Laboratory

Jakub Czyszczonik

25.04.2020

1 Therory

1.1 About AES

AES (Advanced Encryption Standard) is a symmetric block cipher standardized. AES has been adopted by the U.S. government (NIST and NSA) and is now used worldwide. AES use substitution–permutation network instead of Feistel network.

1.2 General Algorithm description

AES have fixed length block size 128 bits, but key can have 128, 192 or 256 bits. Block is represented as a Matrix $4x4$ (16 bajts). Number of transformation matrix is called round. Round number depends on key length 10 for 128 bit, 12 for 192 bit and 14 for 256 bit. For each round is generated round key and key is combined with matrix by bitwise xor operation. Next matrix is transformed by an 8-bit substitution box. Next rows is specially shifted and after that column is mixed.

1.3 Initialization Vector

Initialization Vector (IV) - a string of bits of a fixed length as additional input to cryptographic algorithms. Is required to be random or pseudorandom. Randomization is crucial for encryption schemes to achieve semantic security. If the IV is a nonce, that means it is a number used once.

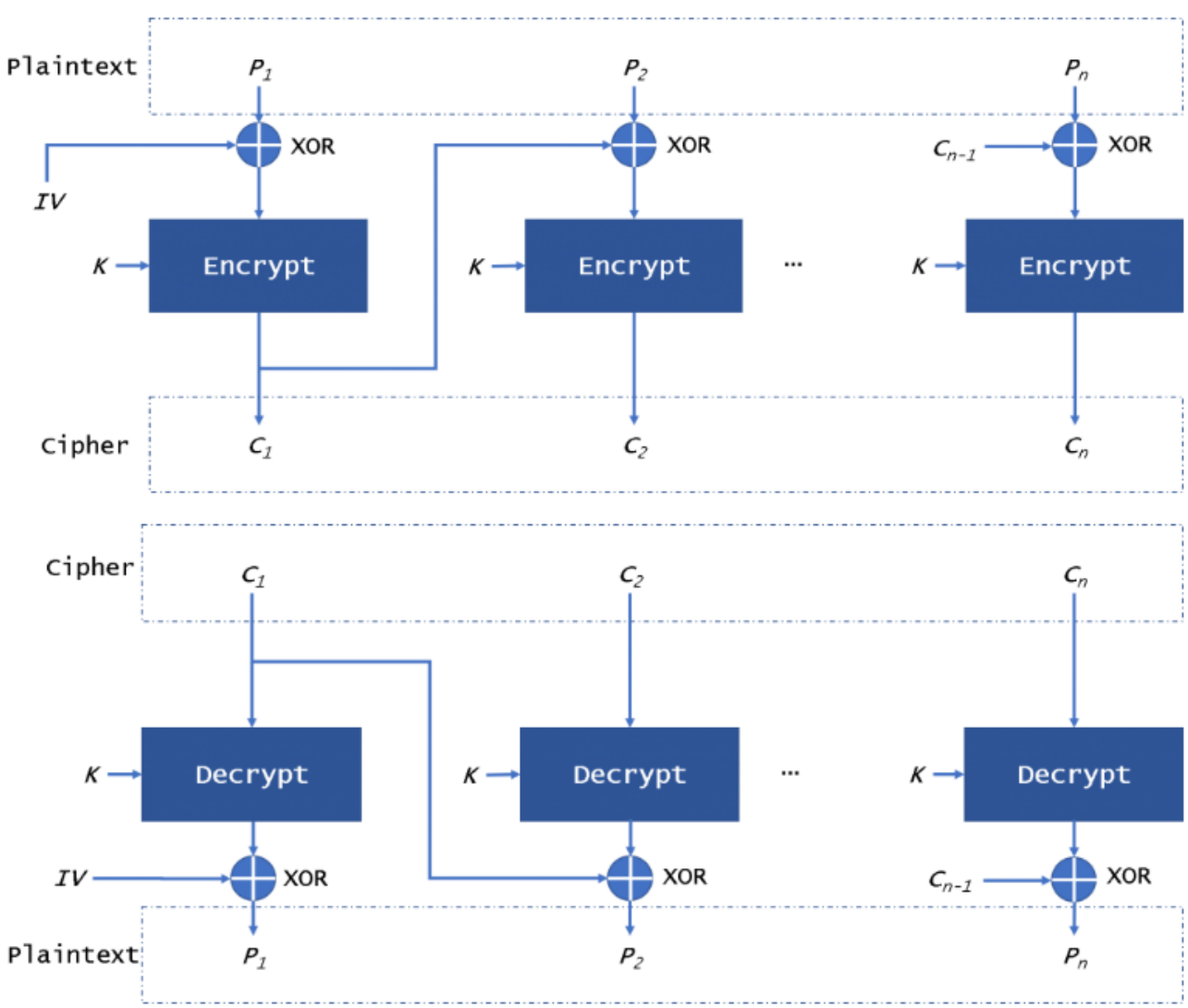
1.4 AES vs DES

	AES	DES
Develop year	2001	1977
Principle	substitution–permutation network	Feistel network
Plaintext size	128, 192, 256 bits	64 bits
Key size	128, 192, 256 bits	64 bits
Ciphertext size	128, 192, 256 bits	56 bits
