CS 4173/5173
COMPUTER SECURITY

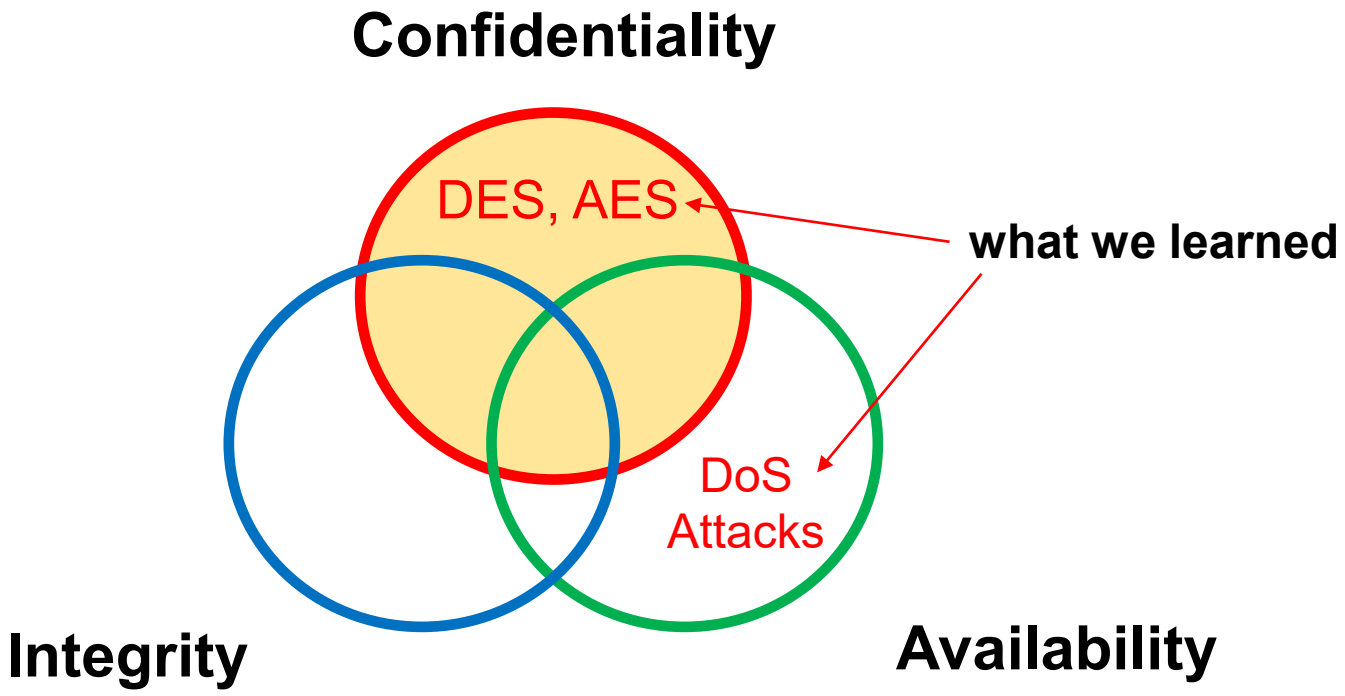
Message Authentication

OUTLINE LAST TIME

- Mode of operations
  - ECB, CBC, …
- Triple DES
  - Meet-in-the-middle attack
  - Procedure of triple DES

MESSAGE MANIPULATION

- We can see that bits in a plaintext can be easily manipulated, then how to detect such manipulation?



MESSAGE AUTHENTICATION/INTEGRITY

- Encryption easily provides confidentiality of messages
  - only the party sharing the key (the "key partner") can decrypt the ciphertext
- How to use encryption to authenticate messages and verify the integrity? That is,
  - prove the message was created by the key partner
  - prove the message wasn't modified by someone other than the key partner

