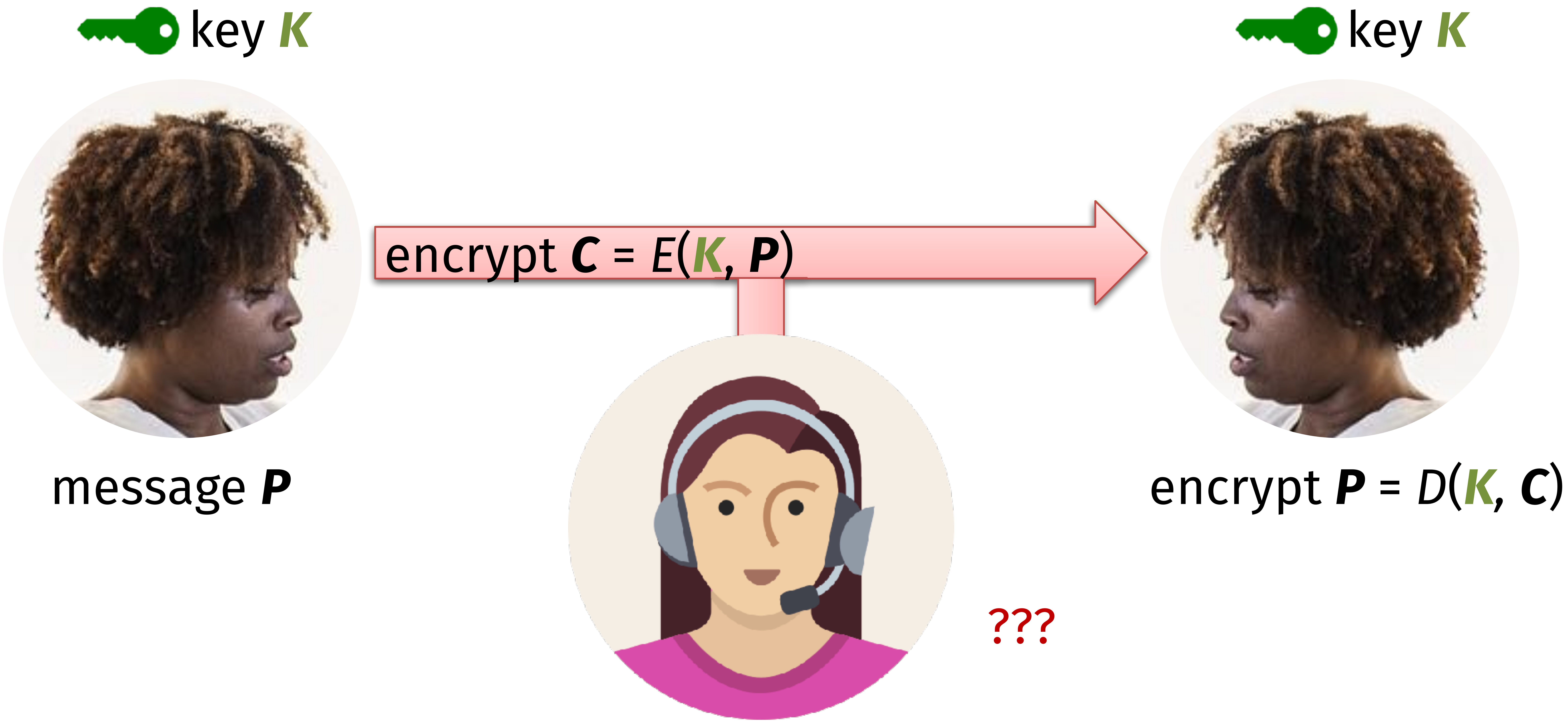
Lecture 8: Data at rest — putting everything together

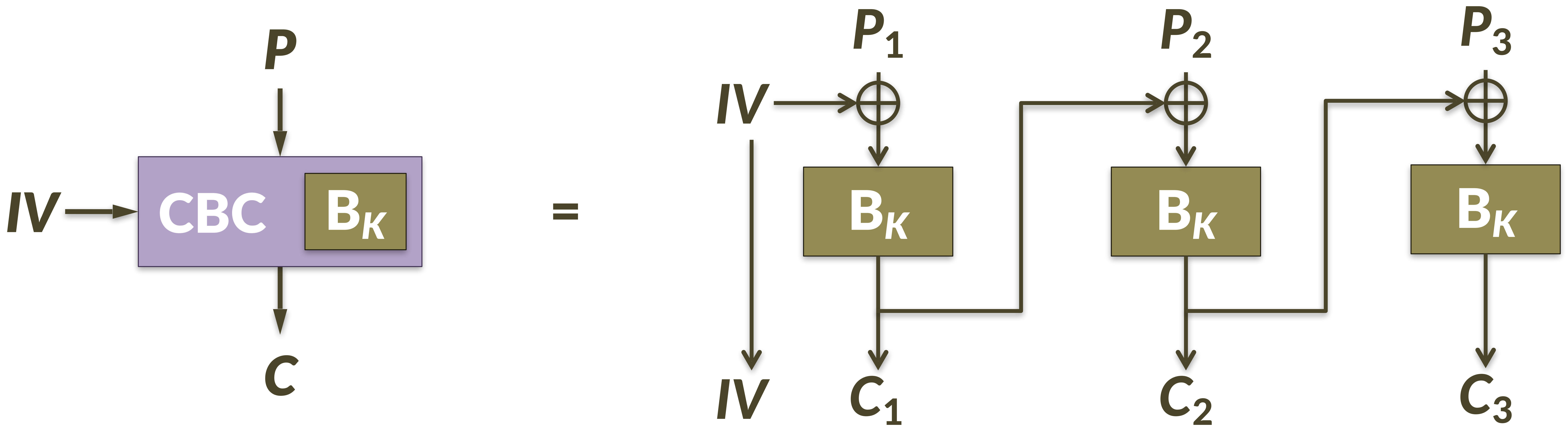
Posted on piazza.com

- Lab 4: due on Monday 2/18 at 11pm
- Midterm next Thursday, February 21
  - Office hours change next week: Tuesday at 9am-noon
  - Posted last year’s exam
  - Tomorrow’s discussion session will focus on test prep

Last time: Protecting *privacy* of data at rest



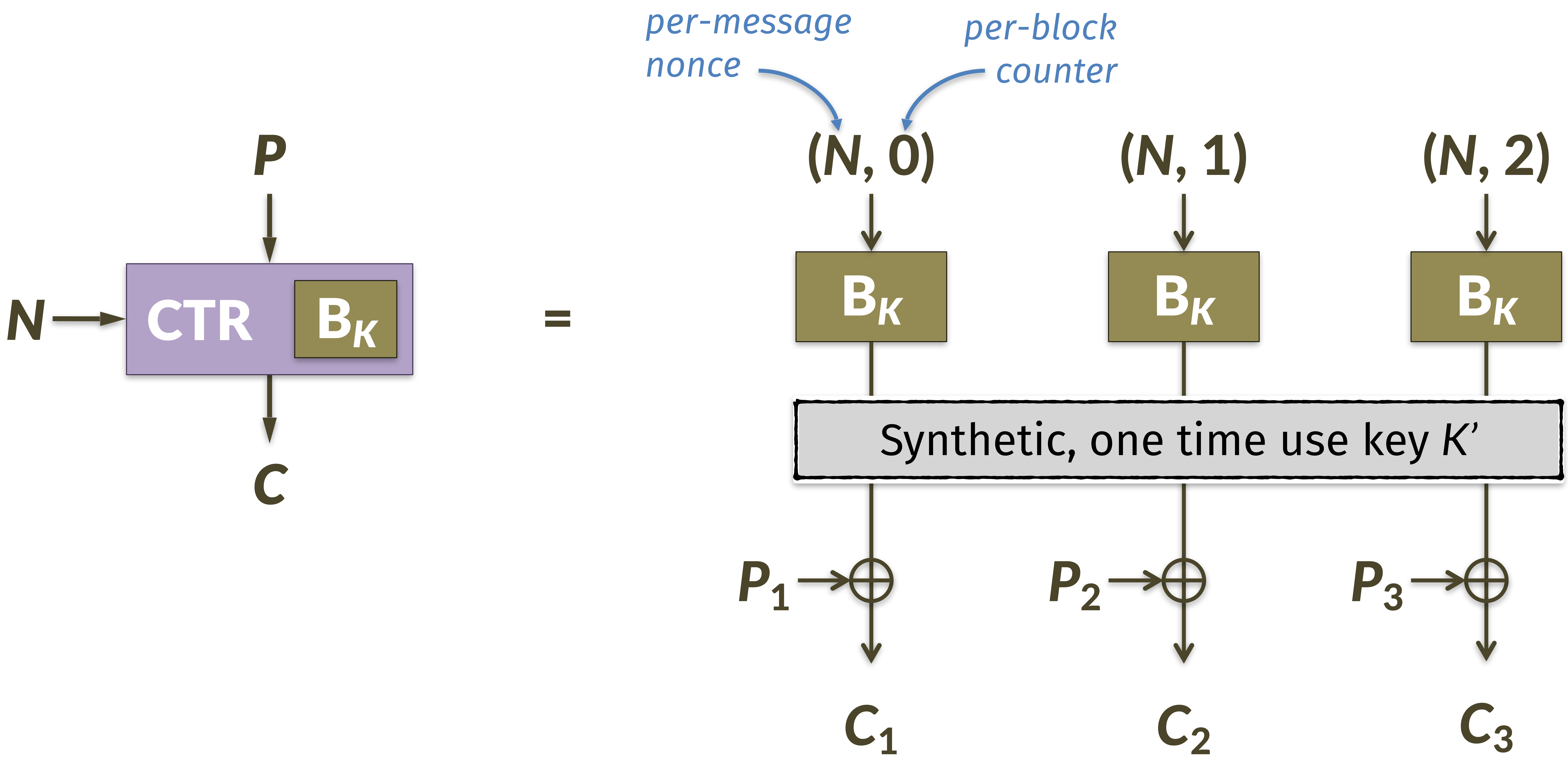
Cipher block chaining (CBC) mode



Two differences from CBC-MAC:

