CISC 468: CRYPTOGRAPHY

LESSON 7: BLOCK CIPHER MODES OF OPERATION

Furkan Alaca

TODAY, WE WILL LEARN ABOUT...

How to use block ciphers to encrypt data that is:

- Larger than the block size
- Not an exact multiple of the block size
- Smaller than the block size

READINGS

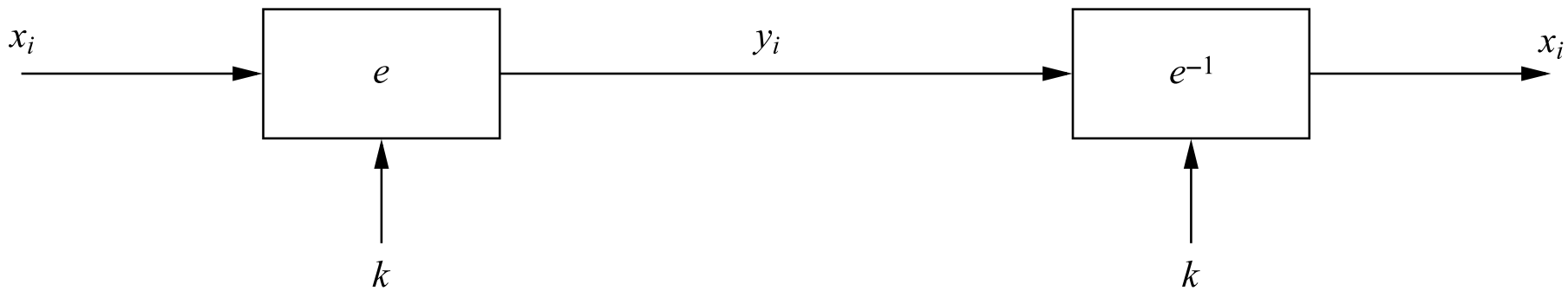
- Section 5.1: Encryption with Block Ciphers: Modes of Operation, Paar & Pelzl

MODES OF OPERATION

- In practice, we usually want to encrypt plaintext messages consisting of more than one 16-byte (128-bit) block
- *Modes of operation* use block ciphers to encrypt plaintext consisting of more than one block
- Popular examples include:
  - Electronic Code Book (ECB) mode
  - Cipher Block Chaining (CBC) mode
  - Cipher Feedback (CFB) mode
  - Output Feedback (OFB) mode
  - Counter (CTR) mode

