POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Information Security

Symmetric Ciphers

Operation modes, Padding and Key length

Lecturer (Academic year 2016-2017):
Marisa Maximiano

UNDERGRADUATE DEGREE IN
COMPUTER ENGINEERING

Summary

- Symmetric Ciphers
  - Block Ciphers principles
  - Operation modes
    - Electronic code book
    - Cipher block chaining mode
    - Cipher feedback mode
    - Output feedback mode
    - Counter mode
  - Padding
  - Key length
- Symmetric algorithms in .NET framework

Symmetric ciphers

Block Ciphers

- The data is **divided into fixed-size blocks** and encrypted one block at a time

Stream ciphers

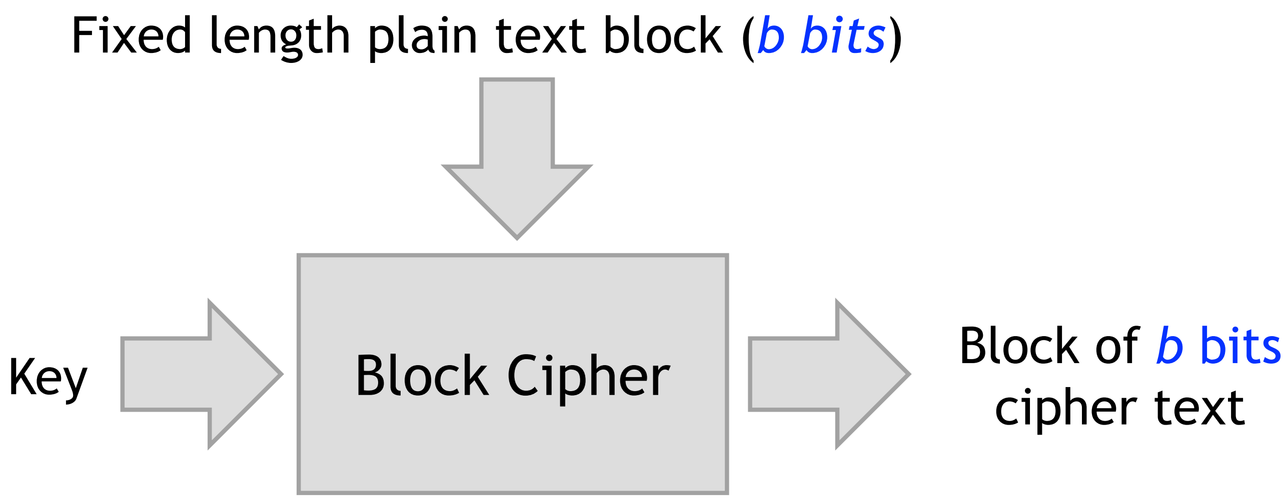
- The data is encrypted one bit at a time
- It uses an infinite **stream** of pseudorandom bits as the key
- For a stream cipher implementation to remain secure:
  - its pseudorandom generator should be unpredictable
  - key should never be reused

Block ciphers

Most of the algorithms currently used implement block ciphers

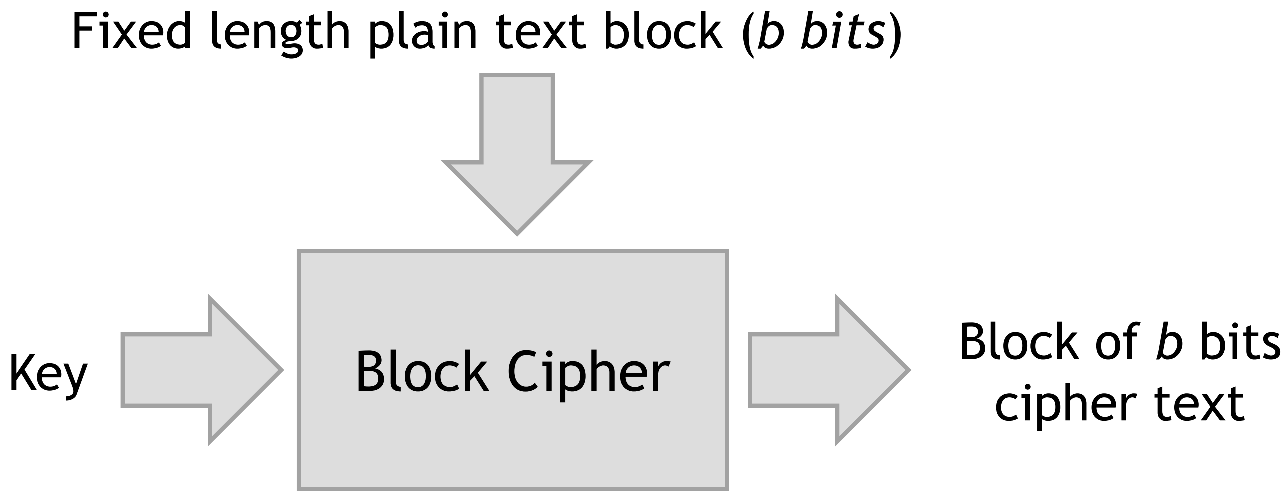
Operation modes

- **Key points**
  - To remember…



Operation modes

- **Key points**
  - To remember…



If plain text greater than *b* bits, block cipher breaks it in *b-bit* blocks

Operation modes

- **Key points**
  - To remember...

