Report

The architecture and the deign

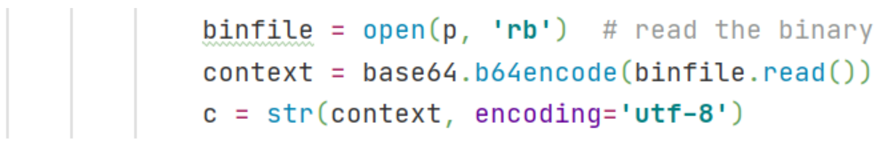
This experiment is used to compare AES and DES, so the application has AES or/and DES encryption and decryption functions.

The user drags the files, enters key, and clicks encrypt or decrypt button to get the encrypt or decrypt files, which will be stored in the same directory as the code. Supposing that the origin file' name is a. The encrypt file is named The aes/des text of a.txt, and the decrypt file is named The origin text (AES/DES) of a.

About the design of the code, there are four different methods to decrypt AES and DES respectively. In addition, each interface corresponds to a method.

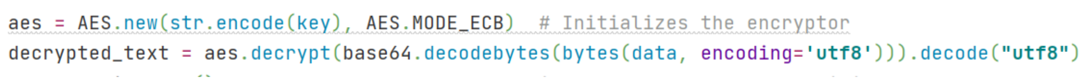
The implement of the application

This application windnd, tkinter, pillow, base64, time and Crypto library. The basic UI interface is implemented through Tkinter. Windnd is used for dragging files, and pillow is used for processing photos. In addition, time is used to record the time and compare them.
Base64 and Crypto are the core library for this application.



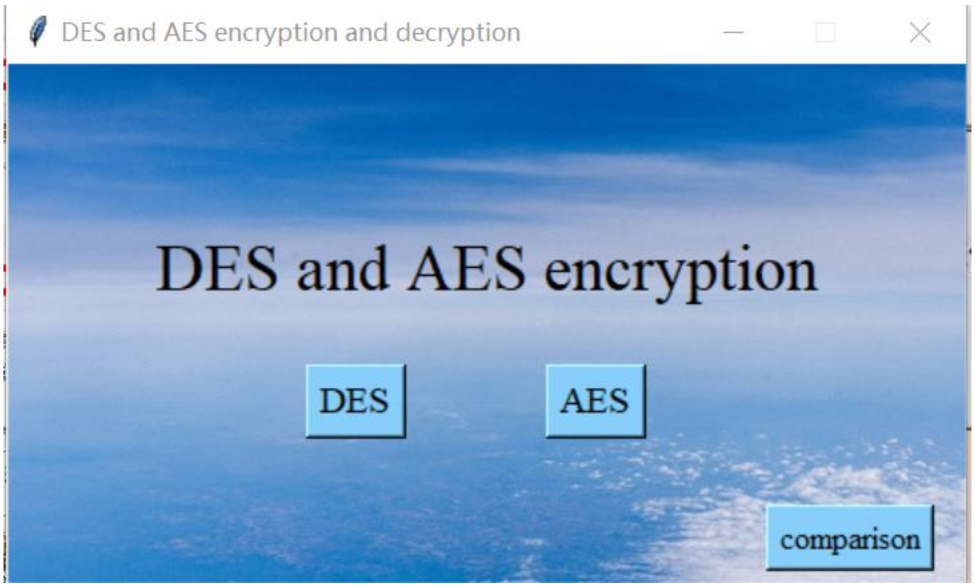
```
binfile = open(p, 'rb')  # read the binary
context = base64.b64encode(binfile.read())
c = str(context, encoding='utf-8')
```

Base64 stores the contents of a file as Bytes by decoding, which is very vital for encrypting or decrypting.
Crypto is the core library to implement DES or AES algorithm. Below is the example of implement decryp

