Confidentiality

Overview

What do we mean by confidentiality?

- The protected asset shall not be disclosed to unauthorized entities

Question: Are the following statements confidentiality requirements?

1. Having a password file and trying to protect it from not being disclosed

2. A film production company tries to prevent piracy (i.e. prevent users from obtaining the movies without paying)

Overview

What do we mean by confidentiality?

- The protected asset shall not be disclosed to unauthorized entities

Question: Are the following statements confidentiality requirements?

1. Having a password file and trying to protect it from not being disclosed
   Answer: yes, we say that the file must remain confidential

2. A film production company tries to prevent piracy (i.e. prevent users from obtaining the movies without paying)

Overview

What do we mean by confidentiality?

- The protected asset shall not be disclosed to unauthorized entities

Question: Are the following statements confidentiality requirements?

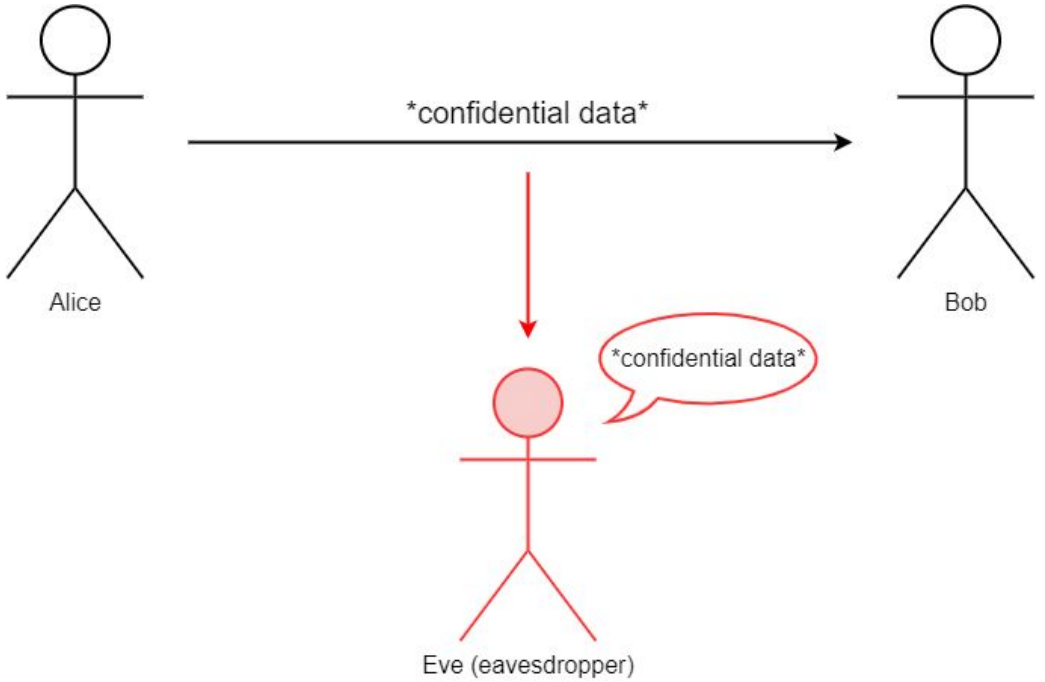
1. Having a password file and trying to protect it from not being disclosed
   Answer: yes, we say that the file must remain confidential

2. A film production company tries to prevent piracy (i.e. prevent users from obtaining the movies without paying)
   Answer: no (in fact, this is a possession/control requirement)