Many block ciphers have a Feistel structure

- **Feistel structure**
  - Number of identical rounds of processing
  - In each round:
    - a **substitution** is performed on one half of the data being processed
    - followed by a **permutation** that interchanges the two halves
  - The original key is expanded so that a different key is used for each round

Operation modes

- **Key points**
  - To remember…

When multiple blocks are encrypted using the same key some **security** issues arise!

Five operation modes were defined by NIST

To enhance the cryptographic effect

Operation modes

Five modes of operation have been standardized by NIST for use with **symmetric block ciphers** such as DES and AES, etc.

Operation modes

Mode of operation is a technique for enhancing the effect of a cryptographic algorithm

1. **Electronic codebook (ECB)**
2. **Cipher block chaining (CBC) mode**
3. **Cipher feedback (CFB) mode**
4. **Output feedback (OFB) mode**
5. **Counter (CTR) mode**

Operation modes

- **1. Electronic Code Book (ECB)**

✓ Good performance
✓ It targets small amount of data (e.g. secret keys)

✓ Attention: the same plain text always produces the same cipher text!

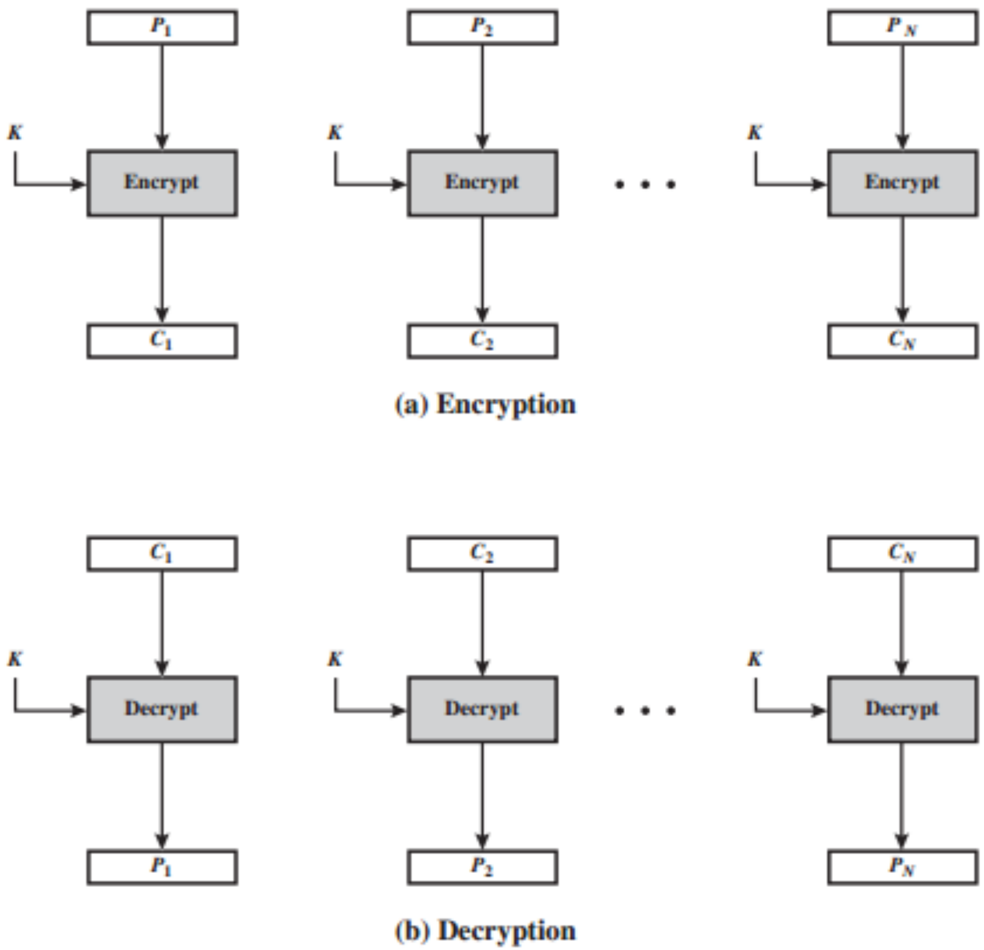


Fig. – Electronic Code Block mode

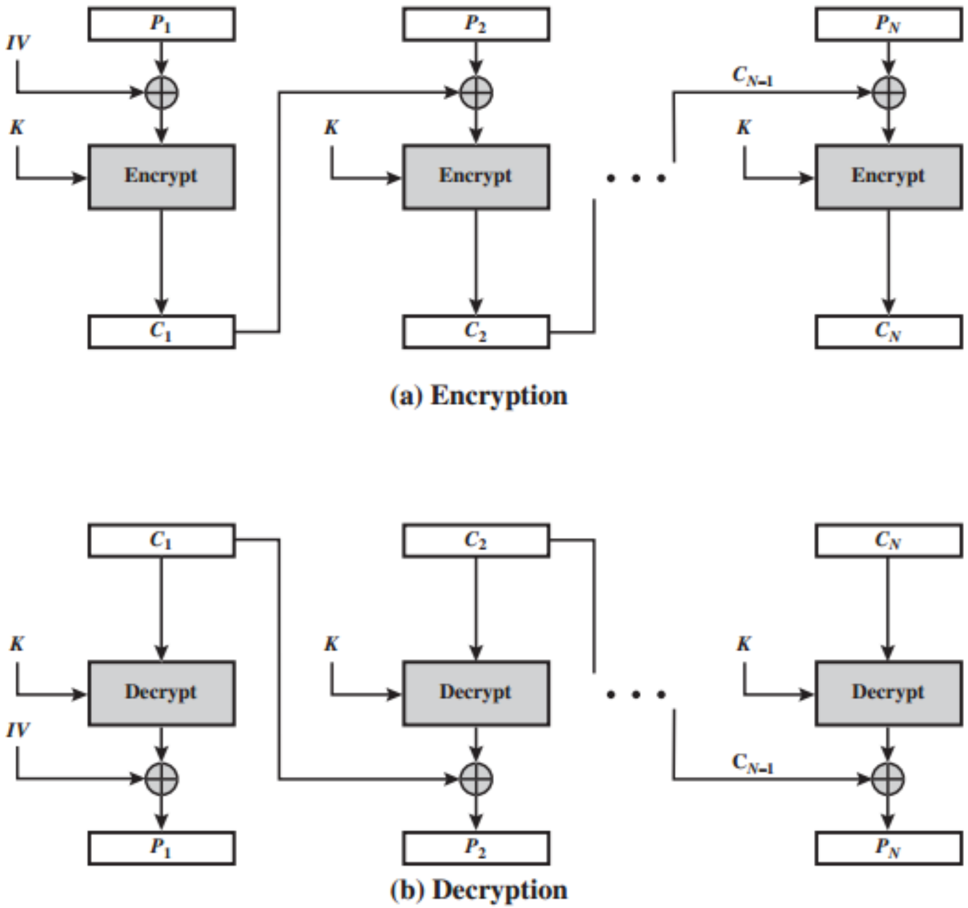
Operation modes

- **2. Cipher block chaining (CBC) mode**
  - Solve the security issue of ECB mode (same cipher text)
  - CBC: technique in which the same plain text block, if **repeated**, produce different cipher text blocks
  - Pain text is XORed with last block cipher text

Operation modes

- **2. Cipher block chaining (CBC) mode**

Fig. – Cipher block chaining mode



- ✓ Here is the need for the initialization vector …
- ✓ The IV has the same length as the cipher text block
- ✓ IV should be secret

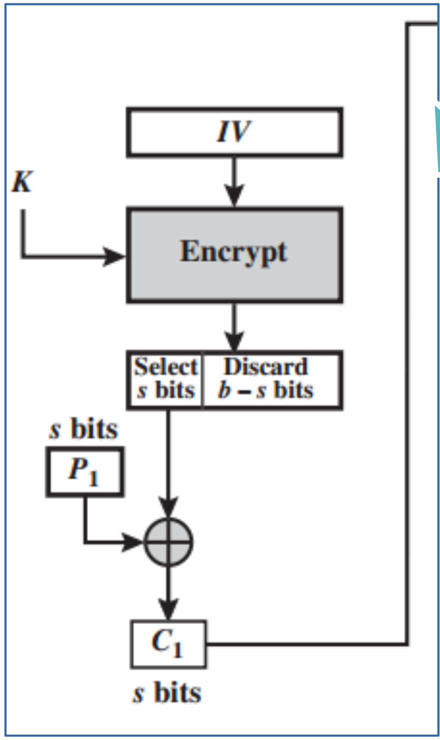
Operation modes

- 3. Cipher feedback (CFB) mode

✓ Cipher text **depends of current** and **previous** cipher texts

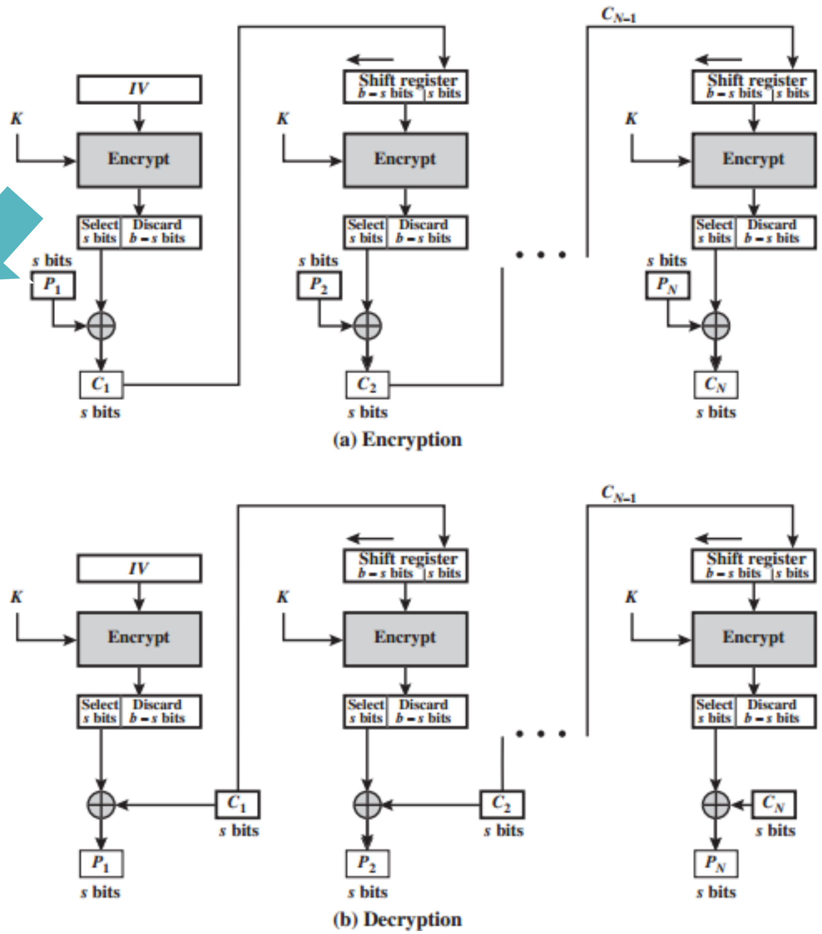
✓ Second block ciphering starts when the shift register is full (when previous block has been completely processed!)

