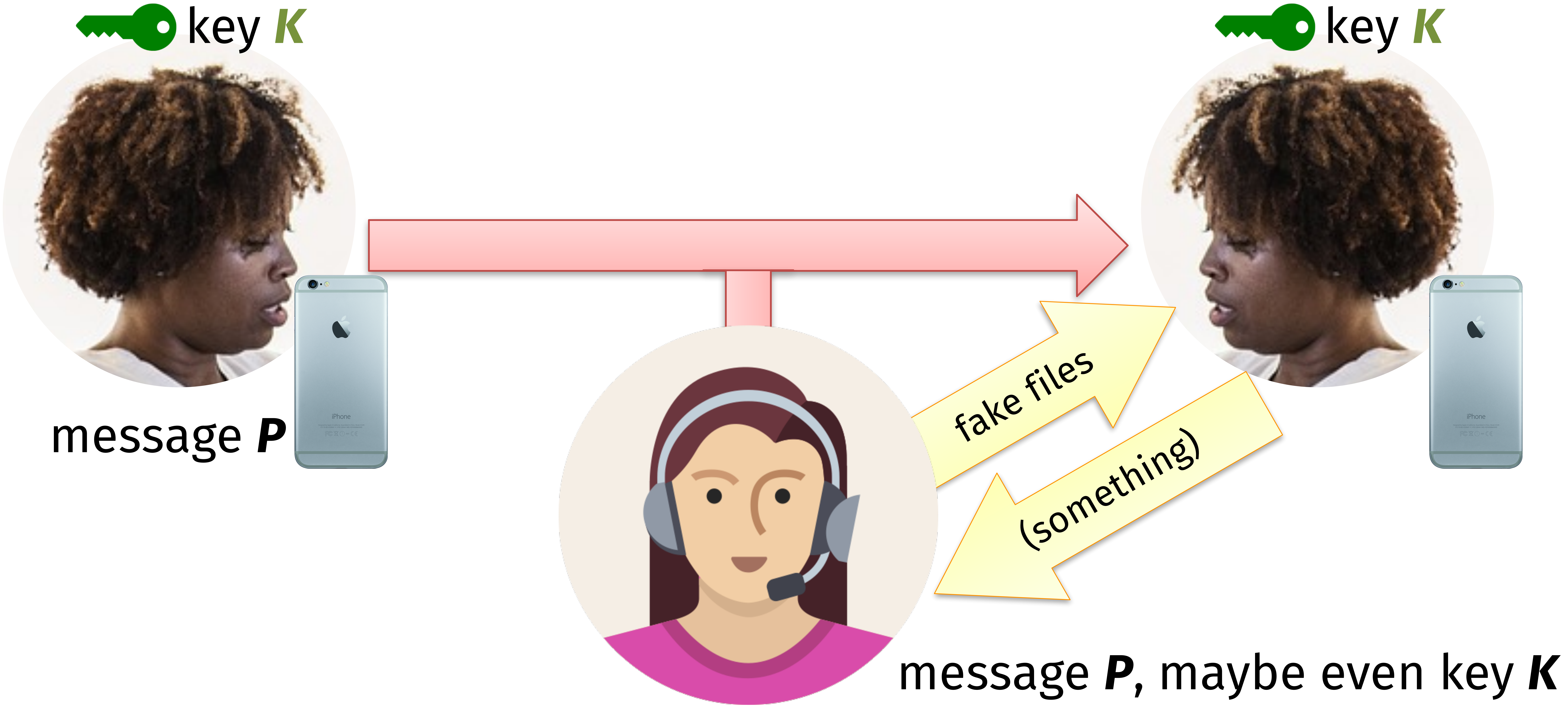
Review: Protecting data confidentiality



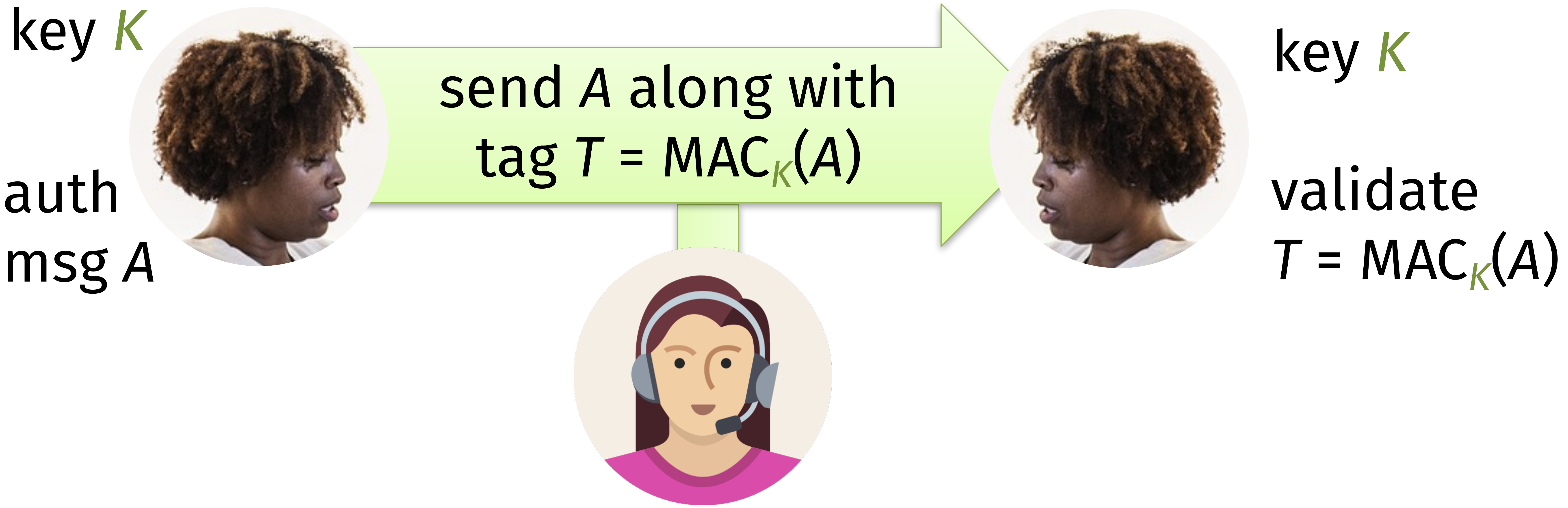
"Confidentiality xor authenticity is **not possible**.
If you don't have both, often you don't have either."

–Prof. Matthew Green, Johns Hopkins

Lack of authenticity → lack of confidentiality



Message authentication code (MAC)