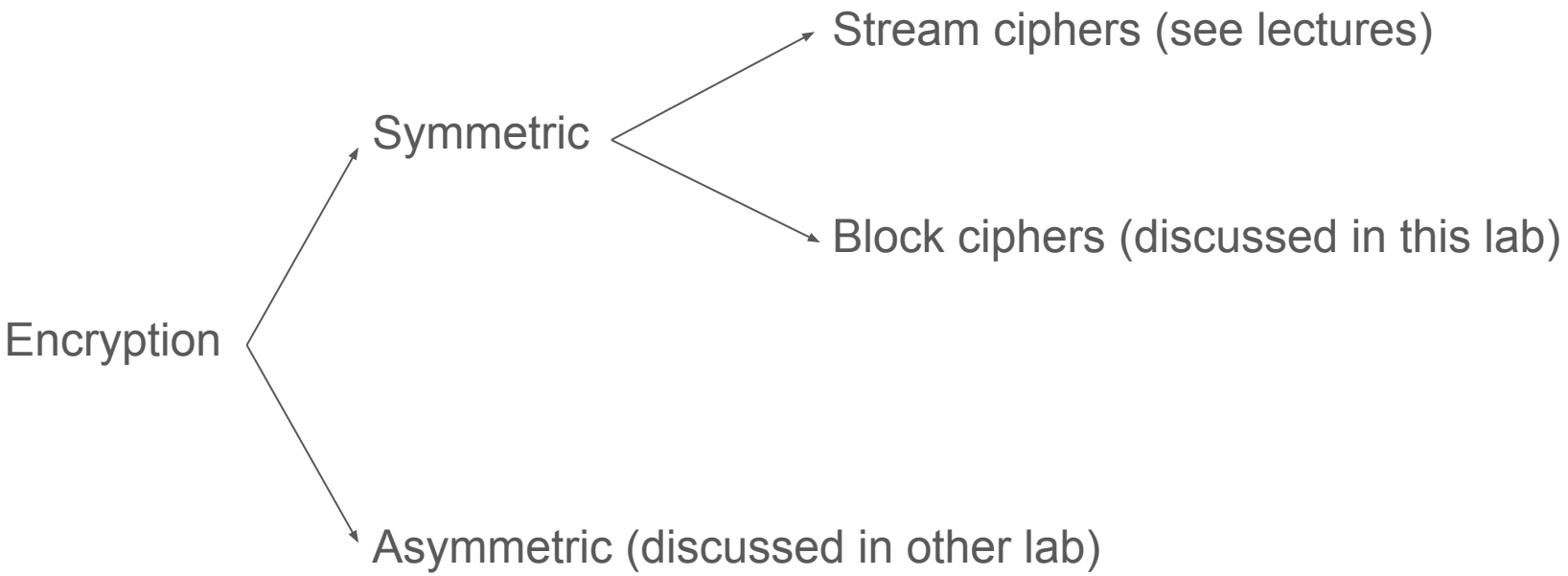
Problem representation

- Alice and Bob are two honest principals
- Alice sends data to Bob over an **unprotected** channel
- Eve, the adversary, can listen to the communication since the channel is unprotected
- **Problem** when data is required to remain confidential