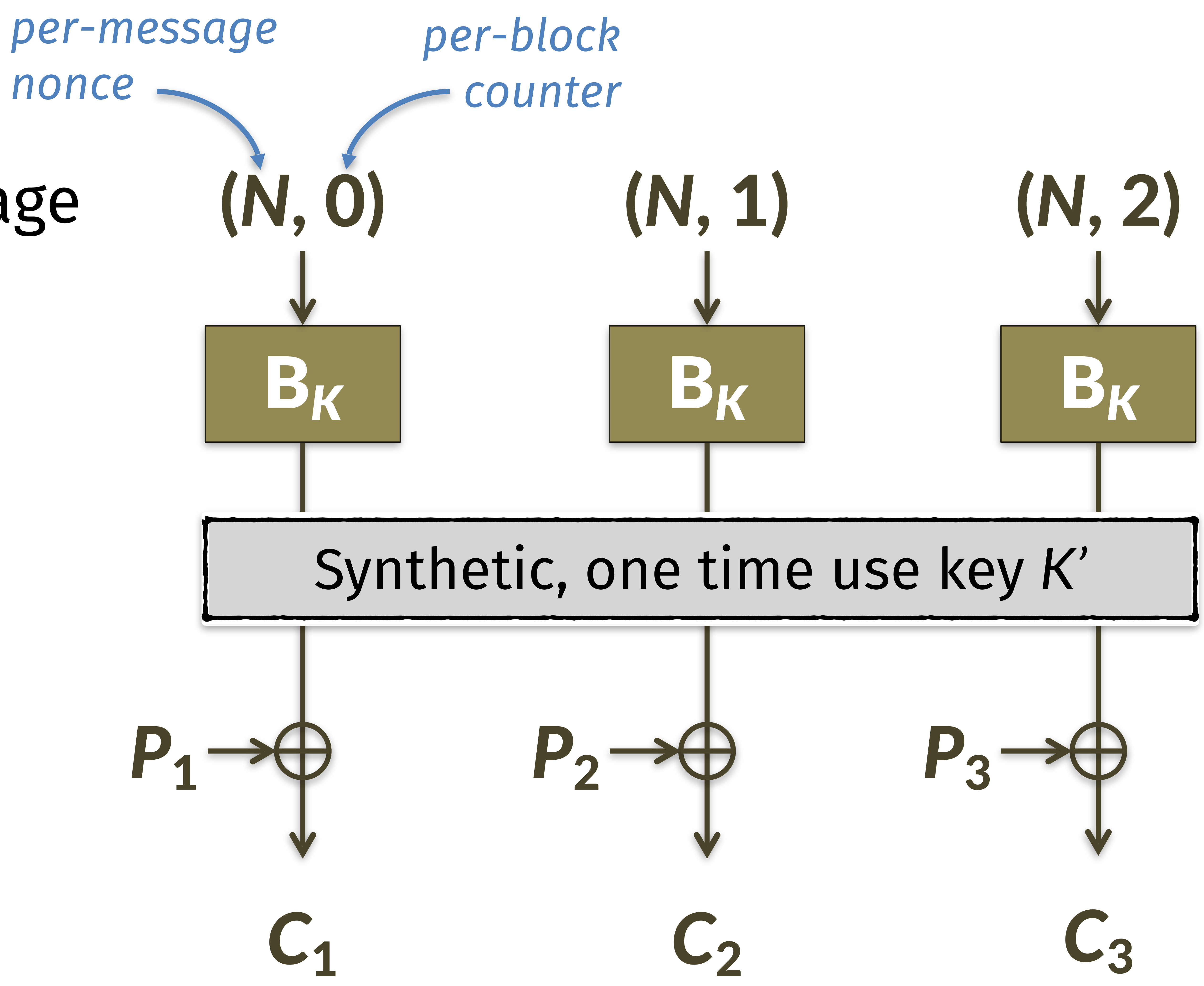
1. All blocks are output
2. First block is protected by a public, randomly chosen *initialization vector*

Counter (CTR) mode

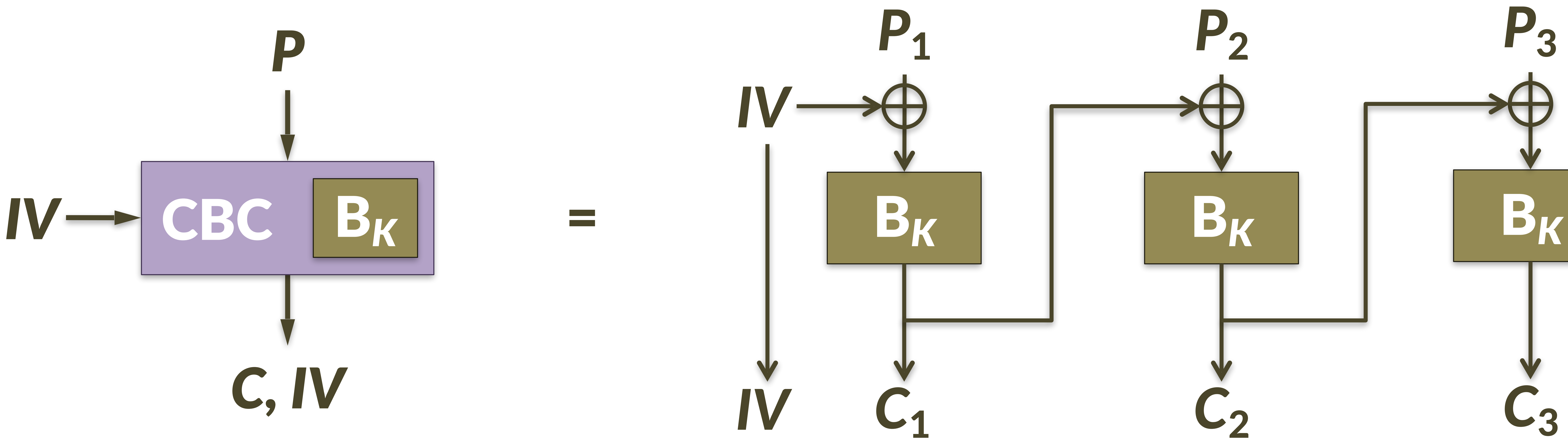


Padding in CTR?

- CTR mode produces a keystream to XOR with message
- If you don't need the full keystream, just discard it
- No need to pad in CTR



Padding in CBC?

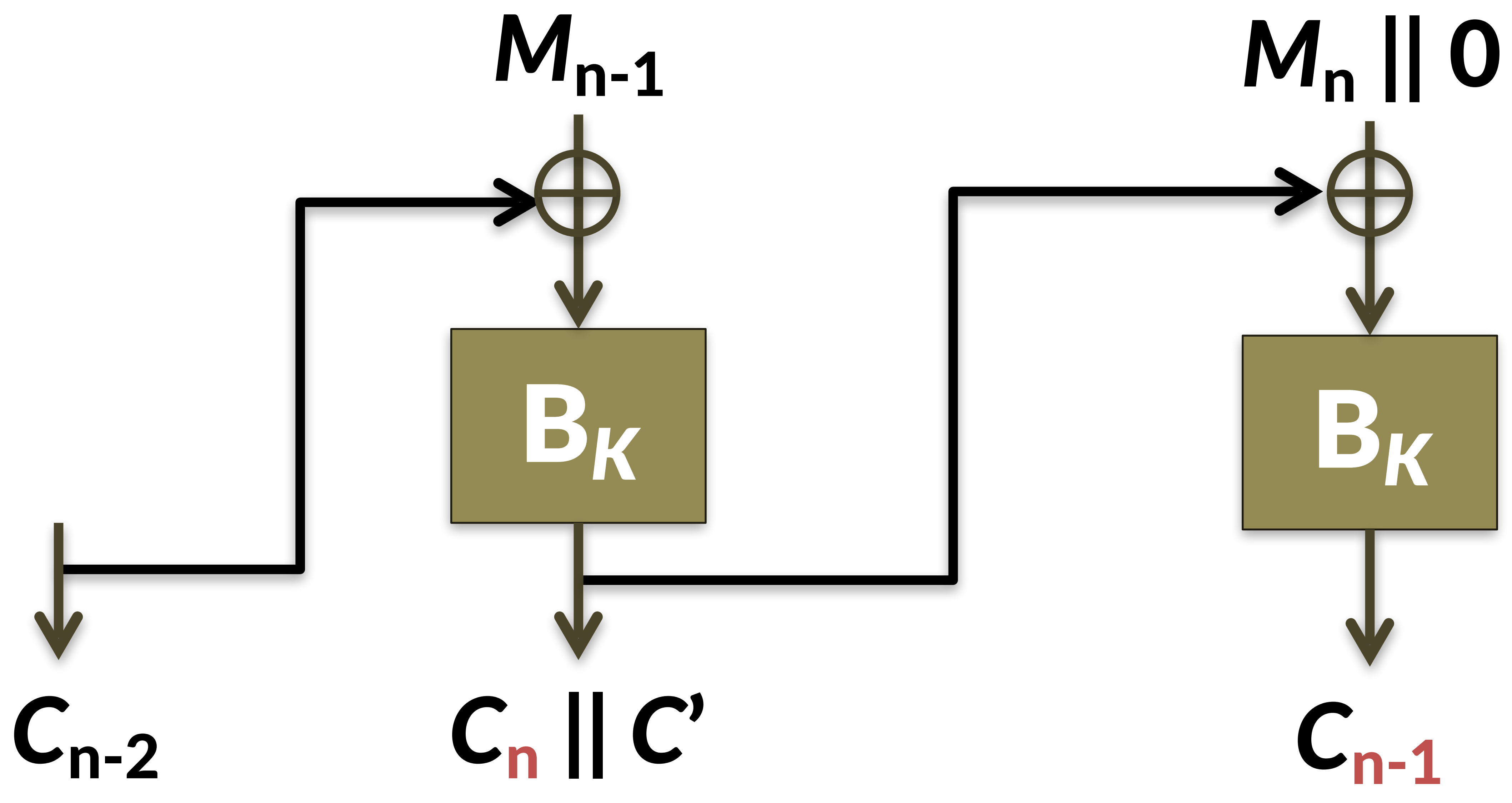


- Not as simple: B_K requires exactly 1 block of text, which means the XOR needs two inputs that are 1 block long
- Seems like padding P_3 is necessary...

Ciphertext stealing for CBC

How to encrypt

- Pad the final block with 0s
  (on its own, this is not invertible)
- Output the entire final block
- For the second-to-last block,
  only output the first $|M_n|$ bytes