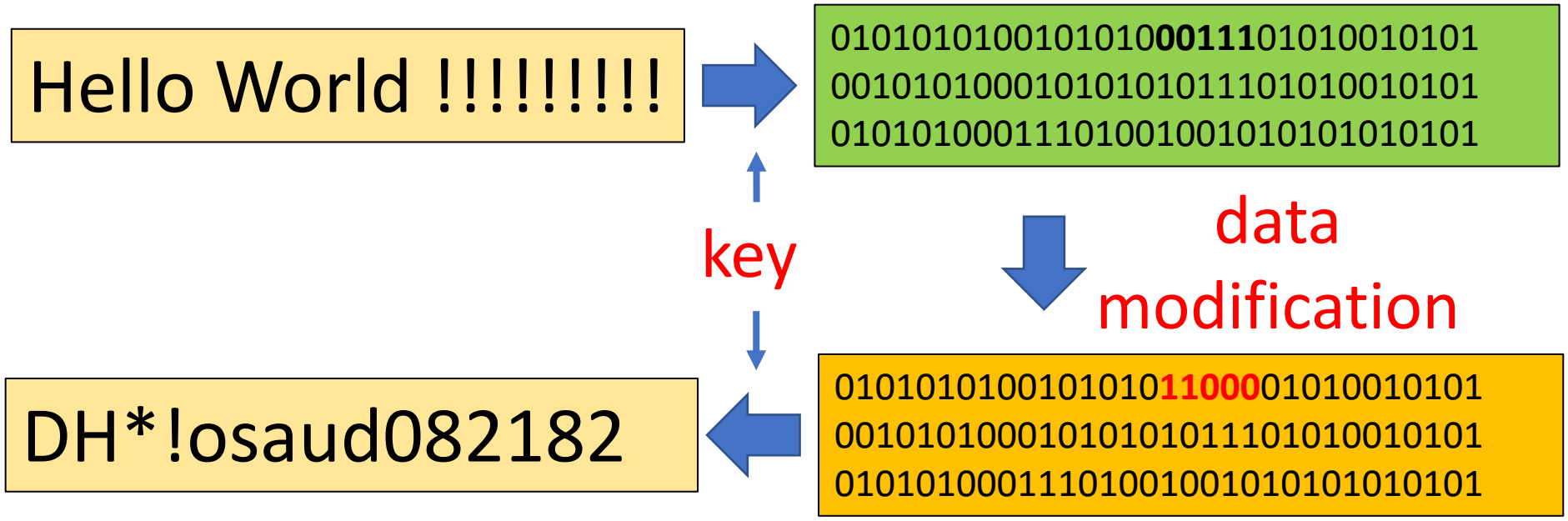
APPROACH #1

- If the decrypted plaintext looks plausible, then conclude ciphertext was produced by the key partner
  - i.e., illegally modified ciphertext, or ciphertext encrypted with the wrong key, will probably decrypt to random-looking data

- Question: is it easy to verify data is plausible-looking?

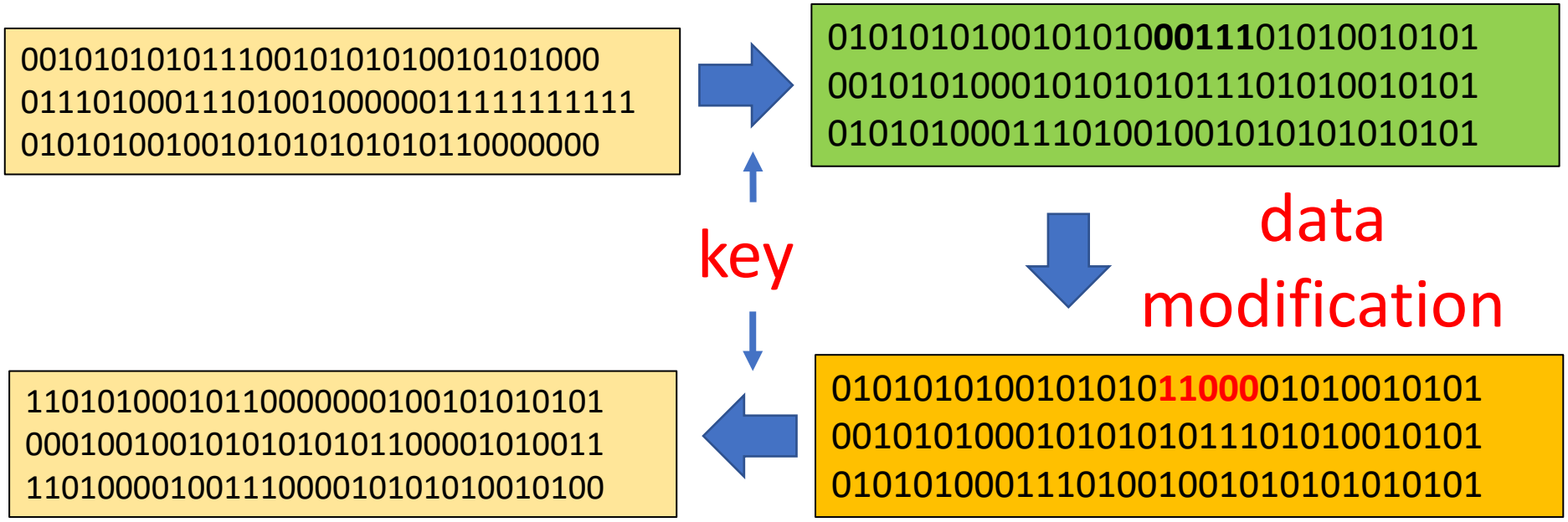
APPROACH #1 (CONT'D)

