(Advanced) Computer Security!

Cryptography

Symmetric Key Cryptography

(part I)

Prof. Travis Peters

Montana State University

CS 476/594 – Computer Security

Spring 2021

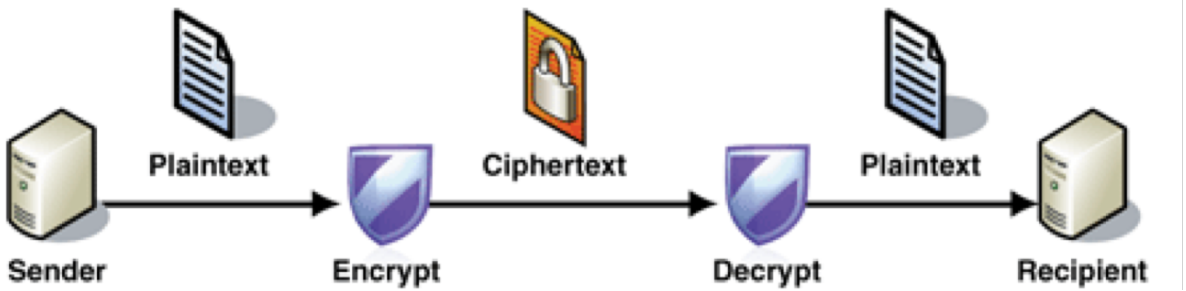
https://www.traviswpeters.com/cs476

Today

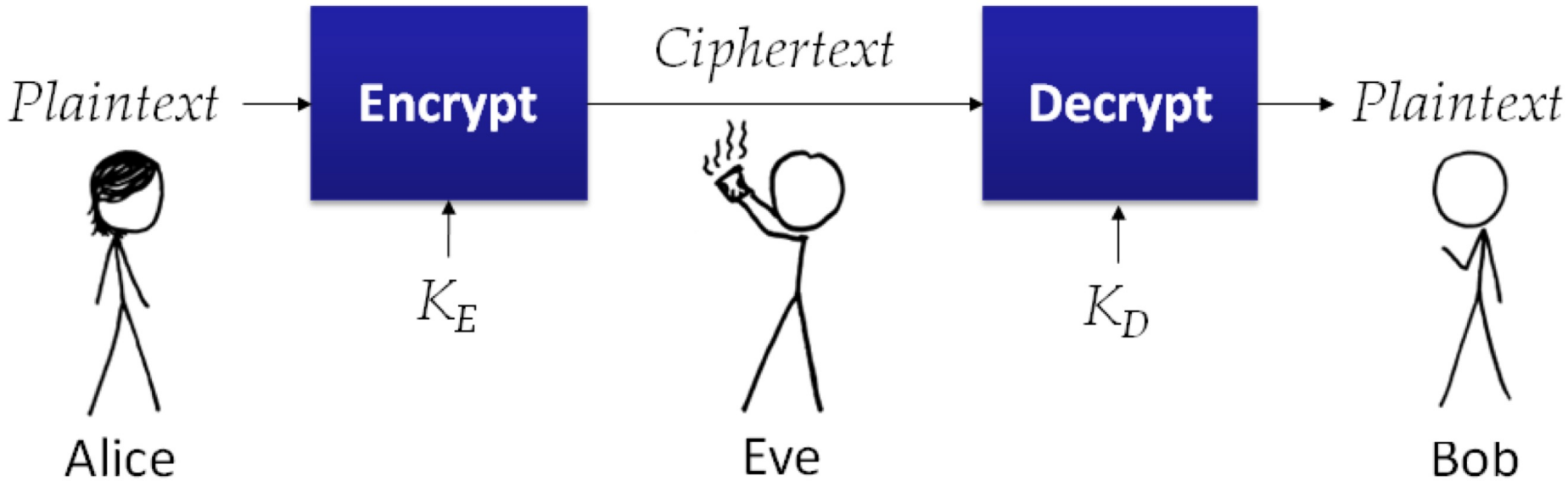
Reminder!
Please update your Slack, GitHub, Zoom
(first/last name, professional photo/background)

- Announcements
  - 'Intro to Crypto' due today

- Learning Objectives
  - Block Ciphers: DES, AES
  - Modes of Operation: ECB, CBC, CFB, OFB, CTR
  - Padding & IVs
  - (Programming using Crypto APIs?)



Intro to Symmetric Key Crypto



$$Cipertext = Encrypt_{K_E}(Plaintext)$$

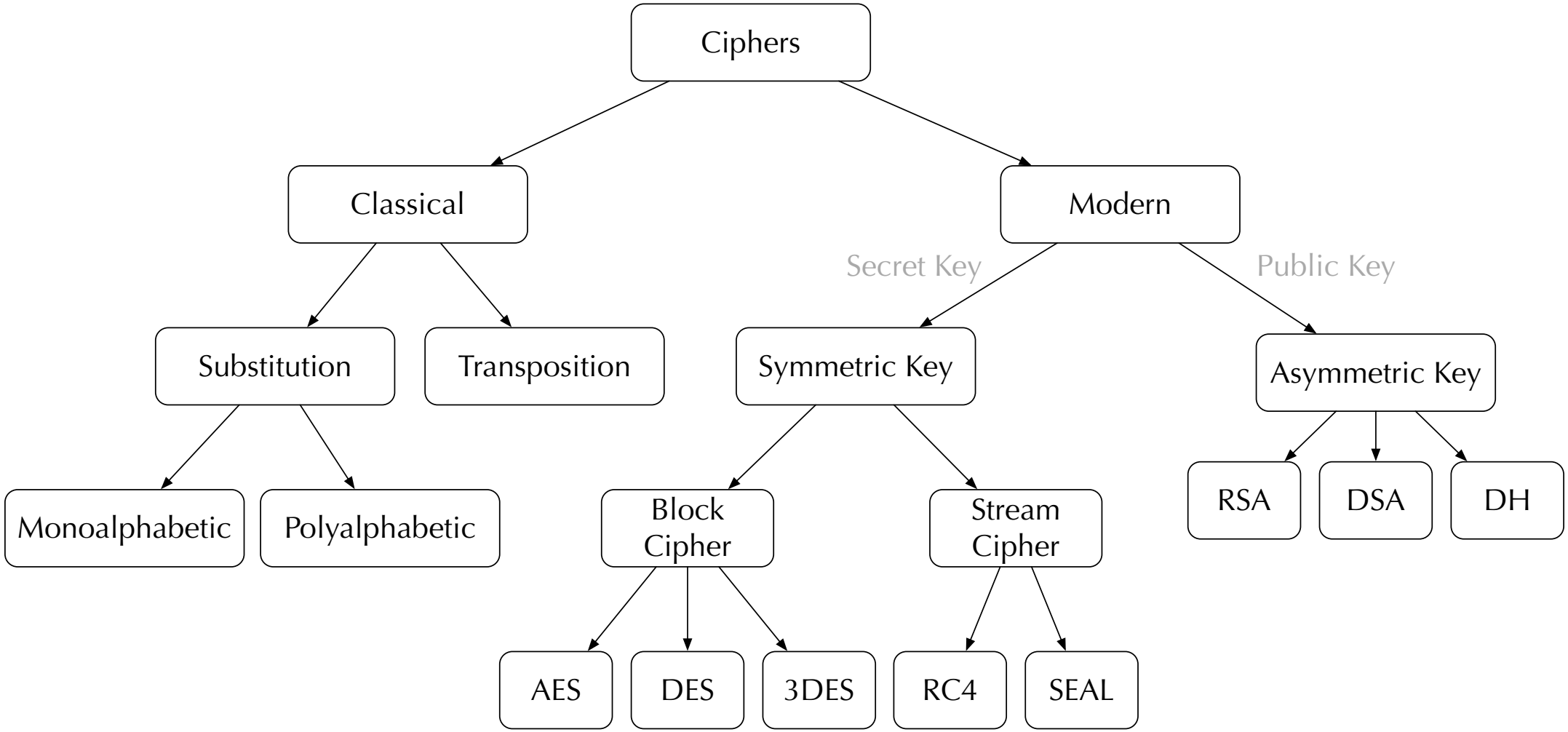
$$Plaintext = Decrypt_{K_D}(Cipertext)$$