ELECTRONIC CODEBOOK (ECB) MODE

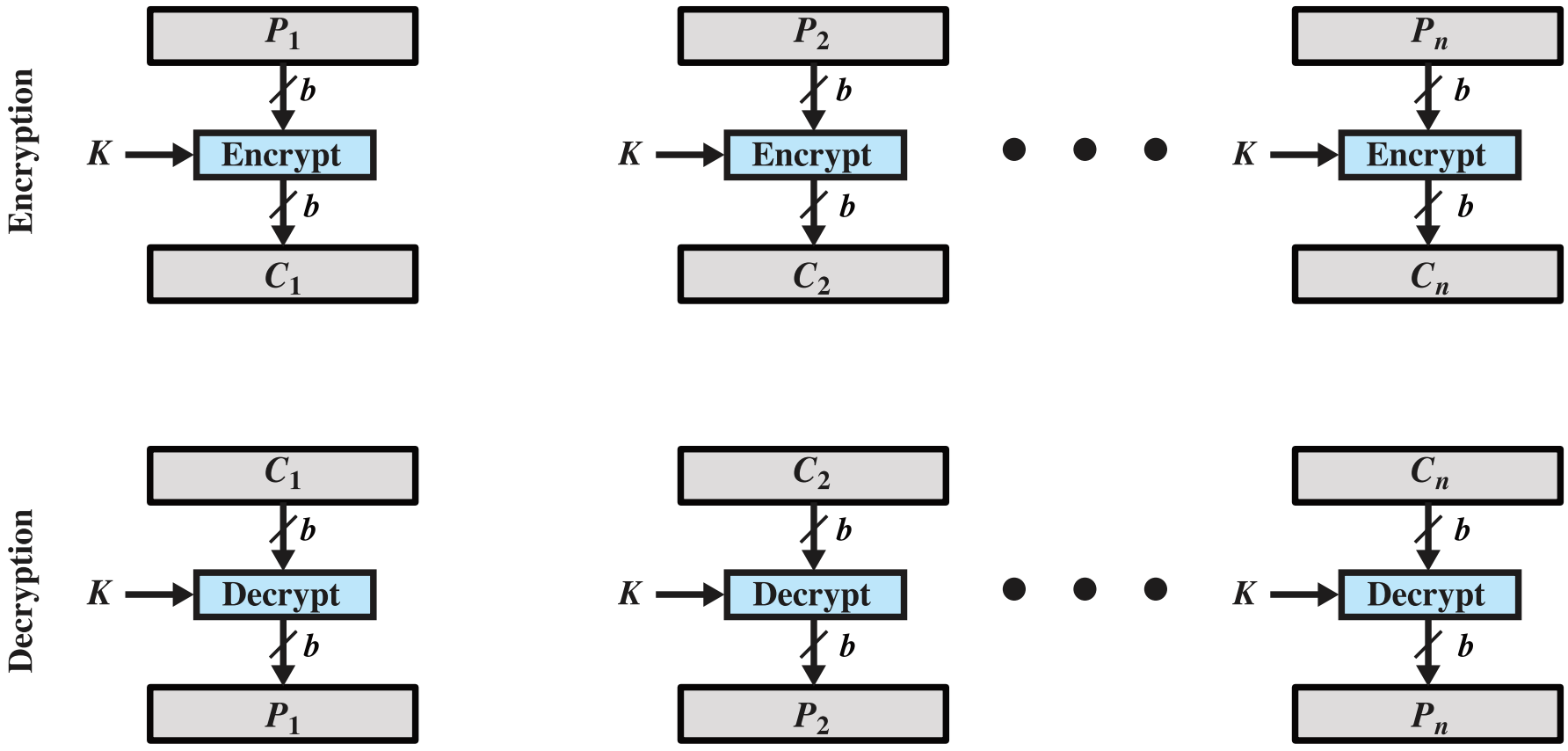
- Simplest mode: Divide plaintext x into b-bit blocks
- Encrypt each block of plaintext x_i using secret key k
- Decrypt each block of plaintext y_i using secret key k



PADDING

- *Padding* is required to be able to handle plaintext messages that are not an exact multiple of the block size
- Simplest method: Append a `1` bit to plaintext and as many `0` bits as are required to reach a multiple of the block length
- If plaintext is exact multiple of the block size, an extra block consisting of only padding bits must be sent
  - Why?

ECB MODE



ANALYZING SECURITY OF ECB

- ECB is like a gigantic codebook: Using a specific key, each possible block of plaintext is always mapped to the same block of ciphertext
  - Remember, a block cipher is a bijective function

ANALYZING SECURITY OF ECB (CONT'D)

- By analyzing ciphertext, an attacker can determine:
  - If the message contains repeated plaintext blocks, by identifying repeated ciphertext blocks
  - If the same message has been sent more than once (i.e., even if there are no repeating blocks within the message)
- An attacker with control over the transmission channel (e.g., admin access to WiFi router) may tamper with the message by reorganizing the ciphertext blocks

ECB: SUBSTITUTION ATTACK

- Consider the banking protocol message structure below
- An attacker that can tap the encrypted communication can:
  - Create accounts at bank A and bank B, and transfer different amounts of money in both directions
  - Collect the encrypted messages and observe ciphertext blocks that change, stay the same, or are moved around
  - Identify the ciphertext block representing their account number, and swap it into other intercepted transactions