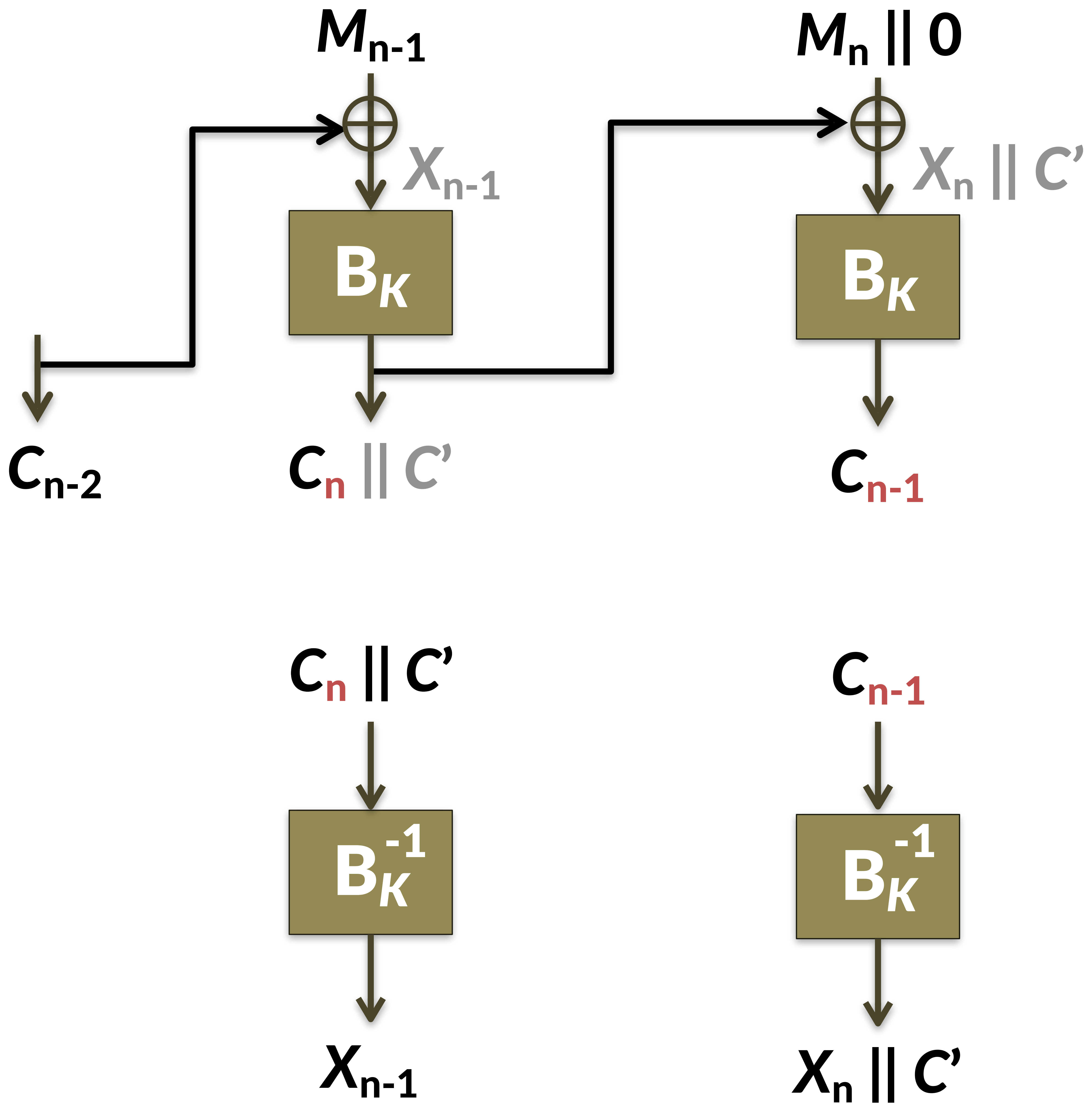
Ciphertext stealing for CBC

<u>How to encrypt</u>

- Pad the final block with 0s
  (on its own, this is not invertible)
- Output the entire final block
- For the second-to-last block, only output the first $|M_n|$ bytes

<u>How to decrypt</u>

- First decrypt the last block
- Data after the first $|M_n|$ bytes == C'
- Now can decrypt the penultimate block

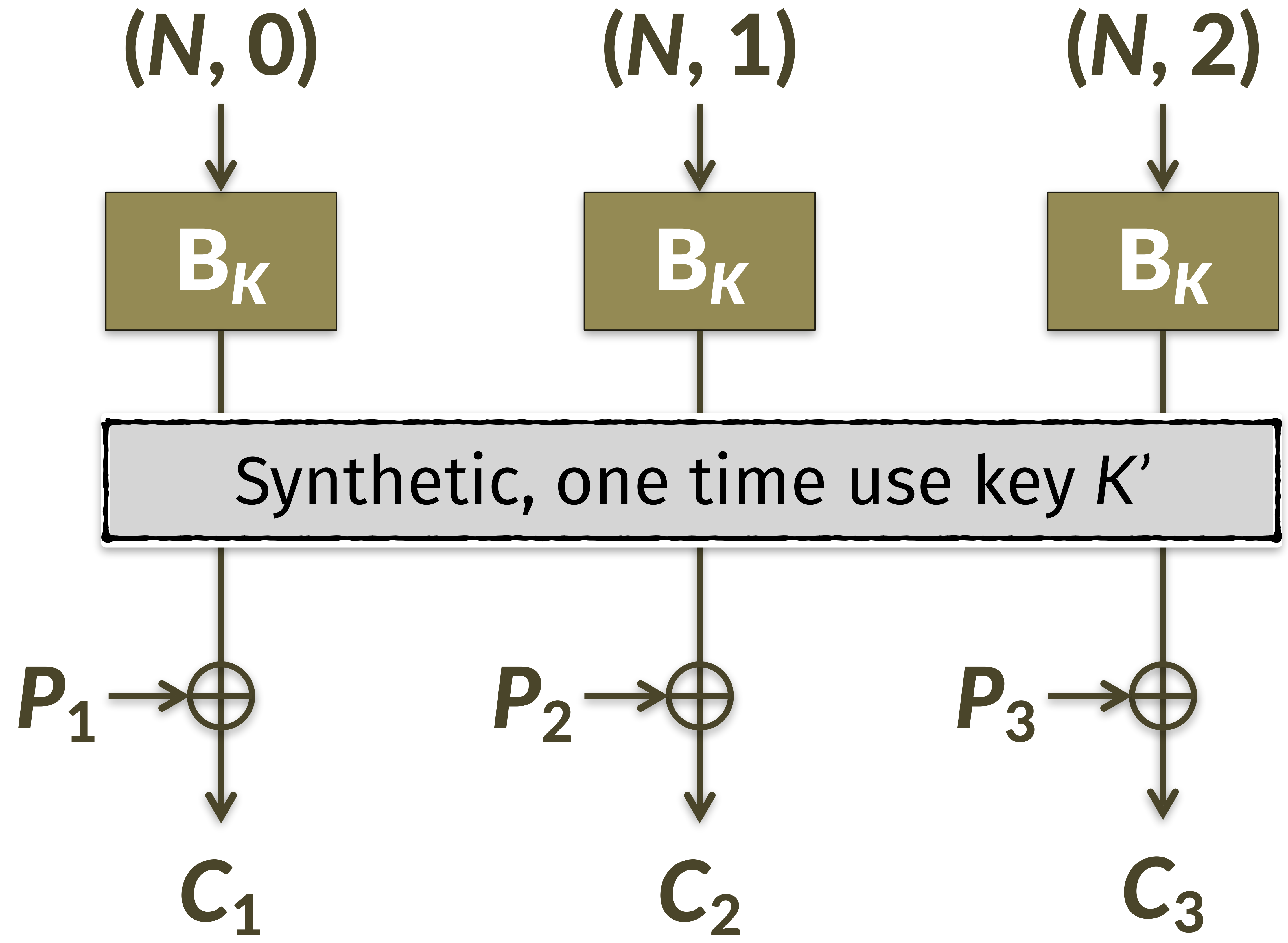


Privacy → Authenticity?

Q: Why don't our existing encryption schemes provide authenticity?

A: Encryption schemes can be malleable

- ECB: Blocks are independent
- CBC: One bit flip in C → one bit flip in next block of P (and destroying the current one)
- CTR: One bit flip in C → one bit flip in same block of P since it is a one-time pad



Today's objectives

- See *where* + *how* symmetric encryption primitives are used in practice
- See *why* they improve privacy

Data at rest protection on laptops

- Goal:
  - Data on a hard disk cannot be tampered with or exfiltrated
  - Even if the laptop is left unattended, lost, or stolen
- Several products
  - Microsoft Bitlocker, standard on Windows 8 & 10
  - Apple FileVault, on by default from Mac OS X Yosemite (10.10)
  - Linux dm-crypt
  - Third-party products like TrueCrypt and SecureDoc (https://en.wikipedia.org/wiki/Comparison_of_disk_encryption_software)

Data at rest protection on laptops: 4 components

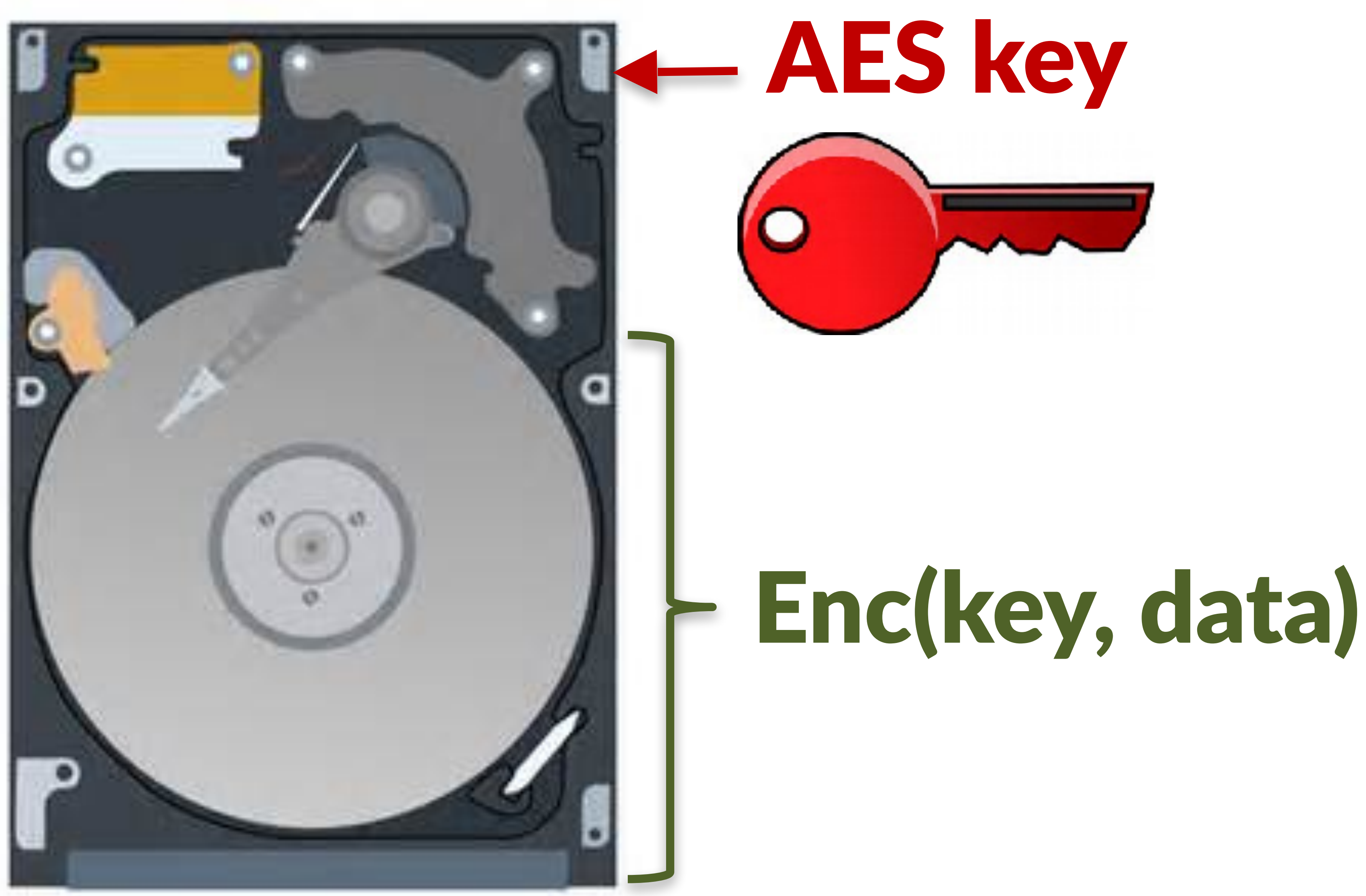
1. **Disk**
   - Long-term storage
   - Always encrypted