- Approach may work when the message is plaintext

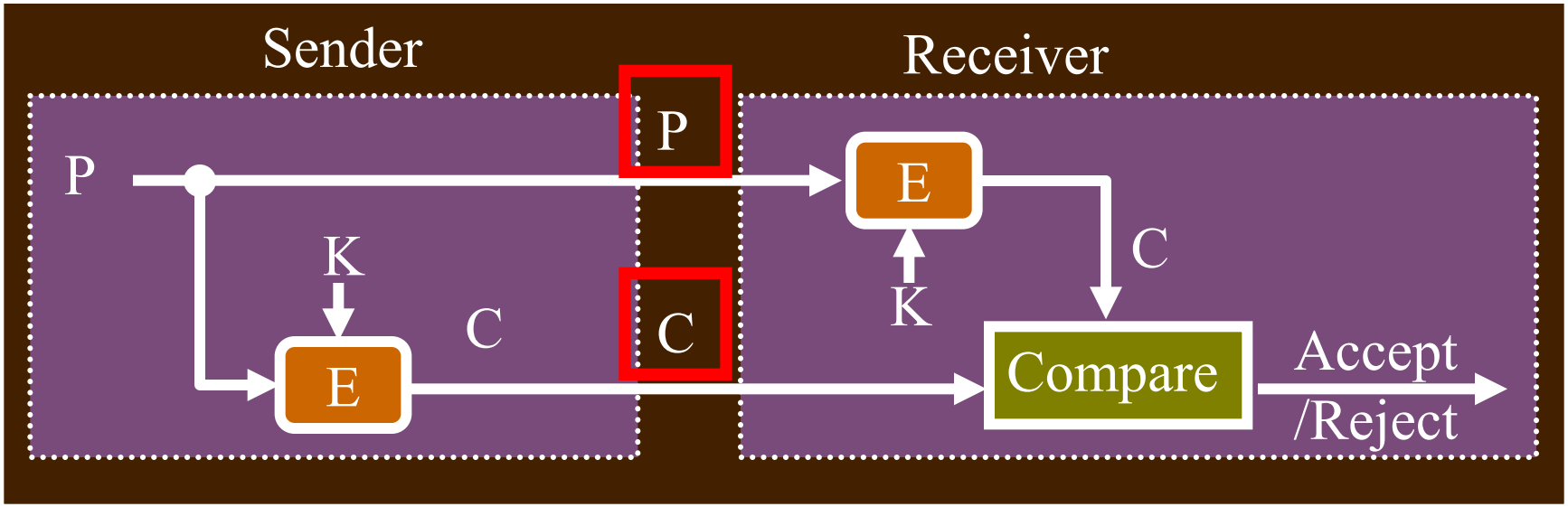


APPROACH #1 (CONT'D)

- But, what if the plaintext itself is random-looking?



APPROACH #2: PLAINTEXT+CIPHERTEXT



- Send plaintext and ciphertext
  - receiver encrypts plaintext, and compares result with received ciphertext