In Symmetric Key Encryption,

$$K_E = K_D$$

A Taxonomy of Encryption Methods

A High-Level Overview

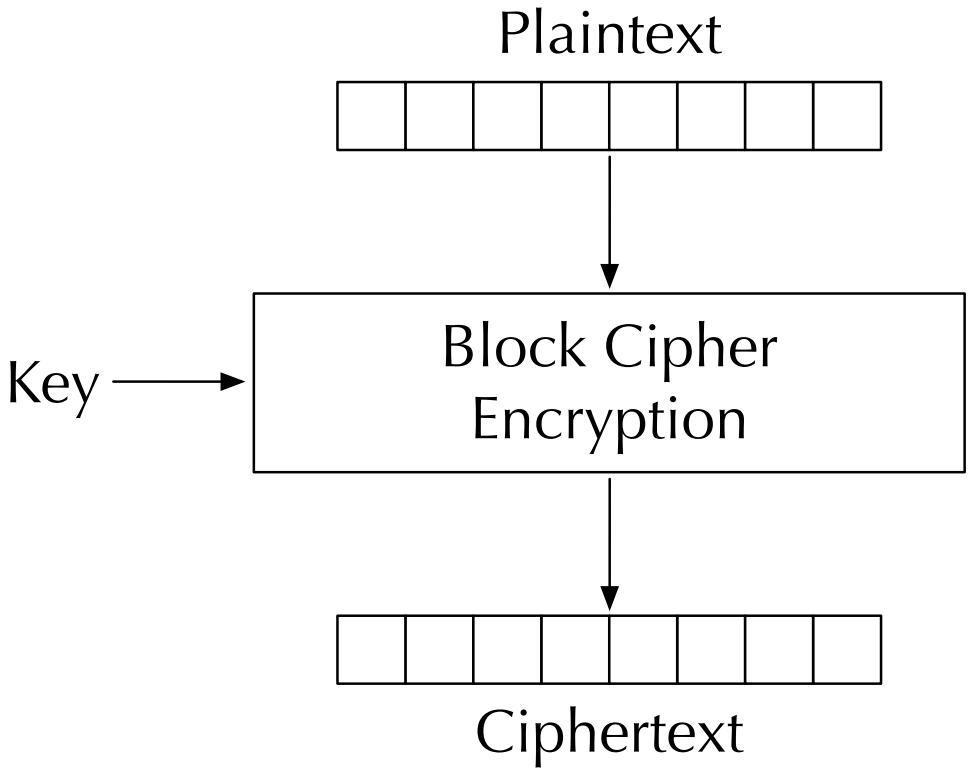


Block Ciphers & Modes of Operation

- Block Ciphers: DES, AES
- Modes of Operation: ECB, CBC, CFB, OFB, CTR

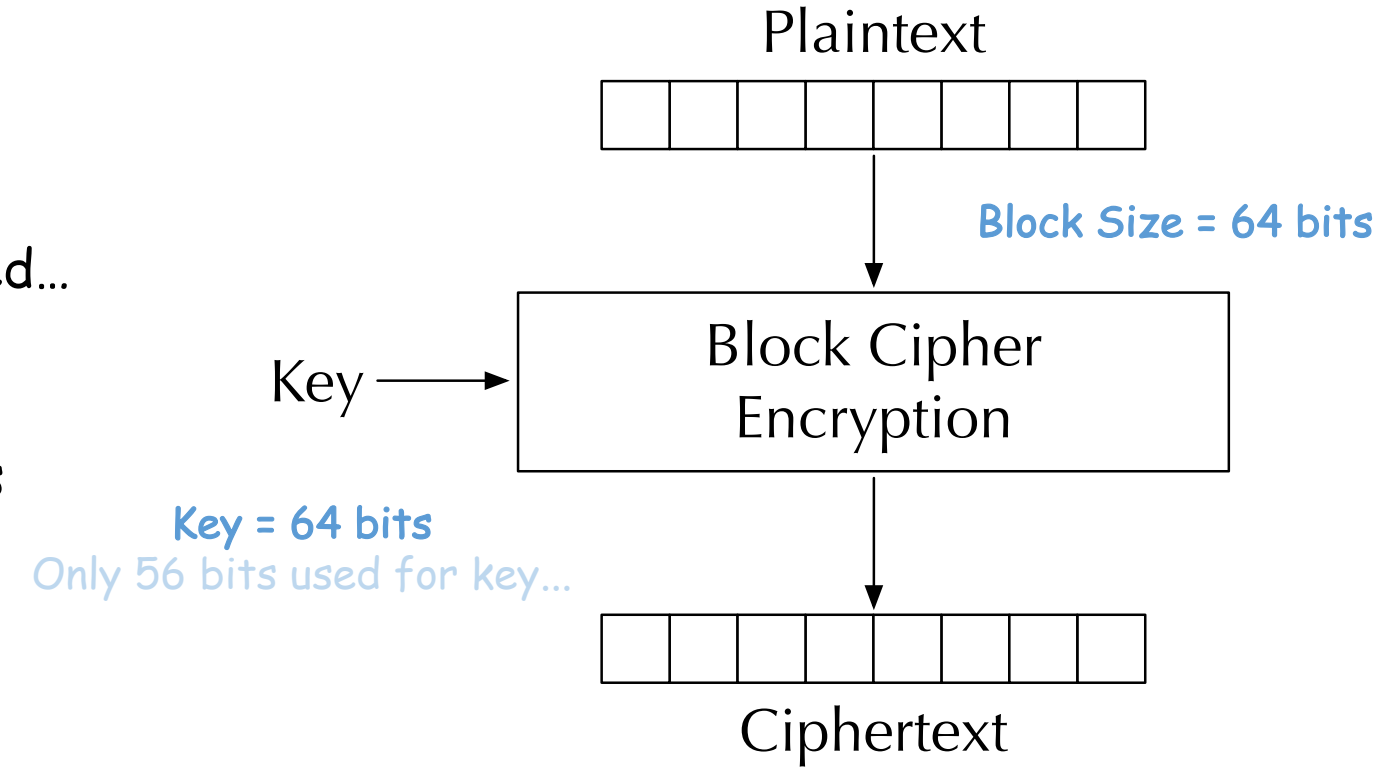
Block Ciphers

- Block cipher = a symmetric encryption which operates on blocks of data
  - e.g., 64-bit blocks, 128-bit blocks
- Break larger messages into small, fixed-size blocks
- Takes one block (**plaintext**) at a time and transforms it into another block of the same length (**ciphertext**) using a **secret key**
- **Decryption** is performed by applying the reverse transformation to ciphertext blocks
- **Important Property:** Even small differences in plaintext result in different ciphertexts



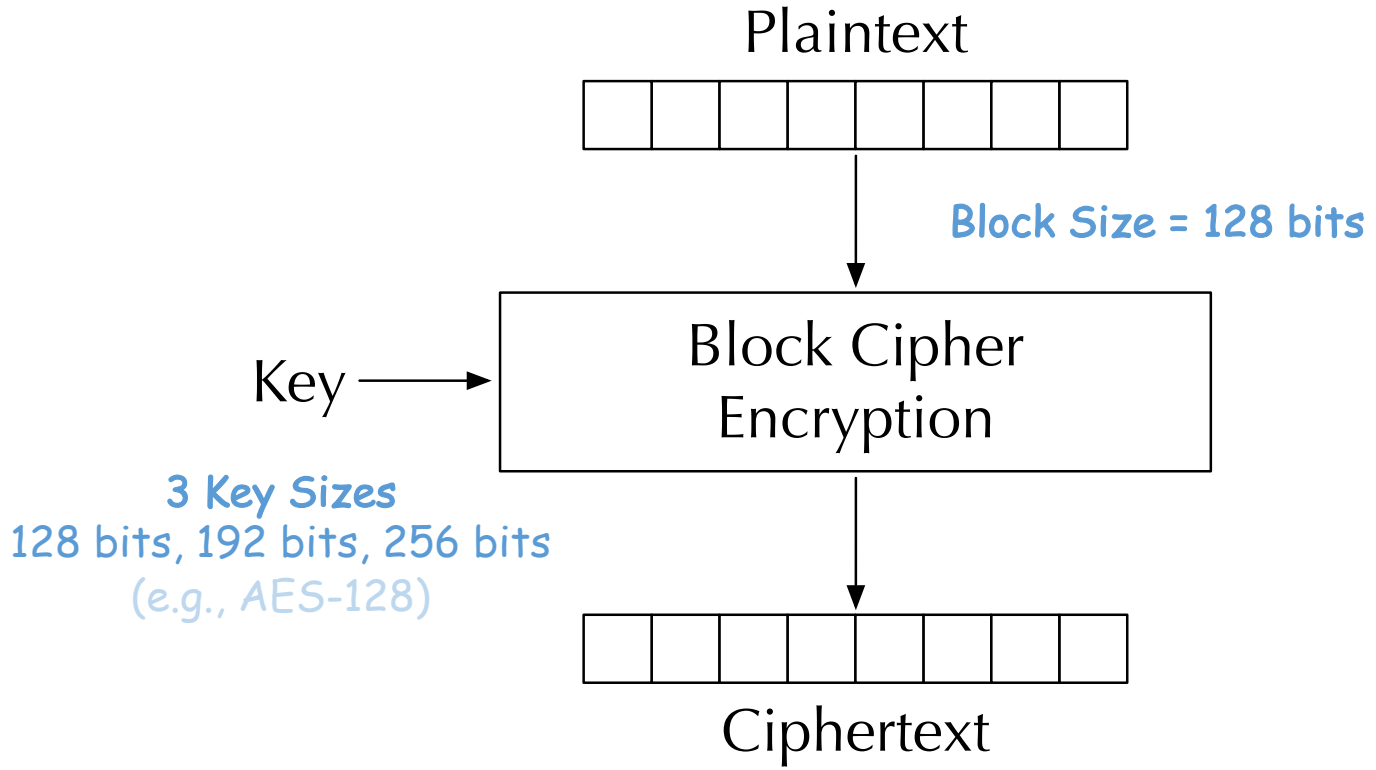
Data Encryption Standard (DES)

- Based on a **Feistel Cipher**
- Problem: key size!
- Theoretical attacks have been identified…
  None practical enough to cause major concerns today
- Triple DES (3DES) designed to address DES's key size problem
- Continued support mostly for legacy reasons -> **AES is preferred today**



Advanced Encryption Standard (AES)

- Winner of a NIST competition (no gov. intervention…)
- Faster than 3DES, variable key lengths, 128-bit blocks
- Underlying design based on substitution-permutation network -> multiple rounds of byte substitutions, shifts, mixing, and adding



Using Block Ciphers

- How to encrypt data using block cipher...
  - if data is BIGGER than the block cipher?
  - if data is SMALLER than the block cipher?
  - if data size is a multiple of the cipher block size?