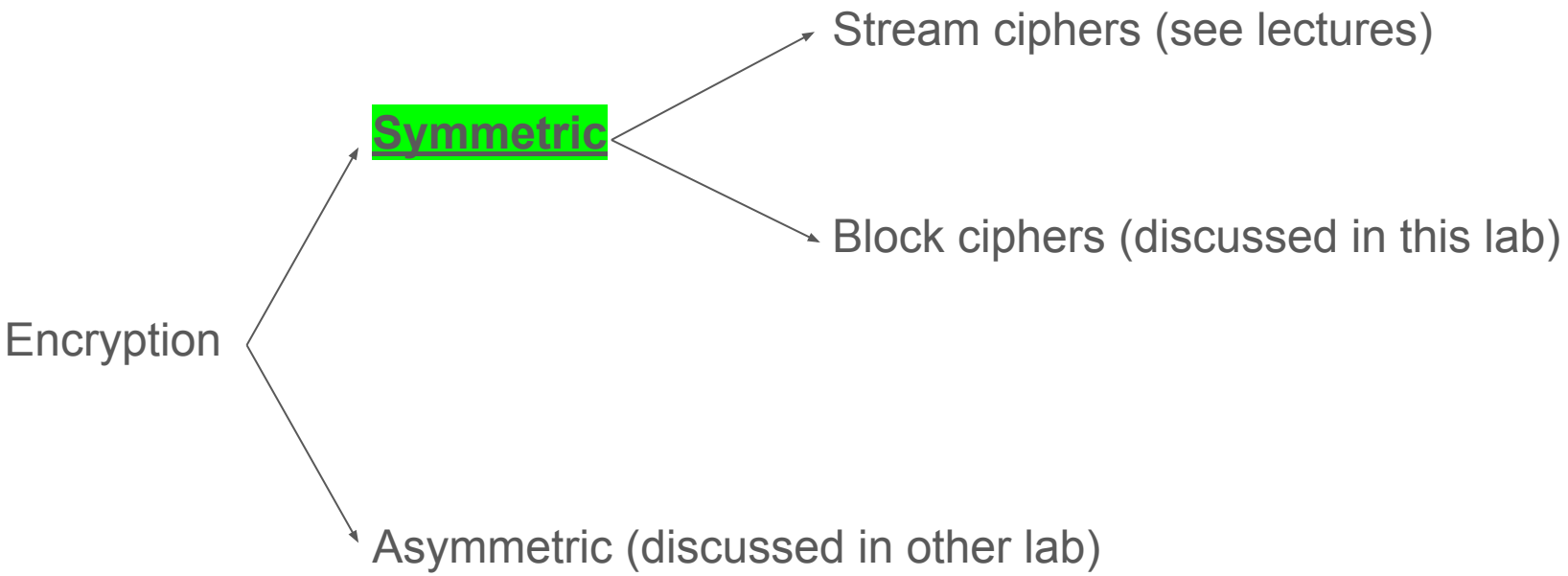
Example of eavesdropping?



Encryption classification



Encryption classification



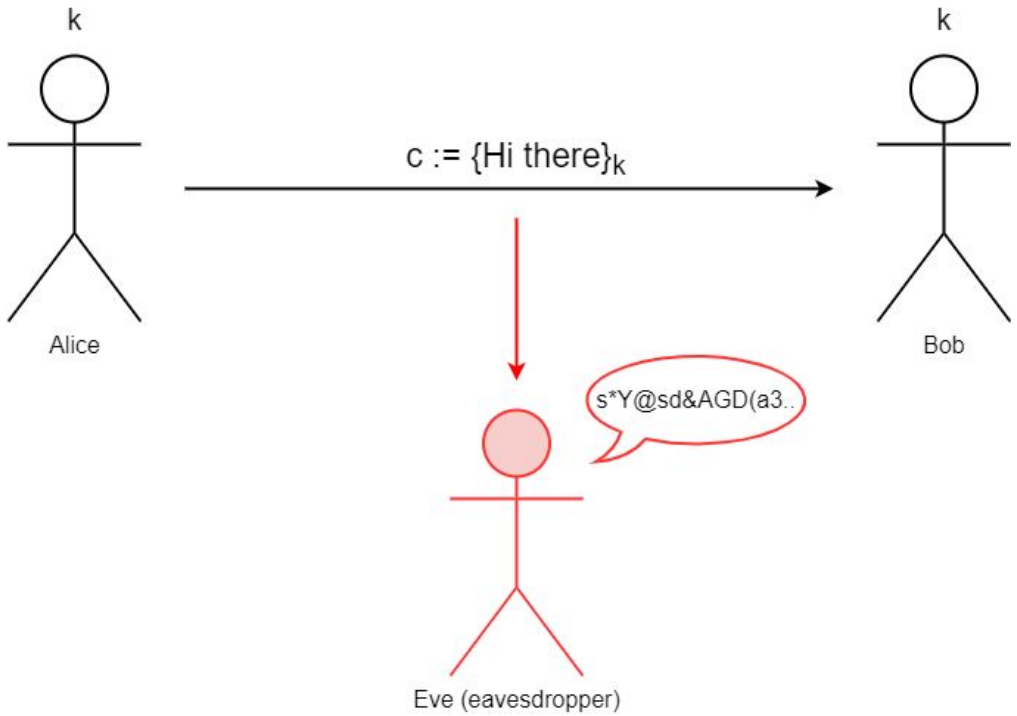
Confidentiality through symmetric encryption