Rounds	128 bits = 10, 192 bits = 12, 256 bits = 14	fixed number of rounds i.e 16
Security	is much more secure and it widely used.	is less secure and hardly used now
Speed	is much faster than DES.	is comparatively slower than AES

1.5 Encryption modes

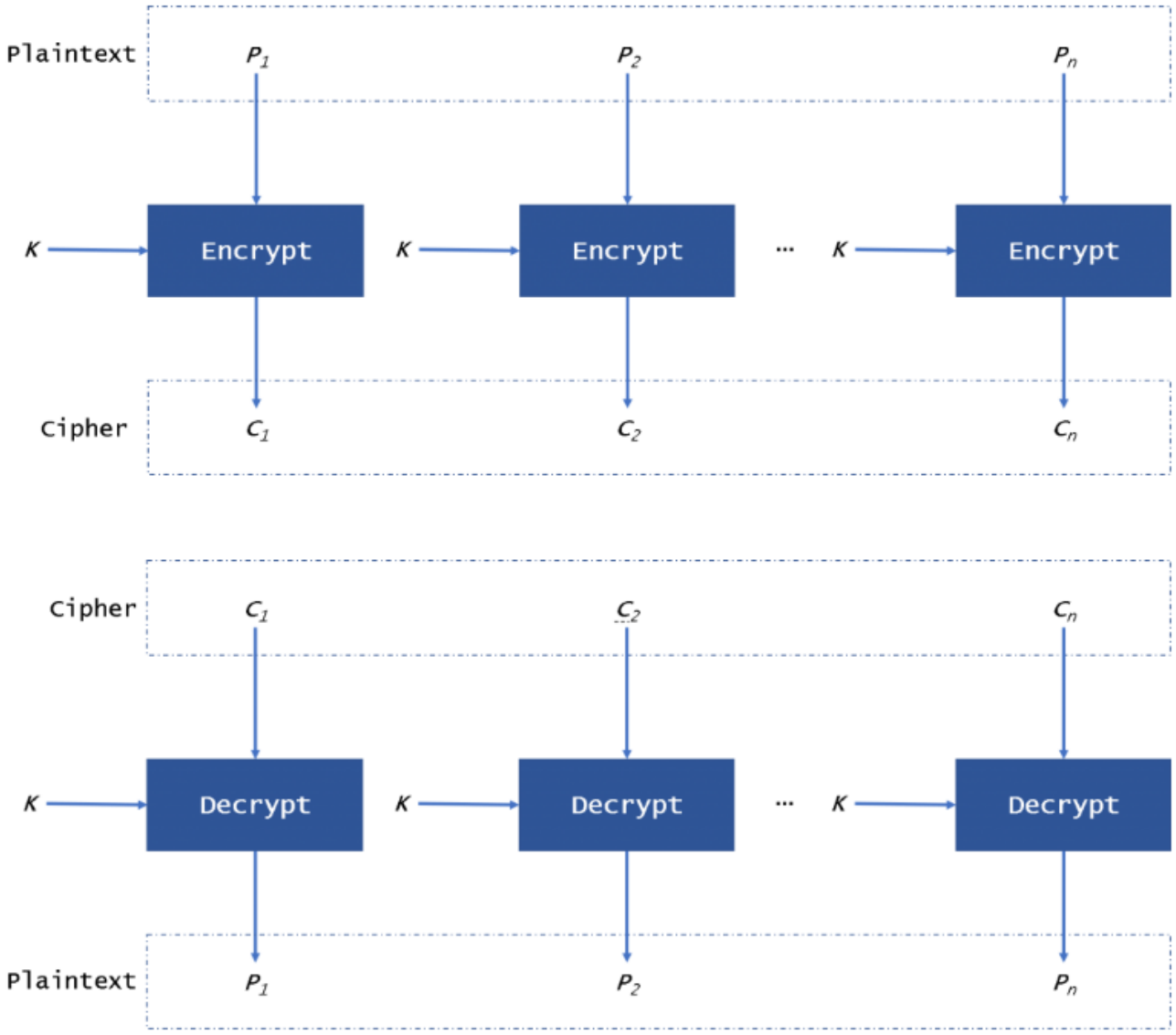
CBC (Cipher Block Chaining): An IV-based encryption scheme, the mode is secure as a probabilistic encryption scheme, achieving indistinguishability from random bits, assuming a random IV. Encryption inefficient from being inherently serial. Widely used, the mode's privacy-only security properties result in frequent misuse. Requires the plaintext to be padded to a multiple of 16 bytes.