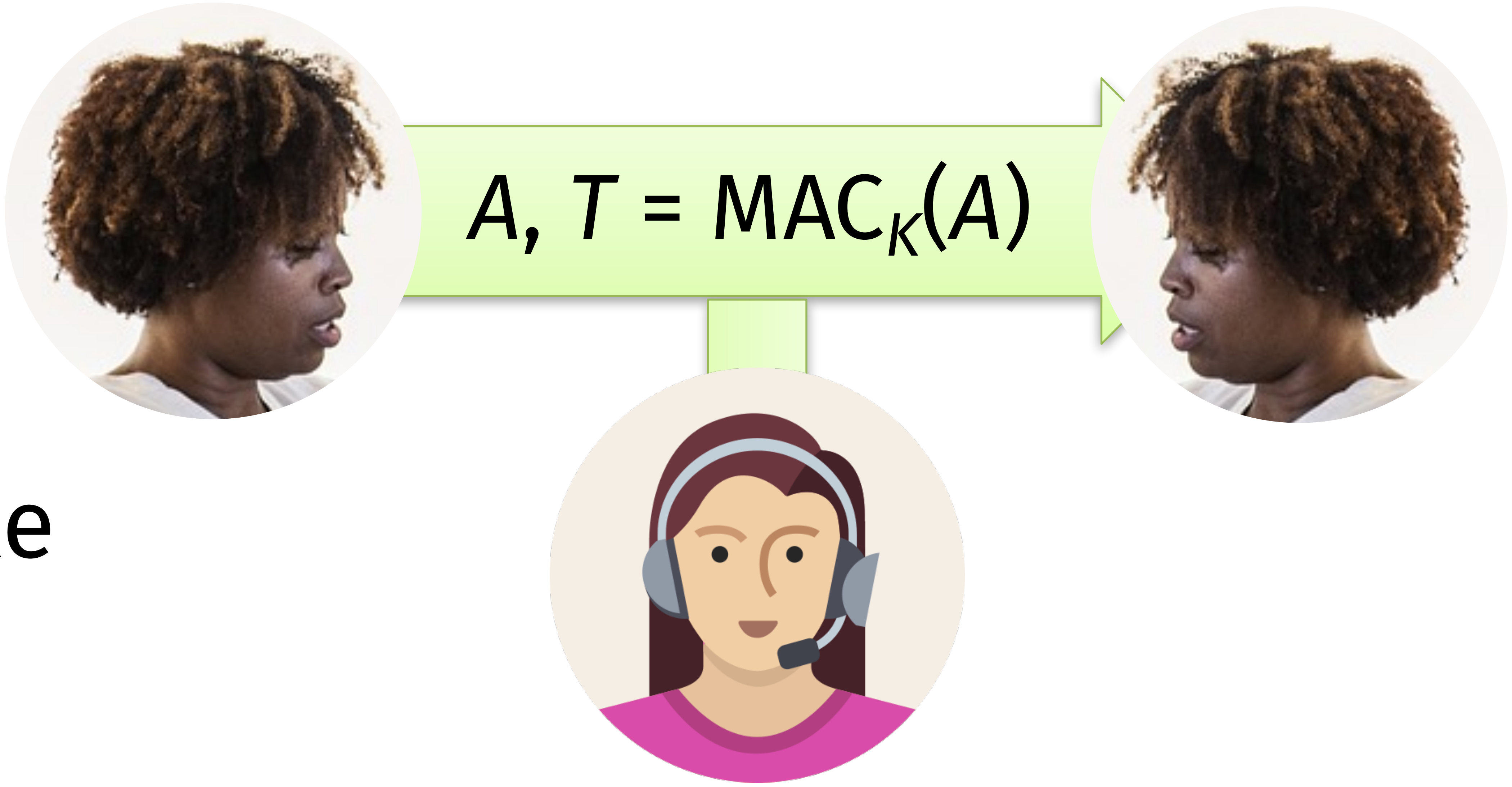


MACs stop an actively malicious Mallory from:
- injecting a new message and tag (A^*, T^*)
- tampering with an existing one

What cryptographic authenticity will *not* do

- *Hide message contents*:
  Need encryption for that

- *Thwart replay attacks*:
  A higher-level protocol needs to handle this, say via nonces or timestamps