Encryption Modes

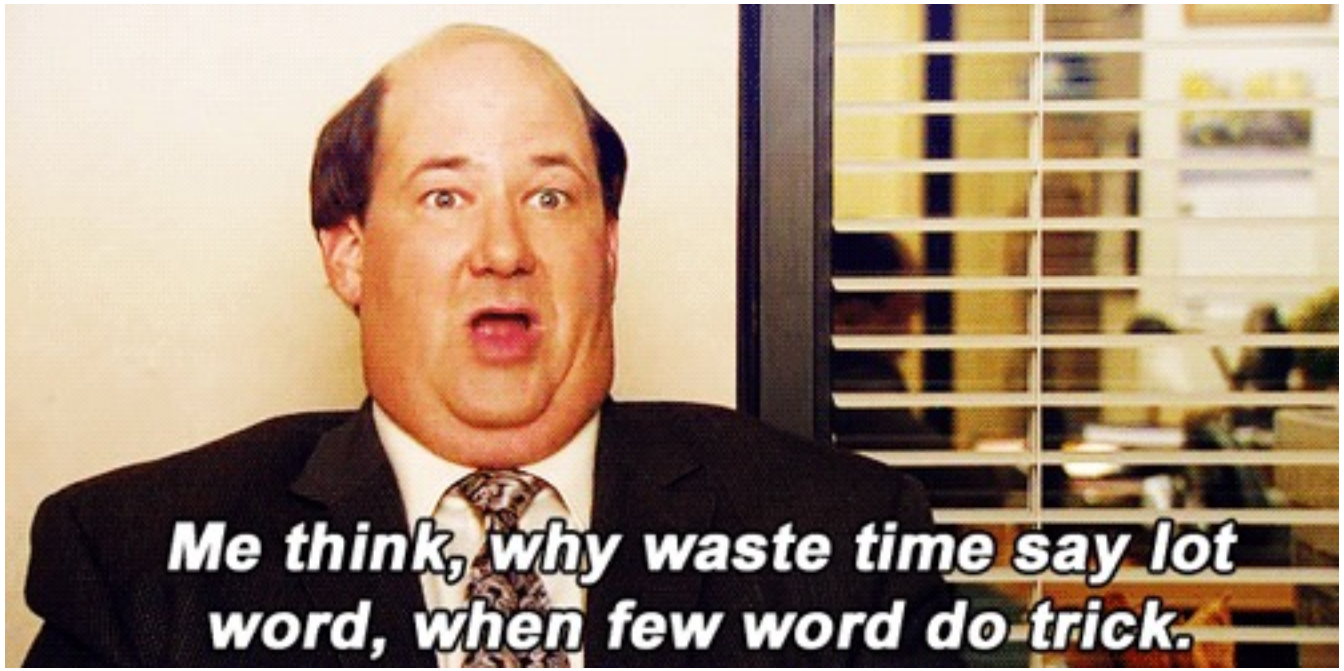
- An **encryption mode** or **mode of operation** refers to the many ways to make the input of an encryption algorithm **different**.

- **Examples**:
  - Electronic Codebook (ECB),
  - Cipher Block Chaining (CBC),
  - Propagating CBC (PCBC),
  - Cipher Feedback (CFB),
  - Output Feedback (OFB),
  - Counter (CTR),
  - …

WARNING!

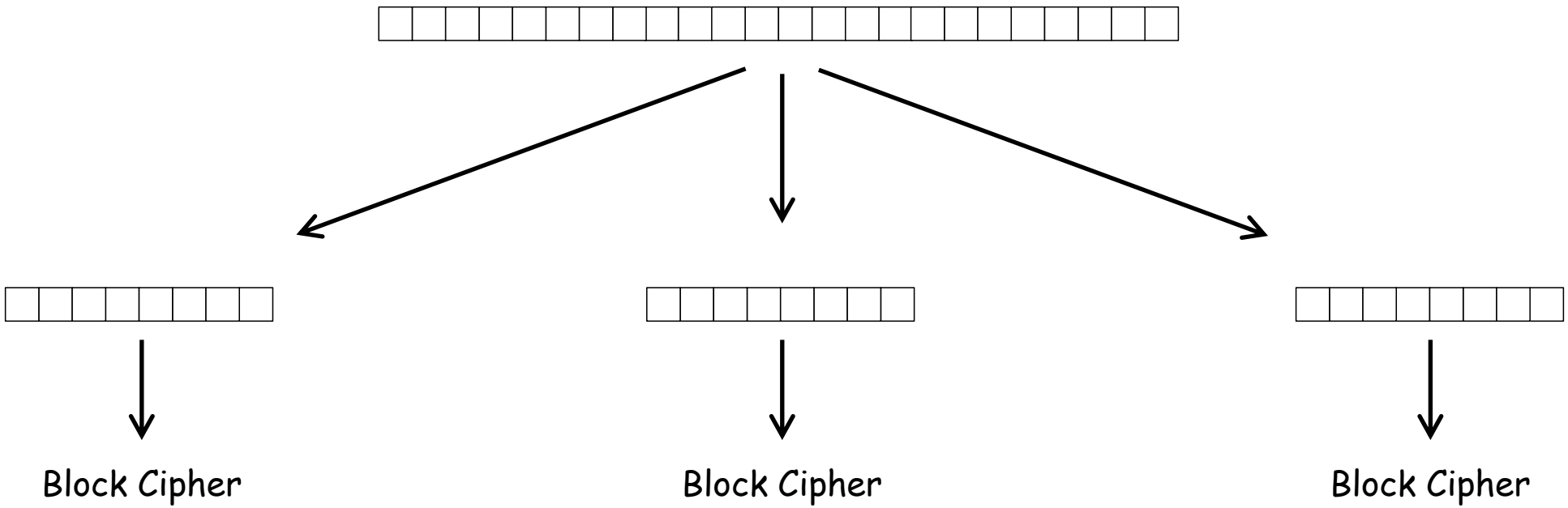
If we aren't careful about how we conduct encryption operations, we may accidentally reveal information about the plaintext…

IDEA: Break large input up into blocks



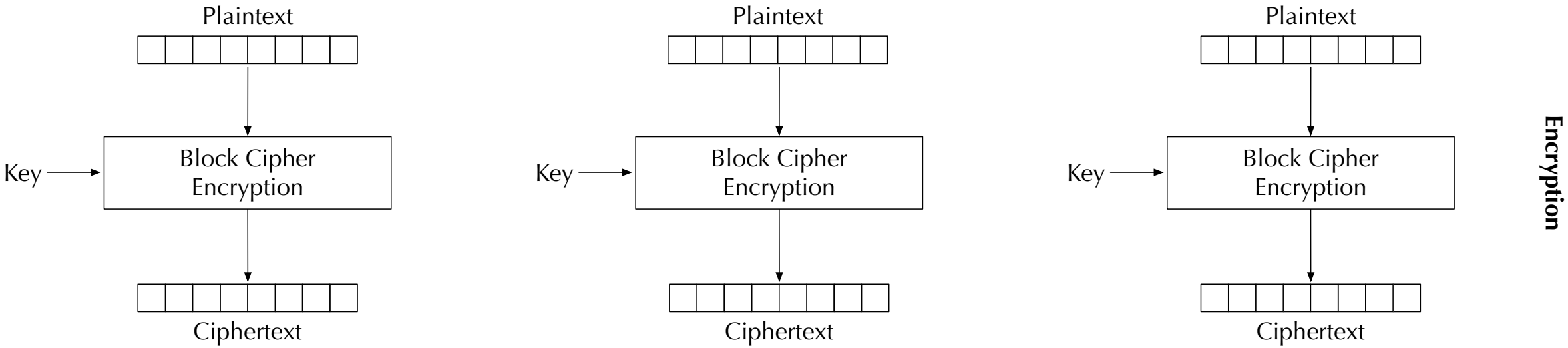
IDEA: Break large input up into n-bit blocks