The plaintext is divided into blocks and needs to add padding data. First, we will use the plaintext block xor with the IV. Then CBC will encrypt the result to the ciphertext block. In the next block, we will use the encryption result to xor with plaintext block until the last block. In this mode, even if we encrypt the same plaintext block, we will get a different ciphertext block. We can decrypt the data in parallel, but it is not possible when encrypting data.



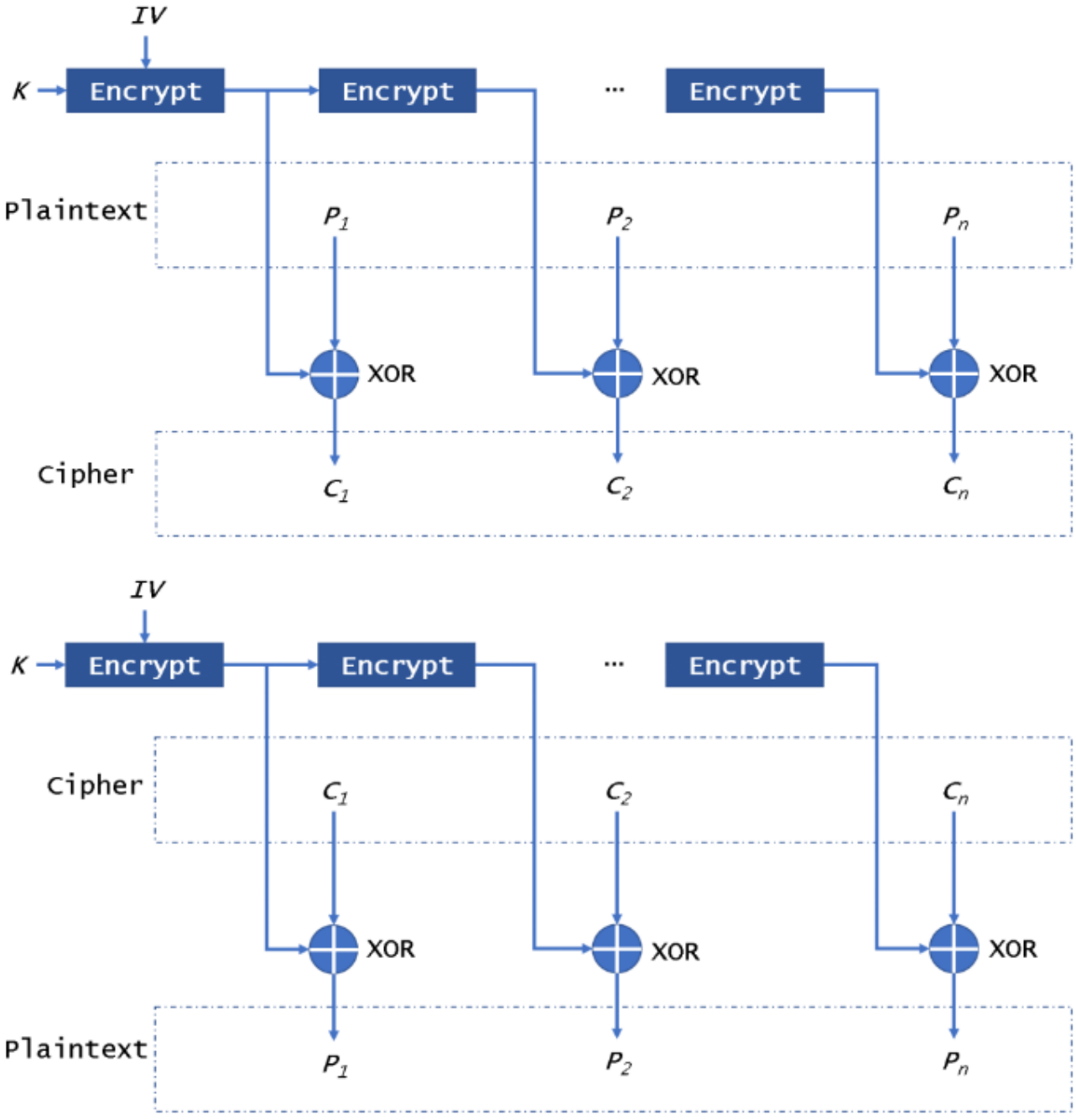
Rysunek 1: CBC

ECB: A blockcipher, the mode enciphers messages that are a multiple of n bits by separately enciphering each n-bit piece. The security properties are weak, the method leaking equality of blocks across both block positions and time. It does not use any kind of IV, nonce or salt.



