

# Image Encryption and Decryption Tool

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Prerequisites</b>	<b>2</b>
<b>3</b>	<b>Installation</b>	<b>2</b>
<b>4</b>	<b>Running the Script</b>	<b>2</b>
4.1	Step 1: Create a Virtual Environment . . . . .	2
4.2	Step 2: Activate the Virtual Environment . . . . .	2
4.3	Step 3: Install Required Libraries . . . . .	2
4.4	Step 4: Run the Script . . . . .	2
<b>5</b>	<b>Features</b>	<b>3</b>
<b>6</b>	<b>Example Workflow</b>	<b>3</b>
6.1	Step-by-Step Example . . . . .	3
<b>7</b>	<b>Visual Examples</b>	<b>4</b>
7.1	Comparison of ECB and CBC Modes . . . . .	4
7.2	Decryption Examples . . . . .	5
<b>8</b>	<b>About the Code</b>	<b>6</b>
8.1	Technical Details . . . . .	6
<b>9</b>	<b>Output Files</b>	<b>6</b>

# 1 Introduction

This document provides a detailed guide for using the Image Encryption and Decryption Tool. The tool allows users to encrypt and decrypt grayscale images using AES encryption in ECB and CBC modes. It supports key generation, encryption, and decryption with automatic file naming based on the encryption method and key size.

## 2 Prerequisites

Before running the script, ensure the following Python libraries are installed:

- `opencv-python`
- `numpy`
- `pycryptodome`

## 3 Installation

To install the required libraries, use the following command:

```
pip install opencv-python numpy pycryptodome
```

## 4 Running the Script

If you encounter issues running the script directly, follow these steps to set up a virtual environment:

### 4.1 Step 1: Create a Virtual Environment

Run the following command to create a virtual environment:

```
python -m venv myenv
```

### 4.2 Step 2: Activate the Virtual Environment

- On Linux/Mac:

```
source myenv/bin/activate
```

- On Windows:

```
myenv\Scripts\activate
```

### 4.3 Step 3: Install Required Libraries

Inside the virtual environment, install the required libraries:

```
pip install opencv-python numpy pycryptodome
```

### 4.4 Step 4: Run the Script

Run the script using the following command:

```
python image_cipher.py
```

## 5 Features

- **Key Generation:** Generate 128, 192, or 256-bit keys for ECB and CBC modes. Keys are saved as .bin files.
- **Image Encryption:** Encrypt images using AES in ECB or CBC mode. The output file is automatically named based on the encryption method and key size.
- **Image Decryption:** Decrypt images using the corresponding key and IV (for CBC). The output file is automatically named based on the decryption method and key size.

## 6 Example Workflow

### 6.1 Step-by-Step Example

#### 1. Generate a 128-bit CBC key:

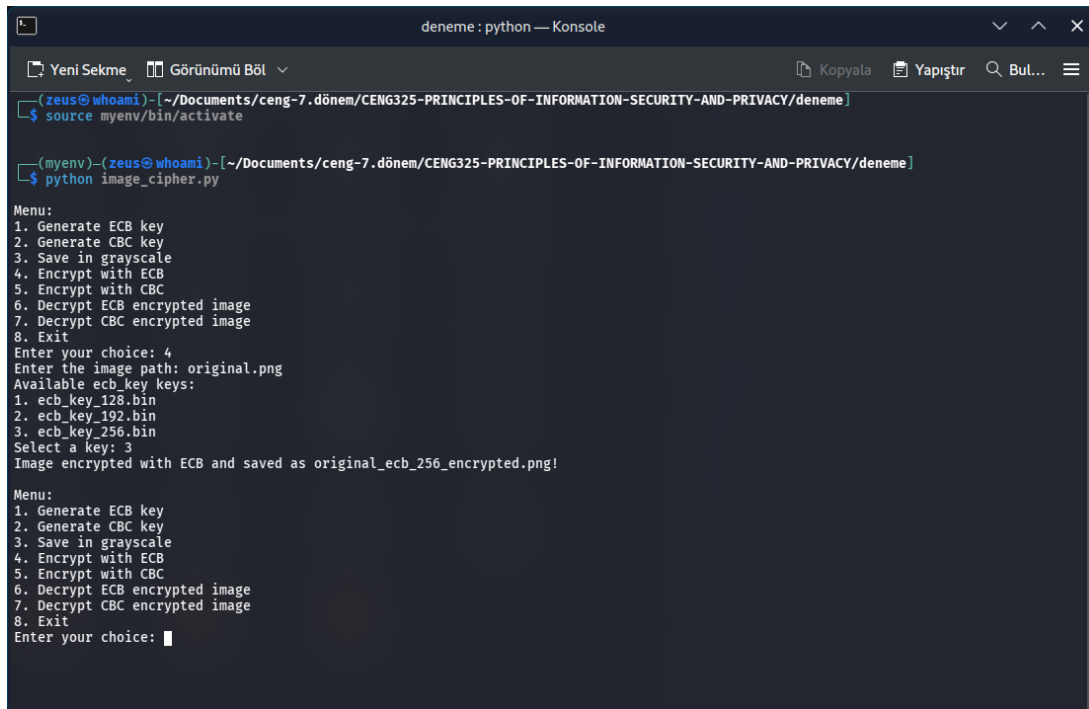
- The key will be saved as `cbc_key_128.bin`.
- The IV will be saved as `cbc_iv_128.bin`.