2. **Memory**
   - Ephemeral storage
   - Erased upon shutdown*

3. **CPU**
   - Performs crypto quickly

4. **Trusted Platform Module**
   - Stores crypto keys**

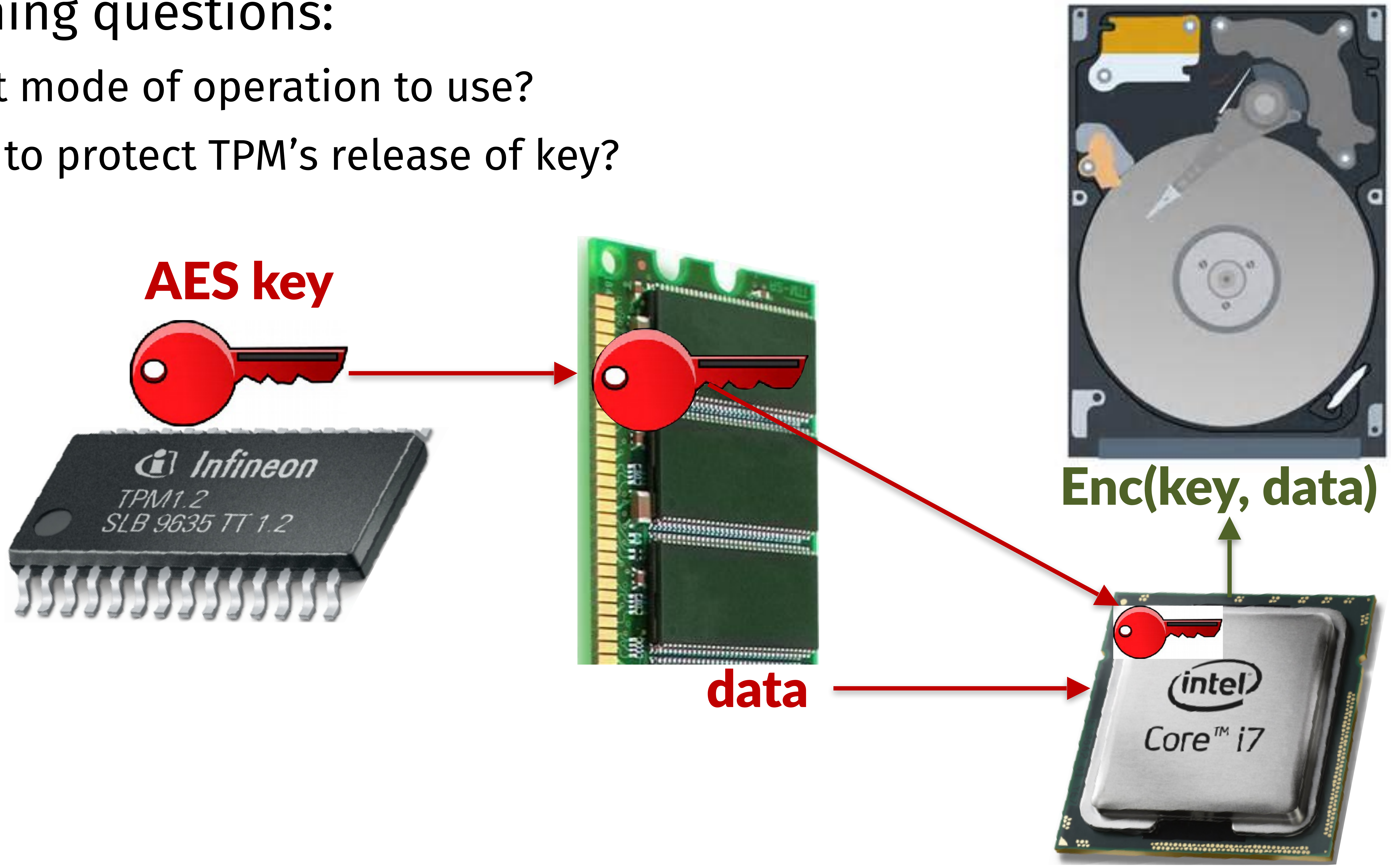
Disk encryption: Attempt 1



Disk encryption: Attempt 2

Remaining questions:

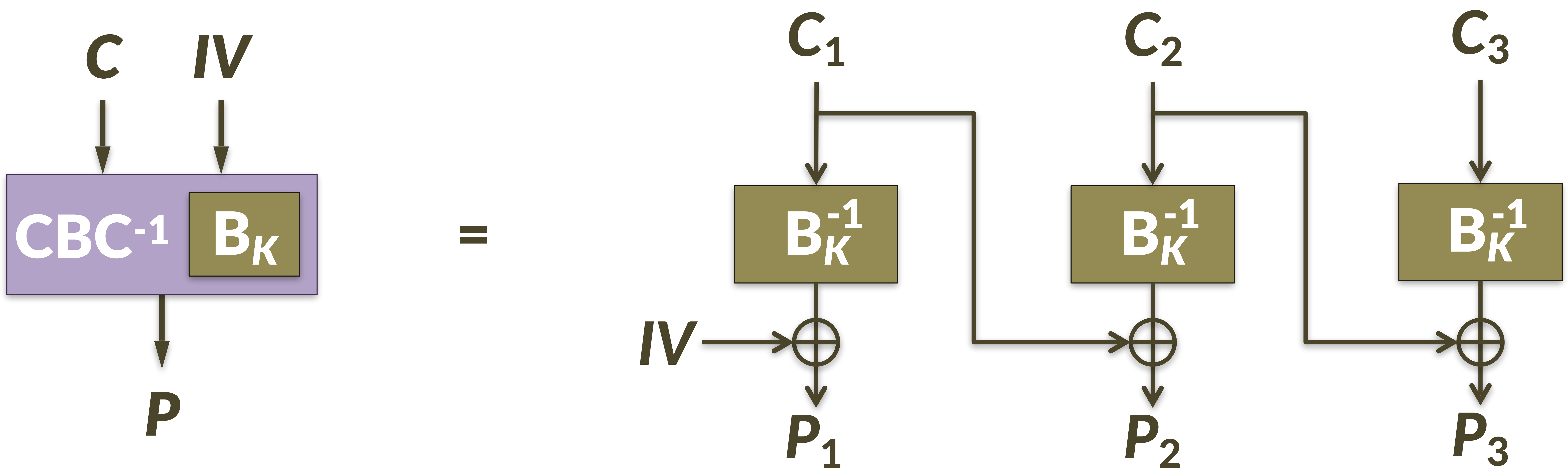
1. What mode of operation to use?
2. How to protect TPM's release of key?



Use CBC mode?

Ferguson (2006): Lacks diffusion in the CBC decryption operation.

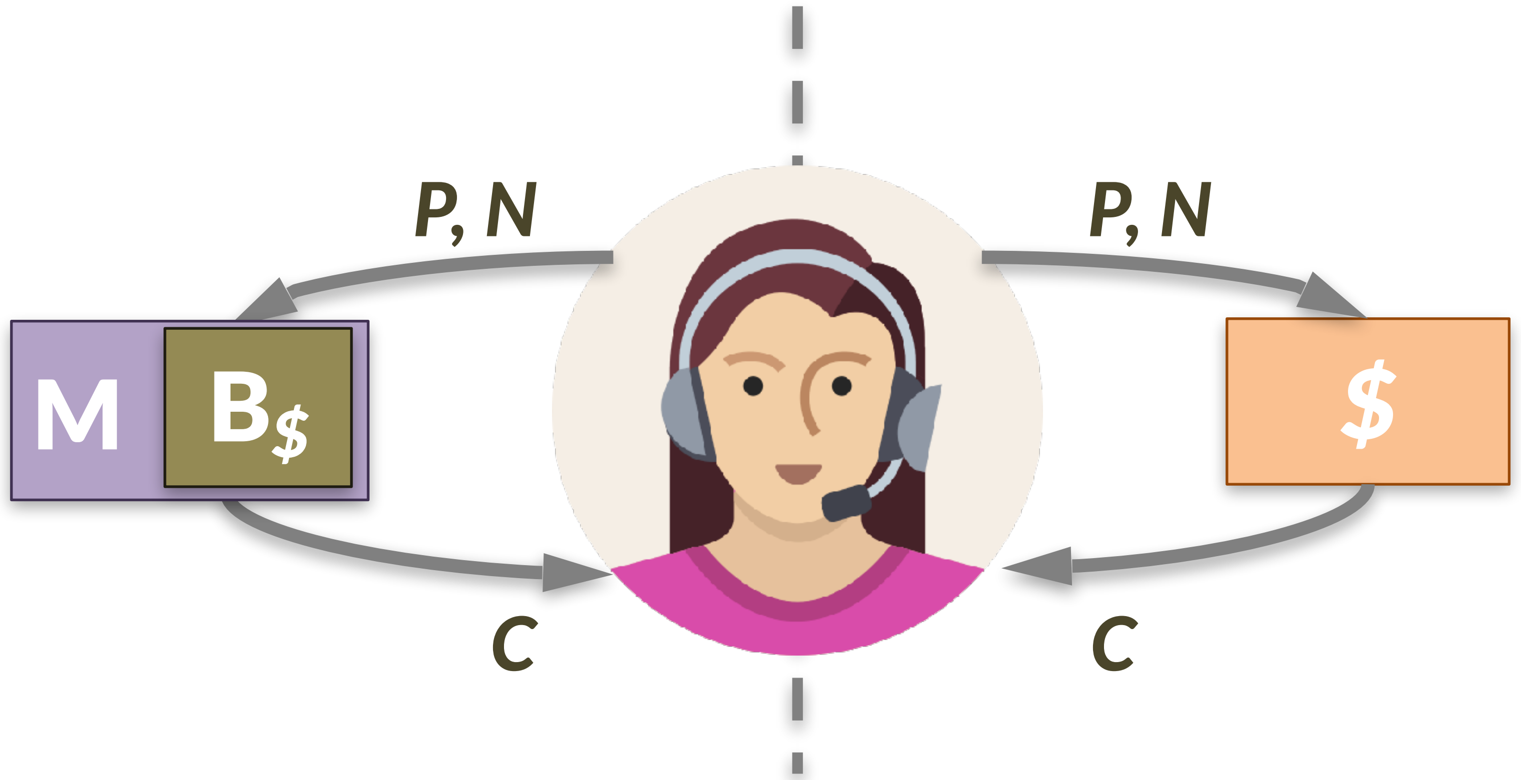
If the attacker introduces a change Δ in ciphertext block i, then plaintext block i is randomized, but plaintext block i+1 is changed by Δ. In other words, the attacker can flip arbitrary bits in one block at the cost of randomizing the previous block.



