Lab - Symmetric Cryptography

In this guide we will develop programs that use cryptographic methods, relying in the Python3 Cryptography module. The module can be installed using the typical package management methods (e.g, `apt install python3-cryptography`), or using the `pip3` tool (e.g. `pip3 install cryptography`).

It will be useful to visualize and edit files in binary format. For that purpose, if you are using Linux, you may install `GHex` or `hexedit` from the repositories.

We will be exploring the low level interface of the `python cryptography` library, for educational purposes. If you plan to use this library in real world application, stay with the Fernet interface.

As the documentation clearly states:

```
This is a “Hazardous Materials” module. You should ONLY use it if you’re 100% absolutely sure that you know what you’re doing because this module is full of land mines, dragons, and dinosaurs with laser guns.
```

Symmetric Cryptography

Symmetric cryptography is used by creating an object that represents a given cipher, with some parameters specifying the mode, as well as a key. The cipher object presents an `encryptor` method, that is applied (`update`) to the text in chunks (may require alignment with the cipher block size). After the text is ciphered, a `finalize` method may be used. Decryption is done in a similar way.

For more information please check the documentation available at the Cryptography.io Hazmat section.

File encryption

Create a function to encrypt the contents of a file, whose name should be provided by the user. The key should be random, or provided by the user. The user must provide (as program parameters or by request or any other suitable method): (i) the name of the file to encrypt, (ii) the name of the file to store the cryptogram, (iii) the name of the encryption algorithm. Check the documentation and implement functions using multiple ciphers. We recommend `AES` and `ChaCha20`.

If you need to save multiple fields to the same file (e.g, salt, cryptogram), save these as fixed length fields to the beginning of the file. A simpler alternative is to use Base64 to convert the objects to text (`base64` module), and then use a delimited such as `\n`.

Note 1: Take in consideration that data may need to be encrypted in blocks, and the last block may require padding. Please see the `PKCS#7` padding method in the documentation.

Questions:

- What is the output of some encrypted data?
- Can you determine the structure of the text?
- What are the lengths of the text and the cryptogram?
- What is the impact of using different keys sizes (e.g. 16 vs 32 bytes)?

File decryption

Alter your program by adding a function that decrypts a user file. For this functionality, the user must provide the following (as program parameters or by request or any other suitable method): (i) the name of the file to decrypt, (ii) the name of the file to store the decryption result. The key must be requested.

Take care of removing the padding (if present!)

Questions:

- What is the impact of padding?
- Is padding visible in the decrypted text?
- What happens if the cryptogram is truncated?
- What happens if the cryptogram lacks some bytes in the beginning?

Cipher modes

Initialization Vector

Some cipher modes use feedback to add more complexity to the cryptogram, namely `CFB`, `OFB` and `CFB`. Feedback implies the use of an Initialization Vector (`IV`), which must be provided when initializing an object for one of such cipher modes. Note that the IV used to encrypt some data must also be provided when decrypting it, therefore, the IV is usually sent in clear text.

Alter your program so the user, when requesting an encryption operation, also indicates the cipher mode to be used. This should be applied both to encryption and decryption.

Note that the `IV` is only used on cipher modes with feedback, which is not the case of ECB. Your program must be able to handle encryption using cipher modes both with and without feedback.

Hint: Use the `secrets` module to obtain securely random IVs.

Questions:

- What length should the `IV` be?
- For each cipher mode, what is the impact of repeating the IV, while changing the Key?
- For each cipher mode, what is the impact of repeating the Key, while changing the IV?

Patterns

In this exercise we will analyze the impact of `ECB` and `CBC` cipher modes in the reproduction of patterns in the original document into the encrypted document. For this, we are going to encrypt an image in the bit map (BMP) file format, after which we are going to visualize the contents of the obtained encrypted file and compare it with the original image. In order we can visualize the contents of the encrypted file we must replace the first 54 bytes with the first 54 bytes from the original file (these bytes constitute the header of BMP formatted files, which is necessary so the file can be recognized as a BMP formatted file).