✓ If *tx* error, big impact (on all following blocks)!



Transmit C_1

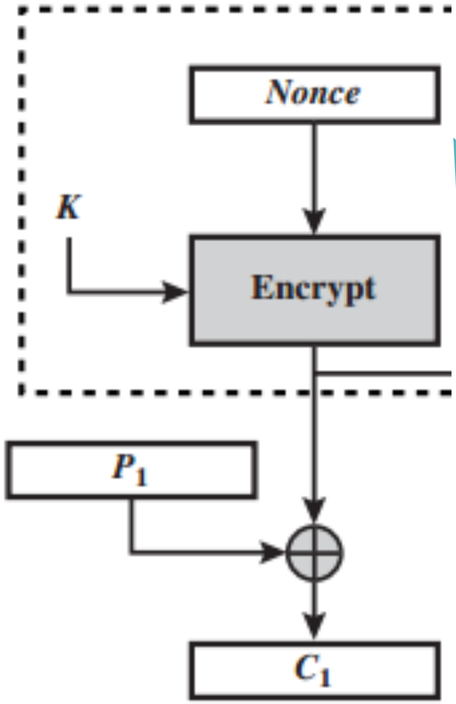
S–bit Cipher Feedback (CFB) Mode



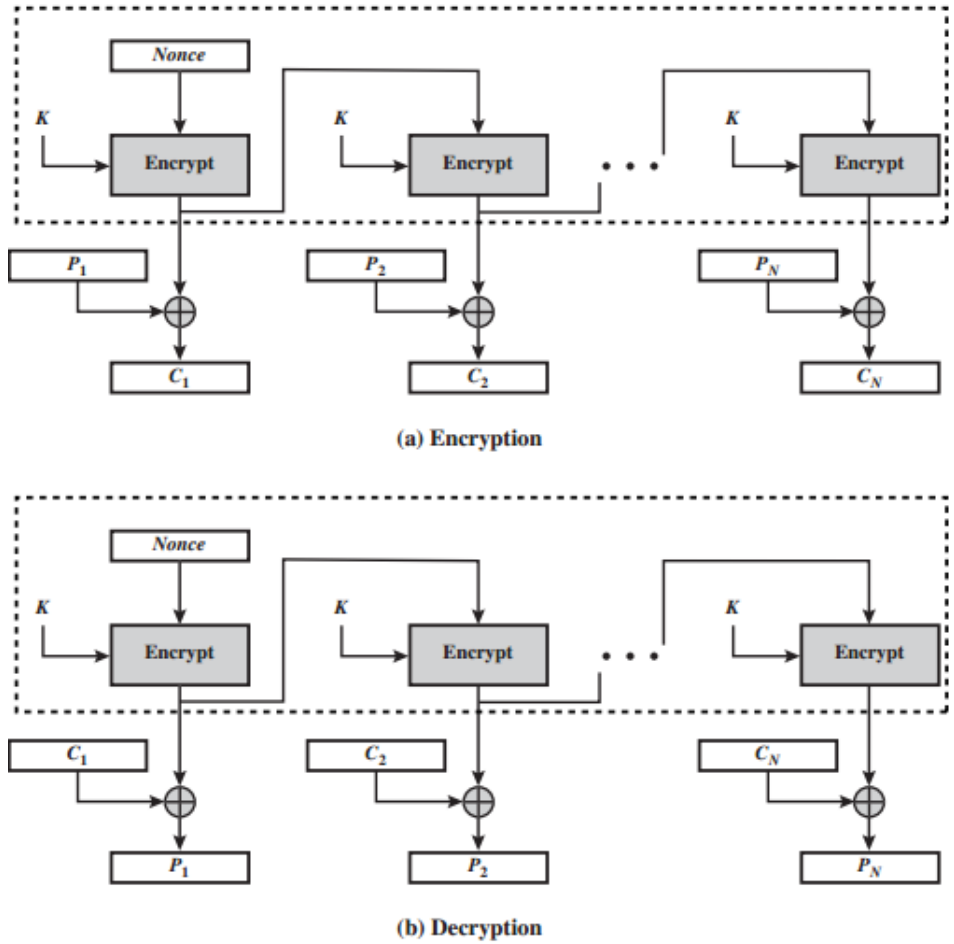
Operation modes

- 4. Output feedback (OFB) mode

> ✓ If *tx* error, little impact!
>
> ✓ Only affects current block



Output feedback mode