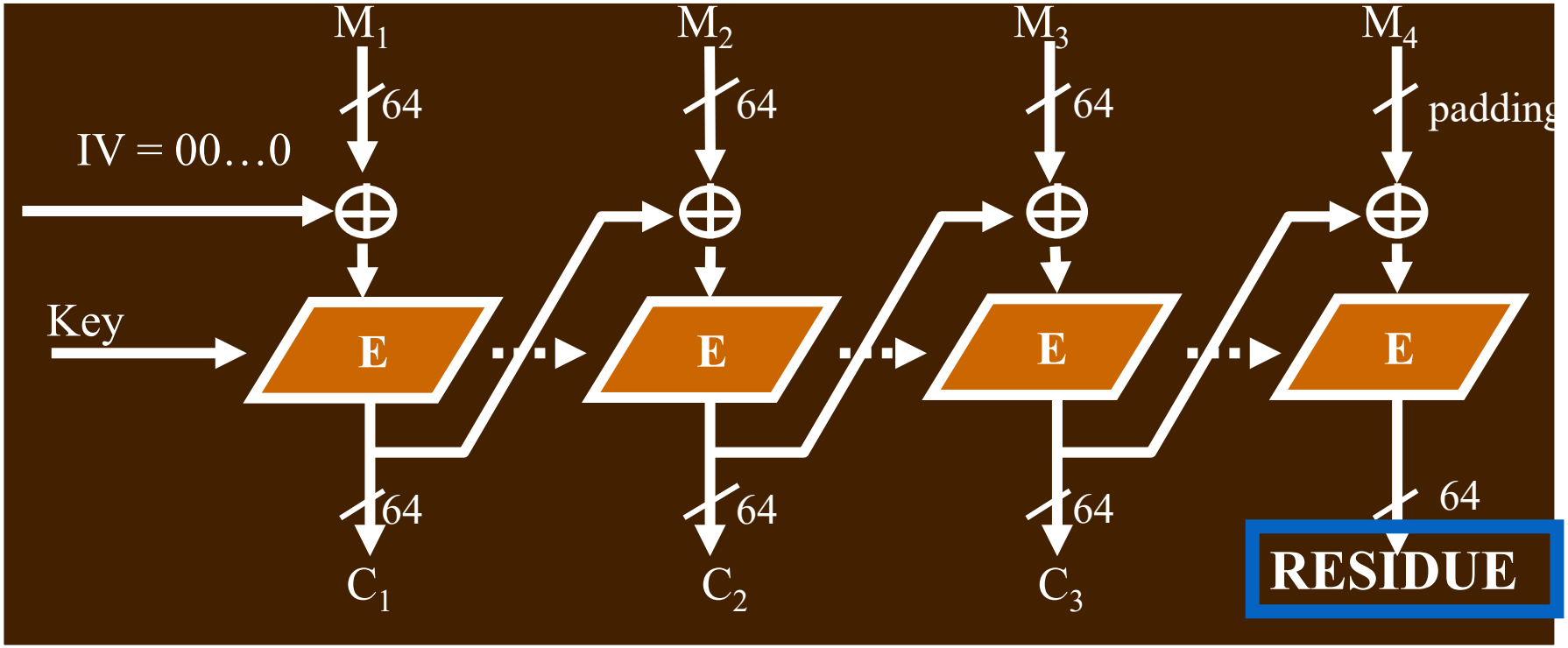
APPROACH #2: ANALYSIS

- Can the receiver detect the following?
  - An attacker modifies the P part.
  - An attacker modifies the C part.
  - The attacker modifies both P and C parts.

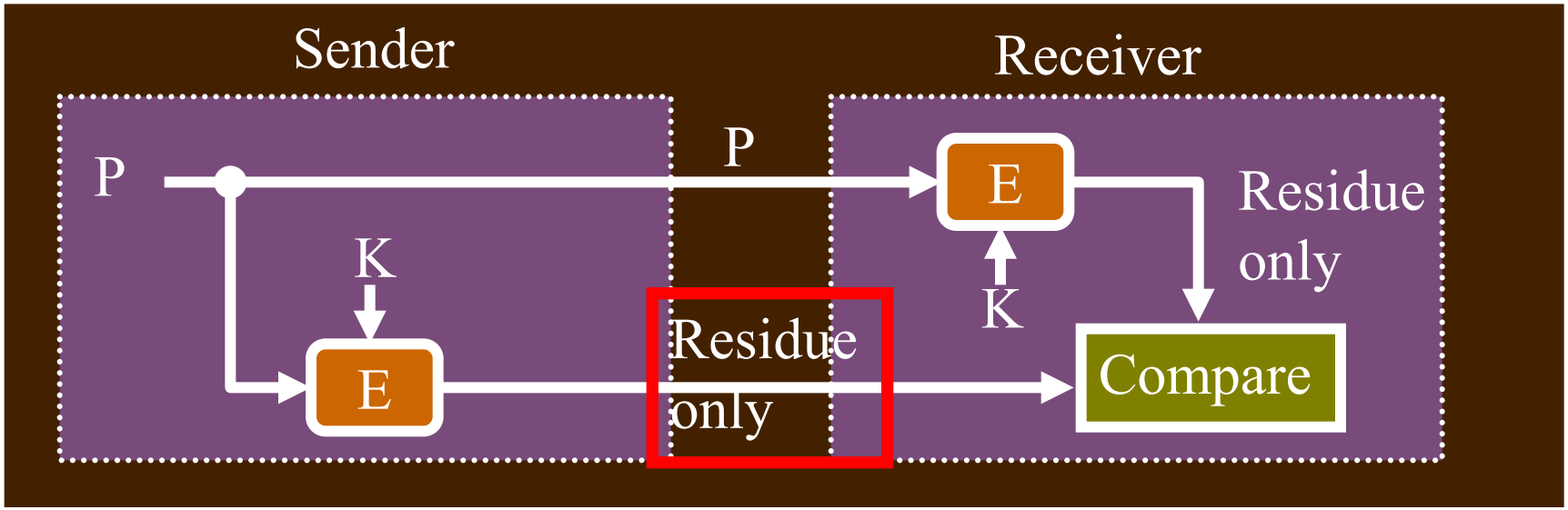
- Disadvantages?