Definition: Message authentication code

Algorithms

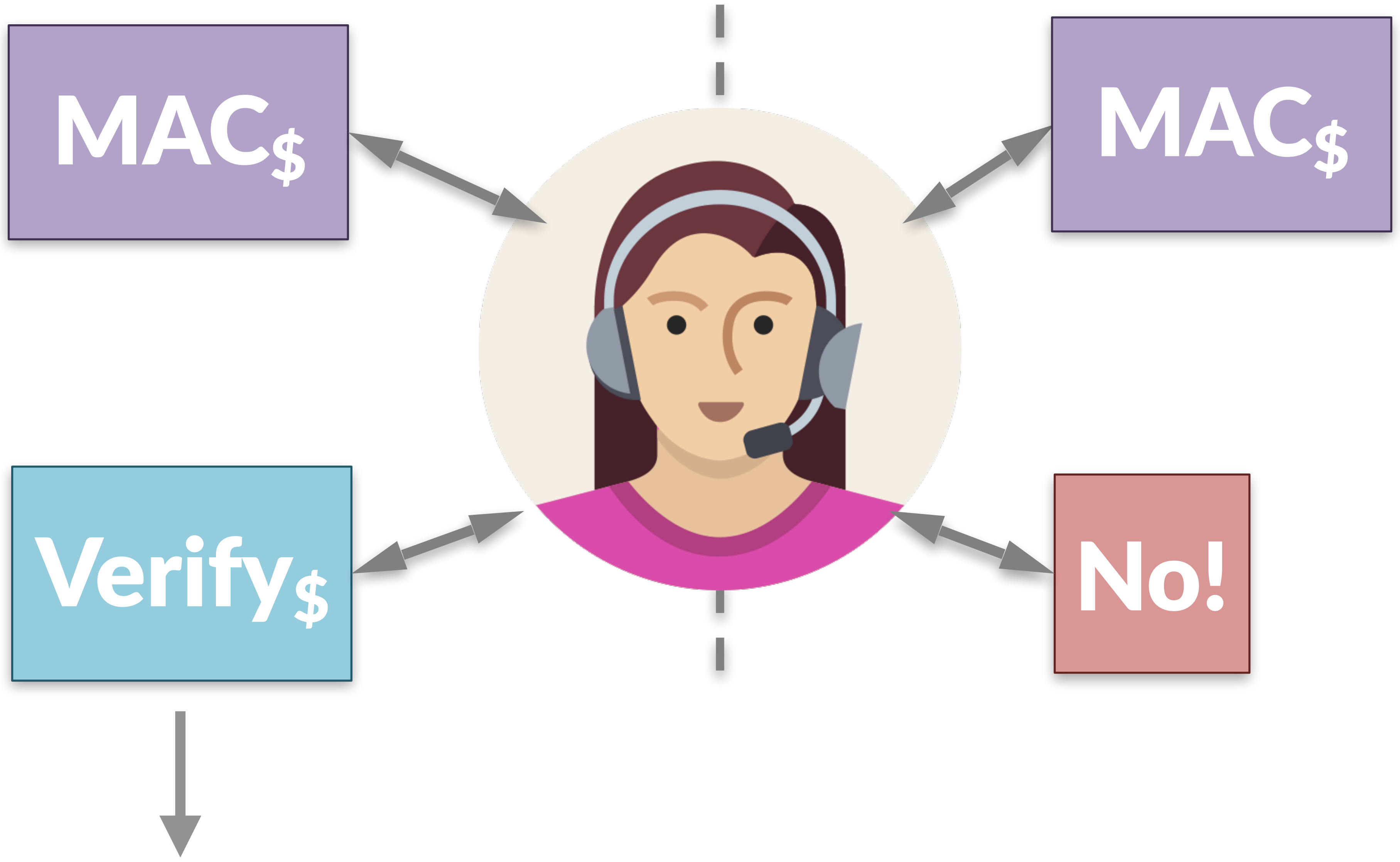
- **KeyGen:** choose key $K \leftarrow \{0,1\}^\lambda$
- **MAC**$_K$($A \in \{0,1\}^\alpha$) → tag $T \in \{0,1\}^\tau$
  - Can be randomized
  - But usually deterministic
  - Prefer short tags: $\tau < \alpha$
- **Verify**$_K$(A, $T \in \{0,1\}^\tau$) → *yes*/*no*

Requirements

- **Performance:** All algorithms are efficiently computable
- **Correctness:** For all K, tags made by MAC$_K$ are accepted by Verify$_K$
- **Security (informal):** Even after observing many (A, T) pairs, Mallory cannot *forge* a *new* one

Formalizing security via existential unforgeability

We say that a MAC satisfies ***(q, t, ε)-existential unforgeability against a chosen message attack*** if all adversaries Mallory that make ≤ q queries and run in time ≤ t can forge a message with probability < ε



Restriction: Mallory cannot verify a MAC tag that Alice produced

Block cipher → MAC

- For our first MAC, let's restrict $|A| = |T|$ = block length of a block cipher
- In this case, simply applying the block cipher suffices to build a MAC!