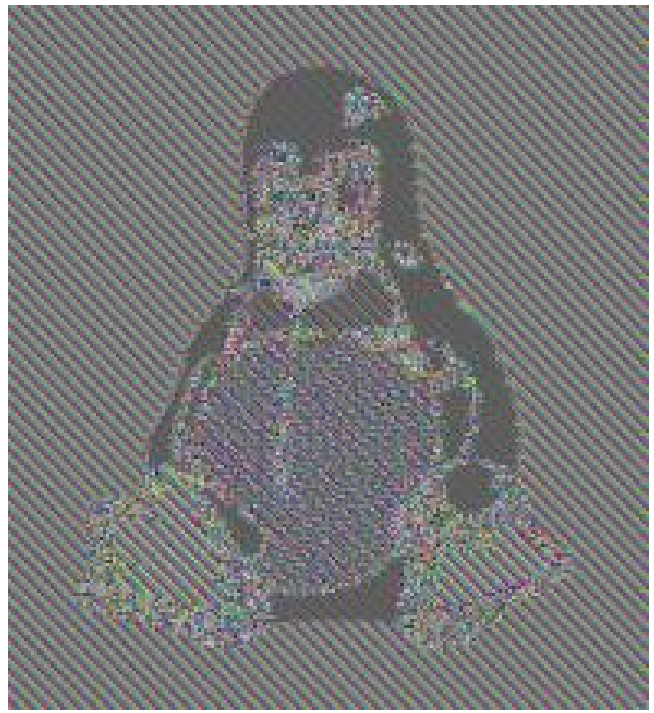
Block #	1	2	3	4	5
	Sending Bank A	Sending Account #	Receiving Bank B	Receiving Account #	Amount $

ECB: SUBSTITUTION ATTACK (CONT'D)

- Substitution attacks on ECB work on all block ciphers
  - *Message Authentication* needed to protect integrity
- Indistinguishability property is not fulfilled, since the attacker can infer repetition in the plaintext by observing repetition in the ciphertext
  - This can defeat data confidentiality, via statistical analysis

ECB: CRYPTANALYSIS OF AN ENCRYPTED BITMAP

(a) Original image



(b) Encrypted with ECB mode

ECB: CRYPTANALYSIS OF AN ENCRYPTED BITMAP (CONT'D)

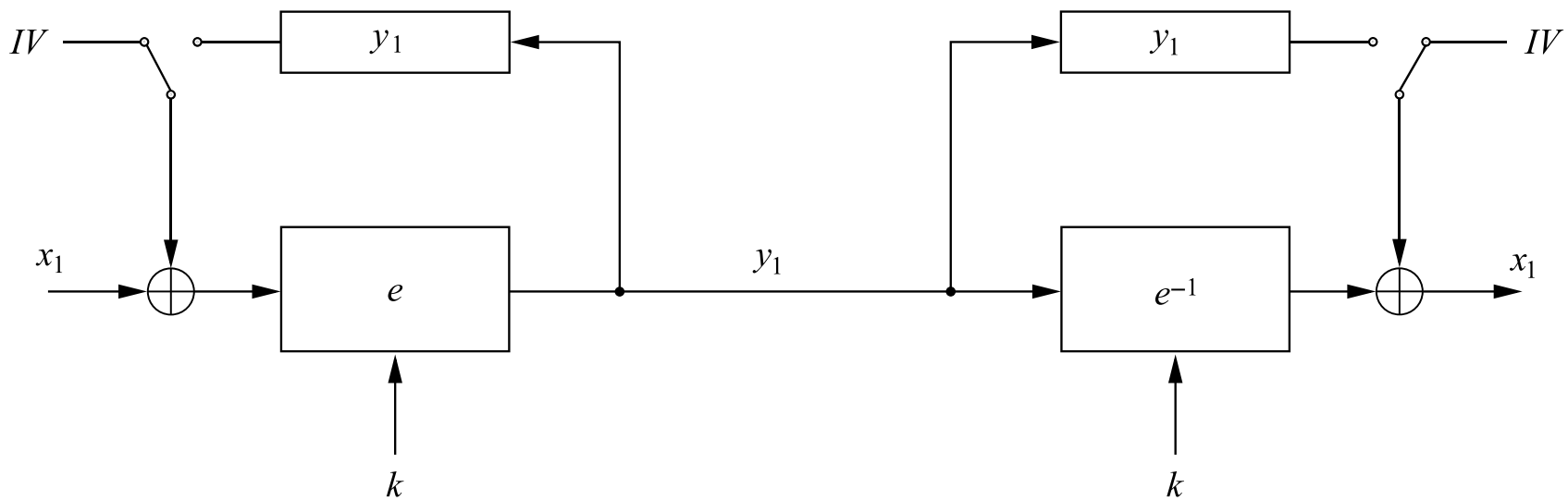
- The weakness exploited in this attack is similar to that of the substitution attack: statistical properties in the plaintext are preserved in the ciphertext
  - This is because ECB results in *deterministic enryption*
- To defeat these attacks, we require *probabilistic encryption*:
  - Repeated plaintext blocks in the same message should be mapped to different ciphertext blocks
  - The same message encrypted more than once should result in a different ciphertext