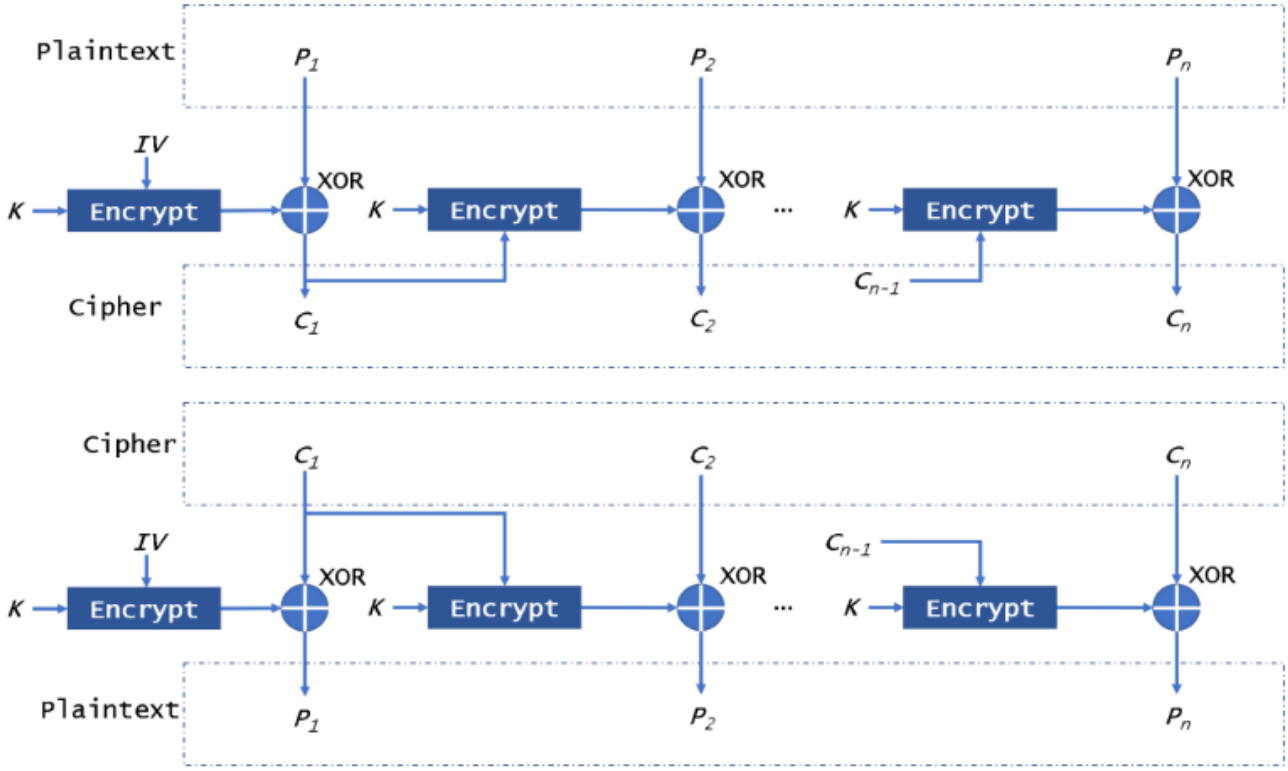
Rysunek 2: ECB

OFB (Output FeedBack): An IV-based encryption scheme, the mode is secure as a probabilistic encryption scheme, achieving indistinguishability from random bits, assuming a random IV. Can be used also as a stream encryptor. It also doesn't need padding data. An IV is encrypted, then it is XOR-ed with a block of plaintext to produce a block of ciphertext. The encrypted IV is passed as an IV for the next block in the chain.