APPROACH #3: USE RESIDUE

- Encrypt plaintext using DES CBC mode, with IV set to zero
  - the last (final) ciphertext output block is called the *residue*



APPROACH #3… (CONT'D)



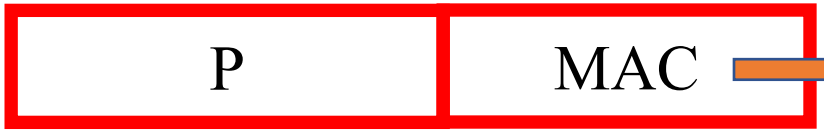
- Transmit the plaintext and this residue
  - receiver computes same residue, compares to the received residue
  - forgeries / modifications highly likely to be detected

APPROACH #3: ANALYSIS

- Can the receiver detect the following?
  - An attacker modifies the P part.
  - An attacker modifies the C part.
  - The attacker modifies both P and C parts.

- Disadvantages?

MESSAGE AUTHENTICATION CODE (MAC)



MAC is a short piece of information used to authenticate a message, usually appended to the end of the message

- The residue is a message authentication code (MAC) or message integrity code (MIC)
  - More specifically, called CBC-MAC. (recall that CBC is a mode of operation to chain cipher blocks together).
- Many other ways to generate MAC.

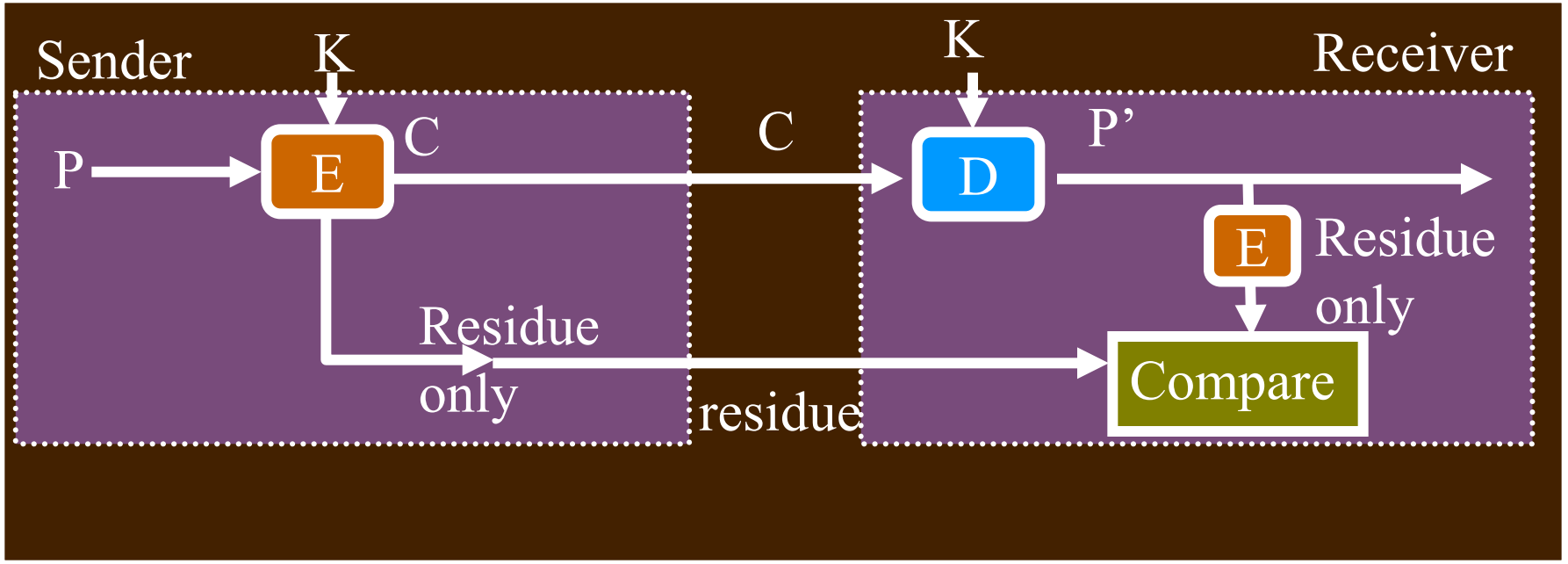
CONFIDENTIALITY AND AUTHENTICITY

- So far we've got
  - confidentiality (encryption),

  or…
  - Authenticity/integrity (CBC-MAC)

- Can we get both at the same time with one cryptographic operation?