Disk encryption: Threat model

Attacker can:

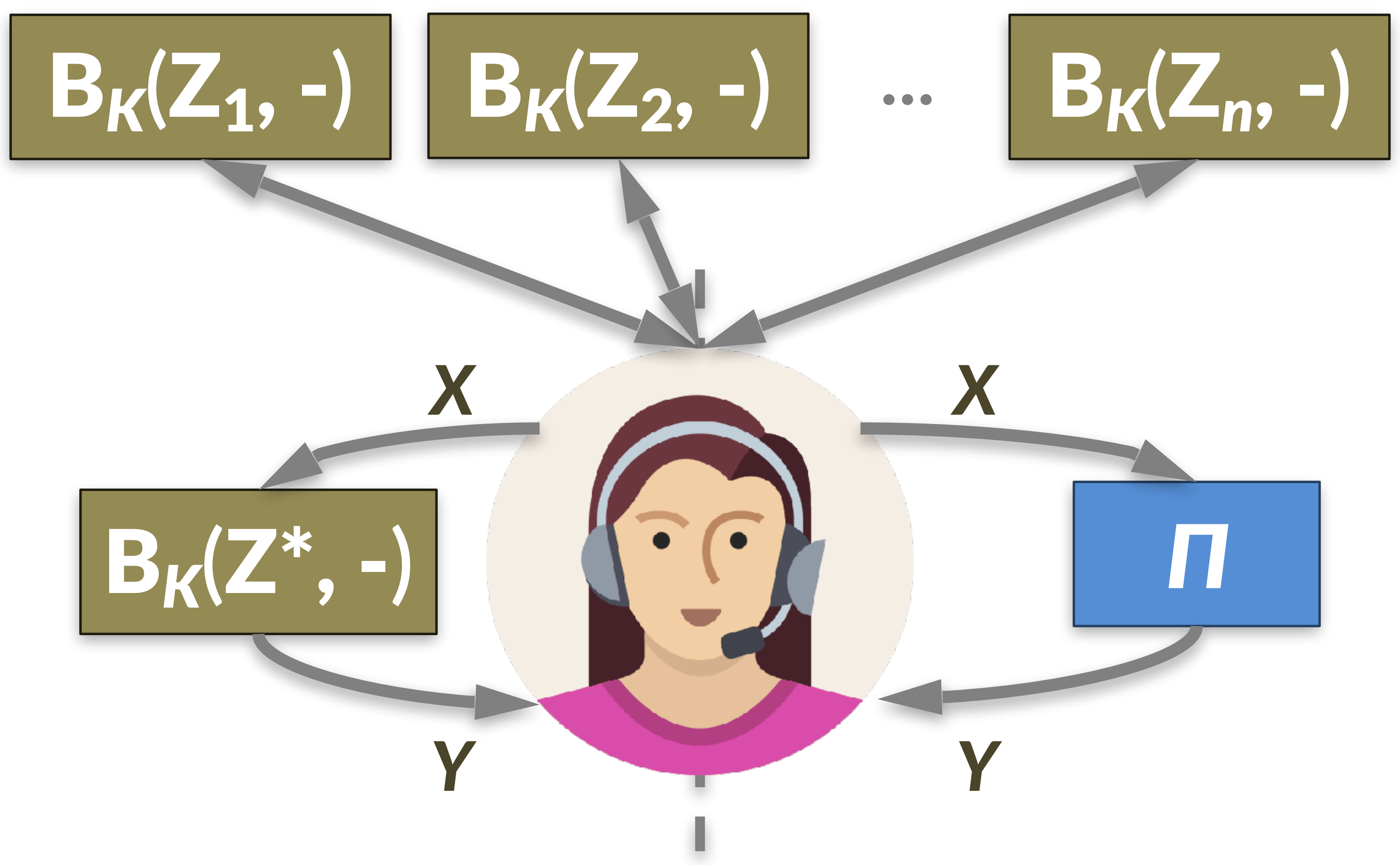
1. Read contents of disk at any time
2. Request disk to encrypt files of its choosing
3. Modify files/sectors on disk
   and request their decryption



Think: persistent malware!

Consequences of threat model:

- Encrypt each sector of disk individually
- No two sectors should be processed identically → Tweakable encryption



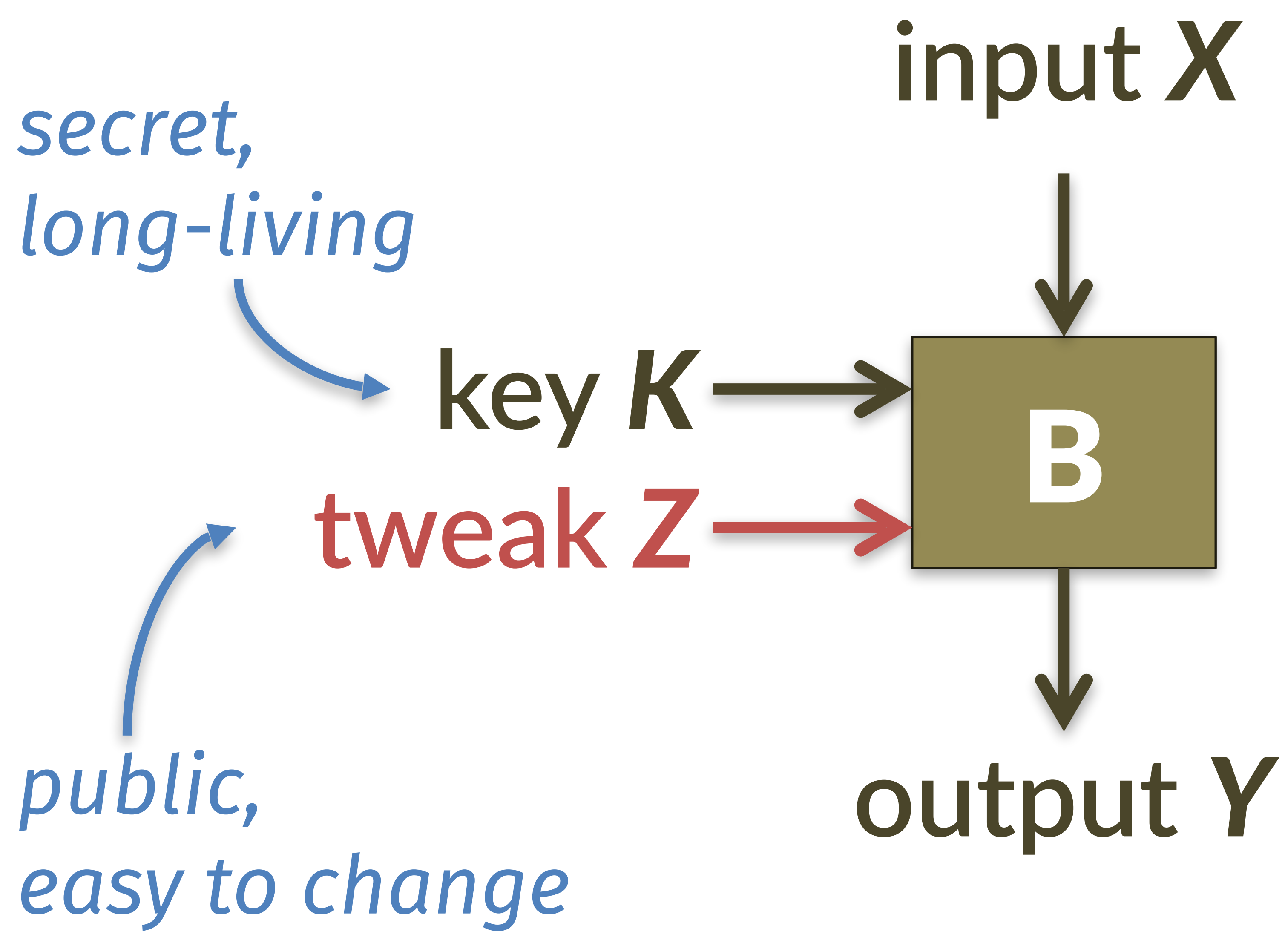
Tweakable encryption

- Encryption extends block ciphers in two ways
  - Add nonces for variety
  - Provide a mode of operation that supports long messages
- Tweakable block ciphers incorporate the first goal directly into BCs

Tweakable block ciphers

History

- First proposed concretely by the "Hasty Pudding Cipher," a first-round submission to the AES competition
- Codified later by Liskov, Rivest, Wagner [Crypto 2002]

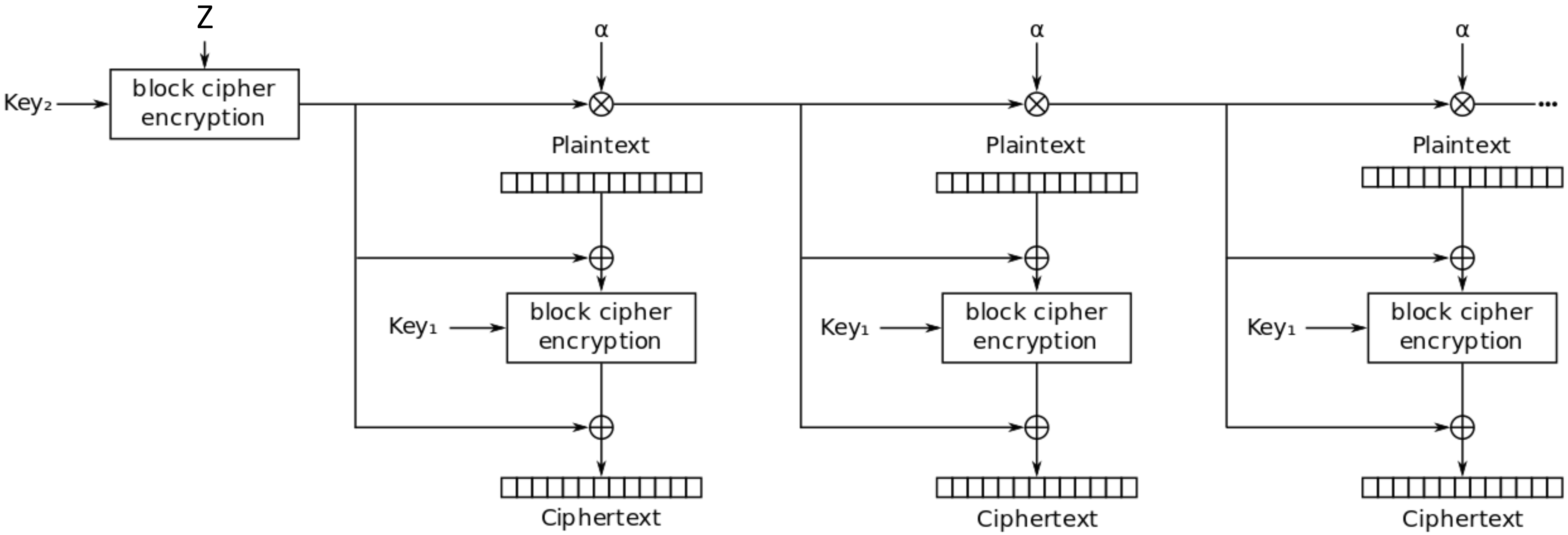


Objectives

- *Variety:* tweak is public, yet cipher with different tweaks act in an "uncorrelated" manner
  - Even a good block cipher B_K can be broken with access to B_{K^*}
  - For a TBC, access to $B_{K,Z}$ provides no help in breaking B_{K,Z^*}
- *Agility:* faster to change tweak than key (avoids key setup/expansion)