#### 2. Encrypt an image (`image.png`) with the CBC key:

- The output file will be named `image_cbc_128_encrypted.png`.

#### 3. Decrypt the encrypted image:

- The output file will be named `image_cbc_128_encrypted_cbc_128_decrypted.png`.



```
deneme : python — Konsole
Yeni Sekme Görünümü Böl
Kopyala Yapıştır Bul...
(zeus@whoami)-[~/Documents/ceng-7.dönem/CENG325-PRINCIPLES-OF-INFORMATION-SECURITY-AND-PRIVACY/deneme]
$ source myenv/bin/activate
(myenv)-(zeus@whoami)-[~/Documents/ceng-7.dönem/CENG325-PRINCIPLES-OF-INFORMATION-SECURITY-AND-PRIVACY/deneme]
$ python image_cipher.py

Menu:
1. Generate ECB key
2. Generate CBC key
3. Save in grayscale
4. Encrypt with ECB
5. Encrypt with CBC
6. Decrypt ECB encrypted image
7. Decrypt CBC encrypted image
8. Exit
Enter your choice: 4
Enter the image path: original.png
Available ecb_key keys:
1. ecb_key_128.bin
2. ecb_key_192.bin
3. ecb_key_256.bin
Select a key: 3
Image encrypted with ECB and saved as original_ecb_256_encrypted.png!

Menu:
1. Generate ECB key
2. Generate CBC key
3. Save in grayscale
4. Encrypt with ECB
5. Encrypt with CBC
6. Decrypt ECB encrypted image
7. Decrypt CBC encrypted image
8. Exit
Enter your choice: █
```