Use the program you developed in the previous sections to encrypt the file `p_ori.bmp` using the `ECB` cipher mode and a cryptographic algorithm of your choice. Using the `dd` application copy the copy the first 54 bytes from the original image file into the first 54 bytes of the encrypted file:

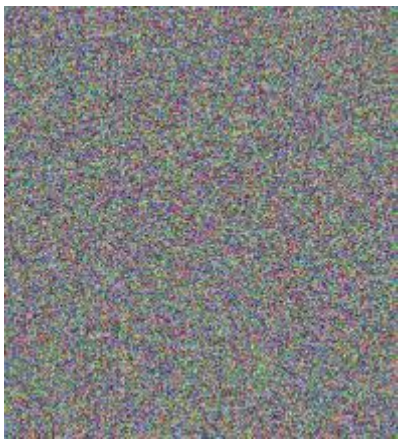
```
$ dd if=p_ori.bmp of=p_enc.bmp ibs=1 count=54 conv=notrunc
```

Using a program to visualize images, open the original image and the encrypted image and compare them. What do you observe?

Repeat all the above operations, now using the `CBC` cipher mode instead of `ECB` cipher mode, and using the same algorithm. Then, compare the original image with the obtained encrypted image. What do you observe?

Repeat the experience, using the same cipher modes, but varying the algorithm.

A typical image encrypted with feedback or without (mouse over) feedback will be:



(Images by Larry Ewing)

Questions:

- What do you conclude from the experiment?
- Can we have an insecure AES cryptogram?

Cryptogram corruption

In this exercise we are going to analyze the impact in a decrypted text caused by errors in the cryptogram, when using `ECB`, `CBC`, `OFB` and `CFB` cipher modes.

Using the program developed in previous sections, encrypt the `p_ori.bmp` file using the `ECB` cipher mode and an algorithm of your choice. Using a binary file editor, change the value of a single bit in some byte of the encrypted image (notice that the first 54 bytes are not part of the image but rather part of the header of the file), for example the byte in position `0x60`.

Decrypt the file with the corrupted bit, using the same cipher mode and algorithm you used to encrypt. Using an binary file editor, open the original file and the decrypted file and compare them. What are your conclusions regarding the impact in the decrypted file produced by the corruption of a single bit in the cryptogram?

Repeat the experience for the remaining cipher modes and, for each of them, analyze the impact on the decrypted image produced by an error in a single bit of the cryptogram. Try to determine which are the cipher modes that produce a bigger impact and those that produce a smaller impact in the decrypted file.

Padding

A block cipher requires input blocks of a fixed size that equals the algorithm block size. However, its improbable that a file to encrypt as a number of bytes that is multiple of the block size of the algorithm to be used, i.e., frequently, the number of bytes that remain for the last block do not equals the block size. To solve this problem, extra bytes are added to have a block with the correct size. These extra bytes are then removed when in the decryption operation.

There are several standards for padding. `PKCS#7` is one of them. The objective for this exercise is that you demonstrate the existence of padding in a encryption operation, and that it is according the `PKCS#7` standard.

Using a binary file editor and the program you developed, idealize an experiment involving encryption and decryption operations with `ECB` cipher mode, `PKCS#5` padding and an algorithm of your choice, that shows the presence of padding and how `PKCS#7` padding is made.

Hint: Do not use padding when decrypting a cryptogram with padding

Questions:

- Is padding required for all ciphers modes?
- What is the impact of padding when selecting the cryptographic primitive?

2021

PREVIOUS
Lab - XSS and CORS

NEXT
Lab - Asymmetric Cryptography

Last updated on 1 Oct 2021

Related

- Assignment 1 - Vulnerable App
- Assignment 2 - Application Exploitation
- Assignment 3 - Binary Vulnerabilities
- Lab - Asymmetric Cryptography
- Lab - Hash Functions