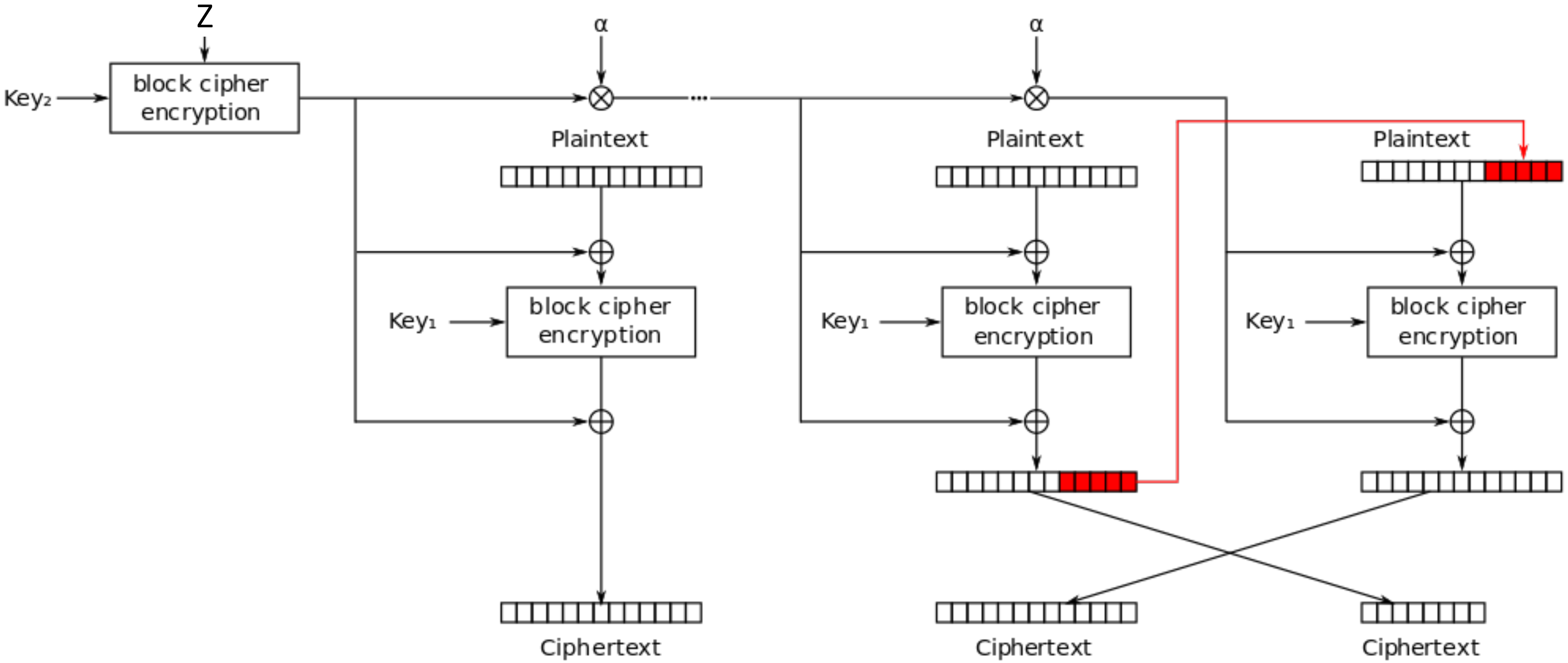
XEX mode

- Tweakable cipher designed in 2004 by Philip Rogaway
- Even-Mansour style with constantly-changing tweak
- Tweak Z = sector number



XEX mode encryption

XTS mode



XEX with tweak and ciphertext stealing (XTS) mode encryption

Key wrapping

- What if multiple users should read the disk?
  - Different user accounts on the machine
  - Recovery key stored in a separate, safe place
- Don’t want to encrypt the entire drive several times!

$\mathrm{Enc}_{K1}(\mathrm{file})$

$\mathrm{Enc}_{K2}(\mathrm{file})$

$\mathrm{Enc}_{K3}(\mathrm{file})$

Key wrapping

- What if multiple users should read the disk?
  - Different user accounts on the machine
  - Recovery key stored in a separate, safe place
- Don't want to encrypt the entire drive several times!
- Key wrapping = protect one key under another
  - Intuitively think of it as $\mathrm{Wrap}_K(K') = \mathrm{Enc}_K(K')$
  - We'll see later that not all encryption works to protect keys
  - Tl;dr: use SIV mode

$\mathrm{Wrap}_{K1}(\mathrm{ek})$

$\mathrm{Wrap}_{K2}(\mathrm{ek})$

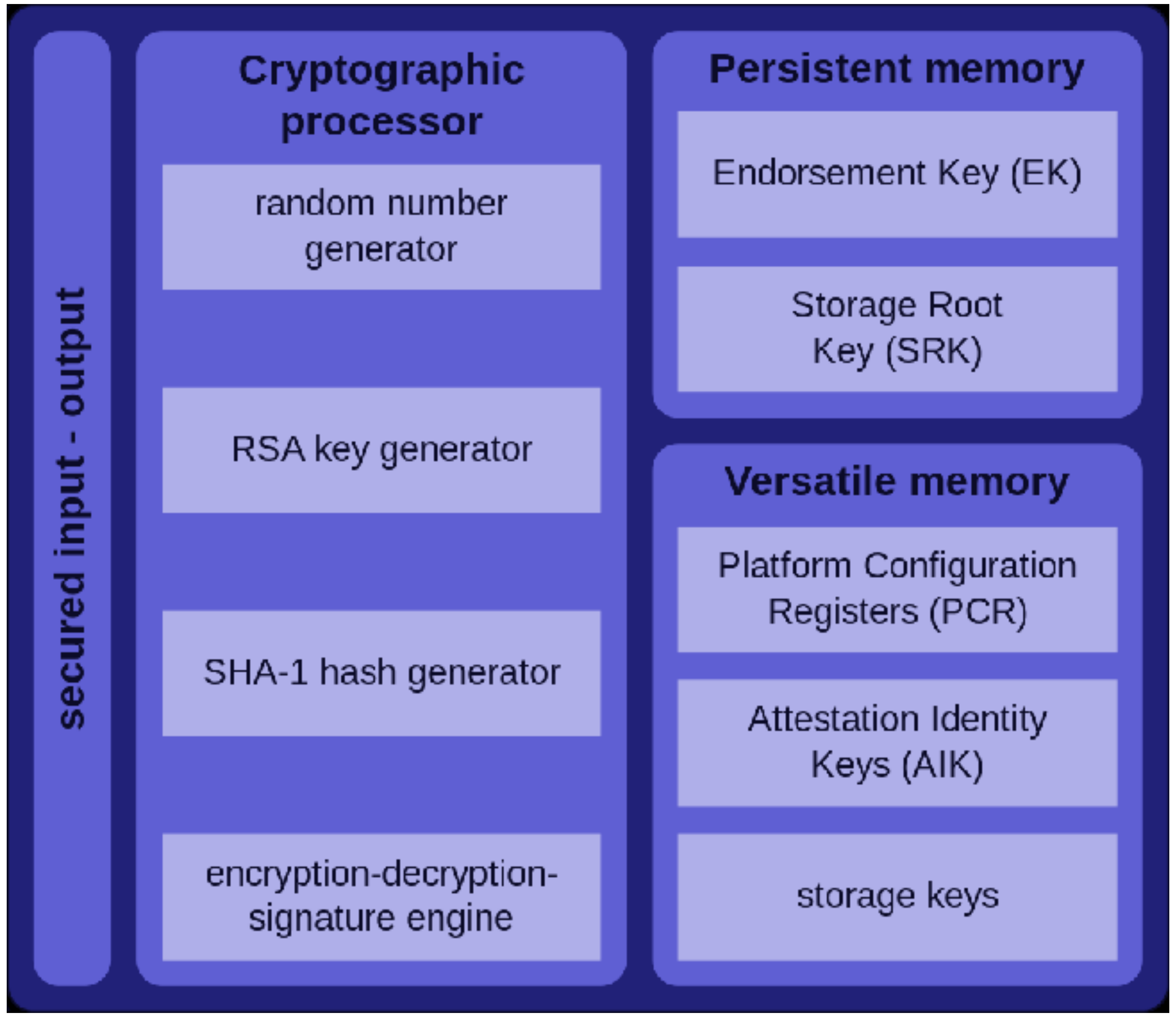
$\mathrm{Wrap}_{K3}(\mathrm{ek})$

$\mathrm{Enc}_{ek}(\mathrm{file})$

Trusted Platform Module

Generate key rather than storing it!

- User's master password (use PBKDF2, which we will discuss later in detail)
- Machine's state ("sealing")



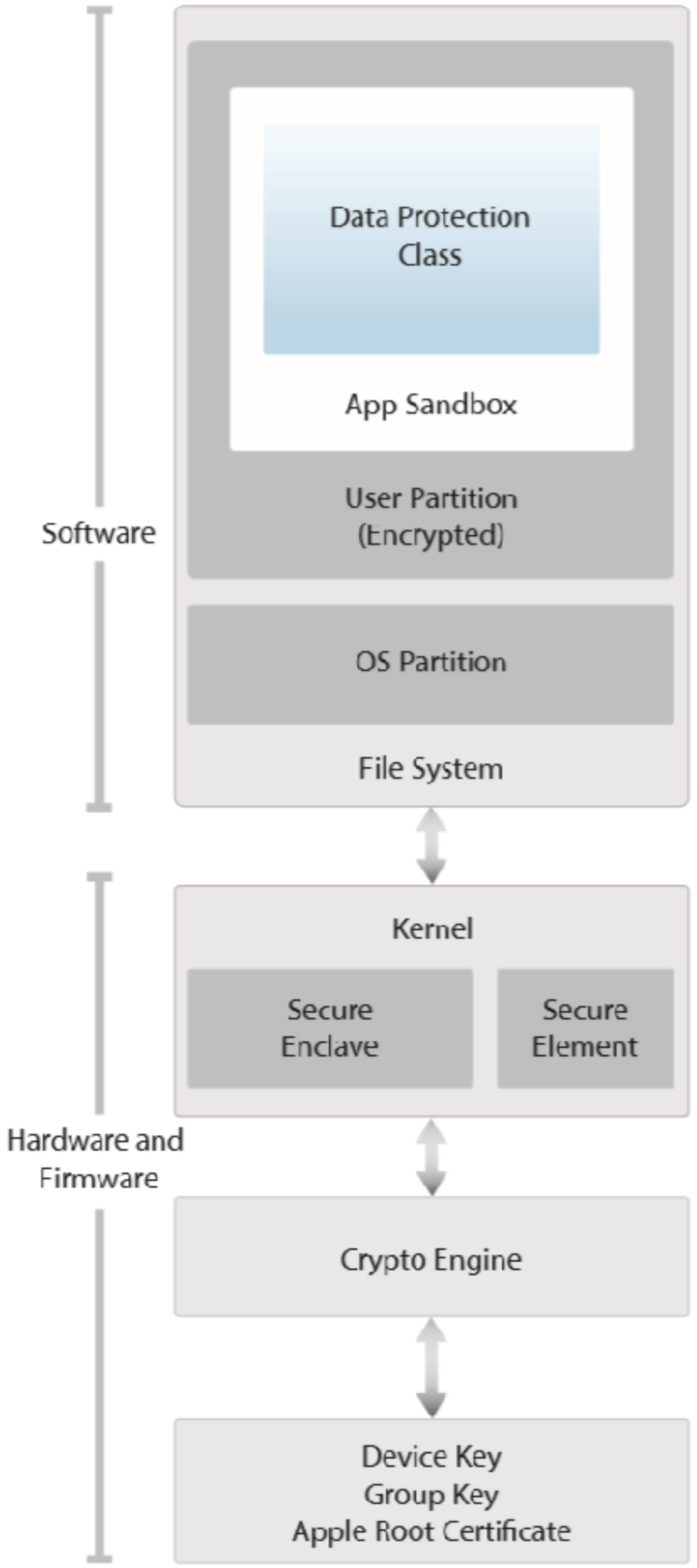
Case study of Apple’s iPhone

- Full disk encryption since iOS 8
- TPM-like hardware protection of key material since iPhone 5s


iOS Security
iOS 9.0 or later

September 2015

Crypto on Apple’s mobile devices



Data for each application always encrypted

All OS components & applications signed by Apple, verified at boot before loading