(a) Encryption

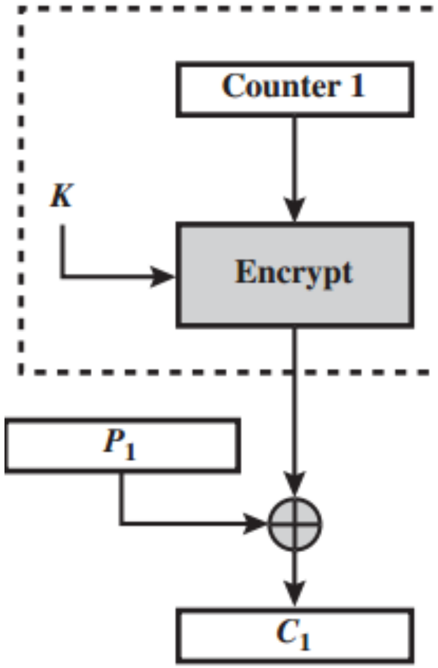
(b) Decryption

Operation modes

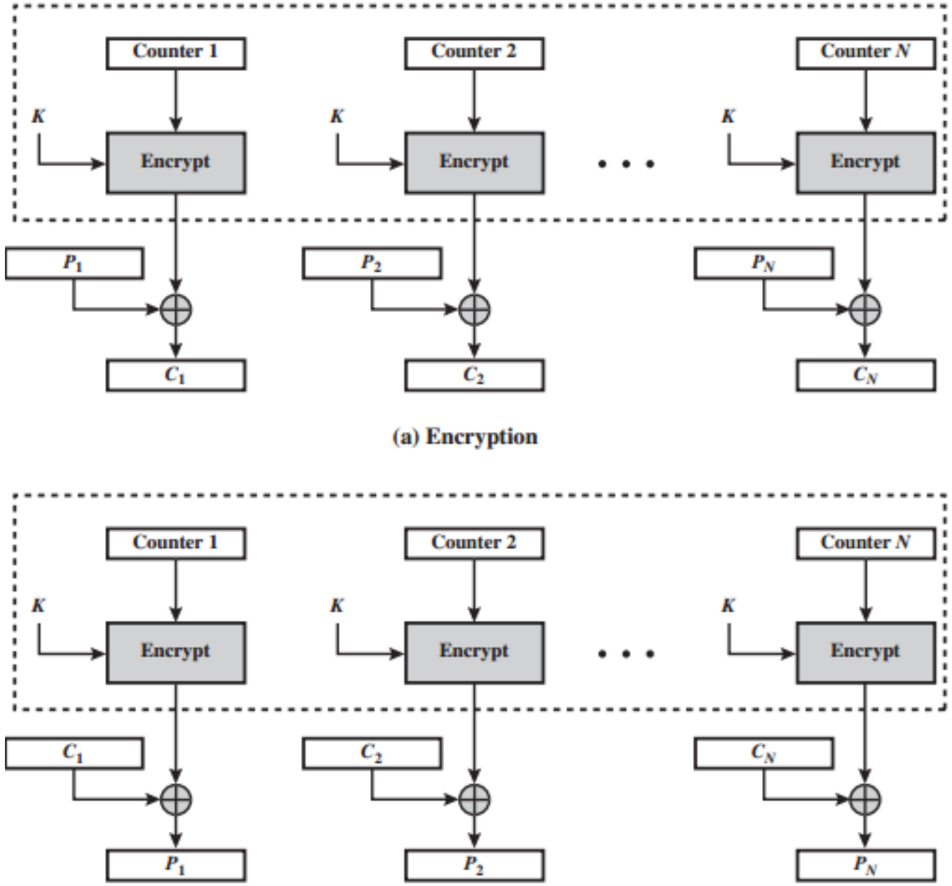
- **4. Output feedback (OFB) mode**
  - A stream cipher eliminates the need to pad a message to be an integral number of blocks
  - It also can operate in real time (a byte is encrypted and sent immediately)