ATTEMPT #1

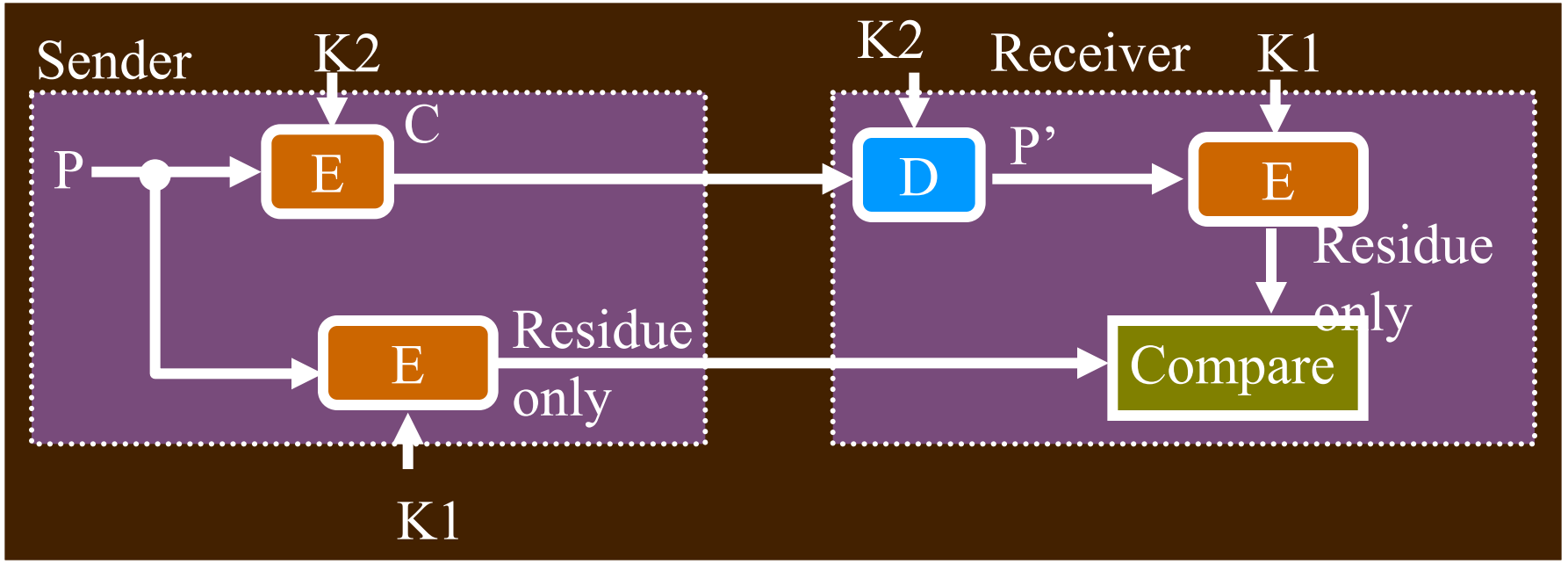


- Is it working?
  - Confidentiality?
  - Integrity?