$$\mathrm{MAC}_K(A) = \mathrm{B}_K(A)$$

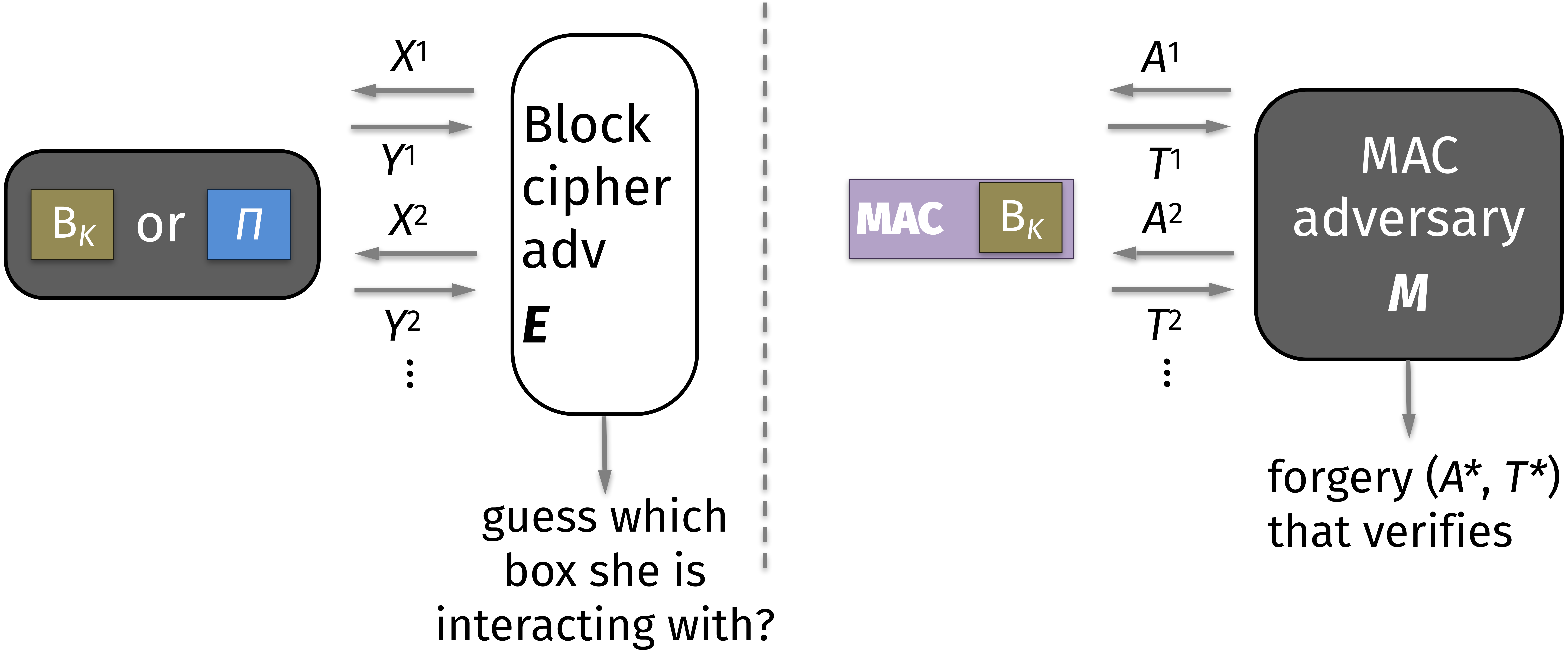
- How do we prove this claim?
  - B_K is pseudorandom, meaning Mallory cannot distinguish it from Π
  - The EU-CMA game is about forgery; it doesn't have an indistinguishability style
- What if we made the MAC from Π rather than B_K?
  - Remember, the output of $\Pi(\mathrm{X})$ doesn't depend on $\Pi(\mathrm{X}')$ for any $\mathrm{X} \neq \mathrm{X}'$

"If an adversary Eve has not **explicitly queried** a [perfect codebook] ***R*** on some point *X*, then the value of ***R***(*X*) is **completely random**… at least as far as Eve is concerned."