Operation modes

- 5. Counter (CTR) mode



Counter mode



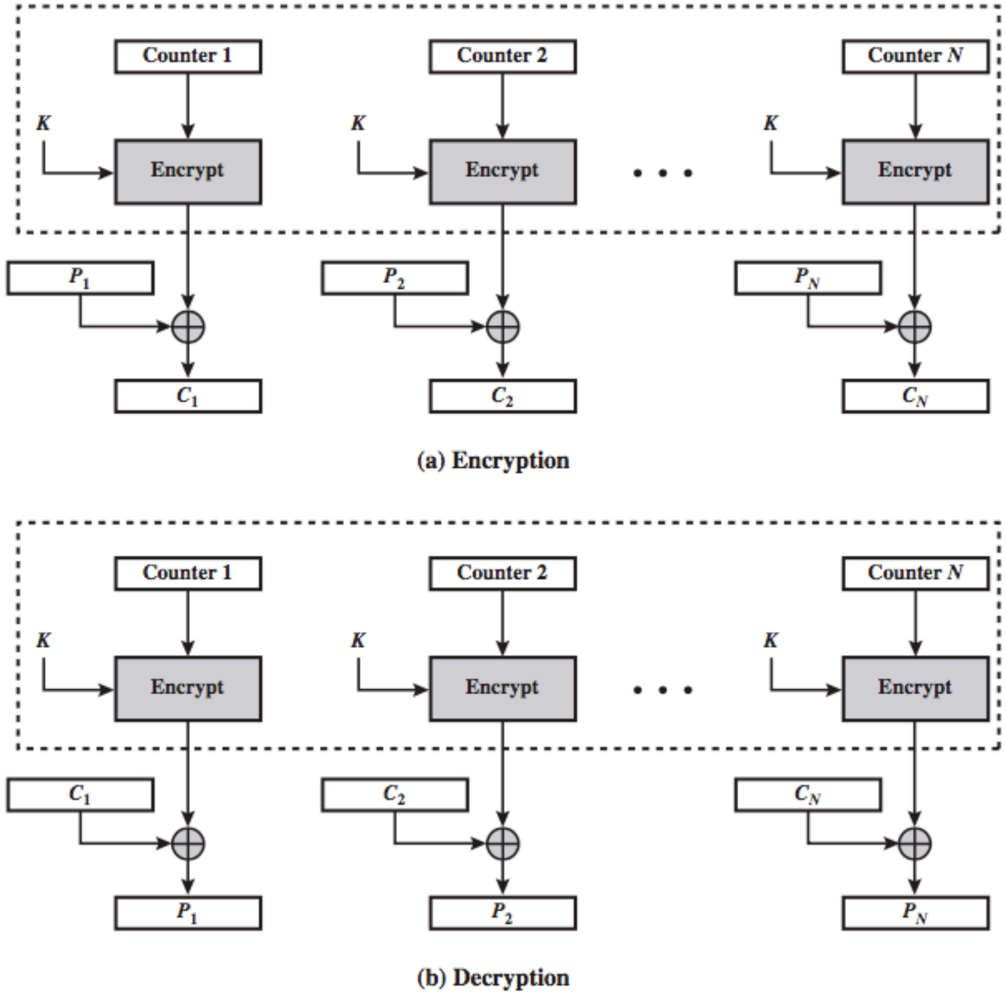
Operation modes

- **5. Counter (CTR) mode**
  - It uses a counter and there is no feedback
  - Needs a counter generation algorithm
  - Easy parallelization
  - Random access. The *th* block of plain text or cipher text can be processed in random-access fashion

Operation modes

- 5. Counter (CTR) mode

Counter (CTR) Mode



Operation modes

- **Block ciphers**
  - Electronic codebook (ECB)
  - Cipher block chaining (CBC)
- **Allow convert a block cipher into a Stream ciphers**
  - Cipher feedback (CFB) – but does not conform with typical stream cipher
  - Output feedback (OFB)
  - Counter (CTR) mode
- Stream cipher eliminates the need to **pad** a message
  - It can also operate in real time (e.g. character stream being transmitted)

Padding

- **When there are not enough bytes to fill the last block** (ex: 8 bytes)
- Consider the last block as: | `FF FF FF FF FF` | ...
- There are 3 bytes left to be filled in the block

- Possible padding (filling) modes are:

None	• no padding
Zeros	• fill with zeros

Example: |`FF FF FF FF` *FF* **00 00 00**|

It may be not reversible if the last byte of data are also zeros

Symmetric ciphers

- Properties:

Properties	Meaning
Block Size	Size of the block to encrypt
IV	Initialization vector
Key	Secret key
Mode	Block encryption mode
Padding	Last block filling mode

- This information has to be known by both parties in a secure communication

Symmetric ciphers

- Properties:
  - This information has to be known by both parties in a secure communication
  - **Secret key** and **initialization vector** are generated by the **encryption algorithm**

Must be shared with the other peer in the secure communication process

  - In case of CBC block encryption mode → IV is mandatory

Symmetric cipher

Symmetric encryption provides **authentication** among those who share the **secret key**

The length of the keys

Length of symmetric keys

- **Security of a cryptographic system**
  - The strength of the algorithm
  - Key length

Assuming algorithm is perfect…

Length of symmetric keys

- **Calculation attack complexity**
  - 8 bit key
    - Key set is 2^8
    - 256 attempts to get the right key

On half of key set: 50% chances

Length of symmetric keys

- **Calculation attack complexity**
  - 56 bit key
    - Key set is 2^{56}
    - Assuming a supercomputer 1.000.000 keys/sec
      - 2.285 years to get the right key
  - 64 bit key
    - Key set is 2^{64}
    - 585.000 years to get the right key

Length of symmetric keys

- **Calculation attack complexity**
  - 128 bit key
    - Key set is 2^{128}
    - 10^{25} years to get the right key

Note: Universe is about 10^{10} years old

Length of symmetric keys

- **A perfect algorithm (as supposed) is not easy to achieve!**
  - Cryptography is a subtle art
    - Ciphers that seem strong are weak
    - Really strong ciphers, with just a little modification become weak…
    - An **old and very tested cipher is better** than a new one claiming for enormous key set and super complex algorithm…

Length of symmetric keys

- **A perfect algorithm (as supposed) is not easy to achieve!**
  - Example:
    - Blowfish was created in the time of DES
    - Blowfish was the predecessor of twofish which was finalist with AES
    - No one knows Blowfish cryptanalysis efforts

Which one should we choose?

DES or Blowfish?

Length of symmetric keys

- **Launching a brute-force attack**
  - We need to have at least
    - A block of cipher text
    - A block of plain text
    - Algorithm name

- Brute force fits in parallel machines very well
- A subset for each processor

Length of symmetric keys

- **Launching a brute-force attack**
  - So, do not forget
    - If message value is $1.50
      - It is not worth to spend $1.000 in a machine
    - If message value is $100.000.000
      - It is worth to spend $10.000.000 in a machine

Usually, the value of information drops rapidly with time!

Length of symmetric keys

- **Breaking ciphers by software**
  - By software, efficiency drops thousands times (vs hardware)

No one seeks efficiency! They seek no charge!!

Idle computers can be used

If success great, otherwise nothing was lost

Length of symmetric keys

- **Breaking ciphers by software**
  - What about the lucky factor?
    - Several persons or groups around the planet
      - May be someone get lucky
    - May neural networks help?

Length of symmetric keys

- **Breaking ciphers by software**
  - May virus help ?
    - It is hard to convince people to join us
    - We could try breaking into their machines
    - Time-consuming and we might get arrested
    - Just work when processor is idle…
    - Micro computers are idle for 70% to 90% of the time!

POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Symmetric algorithms

.NET framework

Symmetric algorithms (.NET Framework)

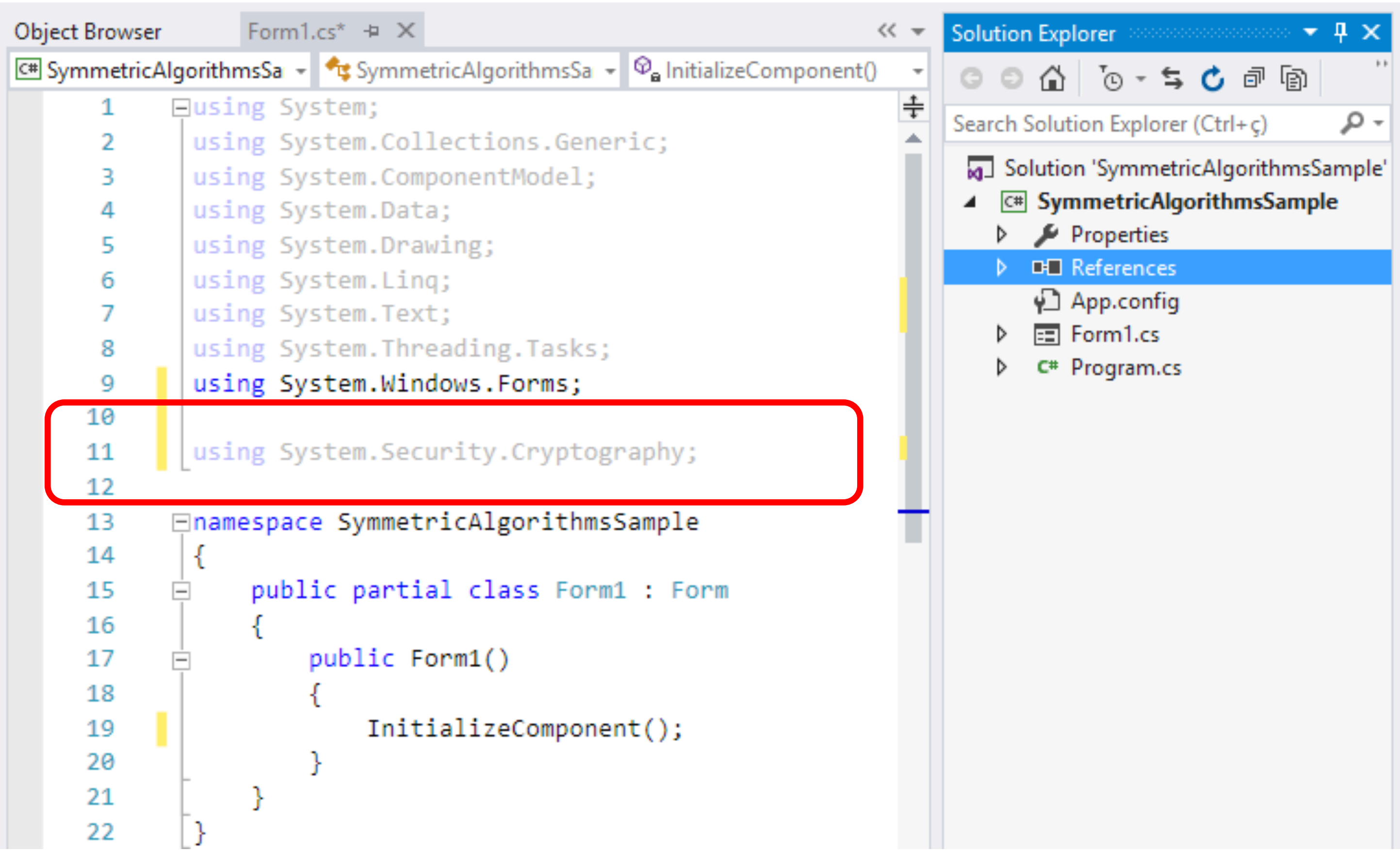
- What can be achieved
  - Information privacy
  - Users authentication
  - Information integrity
  - Non-repudiation
  - Resources access control
  - Service availability

Symmetric algorithms (.NET Framework)

- What can't be done
  - Wrong security policies (e.g. "weak passwords")
  - Trust heavily systems/users
  - Attacks based on emotional or psychological actions
  - Attacks based on direct physical threats to humans
  - Non secure software development

Import namespace

- System.Security.Cryptography



INFORMATION SECURITY

System.Security.Crytography namespace

- SymmetricAlgorithm
- AsymmetricAlgorithm
- CryptoStream
- CspParameters
- HashAlgorithm
- RandomNumberGenerator
- ToBase64Transform e FromBase64
- CryptographicException

Available classes may be different depending on the .NET Framework version being used in the project

+info: https://msdn.microsoft.com/en-us/library/system.security.cryptography(v=vs.110).aspx#Classes

Implemented algorithms

- Symmetric algorithms

- DesCryptoServiceProvider
- TripleDESCryptoServiceProvider
- AesCryptoServiceProvider
- RC2CryptoServiceProvider
- RijndaelManaged

Inheritance Hierarchy

System.Object
  System.Security.Cryptography.SymmetricAlgorithm
    System.Security.Cryptography.Aes
    System.Security.Cryptography.DES
    System.Security.Cryptography.RC2
    System.Security.Cryptography.Rijndael
    System.Security.Cryptography.TripleDES

+info: https://msdn.microsoft.com/en-us/library/system.security.cryptography.symmetricalgorithm(v=vs.110).aspx

SymmetricAlgorithm class

- Methods
  - Clear/Dispose (release reserved resources)
  - Create
  - CreateDecryptor (creates decryptor object)
  - CreateEncryptor (creates encryptor object)
  - GenerateKey (generates secret key)
  - GenerateIV (generates initialization vector)