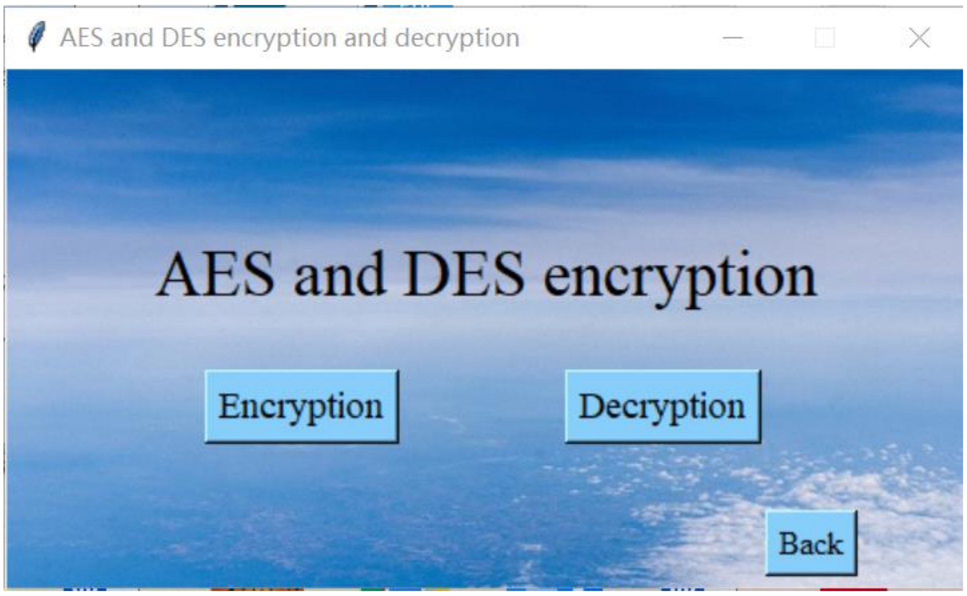


```
aes = AES.new(str.encode(key), AES.MODE_ECB)  # Initializes the encryptor
decrypted_text = aes.decrypt(base64.decodebytes(bytes(data, encoding='utf8'))).decode("utf8")
```