Notations:

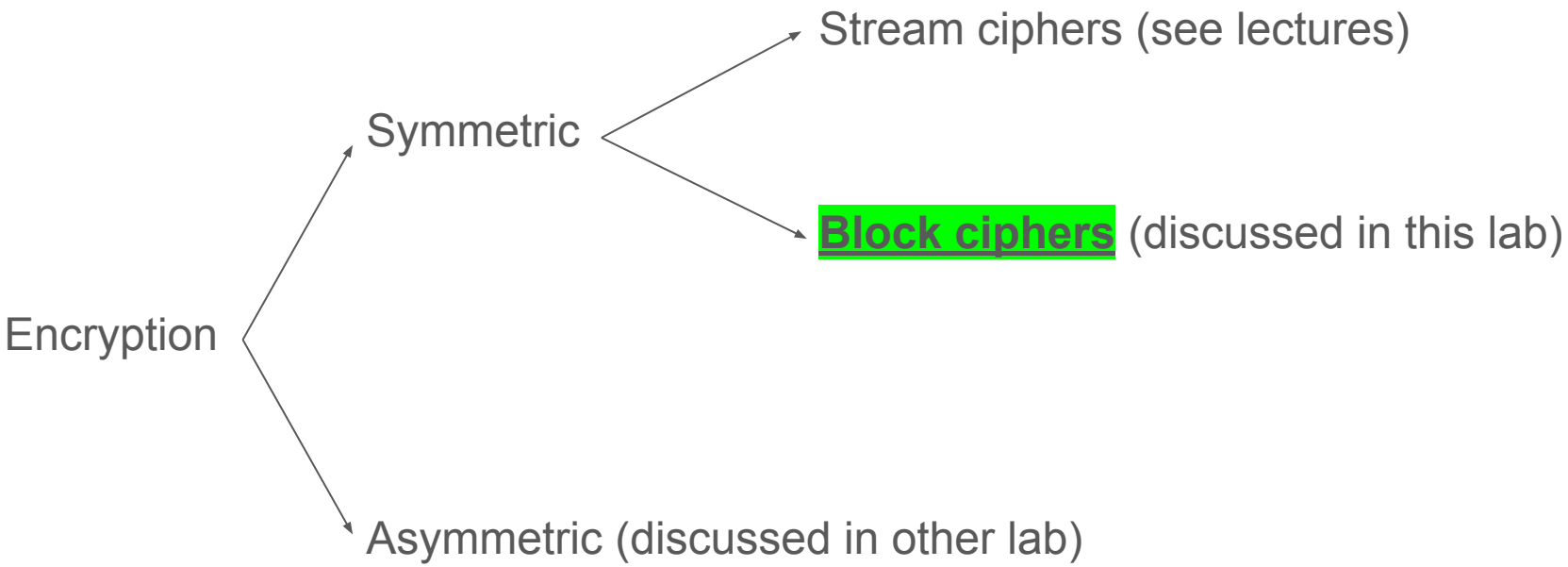
- *c* -> ciphertext
- $\{m\}_k$ -> message *m* encrypted with the secret key *k*

Idea:

- Alice and Bob share the same key *k* (symmetric = same key for encryption and decryption)
- Eve can still intercept traffic, but this time will see a random looking string (**indistinguishable from random**)