Shared information

- Information to be shared/known by both sender and receiver (mandatory):

Parameter	Meaning
key	Secret key
IV	Initialization vector
CipherMode	Block encryption mode
PaddingMode	Last block empty space filling protocol

Secret key and IV

- Are generated by the chosen algorithm
- Must be sent to the other communication peer
- In case block **encryption mode is CBC**, the **initialization vector** must be used

- **Encryption modes:**
  - ECB (Electronic Codebook)
  - CBC (Cipher Block Chaining)
  - CFB (Cipher Feedback)
  - OFB (Output Feedback)

Filling modes

- When data is not enough to fill completely the last block (e.g. 8 bytes left)
  - Consider the following last block: |FF FF FF FF FF| …
  - There are 3 bytes missing to fill completely the block

- **Filling Modes**
  - **None** (no padding)
  - **Zeros** (fill with zeros)

  - Example: |FF FF FF FF FF 00 00 00|
  - May be not reversible if the last bytes of (user) data include zeros

Filling modes (cont.)

- **PKCS7** (fill with number of bytes left to the block end)
  - Example: |FF FF FF FF FF **03 03 03**|
  - [java: PKCS5]

- **ISO10126** (fill with random numbers, except last byte that is used to identify the number of random bytes inserted)
  - Example: |FF FF FF FF FF **AA BB 03**|
  - [== Java]

Note: If the last block has 8 bytes, it is fully filled with padding

Symmetric algorithm - example

- Initial steps

```
// Define algorithm to use
SymmetricAlgorithm sa = TripleDESCryptoServiceProvider.Create();


// Generate secret key and initialization vector (IV)
// methods called in object creation time
sa.GenerateKey();
sa.GenerateIV();

// Define operation mode
sa.Mode = CipherMode.CBC;

// Define filling (padding) mode
sa.Padding = PaddingMode.PKCS7;
```

Encrypt / Decrypt use: CryptoStream

CrytoStream class

- This class can be used to encrypt information (**write mode**) or to decrypt (**read mode**)

Constructors

	Name	Description
	CryptoStream(Stream, ICryptoTransform, CryptoStreamMode)	Initializes a new instance of the CryptoStream class with a target data stream, the transformation to use, and the mode of the stream.

- **Stream**: buffer to store information to encrypt or decrypt
- **Transform**: allows to know which function the class is going to play
- **Mode**: action to perform:
  - Write: *write = encrypt*
  - Read: *read = decrypt*

CrytoStream class

- **Properties**
  - Length (stream size in bytes)
- **Methods**

Clear	releases allocated resources
Close	closes the stream and releases allocated resources
Flush	writes data to the corresponding buffer and cleans the buffer and the stream
FlushFinalBlock	updates the corresponding buffer according to its state and afterwards cleans the buffer of that stream
Read	reads a sequence of bytes
Write	writes a sequence of bytes

CrytoStream class

- **Encryt**

```
MemoryStream ms = new MemoryStream();
CryptoStream cs =
        new CryptoStream(ms, sa.CreateEncryptor(),
                              CryptoStreamMode.Write);
cs.Write(plainbytes, 0, plainbytes.Length);
cs.Close();
```

Encrypted Information:

```
ms.ToArray()
```

CrytoStream class

- **Decrypt**

```
MemoryStream ms = new MemoryStream(cipherData);
CryptoStream cs = new CryptoStream(ms, sa.CreateDecryptor,
CryptoStreamMode.Read);

byte[] plainbytes = new byte[ms.Length];

int numPlainBytes = cs.Read(plainbytes, 0, plainbytes.Length);
cs.Close();

Array.Resize(ref plainbytes, numPlainBytes);
```

Decrypted Information:

`plainbytes`

System.Text.Encoding class

- Conversion: string <-> bytes
  - UTF8
    - GetString(bytes)
    - GetBytes(string)
  - Base64
    - Convert.ToBase64String(bytes)
    - Convert.FromBase64String(string);

System class

+info: https://msdn.microsoft.com/en-us/library/system.bitconverter(v=vs.110).aspx

- **BitConverter**
  - Converts base data types to an array of bytes, and an array of bytes to base data types

  - **ToString(Byte[])**
    - Converts an array of bytes to hexadecimal format (hexadecimal string representation)
    - Ex: “4E-61-BC-00”

```
byte[ ] arrayOne = { 0, 1, 2, 4, 8, 16, 32, 64, 128, 255 };
Console.WriteLine( BitConverter.ToString( arrayOne) );
//output
00-01-02-04-08-10-20-40-80-FF
```

CryptographicException class

- The exception that is thrown when an error occurs during a **cryptographic operation**

- Exception's may occur when:
  - Choosing some specific **encryption modes** with symmetric algorithms (e.g. Mode OFB with all algorithms)
  - Choosing some filling/padding modes with symmetric algorithms (e.g. Rijndeal with passing mode None)

Bibliography

[1] W. Stallings, Cryptography and Network Security: Principles and Practice, 5th Edition, Prentice Hall, 2010 (Chapter 6).

[2] Peter Thorsteinson, G. Gnana Arun Ganesh, .NET Security and Cryptography.

Note:
This presentation is (almost entirely) based on the book [1].
It also uses some adaptations from Prof. Nuno Costa and Vitor Fernandes from previous academic years lecturers.

Slides about security and cryptography in the .NET Framework were based on book [2]
With adaptations from Prof. Rui Ferreira previous academic years lecturers.