ATTEMPT #1: ANALYSIS

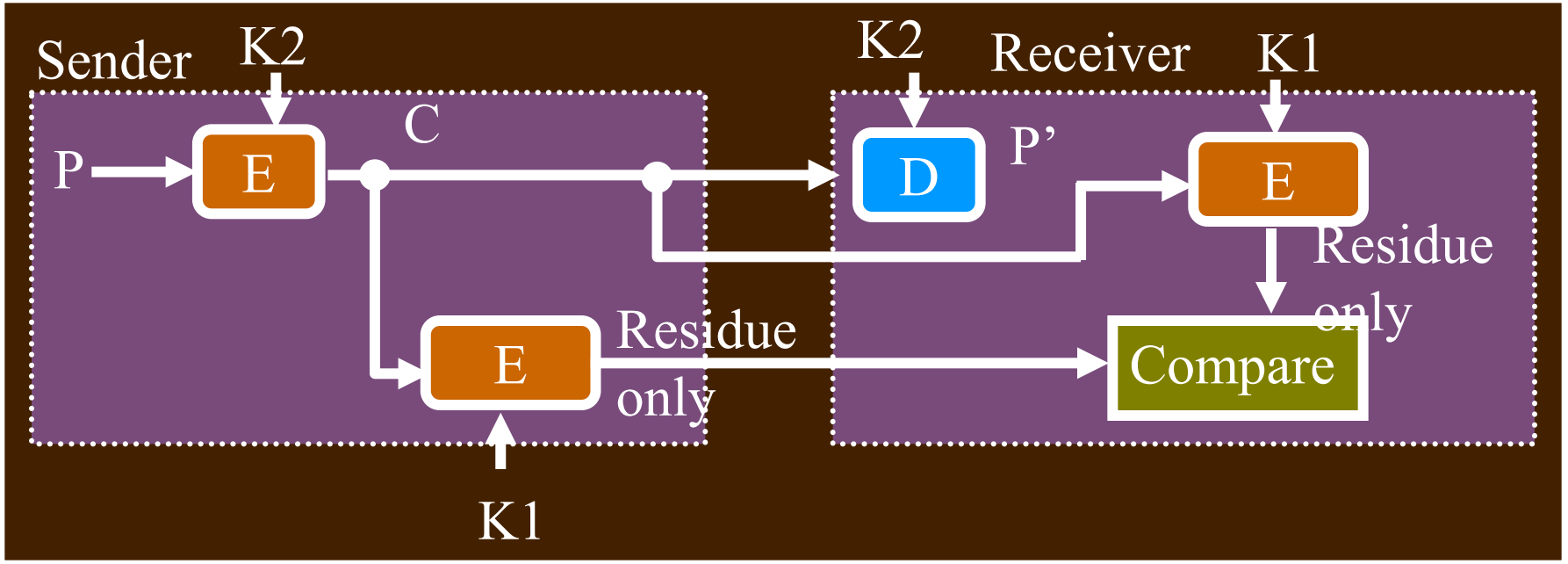
- Confidentiality – Yes
- Integrity – No
  - The attacker just needs to transmit the last block twice.

ATTEMPT #2: ENCRYPT-AND-AUTHENTICATE



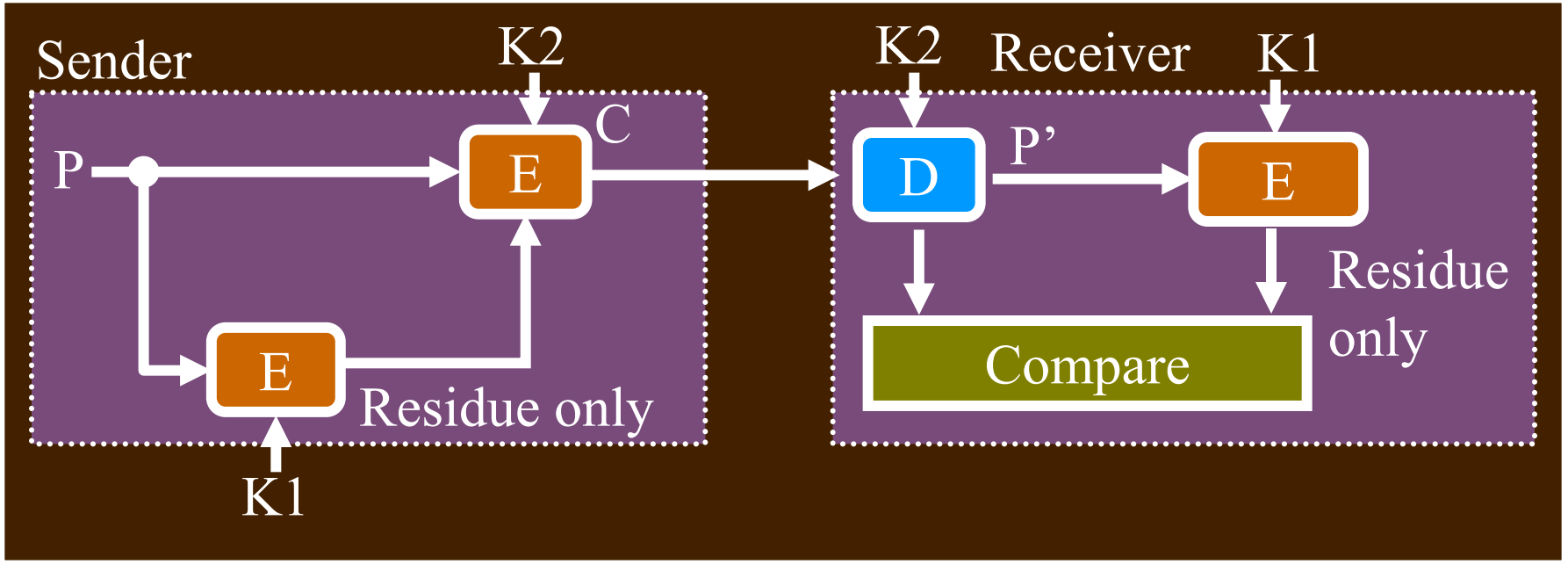
- Is it working?
  - Confidentiality?
  - Integrity?