where n depends on the size of the block cipher

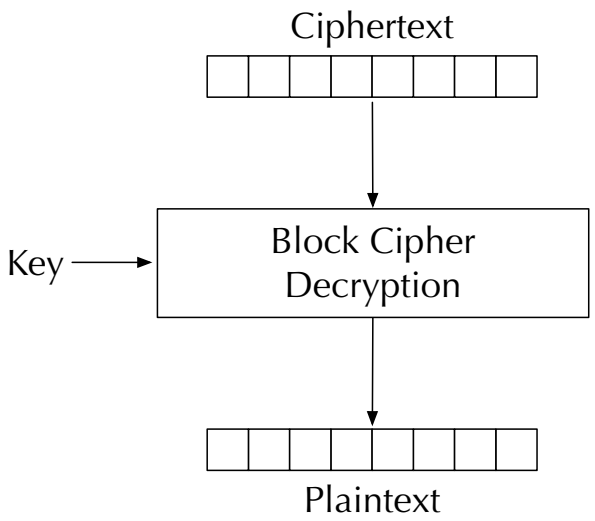


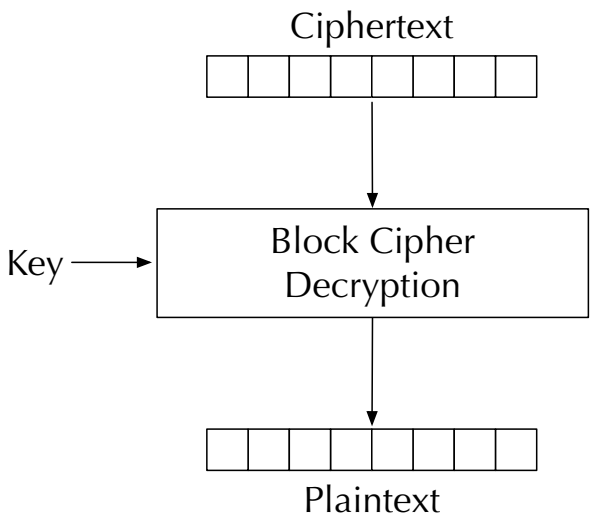
Electronic Codebook (ECB) Mode

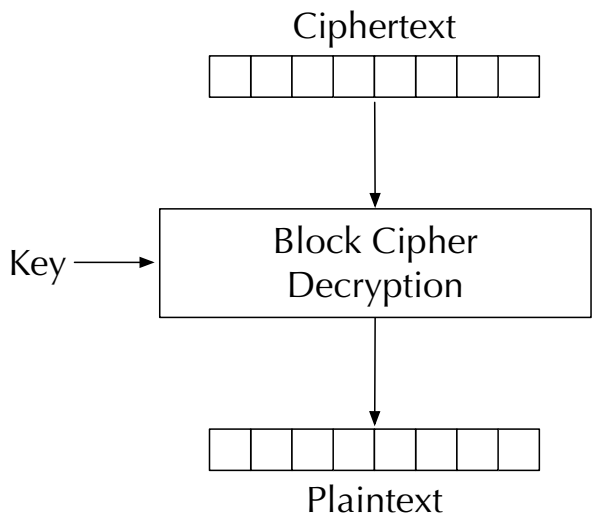
Electronic Codebook (ECB) Mode



Electronic Codebook (ECB) Mode



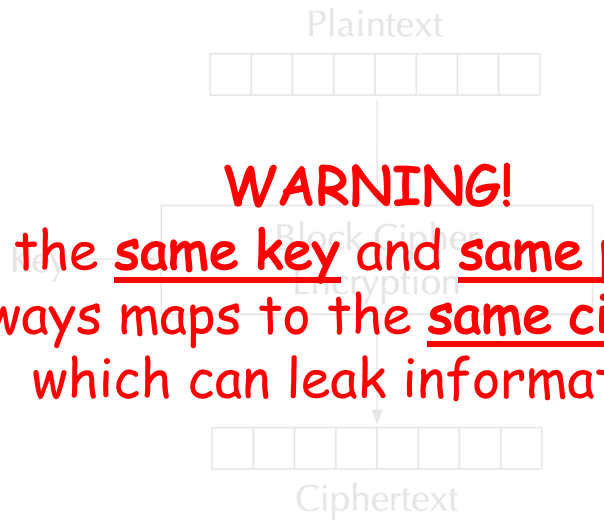


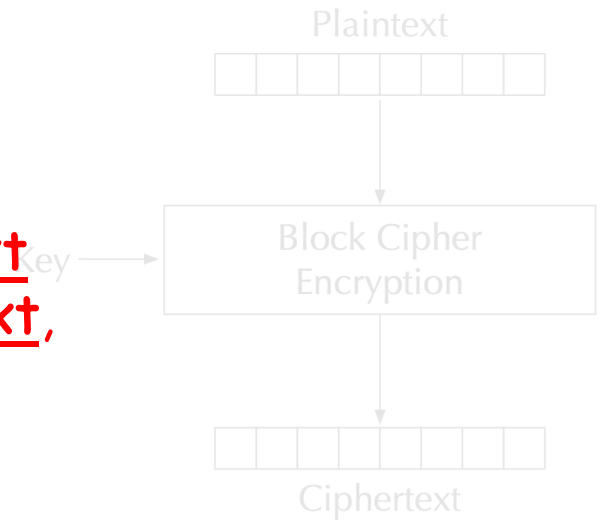


Decryption

WARNING!

