Andrew Sharkawy

Assignment 2

PDF For assignment

My code explained and thought process:

Cse108_hw2_asharkaw.py:

The code starts by defining some variables, including the path to the cipher text file (cpath), an array to store the frequencies of each byte value (frequencies), and some initial values for other variables.

1. The main function is the entry point of the program. It reads the cipher text file, decrypts it using a key, and performs frequency analysis to determine the key length.
2. The cipher text is read from the file using the read_hex_file function, which converts the hex-encoded text into a byte array.
3. The cipher text is decrypted using the decrypt function. The decryption process XORs each byte of the cipher text with a corresponding byte from the key. The key is a repeated sequence of bytes obtained from the key variable.
4. The program then performs frequency analysis to determine the key length. It iterates over different possible key lengths (l) from 1 to 14 and calculates a score based on the frequencies of characters in the decrypted text. The score is a measure of how closely the frequencies match the expected distribution of English characters.
5. After finding the key length with the highest score, the program divides the decrypted text into streams based on the key length. Each stream represents the bytes that were XORed with a particular byte of the key.

6. The program defines a function computeSumFrequency to calculate the score for a given stream of decoded bytes. It counts the frequencies of each character in the stream and calculates a weighted sum based on the expected frequencies of English characters (q).
7. The program then defines a function bestByte to find the most likely byte value for each stream. It iterates over all possible byte values (0 to 255) and XORs each value with the stream's bytes to obtain a decoded stream. It calculates the score for each decoded stream using computeSumFrequency and selects the byte value with the highest score as the most likely key byte for that stream.
8. Finally, the program prints the resulting key as a list of byte values.

Plaintext.py:

The main function is the entry point of the program. It sets the path to the cipher text file (cpath) and defines the key as a sequence of bytes (key).

1. The program reads the cipher text from the file by calling the read_hex_file function, passing the file path. The function opens the file in text mode and reads its contents, removing any newline characters. The hexadecimal data is then converted into a byte array using the bytes.fromhex method. The resulting byte array is stored in the variable ctext.
2. The program decrypts the cipher text using the provided key by calling the decrypt function, passing the cipher text and the key as arguments. The decrypt function takes each byte of the cipher text (ctext) and performs an XOR operation (^) with the corresponding byte from the key. The key bytes are cycled using the modulo operator (%) to ensure that the key is repeated if it is shorter than the cipher text. The resulting decrypted bytes are stored in the ptext byte array.

3. The decrypted plaintext is then printed to the console by decoding the byte array ptext into a string using the ASCII encoding (ptext.decode('ascii')).
4. The read_hex_file function opens the file specified by the given filepath in text mode and reads its contents. It removes any newline characters and obtains a string of hexadecimal data. The bytes.fromhex method is used to convert the hexadecimal string into a byte array, which is returned from the function.
5. The decrypt function takes the cipher text (ctext) and the key as arguments. It initializes an empty byte array ptext to store the decrypted bytes.
6. The function iterates over each byte of the cipher text using the range function and the len(ctext) as the upper limit. For each iteration, it appends to ptext the result of performing an XOR operation between the current byte of the cipher text (ctext[i]) and the corresponding byte from the key (key[i % len(key)]). The modulo operation ensures that the key bytes are cycled when the key is shorter than the cipher text.
7. Finally, the decrypted byte array ptext is returned from the decrypt function.
8. The if __name__ == '__main__': condition checks if the script is being run directly and not imported as a module. If it is the main script, it calls the main function to start the program.

This script is responsible for analyzing the cipher text, determining the key length, and finding the secret key used for encryption.

Usage

1. Place the cipher text file (cipher.txt) in the same directory as the script.
2. Execute the script using the command python cse108_hw2_asharkaw.py.
3. The script will analyze the cipher text and print the decrypted plaintext.

plaintext.py

This script uses the key obtained from the previous script to decrypt the cipher text and display the plaintext.

Usage

1. Ensure that the cipher.txt file is in the same directory as the script.
2. Execute the script using the command python plaintext.py.
3. The script will read the cipher text, decrypt it using the key, and display the decrypted plaintext.

KEY

The key that is recovered from cse108_hw2_asharkaw.py is printed as a list of integers.

secretKey [174, 34, 0, 175, 47, 184, 153, 17, 221, 0]

[174, 34, 0, 175, 47, 184, 153, 17, 221, 0]

We then convert this key to HEX using the byte to string function and we Get:

<u>KEY = AE 22 00 AF 2F B8 99 11 DD 00</u>

PLAINTEXT:

We then use plaintext.py and use this key to figure out the plaintext for the cipher.txt file.

Plaintext is:

We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

The development of computer controlled communication networks promises effortless and inexpensive contact between people or computers on opposite sides of the world, replacing most mail and many excursions with telecommunications. For many applications these contacts must be made secure against both eavesdropping and the injection of illegitimate messages. At present, however, the solution of security problems lags well behind other areas of communications technology. Contemporary cryptography is unable to meet the requirements, in that its use would impose such severe inconveniences on the system users, as to eliminate many of the benefits of teleprocessing.