Figure 1: Terminal Interface When Running the Program

## 7 Visual Examples

### 7.1 Comparison of ECB and CBC Modes

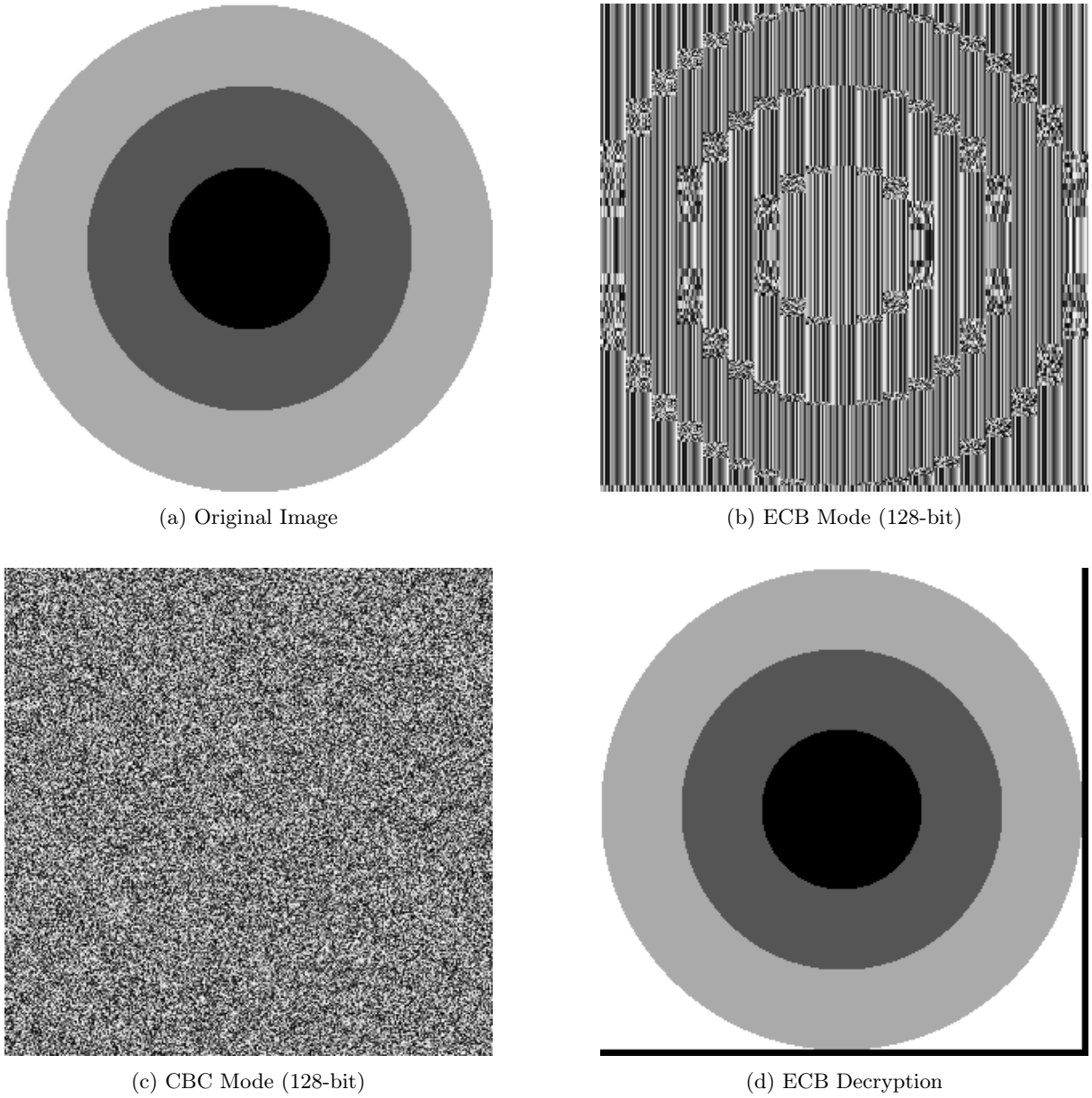
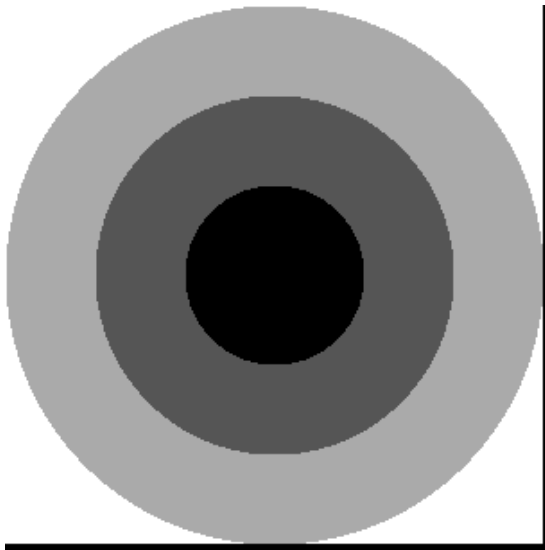


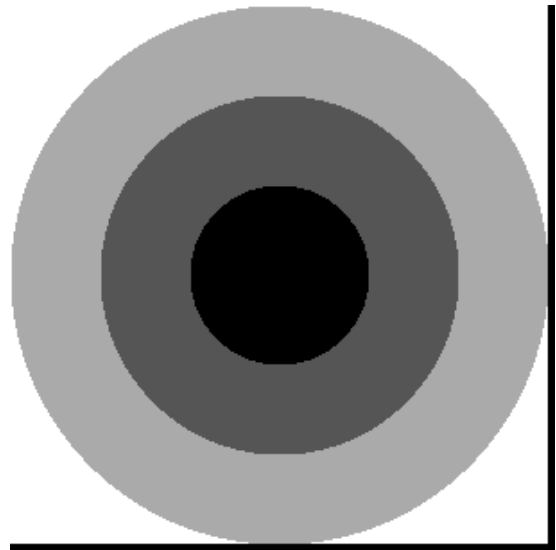
Figure 2: Encryption Modes and Results

**Analysis:** In ECB mode, the structure of the original image can be partially predicted, while this is not possible in CBC mode.

