# CMPG 215 Documentation

# Group 55

Inge Senekal – 41218787

Zohan Cargotis – 42489733

Ethan Sutherland – 38410737

Danie Niebuhr – 37955039

Reuben Thorn – 41767500

# Why we chose Fernet encryption.

Fernet is a cryptographic tool that provides a simple and secure way to encrypt and authenticate data. It is designed to be easy to use, even for developers who are not experts in cryptography. Fernet guarantees that a message encrypted using it cannot be manipulated or read without the key. Fernet is an implementation of symmetric (also known as “secret key”) authenticated cryptography.

# Advantages and disadvantages of Fernet encryption.

## Advantages

* **Security:** Fernet encryption is a method of securing messages that uses very strong and hard-to-crack codes. It uses two types of codes - one to keep the message secret, and another to make sure nobody has tampered with the message. This makes it hard for anyone to read or change the message unless they have the right code to unlock it.
* **Simplicity:** Fernet is a way to make messages secure that's made for people who use Python. It's easy to use and doesn't need someone to be an expert in secret codes. So, even more people can use it to protect their messages.
* **Efficiency:** Fernet encryption uses a type of secret code that is faster and easier for computers to use than another type of secret code. This makes it a good choice for computer programs that need to work fast.
* **Versatility:** Fernet encryption can help protect lots of things on the internet, like passwords and secret information, as well as making sure things stay safe when they're being sent from one place to another. It works with different kinds of computers and systems, so it's easy to use it to keep things safe no matter what kind of computer someone is using.
* **Integration:** Fernet encryption is already part of some tools that people who use Python to make websites like to use, like Flask and Django. This means that if someone wants to use it to protect their website, they don't have to start from scratch. They can also use it with other tools to make sure that things stay safe when they're being sent from one place to another.

## Disadvantages

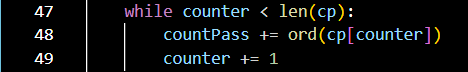
* **Key management:** Fernet encryption uses a type of secret code where the same code is used to hide and show the message. But this can make it harder to keep the code secret and make sure everyone who needs it can get it safely.
* **Lack of forward secrecy:** Fernet encryption can't make sure that if someone gets the secret code, they won't be able to read old or new secret messages that were made with that code.
* **Limited scalability:** Fernet encryption might not work well for big computer programs that need to change the secret code a lot, or that have lots of secret messages to protect.
* **Compatibility issues:** Fernet encryption is made to work with Python, so it might not work with other kinds of computer programs or systems.

# Our own algorithm EFN.

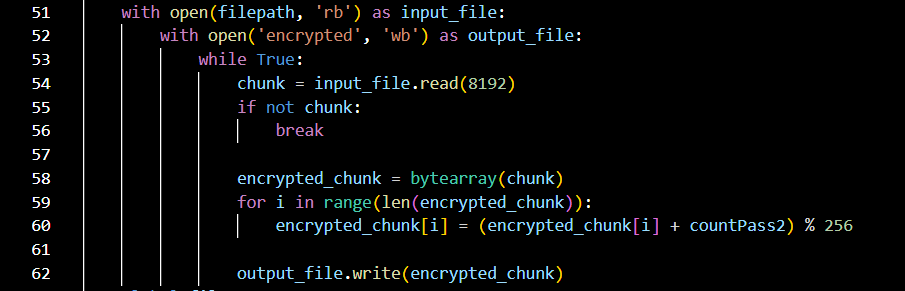
## Description

Encryption for noobs (EFN) is an encryption method that we developed from scratch in python. This encryption encrypts and decrypts based on a password given by the user, thus making it secure because you need a specific password to decrypt an encrypted file much like the Fernet encryption method.

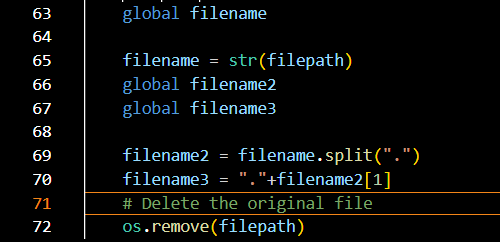
**Encryption**



Here the password is converted to an ordinal number on which the encrypting and decrypting is based on.

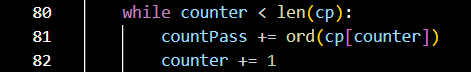


The selected file is opened and turned into encrypted chunks then the encrypted chunks are written into an output file/decrypted file.