ATTEMPT #3: ENCRYPT-THEN-AUTHENTICATE



- Is it working?
  - Confidentiality?
  - Integrity?

ATTEMPT #4: AUTHENTICATE-THEN-ENCRYPT



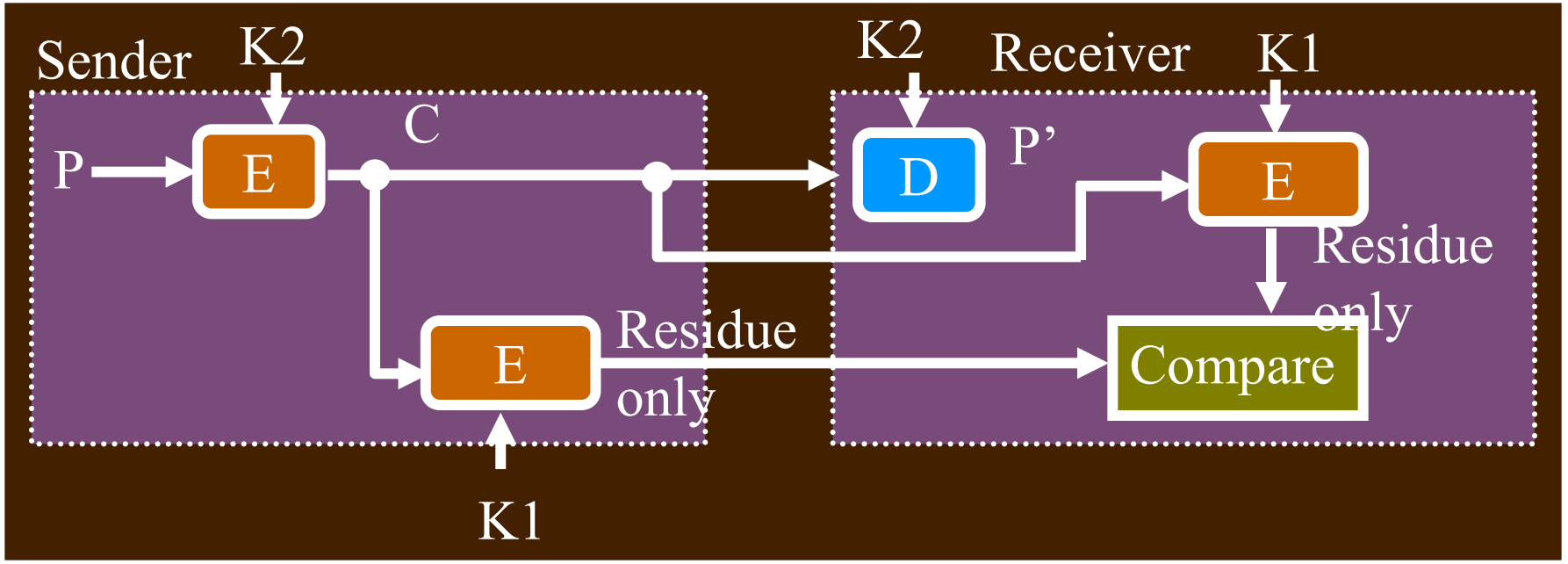
- Is it working?
  - Confidentiality?
  - Integrity?

ATTEMPTS #2-4: ANALYSIS

- Generally, secure combination of encryption and authentication is hard.

- Encrypt-then-authenticate design is a secure combination, generally better than
  - encrypt-and-authenticate
  - authenticate-then-encrypt

DISADVANTAGES



- Encryption-based MAC can be slow
  - e.g., CBC-MAC
- We need two keys K1 and K2.

WHAT WE KNOW

- Achieving confidentiality does not necessarily mean achieving integrity.
  - Fully encrypting a message does not mean data tampering (integrity violation) can be detected.
- MAC is an effective way to protect integrity.
  - **CBC-MAC** in symmetric cryptography **works but is slow**
  - Integrity protection usually incurs overhead.
- Efficient design is needed to achieve both confidentiality and integrity
  - **Fast MAC** design widely used today based on **hash functions**.
    - Hash function based MAC is called **HMAC**