CYPHER-BLOCK CHAINING (CBC) MODE

1. To ensure that the same plaintext block is not mapped to the same ciphertext block twice, we can "chain together" the blocks in a way that makes the output of each encryption function dependent not just on the current plaintext block but on all the previous blocks too
2. To ensure that the same message is never encrypted to the same ciphertext, we can use an *initialization vector* that is randomized each time a secret key is reused for encryption

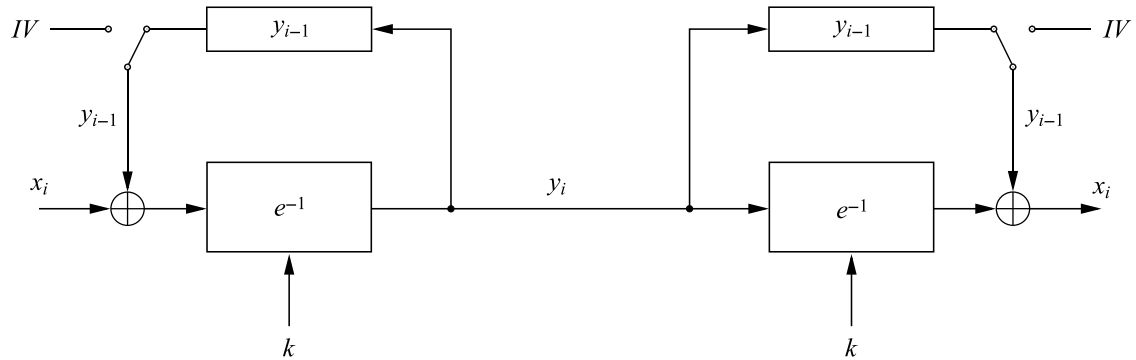
CBC: ENCRYPTING THE FIRST BLOCK

- Sender XORs first plaintext block with *initialization vector* (*IV*)
- Receiver decrypts first ciphertext block and also XORs with IV
- The IV is not secret: It can prepended as-is to the ciphertext
  - But the same IV should never be reused for encryption with the same key