–*Jon Katz and Yehuda Lindell*, Introduction to Modern Cryptography

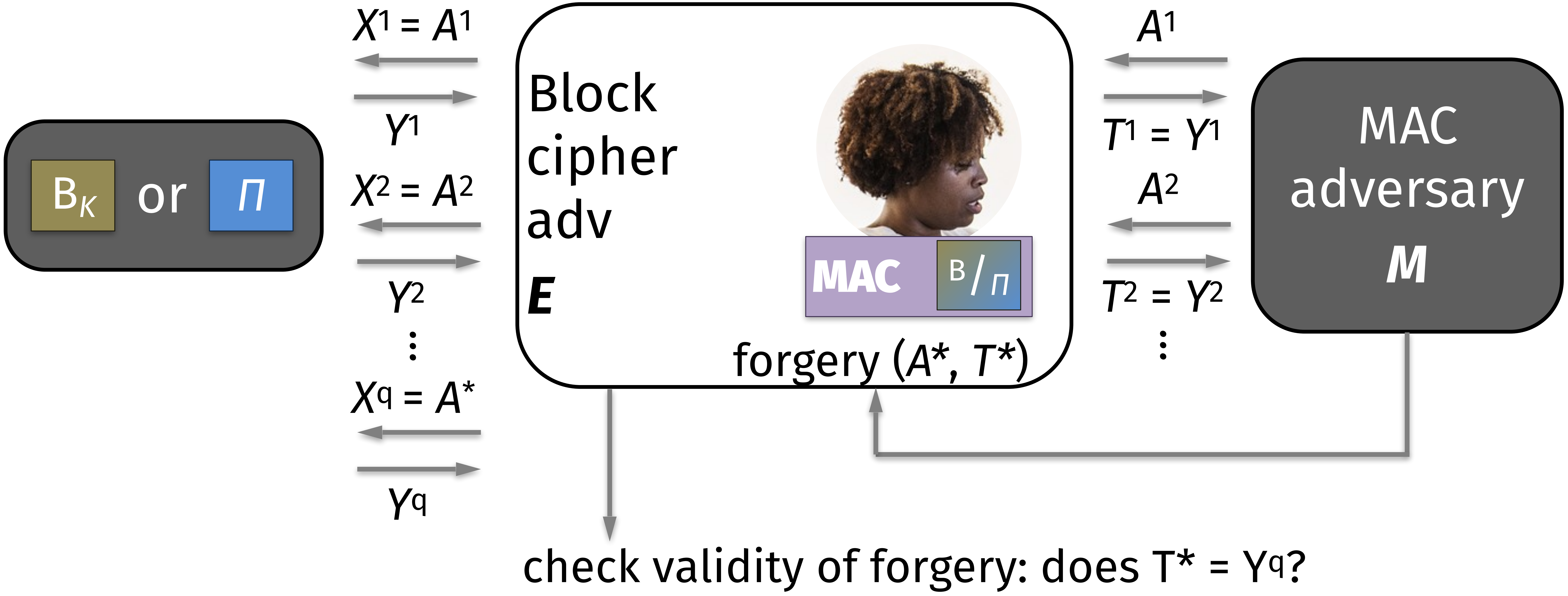
Thm: B_K is pseudorandom → MAC B_K is EU-CMA

Prove the contrapositive: given adversary *Mallory* that *forges* a MAC, we will construct an adversary *Eve* that distinguishes a block cipher from *Π*