Encryption classification



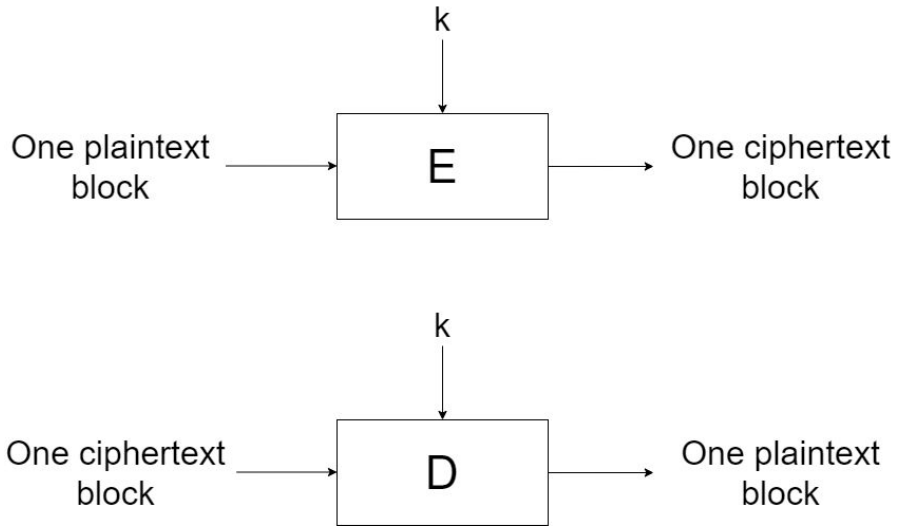
Primitive: block ciphers

Work principle

- Encrypt/decrypt one block at a time
- Block have fixed size (e.g. 16 bytes)

Block ciphers

- AES (Advanced Enc. Standard)
  - **Current standard (to be used)**
  - Key size: 128/192/256 bits
  - Block size: 16 bytes
- DES (Data Enc. Standard)
  - **Old, deprecated standard (not to use)**
  - Key size: 56 bits
  - Block size: 8 bytes
- Many others

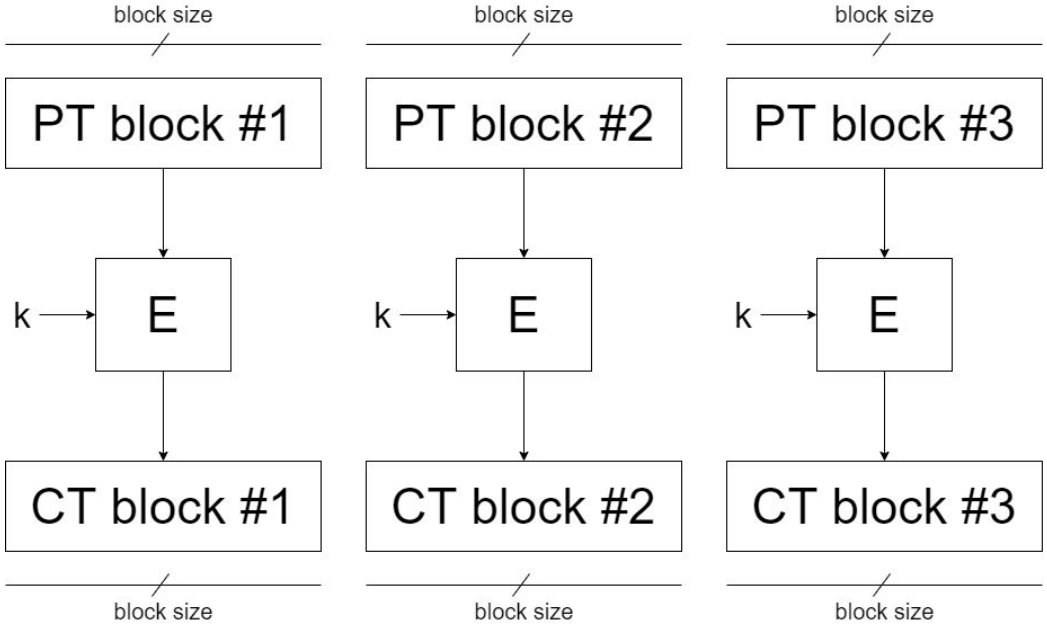


Encryption mode

Having a message larger than the block size, how to extend the algorithm?