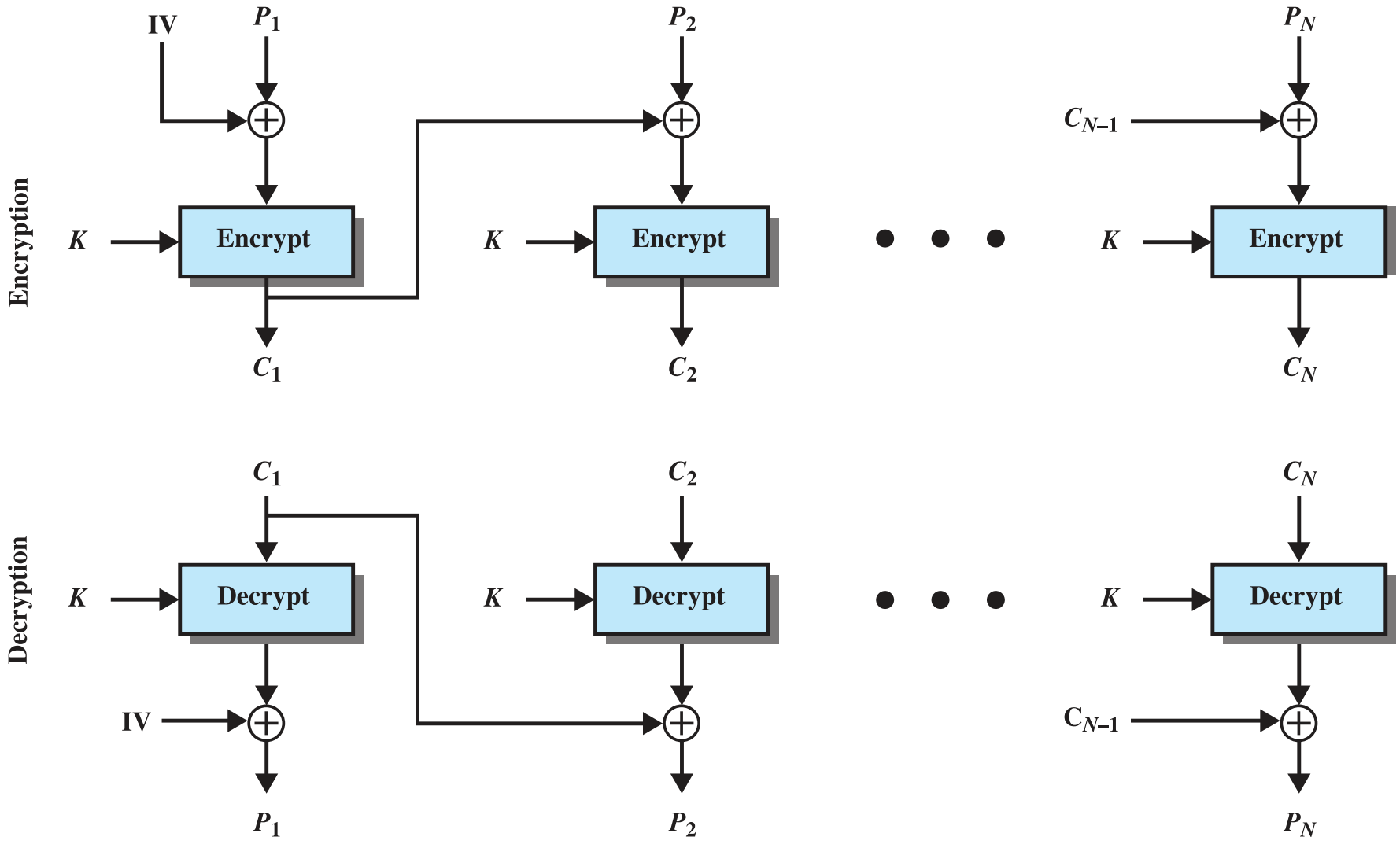


CBC: ENCRYPTING SUBSEQUENT BLOCKS

- Sender XORs each plaintext block x_i with the previous ciphertext block y_{i-1}
- Receiver XORs each decrypted block $e^{-1}(y_i)$ with the previous ciphertext block y_{i-1}
- Each ciphertext block y_i depends on the IV and all plaintext blocks x_j for $j \leq i$

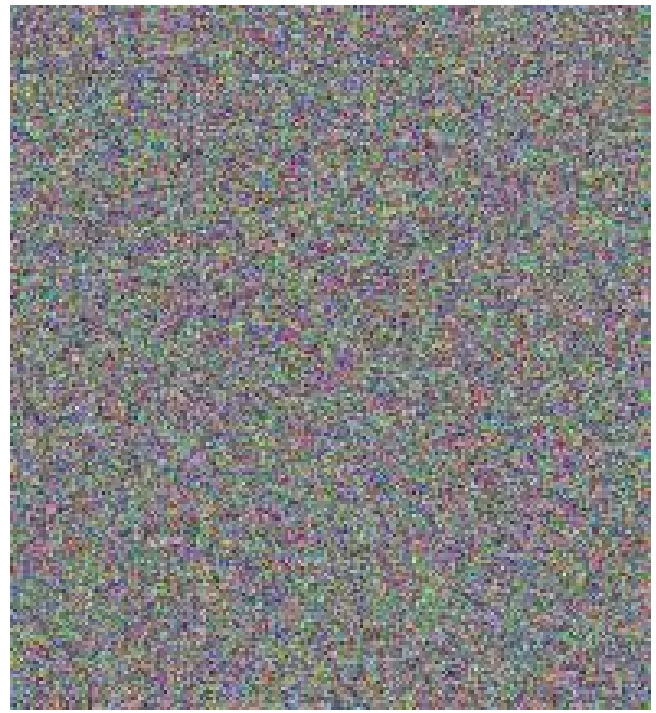


CYPHER-BLOCK CHAINING (CBC) MODE



CBC CRYPTANALYSIS

(a) Original image.