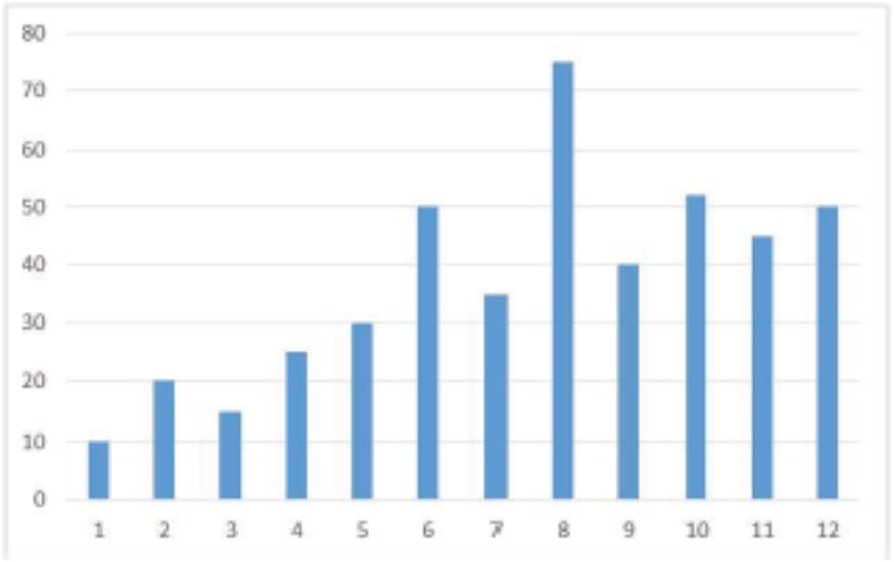
For the **same key** and **same plaintext**
-> always maps to the **same ciphertext**,
which can leak information!



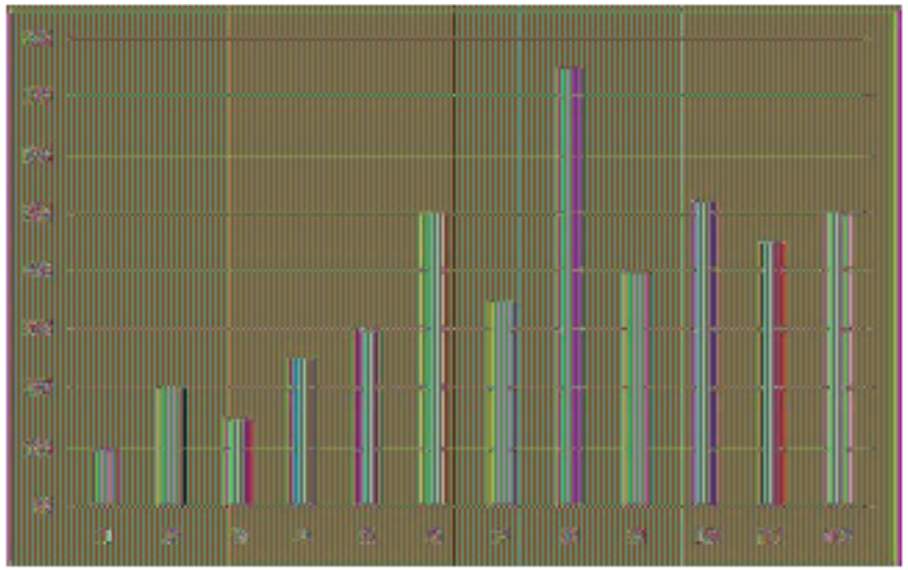


Encryption

Electronic Codebook (ECB) Mode



(a) The original image (`pic_original.bmp`)



(b) The encrypted image (`pic_encrypted.bmp`

WARNING!

For the **same key** and **same plaintext**
-> always maps to the **same ciphertext**,
which can leak information!

You Try!

- Using **openssl enc** command:

```
$ openssl enc -aes-128-ecb -e -in plain.txt  -out cipher.txt \
      -K 00112233445566778899AABBCCDDEEFF

$ openssl enc -aes-128-ecb -d -in cipher.txt -out plain2.txt \
      -K 00112233445566778899AABBCCDDEEFF
```

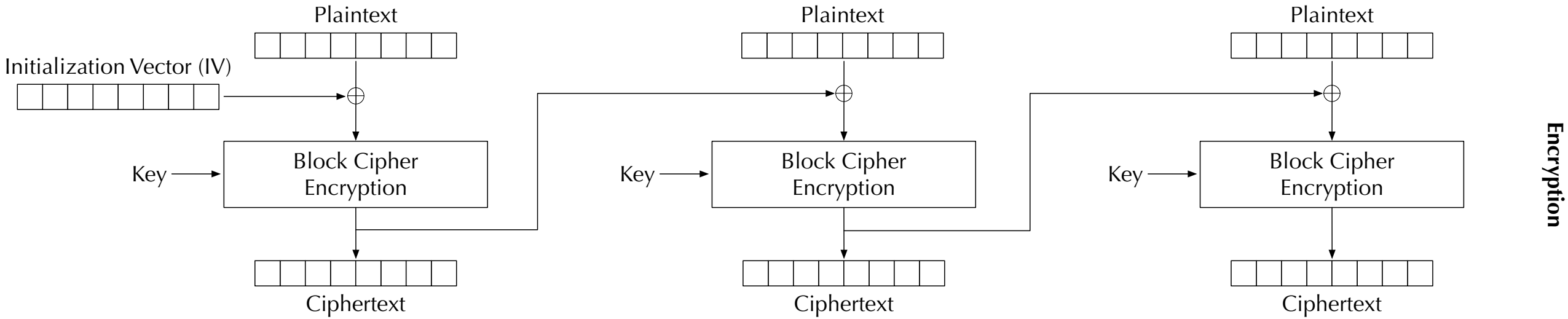
Task:
encrypt a plaintext, then decrypt it

⚠ This is a **terrible** key to use in practice! We only use it here for test purposes.