Example: ECB

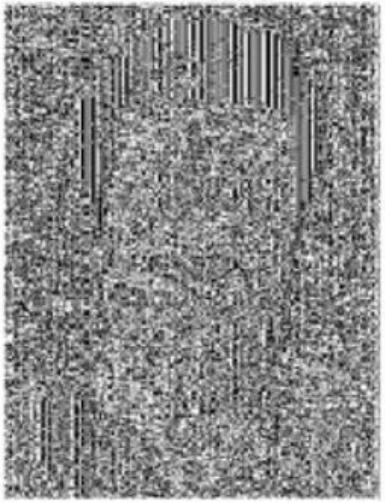
- Split the message into blocks
- Apply the encryption algorithm to each PT block with the key *k*
- Concatenate resulting CT blocks
- Note:
  PT#i = PT#j => CT#i = CT#j



ECB

An example plaintext

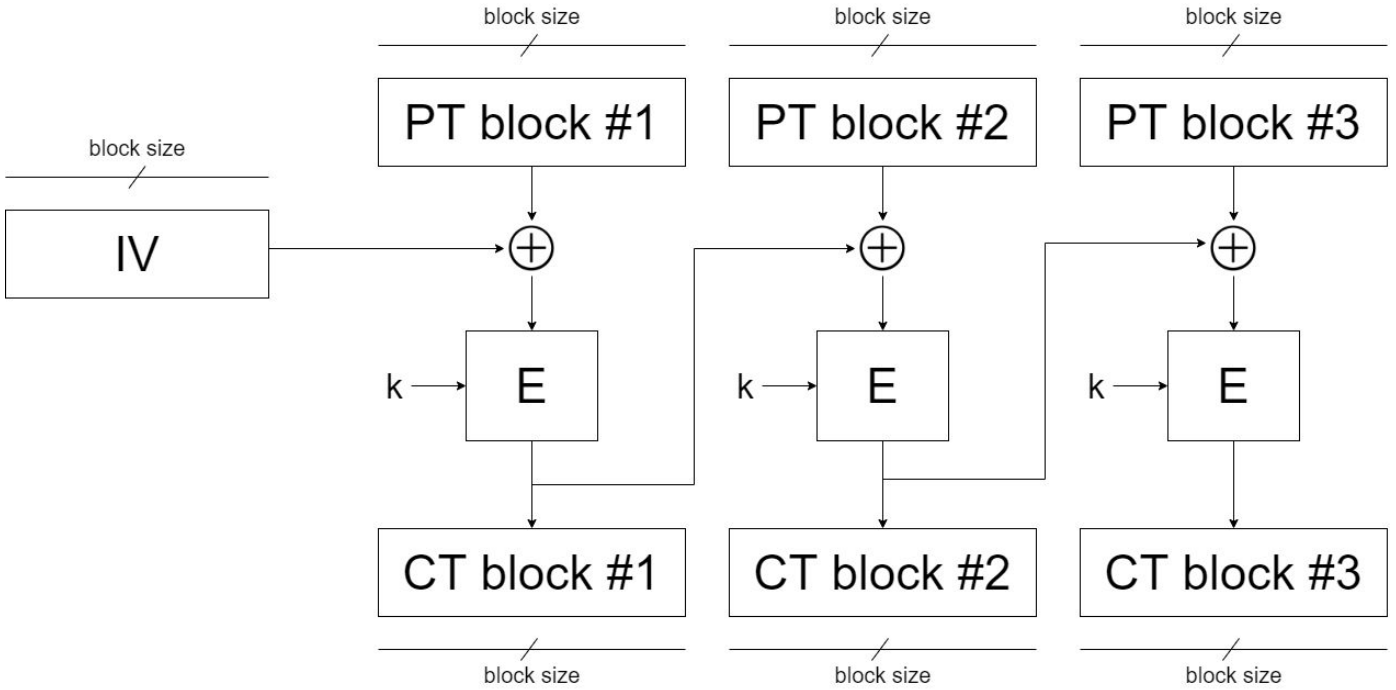