(b) Encrypted with CBC mode.

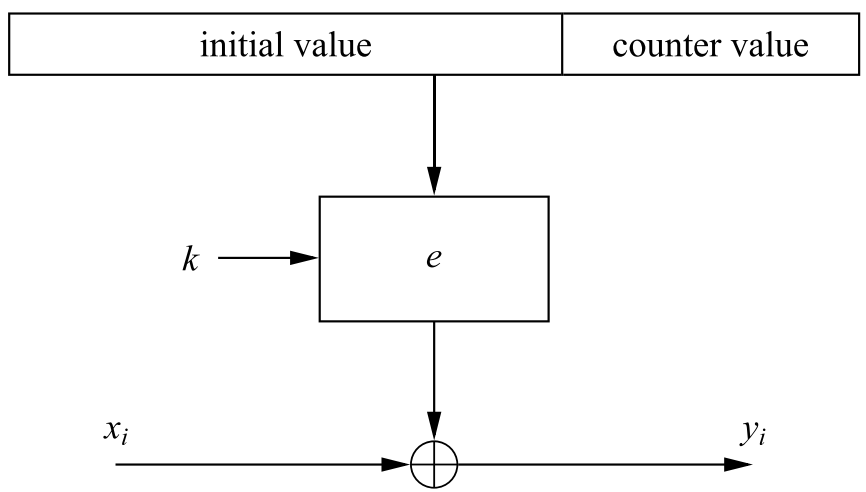
CBC: ATTEMPTING A SUBSTITUTION ATTACK

- If an attacker substitutes block 4 for their own account number, the receiving bank will decrypt blocks 4 and 5 to some random values
  - As long as IV was not reused
- So the attacker will not receive the money, but another random account might
  - We still need a mechanism to protect integrity