- The **-aes-128-ecb** option specifies 128-bit (key size) AES algorithm w/ ECB mode
- The **-e** option indicates encryption
- The **-d** option indicate decryption
- The **-K** option is used to specify the encryption/decryption key

Cipher Block Chaining (CBC) Mode

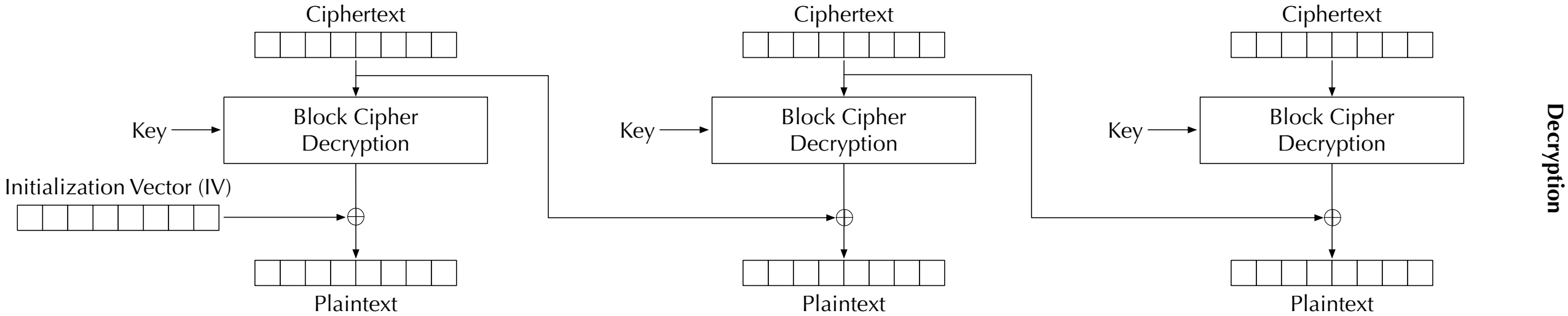
Cipher Block Chaining (CBC) Mode



- CBC introduces **block dependency**; i.e., $C_i = E_K(P_i \oplus C_{i-1})$
- The main purpose of the **initialization vector (IV)** is to ensure that even if two plaintexts are identical, their ciphertexts are different, because different IVs will be used.
- Encryption **cannot** be parallelized
- Decryption **can** be parallelized; i.e., $P_i = D_K(C_i) \oplus C_{i-1}$

Cipher Block Chaining (CBC) Mode

- CBC introduces **block dependency**; i.e., $C_i = E_K(P_i \oplus C_{i-1})$
- The main purpose of the **initialization vector (IV)** is to ensure that even if two plaintexts are identical, their ciphertexts are different, because different IVs will be used.
- Encryption **cannot** be parallelized
- Decryption **can** be parallelized; i.e., $P_i = D_K(C_i) \oplus C_{i-1}$



You Try!

- Using **openssl enc** command:

```
$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher1.txt \
    -K  00112233445566778899AABBCCDDEEFF \
    -iv 000102030405060708090A0B0C0D0E0F

$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher2.txt \
    -K  00112233445566778899AABBCCDDEEFF \
    -iv 000102030405060708090A0B0C0D0E0E
```

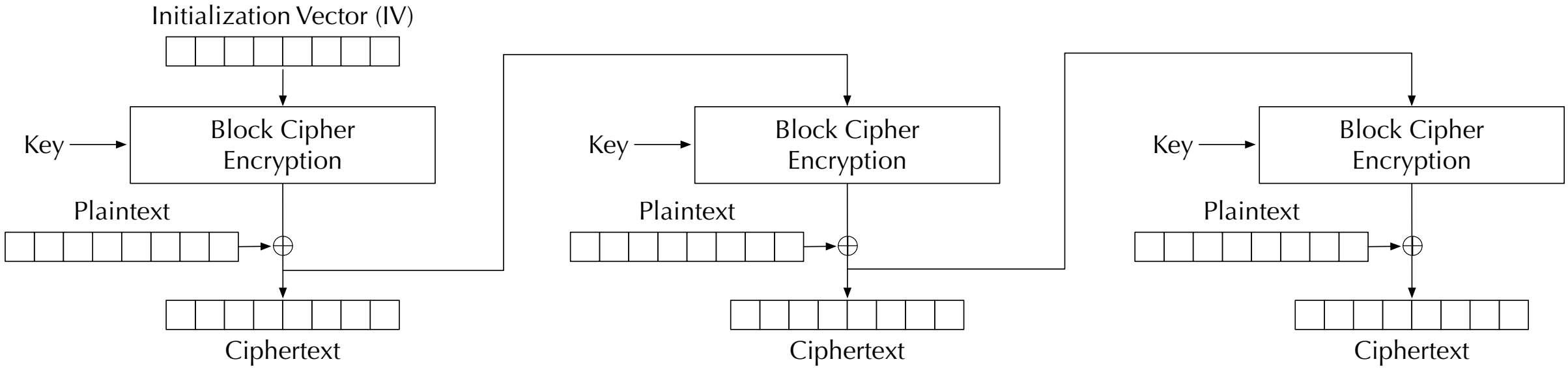
⚠ This is a **terrible** key to use in practice! We only use it here for test purposes.

Task:
encrypt a plaintext, with the same key, but a different IV
---
use xxd to view cipher1.txt & cipher2.txt

- The **-aes-128-cbc** option specifies 128-bit (key size) AES algorithm w/ ECB mode
- The **-e** option indicates encryption
- The **-K** option is used to specify the encryption/decryption key
- The **-iv** option is used to specify the initialization vector (IV)

Cipher Feedback (CFB) Mode

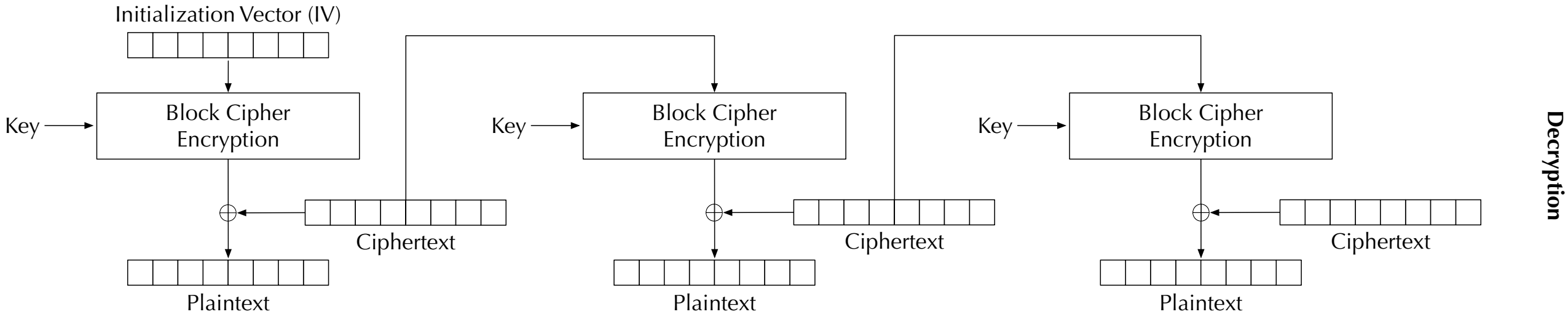
Cipher Feedback (CFB) Mode



- Similar to CBC, but slightly different… a block cipher is turned into a **stream cipher**!
- Ideal for encrypting real-time data.
- Padding not required for the last block.
- Encryption can only be conducted sequentially -> we need to wait for all the plaintext
- Decryption using the CFB mode can be parallelized