Encrypted with AES in ECB mode



courtesy B. Preneel

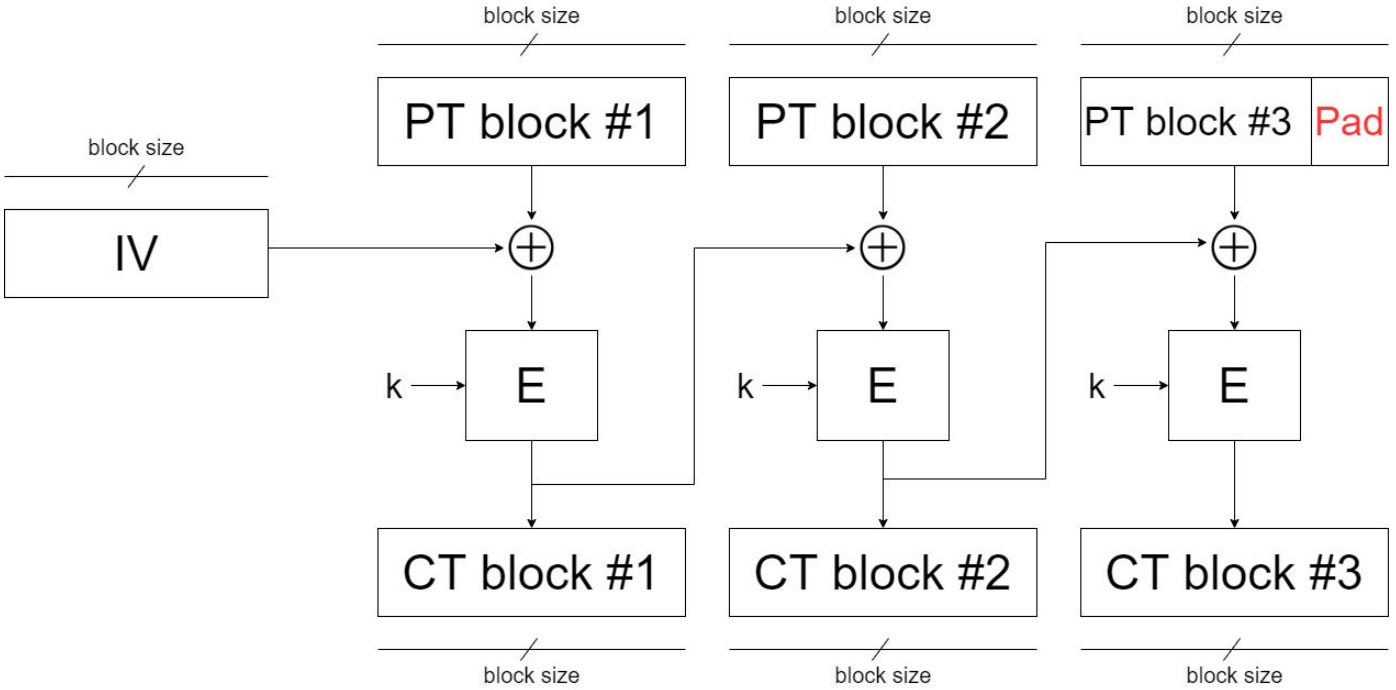
Encryption mode (cont.)

Cipher block chaining (CBC) w/ random IV - guarantees all CT blocks are different



Padding

What if the message is not multiple of the block size?