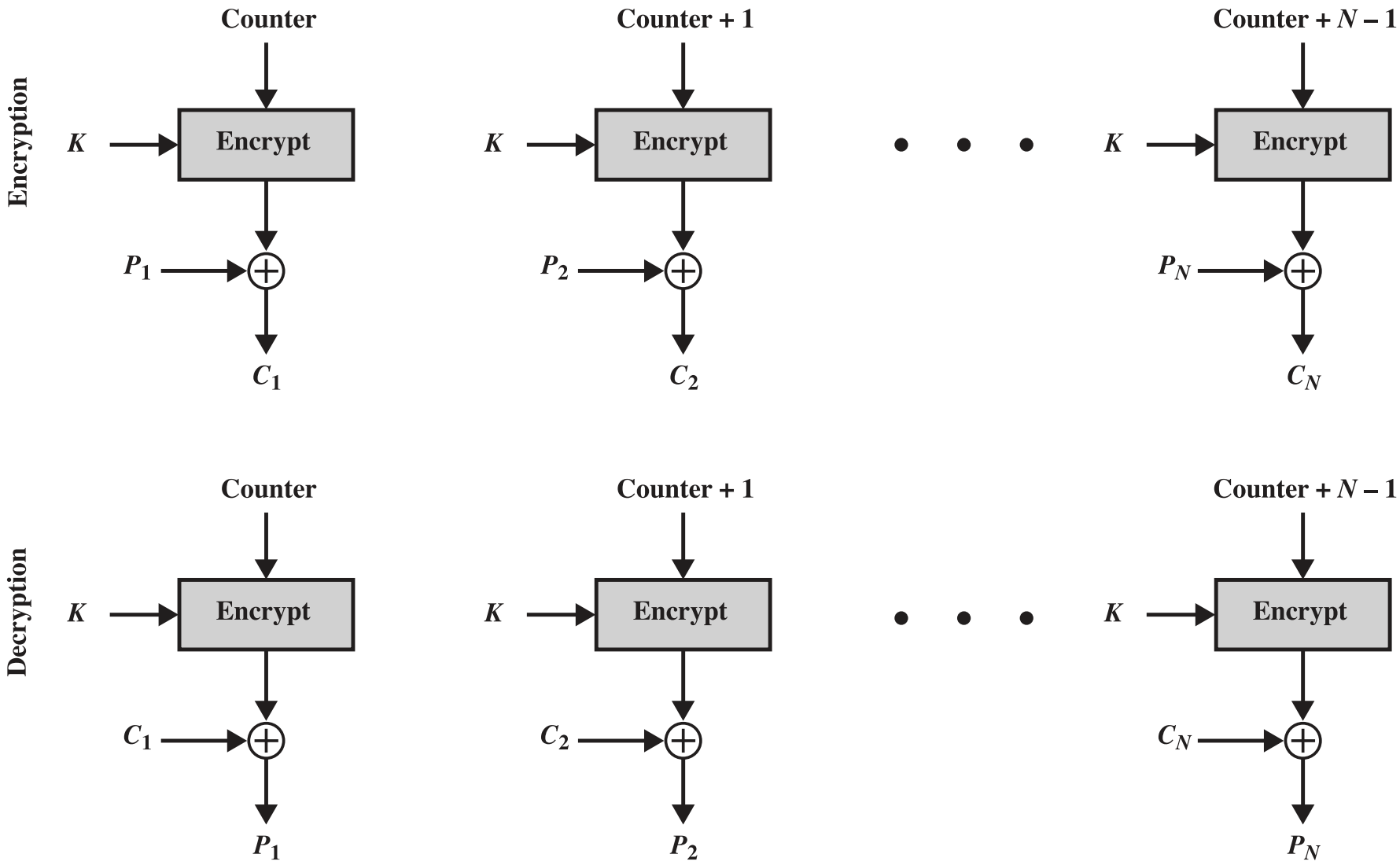
Block #	1	2	3	4	5
	Sending Bank A	Sending Account #	Receiving Bank B	Receiving Account #	Amount $

COUNTER (CTR) MODE

- CTR uses a block cipher to build a stream cipher
- The input to the block cipher is an IV and counter value, and the output is a block of pseudorandom bits
  - Counter is incremented for each subsequent block
- The plaintext block is encrypted by XORing it with the pseudorandom bits



COUNTER (CTR) MODE



COMPARISON: ERROR PROPAGATION

- If a single bit error occurs in y_i, what happens to x_i^d?
  - With ECB and CBC, about half the bits are flipped
  - With CTR, only the single corresponding bit in x_i^d is flipped
- If a single bit error occurs in y_i, what happens to x_{i+1}^d?
  - In ECB and CTR, x_{i+1}^d and onward decrypt correctly
  - In CBC, x_{i+1}^d is corrupted

OTHER MODES: OFB

- Output Feedback (OFB) mode: Builds a stream cipher by encrypting an IV (similar to CTR) and using the pseudorandom block as input to generate the next pseudorandom block

OTHER MODES: CFB

- Cipher Feedback (CFB) mode: Builds a stream cipher by encrypting an IV (similar to CTR) and using the ciphertext block as input to generate the next pseudorandom block

COMPARISON: PARALLELIZATION AND RANDOM ACCESS

Mode	Encryption Parallelizable	Decryption Parallelizable	Random Access
ECB	Yes	Yes	Yes
CBC	**No**	Yes	Yes
CTR	Yes	Yes	Yes
CFB	**No**	Yes	Yes
OFB	**No**	**No**	**No**

CONCLUDING REMARKS

- Block cipher modes of operation are required for encryption/decryption of multi-block messages
  - Each mode has advantages and disadvantages, e.g., for parallelization or error propagation
- To avoid leaking information, IVs must be properly chosen (i.e., randomly) and never reused with the same key
- Among modes we've seen so far (ECB, CBC, CTR, CFB, OFB):
  - CBC can be susceptible to attacks we have not yet covered
  - CTR is preferred
  - None protect against integrity
- Coming later: How to protect integrity using cryptography