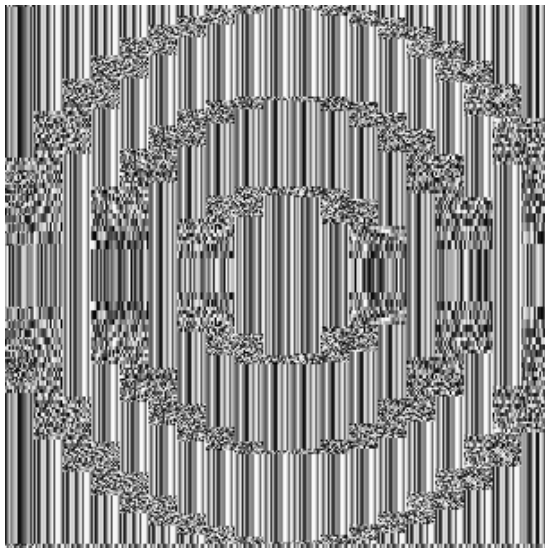
## 7.2 Decryption Examples



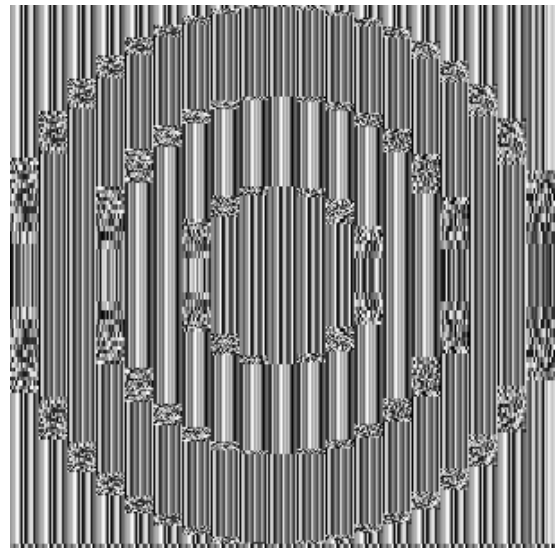
(a) Decryption with Correct ECB Key



(b) Decryption with Correct CBC Key



(c) Decryption with Wrong Mode (ECB→CBC)



(d) Decryption with Wrong Bit Length

Figure 3: Decryption Results

### Note:

- Decryption failed because a different encryption mode was used.
- Decryption failed because a key with different bit length was used.
- Decryption would fail if the key type and bit length were changed.

## 8 About the Code

This script uses the AES encryption algorithm from the `pycryptodome` library. It supports both ECB and CBC modes, with proper padding for images that are not a multiple of 16 pixels in width or height. The script ensures that file names are descriptive and include details about the encryption/decryption process to avoid confusion.

### 8.1 Technical Details

- **AES (Advanced Encryption Standard):** A symmetric block cipher algorithm. It uses keys of 128, 192, or 256 bits in length.
- **ECB (Electronic Codebook) Mode:** Encrypts each block independently. Identical blocks of content are encrypted identically, which may preserve structure in patterned images.
- **CBC (Cipher Block Chaining) Mode:** XORs each block with the previous ciphertext block before encryption. This ensures that identical blocks of content are encrypted differently.
- **IV (Initialization Vector):** A random value used in CBC mode for encrypting the first block.

## 9 Output Files

- **Key Files:**
  - `ecb_key_[bit_size].bin` - ECB keys
  - `cbc_key_[bit_size].bin` - CBC keys
  - `cbc_iv_[bit_size].bin` - CBC initialization vectors
- **Encrypted Files:**
  - `[image_name]_ecb_[bit_size]_encrypted.png` - Images encrypted with ECB
  - `[image_name]_cbc_[bit_size]_encrypted.png` - Images encrypted with CBC
- **Decrypted Files:**
  - `[encrypted_image_name]_ecb_[bit_size]_decrypted.png` - Images decrypted with ECB
  - `[encrypted_image_name]_cbc_[bit_size]_decrypted.png` - Images decrypted with CBC

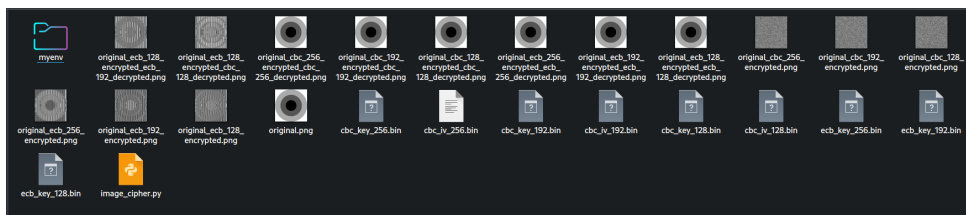


Figure 4: Example Output Files