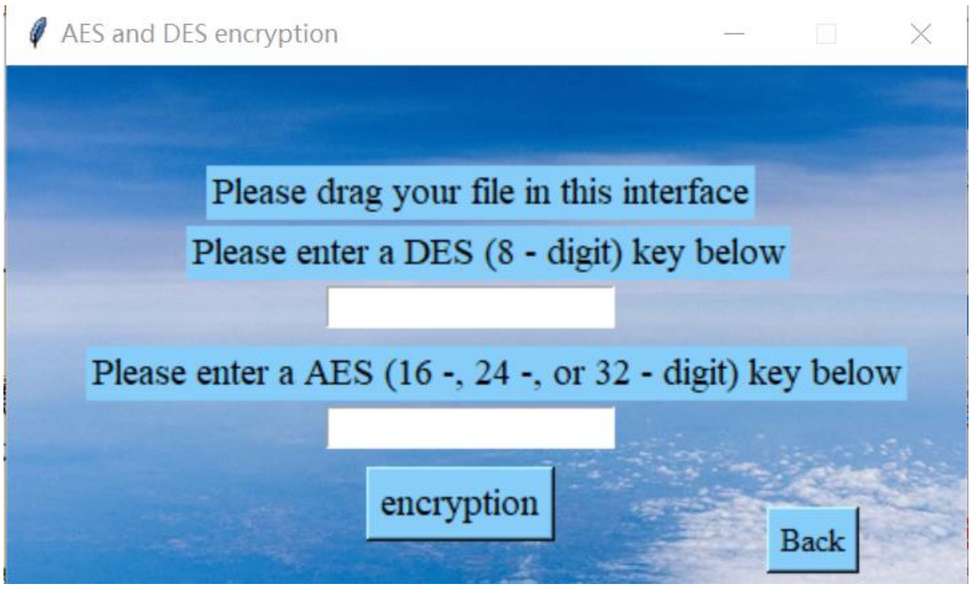
The user interface



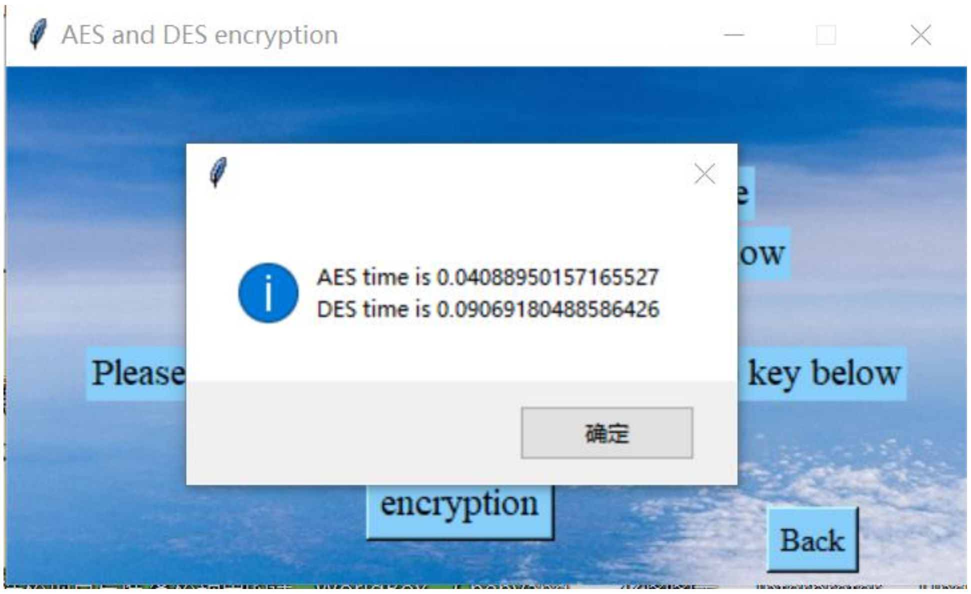
The above is the main interface, which has 3 button, DES, AES, and comparison. All of these three buttons can enter the interface that can encrypt or decrypt. In the following, it is the comparison interface, and DES and AES interfaces are very similar to this.



Encryption and Decryption interface are very similar.

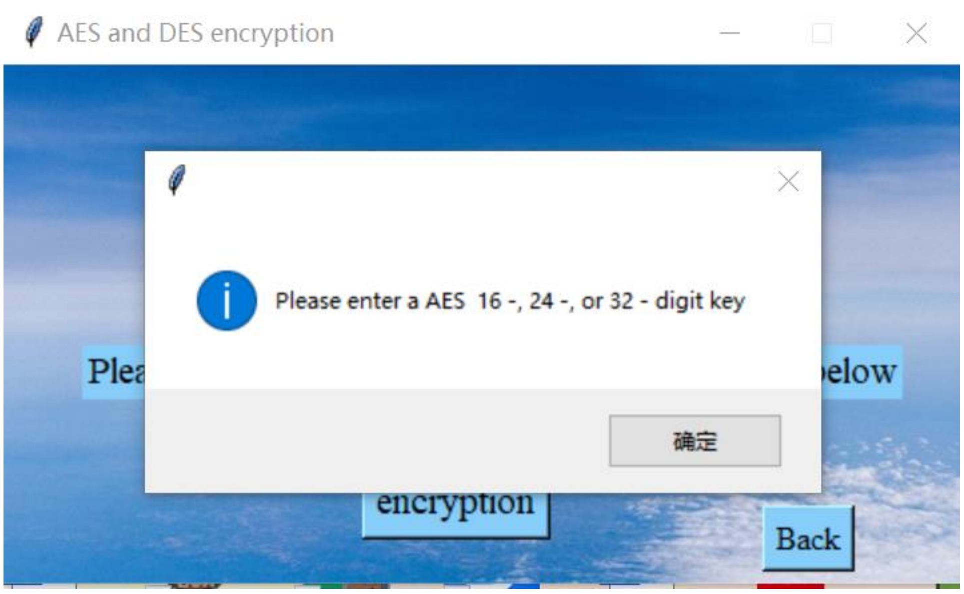

AES and DES encryption
Please drag your file in this interface
Please enter a DES (8 - digit) key below
Please enter a AES (16 -, 24 -, or 32 - digit) key below
encryption
Back

You can drag in as many files as you want to encrypt. For the Comparsion feature, the files are DES and AES encrypted, then the encrypted files are generated and The Times are compared.