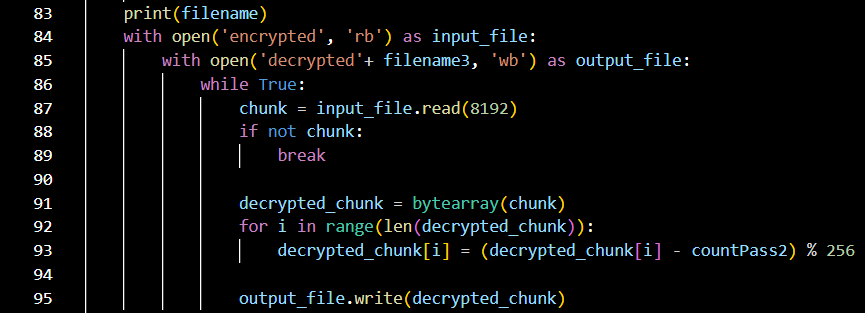


The original files extension is saved here so that when we decrypt the file it is reverted to the same type of file as the original.

**Decryption**



Just as in the encryption the decryption also turns the password into an ordinal number used in the decryption of the file, thus if the passwords don’t match the decryption will be incorrect.



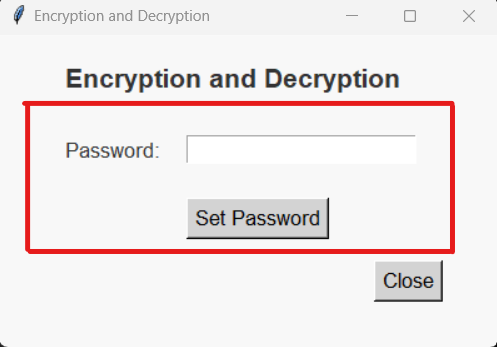
Here the encrypted file is opened, then written into a new file ‘decrypted’ with the original files extension so keep it the same file type. The encrypted chunks are then reverted into original content as decrypted chunks and written into the output file.

## How to use

**Encrypting**

Step1: Open the application.

Step2: Set a password that will be used as a key for encrypting and decrypting.



Step3: Browse for file you want to encrypt.

A screenshot of a computer

Description automatically generated

Step4: Type in same password you set at Step 2.

A screenshot of a computer

Description automatically generated

Step5: Select encryption method.

A screenshot of a computer

Description automatically generated

Step6: Select Encrypt.

A screenshot of a computer

Description automatically generated

Step7: Press Run button.

A screenshot of a computer

Description automatically generated

**To find the encrypted file:** Open File explorer > This PC > Local Disk(C:) > Users > User

**Decryption**

Step1: Browse for file you want to decrypt (Open File explorer > This PC > Local Disk(C:) > Users > User).

A screenshot of a computer

Description automatically generated

Step2: Input same password set while encrypting.

A screenshot of a computer

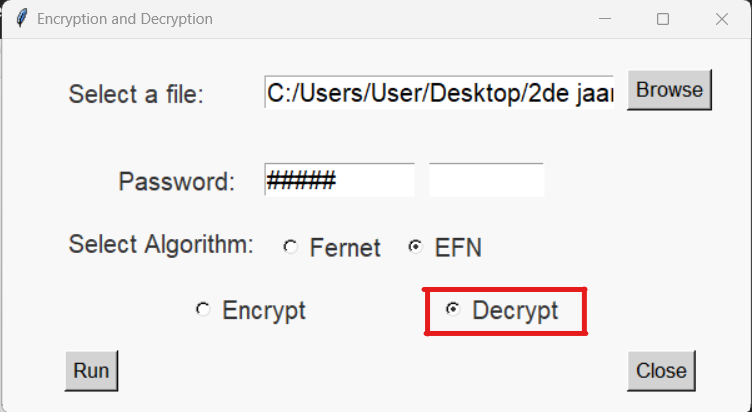
Description automatically generated

Step3: Select Algorithm \*same as the algorithm used to encrypt\*.

A screenshot of a computer

Description automatically generated

Step4: Select Decrypt.



Step5: Press Run Button.

A screenshot of a computer

Description automatically generated

**To find the decrypted file:** Open File explorer > This PC > Local Disk(C:) > Users > User

# References

Cryptography.io. (n.d.). Fernet: Easy-to-use encryption. https://cryptography.io/en/latest/fernet/

Herrera, L. (2018). Practical cryptography with Python. Apress.

Mukherjee, P. (2020). Mastering Python Networking: Your one-stop solution to using Python for network automation, programmability, and DevOps, 3rd ed. Packt Publishing.

Cryptography.io. (n.d.). Fernet: Easy-to-use encryption. https://cryptography.io/en/latest/fernet/#limitations

Szefer, J., & Keller, E. (2018). Cryptography for embedded systems. Springer.

https://www.geeksforgeeks.org/fernet-symmetric-encryption-using-cryptography-module-in-python/