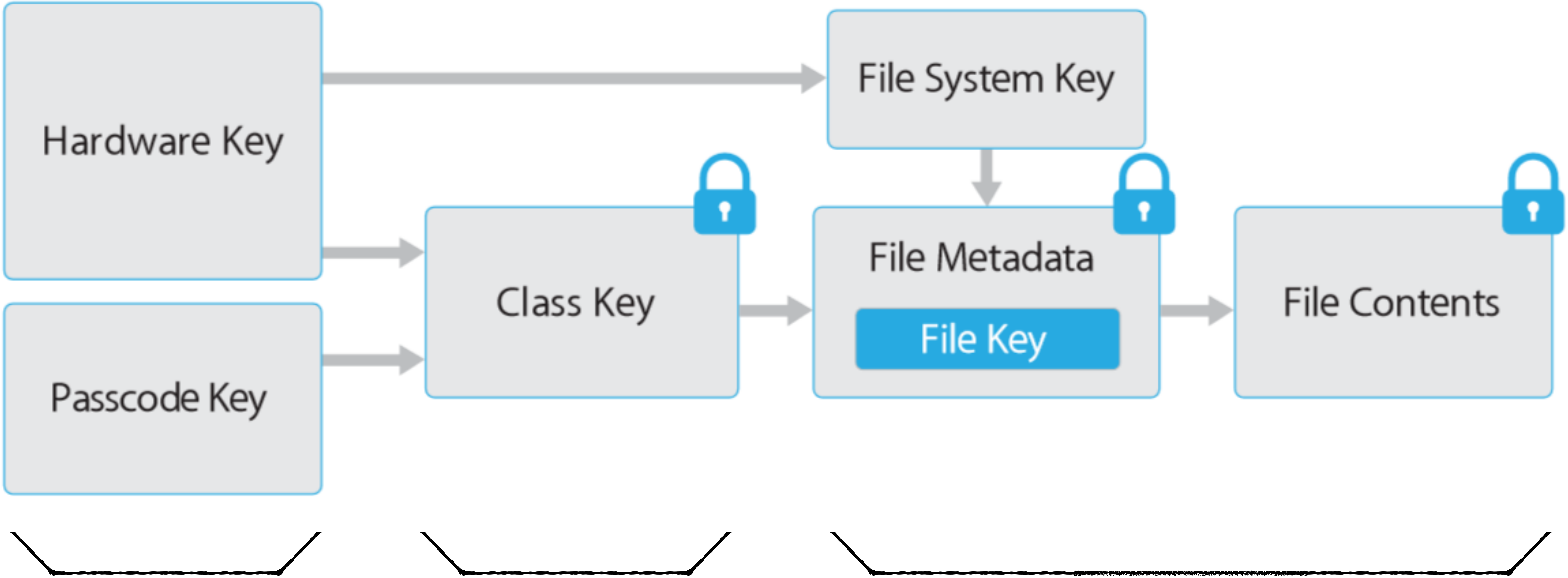
Secure place to store keys & biometrics

Hardware AES-256 engine on RAM-disk path

with what key?

Unique, randomly-chosen string for each device

Hierarchy of keys

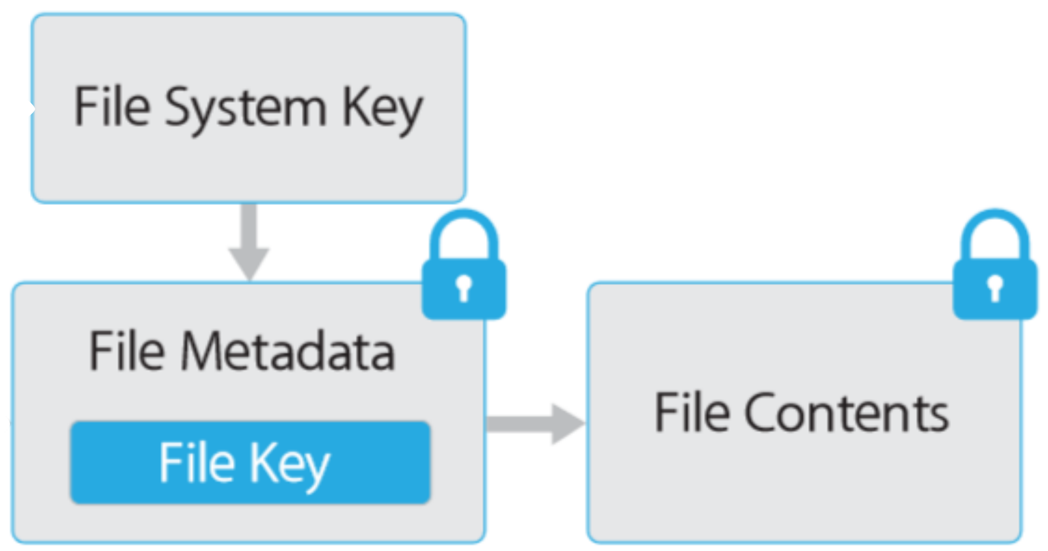


Master key generated from things that you know, are, and have

Derive keys that live only for limited time

Use these to wrap a unique key for each file

Per-file encryption

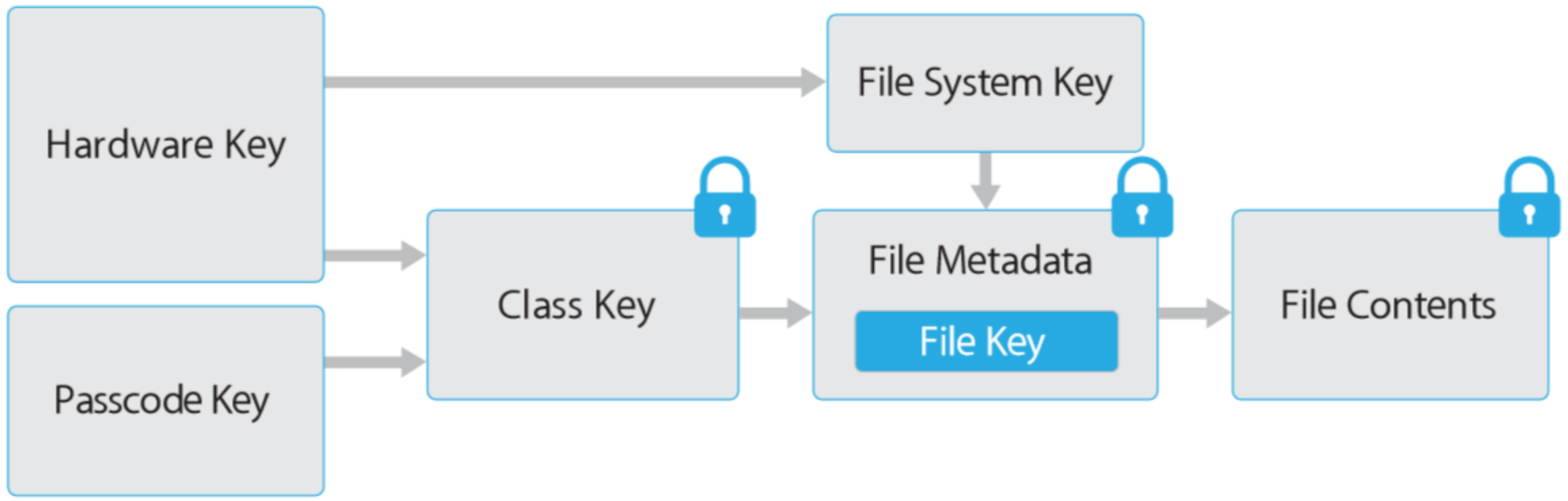


- Each file is encrypted with a unique key
- File encrypted with AES-XTS
- Keywrap goes in the file's metadata
- Secure Enclave includes a hardware chip to generate keys at random
    - Specifically: it uses CTR-DRBG, which we will discuss later in the course

Four classes of data protection

Per-file key is wrapped with 1 of 4 “class keys” based on availability

Availability	Example	Key erased if phone is...
Always	SIM PIN	Wiped
After 1st unlock	Wifi password	Shut down
When locked	Incoming mail	(N/A)
When unlocked	Web passwords & bookmarks	10s after lock (without biometric)



Deriving class keys

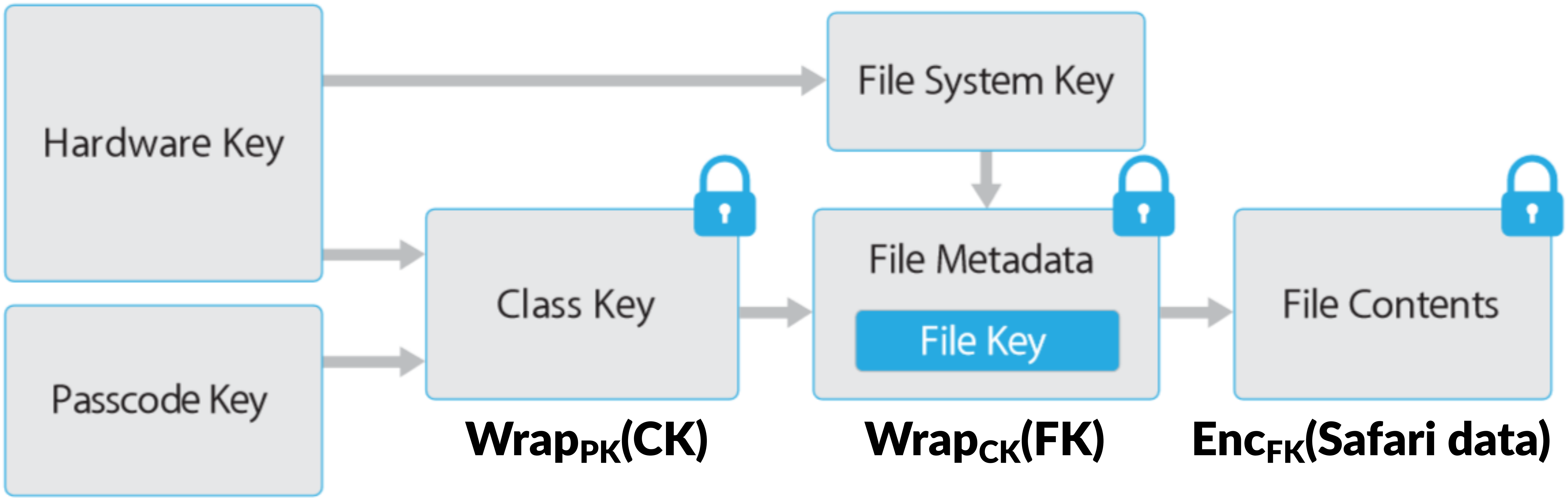
Remember our mantra: ***generate*** key rather than ***storing*** it!