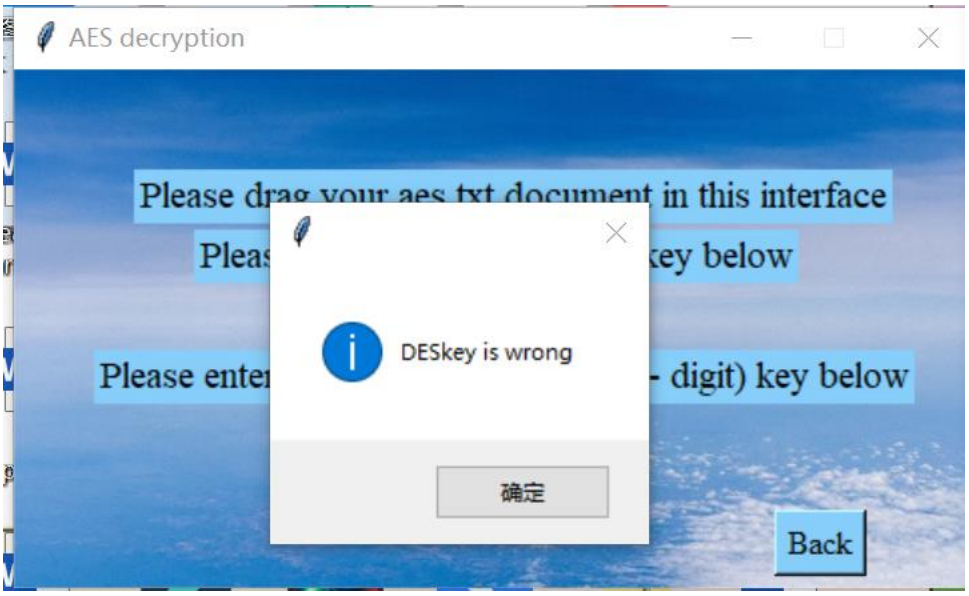
AES time is 0.04088950157165527
DES time is 0.09069180488586426

Note that DESkey must be 8 bits and AESkey must be 16,24, or 32 bits. Otherwise, the program will remind you to enter a key with the correct number of digits


AES and DES encryption

Please enter a AES 16 -, 24 -, or 32 - digit key

This comparsion function is used to compare encryption and decryption times. Consequently, you only can decrypt one AES encrypted file and one DES encrypted file in order to compare their speeds. The app will alert you if you upload the wrong file. The app will also alert you if you enter the wrong key. If you have ever changed the name of an encrypted file, our program will not recognize the encrypted file.

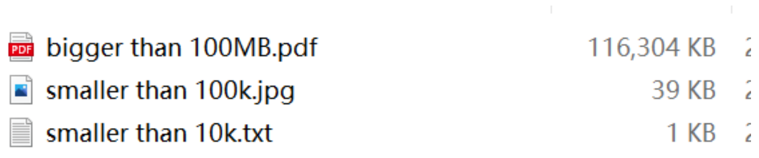

AES decryption

Please drag your aes txt document in this interface

DESkey is wrong

All of these interfaces and operations are very similar for DES and AES functionality.

The experiment



bigger than 100MB.pdf	116,304 KB
smaller than 100k.jpg	39 KB
smaller than 10k.txt	1 KB

At first, I have repeatedly tested the different encryption time and decryption time of the above three sizes of files for AES or DES.