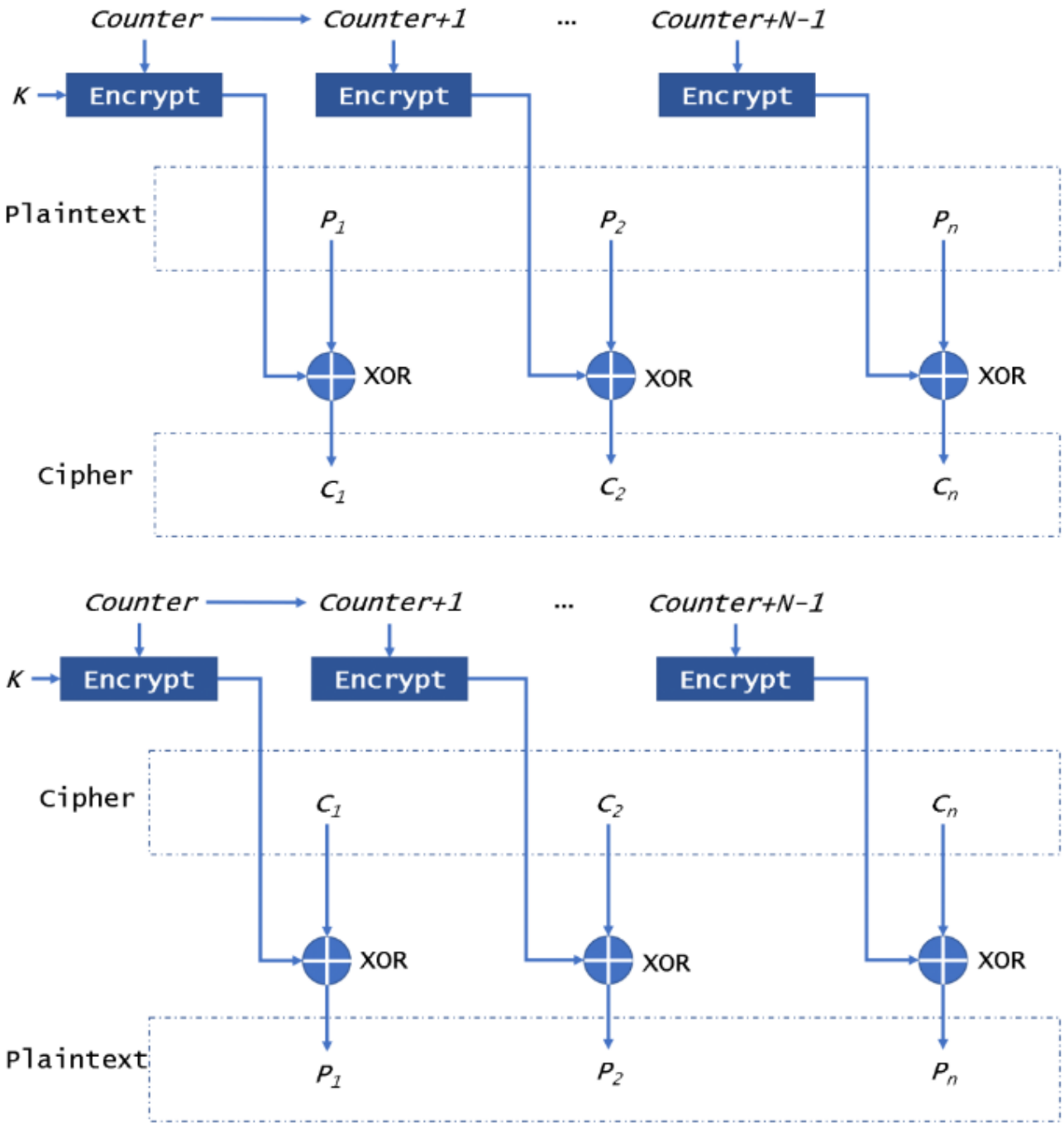
Rysunek 3: OFB

CFB (Cipher FeedBack): An IV-based encryption scheme, the mode is secure as a probabilistic encryption scheme, achieving indistinguishability from random bits, assuming a random IV. An IV is encrypted, then it is XOR-ed with a block of plaintext to produce a block of ciphertext. Obtained block of ciphertext is passed as an IV for the next block in the chain.



Rysunek 4: CFB

CTR (Counter): An IV-based encryption scheme, the mode achieves indistinguishability from random bits assuming a nonce. It also is a stream encryptor. As an input block to the encryptor (Encrypt) is given the value of a counter. The counter is created using a vector. The counter has the same size as the used block. The XOR operation with the block of plain text is performed on the output block from the encryptor. All encryption blocks use the same encryption key. CTR will use the counter to be encrypted every time instead of the IV. So if you could get counter directly, you can encrypt/decrypt data in parallel.



Rysunek 5: CRT

EAX A Encrypt-then-Authenticate-then-Translate mode in this mode we achieves the privacy and authenticity of each block. Ciphertext block contains a message authentication code (MAC). Uses nonce.

2 Summary of Encryption modes

2.1 ECB

PROS:

- Simple
- Fast
- Encryption and Decryption can by computed parallel.