- Wrap sensitive class keys in a *passcode key* derived from:
  - User's alphanumeric pin
  - Unique string fused into the chip at manufacture time, unknown outside Secure Enclave
- Countermeasures to make brute forcing the PIN as difficult as possible
  - Crypto: use 10,000 iterations of a hash to derive the key (~80 ms per guess)
  - Hardware: pause between tries + optionally wipe the phone
- Note: there exist other ways to derive class keys to enable iTunes & iCloud backups + corporate device management

Delays between passcode attempts

Attempts	Delay Enforced
1-4	none
5	1 minute
6	5 minutes
7-8	15 minutes
9	1 hour

Putting it all together



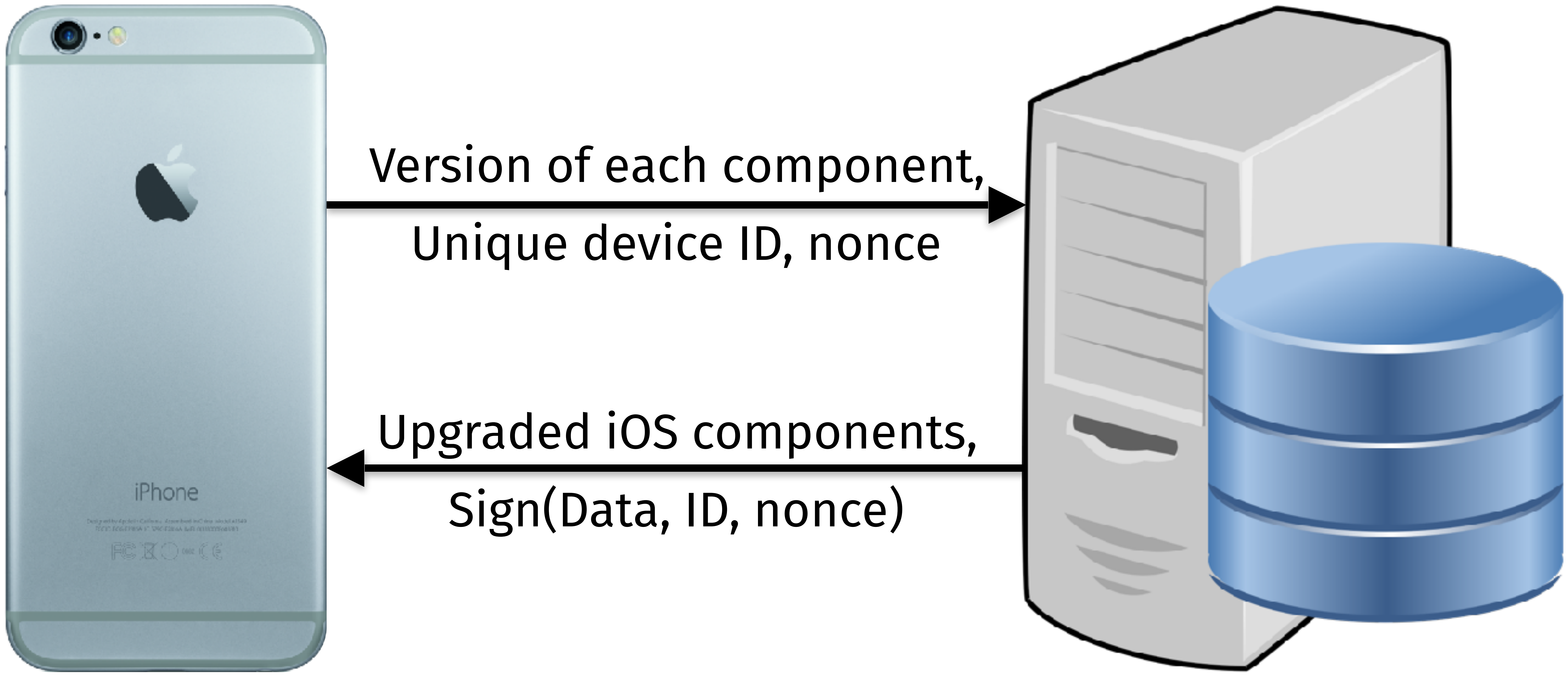
PK = PBKDF2(pin, uid)

When phone is locked

- Without Touch ID: CK is deleted from Secure Enclave's memory
- With Touch ID: Form new keywrap Wrap$_{TouchID}$(CK), then delete CK

P.S.: CK is changed whenever the passcode is changed
P.S.: Memory used by Secure Enclave is itself encrypted w/ ephemeral key

Something different: iOS over-the-air updates



Signature contains:

- Device ID to *personalize* the response to this particular phone
- Nonce to connect the response to the initial request, *prevent replay* attacks

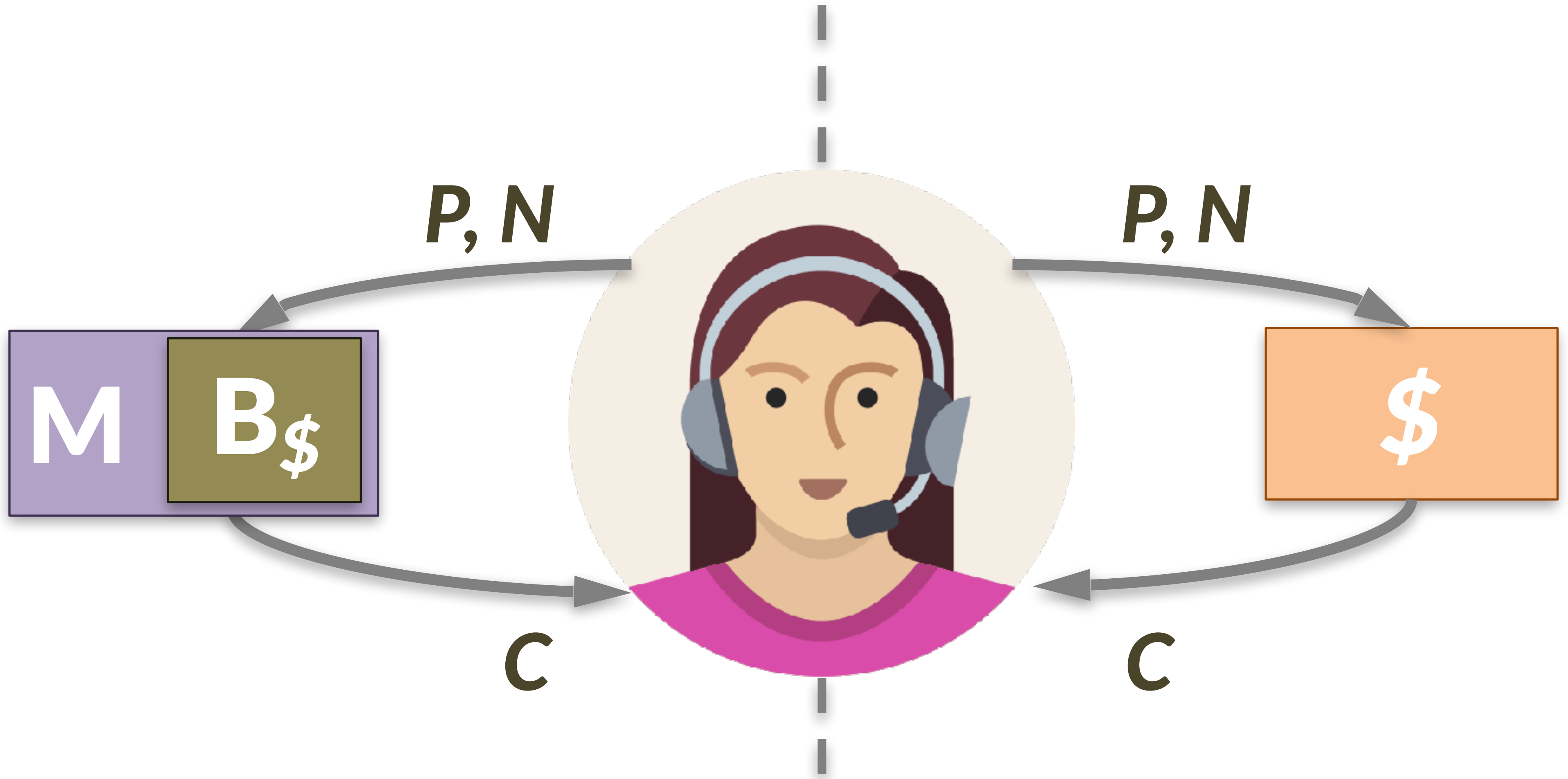
Apple vs. FBI Case

- FBI wanted data on a locked iPhone 5c in its possession, but they did not have the PIN
- key = Hash(pin, uid), so to read the files they could:
  - Pry open the phone to find the uid
  - Brute-force the pin on the phone itself
- FBI wanted Apple to modify its operating system to enable a brute-force search of the PINs:
  - Allow PINs to be submitted via external interface, not by hand
  - Remove the delay between incorrect guesses
  - Remove the "poison pill" wiping of the phone after 10 misses
- FBI wanted Apple to produce & digitally sign this "GovtOS" update, else the phone will reject it
- iPhone 5c doesn't have a Secure Enclave, so delay is software-enforced rather than hardware-enforced

Part 1 recap: protecting data at rest

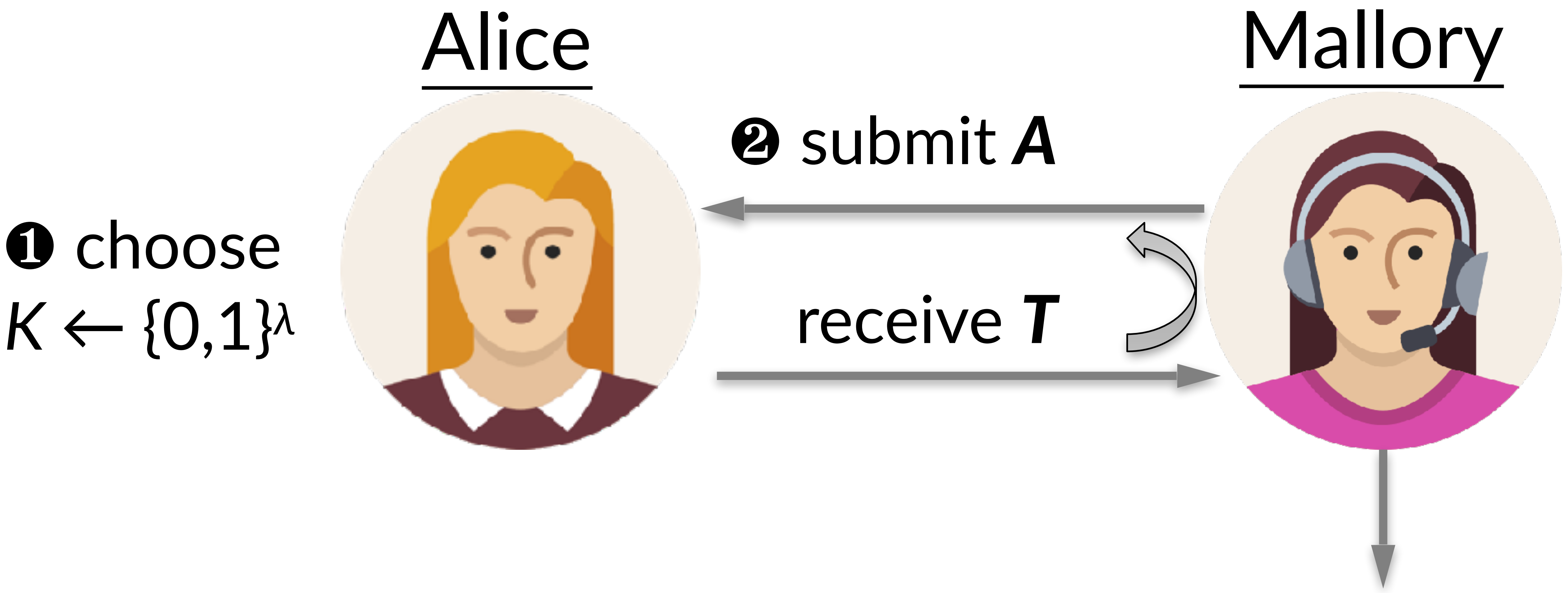
Privacy

IND$-CPA against
nonce-respecting Eve



Authenticity

Even after viewing many (***A***, ***T***) pairs,
Mallory cannot forge a new one



Mallory wins if:
1. It's a valid forgery
2. It's new

❸ output
(***A****, ***T****)