For the bigger than 100MB.pdf

Times	AES(Encrypt)	DES(Encrypt)	AES(Decrypt)	DES(Decrypt)
1	3.15441	4.85832	1.66322	3.37529
2	2.84990	4.66498	1.49341	3.83527
3	3.04737	4.52993	1.55187	3.18523
Average	3.01722	4.68441	1.56950	3.46526

For the smaller than 100k.jpg file

Times	AES(Encrypt)	DES(Encrypt)	AES(Decrypt)	DES(Decrypt)
1	0.001995	0.001994	0.000998	0.001991
2	0.003004	0.003941	0.001990	0.003987
3	0.002088	0.001995	0.000993	0.001993
Average	0.002362	0.002643	0.001327	0.002657

For the smaller than 10k.txt file

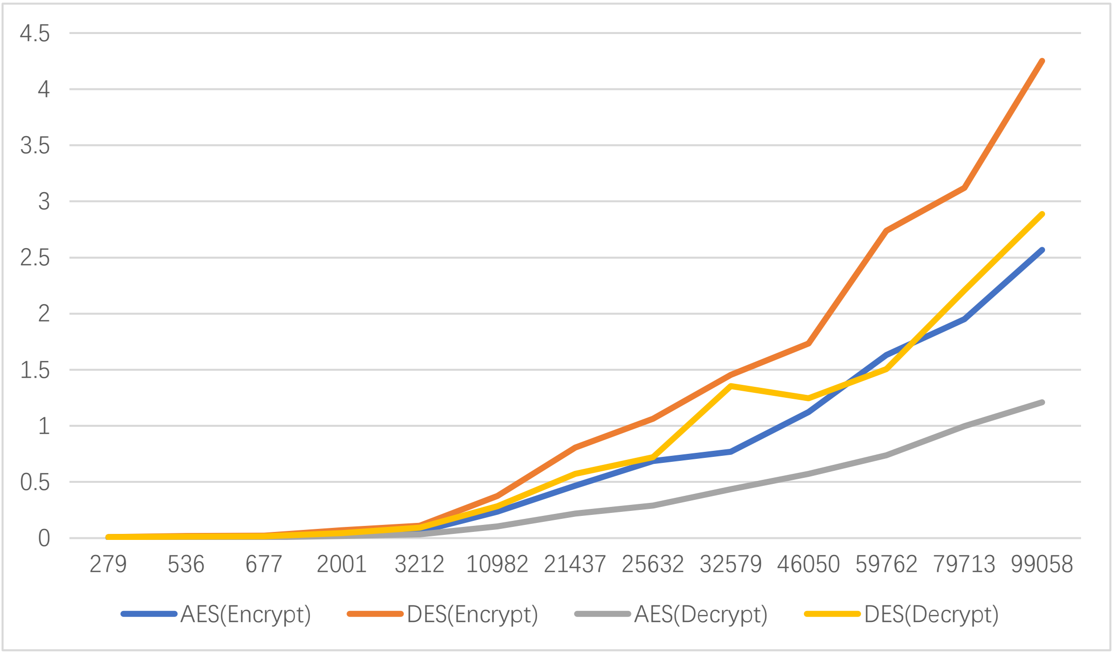
Times	AES(Encrypt)	DES(Encrypt)	AES(Decrypt)	DES(Decrypt)
1	0.000987	0.000996	0.001989	0.001000
2	0.000969	0.000995	0.001054	0.000000
3	0.000996	0.000997	0.000996	0.000998
Average	0.000983	0.000996	0.001346	0.000666

By comparison, the speed difference between AES and DES is not significant for 49KB and 1KB files. However, for files over 100MB, the gap becomes increasingly significant. Since the gap between 39KB and 100MB is too large, we will intercept several points between the two to compare speeds. The following data were averaged through multiple tests

Size(kb)	AES(Encrypt)	DES(Encrypt)	AES(Decrypt)	DES(Decrypt)
279	0.00488	0.00894	0.00299	0.00897
536	0.00997	0.01894	0.00598	0.01396
677	0.01296	0.02293	0.00697	0.01696
2001	0.03884	0.06984	0.01999	0.04580
3212	0.05983	0.10876	0.03291	0.09276
10982	0.23669	0.37489	0.10473	0.28523
21437	0.46600	0.80639	0.21845	0.57338
25632	0.68531	1.06445	0.28918	0.72001
32579	0.76899	1.45484	0.43499	1.35352
46050	1.12204	1.73391	0.57205	1.24721
59762	1.63127	2.73691	0.73813	1.50587

79713	1.95179	3.12153	0.99727	2.20702
99058	2.56772	4.25205	1.21006	2.88737

The following figure is a line chart based on the data in the table.