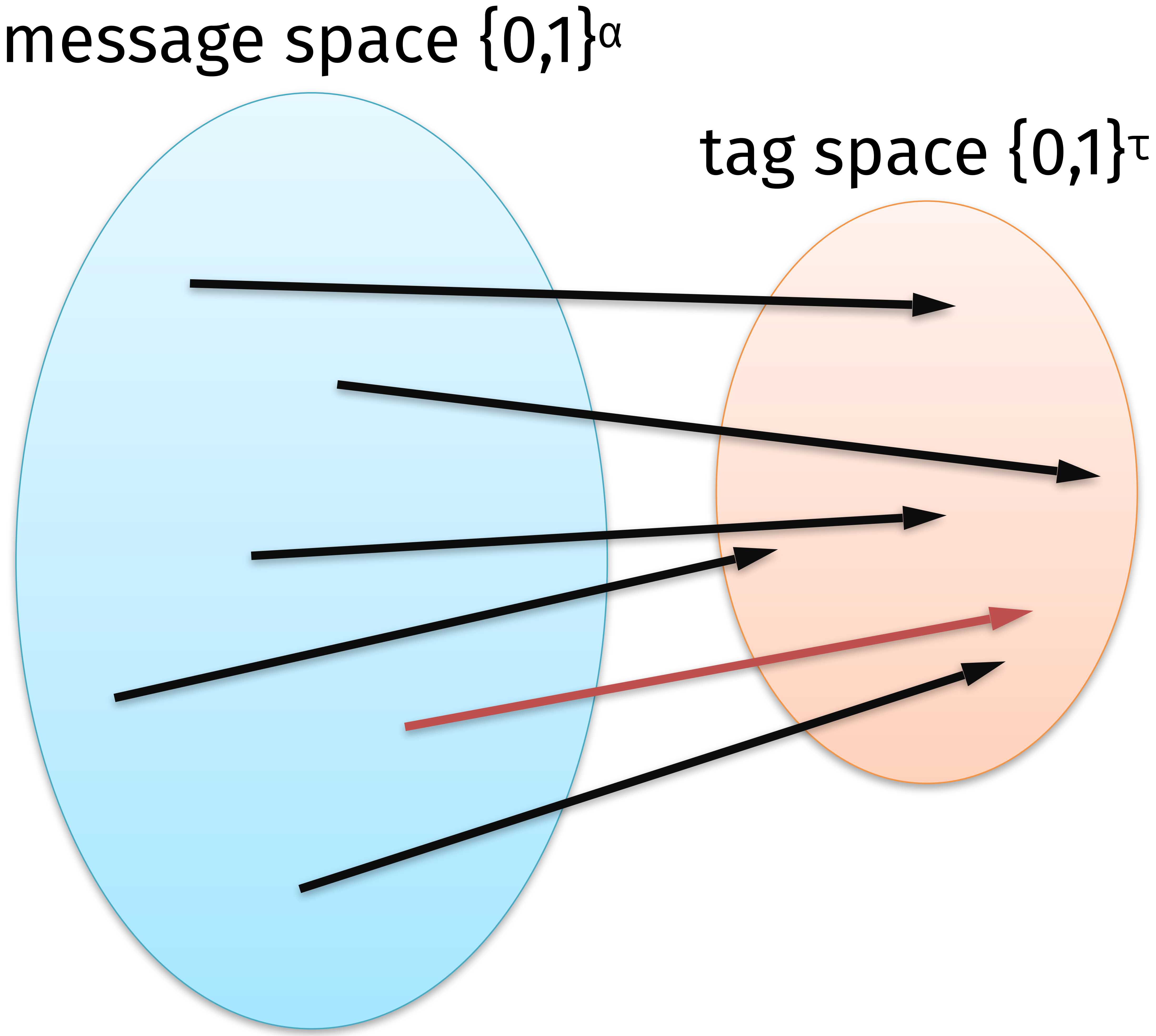
Thm: B_K is pseudorandom → MAC B_K is EU-CMA

Why this works: If E had access to B_K then M can forge. If E had access to Π then Pr[M forges] ≤ $2^{-\tau}$ because $\Pi(A^*)$ is independent of other queries



MACs for longer messages?

- *Performance goal:* minimize space required for MAC tag
- *Security goal:* ensure that MAC remains existentially unforgeable



CBC-MAC: cipher block chaining, revisited

- 1st block simply runs the underlying block cipher (no more nonce/IV!)
- Subsequent inputs to the block cipher depend on both new input + prior output!
- Only the final block tag is revealed $\Rightarrow$ important for performance *and* security