Cipher Feedback (CFB) Mode

- Similar to CBC, but slightly different… a block cipher is turned into a **stream cipher**!
- Ideal for encrypting real-time data.
- Padding not required for the last block.
- Encryption can only be conducted sequentially -> we need to wait for all the plaintext
- Decryption using the CFB mode can be parallelized



You Try!

```
$ echo -n 'abcdefghijklmnopqrstuvwxyz' > plain.txt

$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher1.txt \
    -K  00112233445566778899AABBCCDDEEFF \
    -iv 000102030405060708090A0B0C0D0E0F
$ openssl enc -aes-128-cfb -e -in plain.txt -out cipher2.txt \
    -K  00112233445566778899AABBCCDDEEFF \
    -iv 000102030405060708090A0B0C0D0E0F

$ ls -l *.txt
-rw-rw-r-- 1 seed seed 32 Apr  1 20:02 cipher1.txt
-rw-rw-r-- 1 seed seed 26 Apr  1 20:02 cipher2.txt
-rw-rw-r-- 1 seed seed 26 Apr  1 20:02 plain.txt
```

⚠ This is a **terrible** key to use in practice! We only use it here for test purposes.

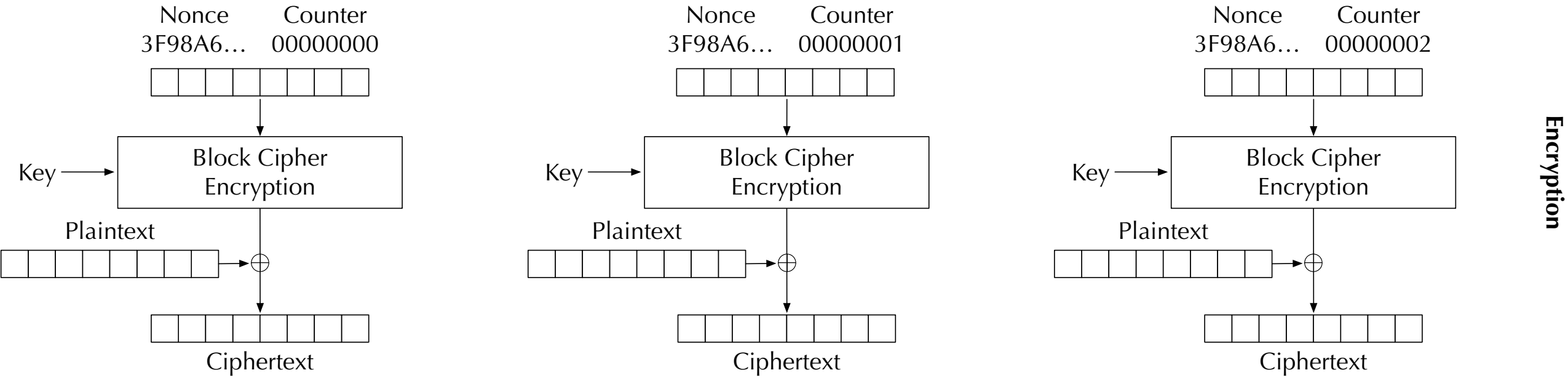
Task:
Compare encrypted plaintext using CBC vs. CFB

(same key, same iv, different mode)

- Plaintext size is 26 bytes
- **CBC mode**: ciphertext is 32 bytes due to padding
- **CFB mode**: ciphertext size is the same as the plaintext (26 bytes)

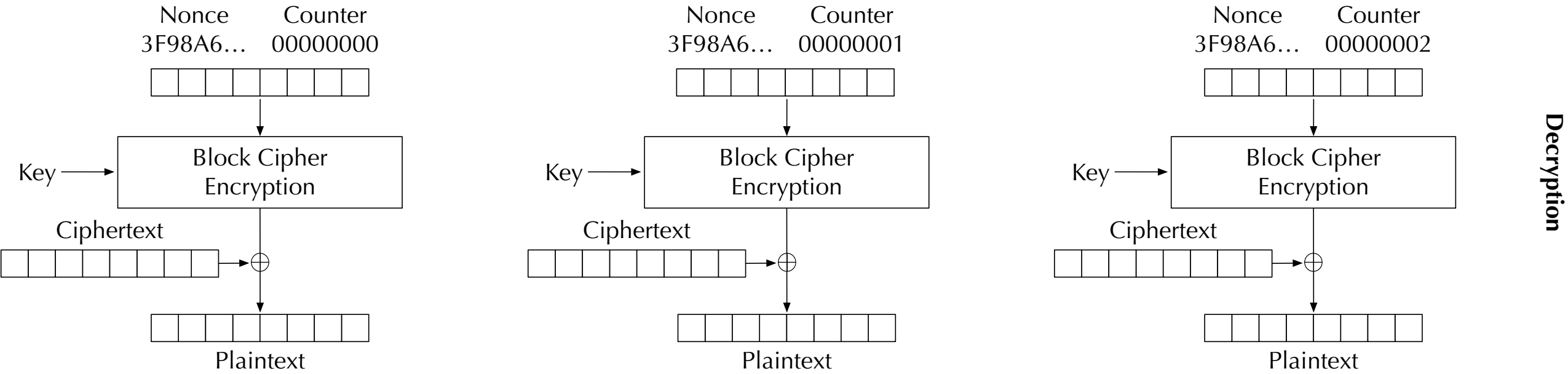
Counter (CTR) Mode

Counter (CTR) Mode



- Use a counter to generate the key streams
- No key stream can be reused; the counter value for each block is prepended with a randomly generated value called **nonce** (same idea as the IV)
- Both encryption and decryption can be parallelized
- The key stream in the CTR mode can be calculated in parallel during the encryption

Counter (CTR) Mode



- Use a counter to generate the key streams
- No key stream can be reused; the counter value for each block is prepended with a randomly generated value called **nonce** (same idea as the IV)
- Both encryption and decryption can be parallelized
- The key stream in the CTR mode can be calculated in parallel during the encryption

Modes for Authenticated Encryption (AE)

- We've now seen various encryption modes: EBC, CBC, CFB, CTR
- None of the encryption modes discussed so far can be used to achieve **message authentication** (only message confidentiality…)
- There are modes of operation that have been designed to combine **encryption** and **message authentication codes (MAC)**
  - GCM (Galois/Counter Mode)
  - CCM (Counter with CBC-MAC)
  - OCB mode (Offset Codebook Mode)