0x3 0x3 0x3

E.g.: of padding 3 bytes w/ PKCS#7

Symmetric encryption with WinRAR

Demo

Symmetric encryption in .NET

```
/************************ ENCRYPTION ***********************/
/************************   SETUP   ***********************/

// Create AES object for encryption
SymmetricAlgorithm aesEncryptAlg = AesManaged.Create();

// Select key size (in bits)
// Note: possible values for AES are 128/192/256
aesEncryptAlg.KeySize = 192;

// Select mode
aesEncryptAlg.Mode = CipherMode.CBC;

// Select padding
aesEncryptAlg.Padding = PaddingMode.PKCS7;

// Generate/set encyrption key
// In this case it is generated randomly, so the value must be saved
aesEncryptAlg.GenerateKey();
savedSymmetricKey = (byte[])aesEncryptAlg.Key.Clone();

// Generate fresh IV, or set value
// In this case it is generated randomly, so the value must be saved
aesEncryptAlg.GenerateIV();
savedIv = (byte[])aesEncryptAlg.IV.Clone();
```

```
/************************ ENCRYPTION ***********************/
/************************ PROCESSING ***********************/

// Create streams
MemoryStream memoryStreamEnc = new MemoryStream();
CryptoStream cryptoStreamEnc = new CryptoStream(memoryStreamEnc,
                                aesEncryptAlg.CreateEncryptor(),
                                CryptoStreamMode.Write);

// Encrypt by writing to the stream
// Note: mandatory to close the stream when finished
cryptoStreamEnc.Write(plaintext_bytes, 0, plaintext_bytes.Length);
cryptoStreamEnc.Close();

// Read the ciphertext and close stream
byte[] ciphertext_bytes = memoryStreamEnc.ToArray();
memoryStreamEnc.Close();

// Return the ciphertext
return ciphertext_bytes;
```

Symmetric encryption in .NET (cont.)

```
/************************ DECRYPTION ************************/
/************************   SETUP   ************************/

// Alloc array for the plaintext
byte[] plaintext_bytes = new byte[ciphertext_bytes.Length];

// Create AES object for decryption
SymmetricAlgorithm aesDecryptAlg = AesManaged.Create();

// Select key size (in bits)
// Note: must be the same algorithm, i.e., choose the same key size
aesDecryptAlg.KeySize = 192;

// Select mode
// Note: must be the same mode used for encryption
aesDecryptAlg.Mode = CipherMode.CBC;

// Select padding
// Note: must be the same padding used for encryption
aesDecryptAlg.Padding = PaddingMode.PKCS7;

// Set decryption key
// Note: must be the same key used for encryption
aesDecryptAlg.Key = savedSymmetricKey;

// Set IV
// Note: must be the same IV used for encryption
aesDecryptAlg.IV = savedIv;
```

```
/************************ DECRYPTION ************************/
/************************ PROCESSING ************************/

// Create streams
MemoryStream memoryStreamDec = new MemoryStream(ciphertext_bytes);
CryptoStream cryptoStreamDec = new CryptoStream(memoryStreamDec,
                                aesDecryptAlg.CreateDecryptor(),
                                CryptoStreamMode.Read);

// Decrypt by reading from the stream
// Note: mandatory to close the stream when finished
cryptoStreamDec.Read(plaintext_bytes, 0, ciphertext_bytes.Length);
cryptoStreamDec.Close();
memoryStreamDec.Close();

// Return the plaintext
return plaintext_bytes;
```

Guide

Project

- New -> Windows Forms Application (.NET Framework)

Visual elements

- Button, Label, TextBox - **from toolbox**
- After adding a visual element - **double click to generate event method (e.g. ButtonEnc_Click (object sender, EventArgs e)**

What to try: change the key size, use different modes, padding, etc.

- Can you find ECB duplicate ciphertext blocks?
- Can you find speed differences when using different key sizes for AES?