CONS:

- Plaintext can be modified by modification ciphertext
- Not resistant to replay attack.
- Doesn't hide data patterns.(Two this same blocks have this same cipthertext)

2.2 CBC

PROS:

- Hide data patterns. (Two this same blocks have this same cipthertext).
- Decryption can by computed parallel.

CONS:

- Encryption doesn't support parallel computation.
- Slow

2.3 CFB

PROS:

- No Padding
- Decryption can by computed parallel.

CONS:

- Encryption doesn't support parallel computation.
- Not resistant to replay attack.

2.4 OFB

PROS:

- No Padding
- Decryption and encryption is performed by this same algorithm.
- Broken block affect only on the current block

CONS:

- Doesn't support parallel computation.

2.5 CRT

PROS:

- No Padding
- Decryption and encryption is performed by this same algorithm.
- Broken block affect only on the current block.
- Supports parallel computation.

2.6 EAX

PROS:

- Message expansion is minimal, being limited to the overhead of the tag length.
- Using CTR mode means the cipher need be implemented only for encryption, in simplifying implementation of some ciphers (especially desirable attribute for hardware implementation).
- The algorithm is on-line, that means that can process a stream of data, using constant memory, without knowing total data length in advance.

CONS:

- Is slower than a well designed one-pass scheme based on the same primitives.

3 The chosen plaintext attack

Assumption: IV is week, so Adversary $\mathcal{A}$ can predict next value of IV. Algorithm:

1. Oracle are initialised.
2. Adversary $\mathcal{A}$ pick random message m of size 16 bytes (which is equal to AES block size).
3. $\mathcal{A}$ ask $\mathcal{O}$ for m encryption. $\mathcal{A}$ obtains cipthertext c and IV iv_0.
4. $\mathcal{A}$ predicts the next IV to be iv_1 and using that compute message $m_f = (iv_1 \oplus (iv_0 \oplus m))$.
5. $\mathcal{A}$ pick random message m_r s.t $m_r \neq m_f$.
6. $\mathcal{A}$ send tuple (m_f, m_r) to $\mathcal{O}$ and obtains c'.

7. $\mathcal{A}$ guessing if $c' = c$ outputs 0, otherwise outputs 1.

Steps 4-7 can be repeated as many times with success probability equal to 1! $\mathcal{A}$ will play game. In this attack very important is predictable IV. Message, cipthertext and IV obtained in steps 2-3 can be used many times. In 4th step adversary predicts next iv and compute message s.t $m_f = (iv_1 \oplus (iv_0 \oplus m))$. Looking inside AES black box we will see in first step (XOR message and IV) that: $m_f \oplus iv_1 = ((iv_0 \oplus m) \oplus iv_1) \oplus iv_1 = (iv_0 \oplus m) \oplus iv_1 \oplus iv_1 = iv_0 \oplus m$. So we obtain this same message and IV used in first encryption in oracle. Encryption key don't changes so result cipther text will be equals to c. Otherwise the another random picked message was encrypted in challenge. $\mathcal{A}$ can guess every challenge with probability equal to 1.

4 Run Description

4.1 Python

I use Python 3.6.9 and libraries: Crypto, jks, unittest

4.2 Keystore

You can generate your own keystore by command:
keytool -genseckey -keystore name.jks -storetype jceks -storepass password1 -keyalg AES -keysize 256 -alias alias -keypass password2
You can also use default keystore.

4.3 Run code

Arguments for running Exercise1 you will see after running:
python3 Exercise1.py -h
Exercise supports string messages and files also. When you don't pass good args, program will run with default configuration (message in console). Exercise2 (CPA) Will be run :
python3 Exercise2.py
Test will be run:
python3 EncryptionTests.py

Example usage:
python3 Exercise1.py –challenge file file0.py file1.py
python3 Exercise1.py message example python3 Exercise1.py –mode CBC –path keystores/default.jks –password default –alias default message example
Hint:
(–) = 2 x -