Three conclusions can be drawn,

1. AES and DES generally slow down as file sizes grow
2. Decryption is faster than encryption for both DES and AES
3. AES is getting faster and faster than DES as file sizes increase

As for the first point, it is very easy to understand. With more information to process, it's normal to slow down. The second point is because all of the size of the encryption file is less than the origin file. To understand the third point, It is important to know how AES works.

At first, the plaintext is cut into 128bits each. For the last remaining less than 128bit plaintext zeros processing. The plaintext of each part is encrypted and the result is finally spliced. These processes are very similar to DES, except that DES cuts every 64 bits. The speed difference is mainly in the number of rounds processed for each plaintext.

As for AES, the rounds can be divided into 3 types, initial round, rounds, and final round. For 128,192, and 256 length keys, the AES algorithm goes through 10, 12, and 14 rounds, respectively (not include initial round). For the initial round, there is only 1 step, addroundkey. For the rounds, it has 4 steps, subbytes, shiftrows, mixcolumns, and addroundkey. The final round includes subbytes, shiftrows, and addroundkey. In fact, these small processes are also very similar to DES.

SubBytes:

The 128-bit plaintext is arranged in a 4*4 matrix. This is a per-byte processing of the text. Each byte in the plaintext corresponds to a value in a Subtitution Box (a 16-by-16 two-dimensional array of constants) that replaces the value in the Substitution box with the value in the original plaintext. For the encrypting, using Subtitution Box, and decryption uses inverse substitution box.

ShiftRows:

The first row stays the same, and then the nTH row goes n to the left.

MixColumns:
For each column of the input array, the matrix is multiplied by a two-dimensional array of constants called the fixed matrix to get the corresponding output column.

AddRoundKey
The 128-bit Key is arranged in a 4*4 matrix, so that each byte in the original text is xor computed with the corresponding byte in the ciphertext, and the Key of each round is processed by keyexpansion. Use an array W of length 4 * 4 * (10+1) bytes to store the keys for all rounds. The value of W{0-15} is equivalent to the value of the original key, which is used for processing for the initial round. Each subsequent element W[I] is evaluated by W[i-4] and W[i-1] until all elements of array W are assigned.

Although the length of the key is larger, AES can process the plaintext in fewer rounds. Because there are fewer copies of plaintext cut. In the case that the number of rounds for processing the key remains unchanged. So the longer the plaintext, the more significant the difference. For smaller files, the difference is not significant. DES may even be slightly faster due to the increase in Key length.

Meanwhile, both DES and AES can be implemented in different modes. All these experiments are based on the ECB model, which is a easiest modes without additional operation. Another model is named CBC, which need to using initialization vector to make the same plain text get different ciphertext, which is safer. In order to compare the results, I will encrypt files larger than 100MB in different modes.

Mode	DES	AES
ECB	4.81	2.67
CBC	4.98	2.72
CFB	17.93	6.33

For ECB and CBC mode, the speed difference is not significant. CFB is a very complex and secure mode, so there is a big difference in speed.

The effort

For assignment 1, I implemented the DES algorithm using strings, which is very slow. This made my initial comparison between DES and AES a thousand times worse. In the Crypto library, DES and AES algorithms are implemented by manipulating bytes, which is what makes them comparable. The initial comparison of AES and DES algorithms implemented by different implementations is complex and meaningless. At the same time, the UI interface background image processing, the new window opened, the old window processing is not very simple.

The conclusion

By implementation, IT can be shown that AES is a faster encryption algorithm. In addition to this, DES is more insecure because of its short keys, short sets of plaintexts, and simple key handling. For DES, every 64 bits will be encrypted once, will be processed many times, too waste of resources, AES is more memory saving. At the same time, AES algorithm is more secure because of the variability of KEY length.

All in all, AES is better.