CSE 539: Applied Cryptography

Project 1: Block Encryption Algorithm Project

Group members:

Atin Singhal (ASU ID: 1217358454)

Prerana Mahalanobis (ASU ID: 1217126352)

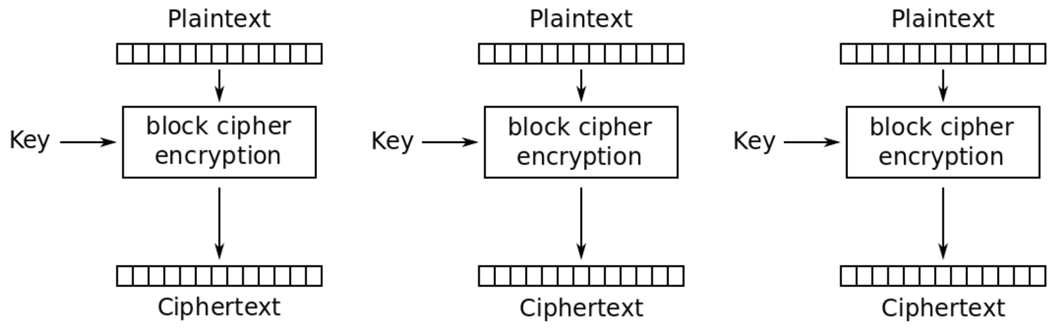
Submitted on: 09/23/2019

Objective:

- Develop a (reversible) 32-bit block encryption algorithm.
- Utilize substitution and permutation in encryption.
- Brute force a block encryption algorithm.

Description:

- The algorithm is for a 32-bit block cipher which uses Electronic Code Book (ECB) mode, i.e. the message is divided into blocks, and each block is encrypted separately.
- The cipher takes two 32-bit quantities as input, a plaintext and a key, then produce a 32-bit output ciphertext.
- The key is an 8-character (32-bits) hexadecimal string (example: "102B9E0F")
- The encryption algorithm is reversible, i.e. it is possible to decrypt the ciphertext when the key is known.



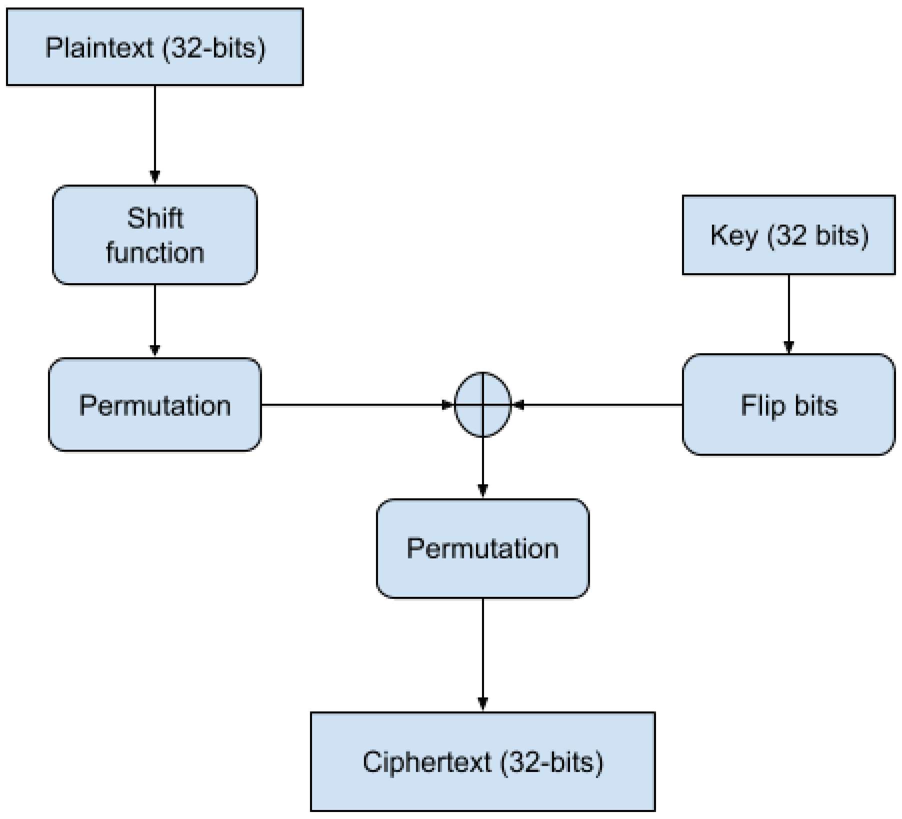
Electronic Codebook (ECB) mode encryption

Algorithm Design:

Encryption:

- The algorithm takes plaintext & key as input.
- A type of substitution cipher in which each character in the plaintext is replaced by a character some fixed number of positions down the ASCII table is used. For this algorithm, the value by which we are shifting the characters is taken from the last hexadecimal bit for the key. For example, if the key is “102B9E0F”, there will be a right shift of 16, A would be replaced by Q, B would become R, and so on.
- The result of the previous operation is passed to a permutation box.
- The bits of the key are flipped and an XOR operation is performed between the resultant of previous operation & the flipped key.
- The result of the previous operation is again passed through a permutation box for bit-shuffling.
- The string obtained is the cipher text.

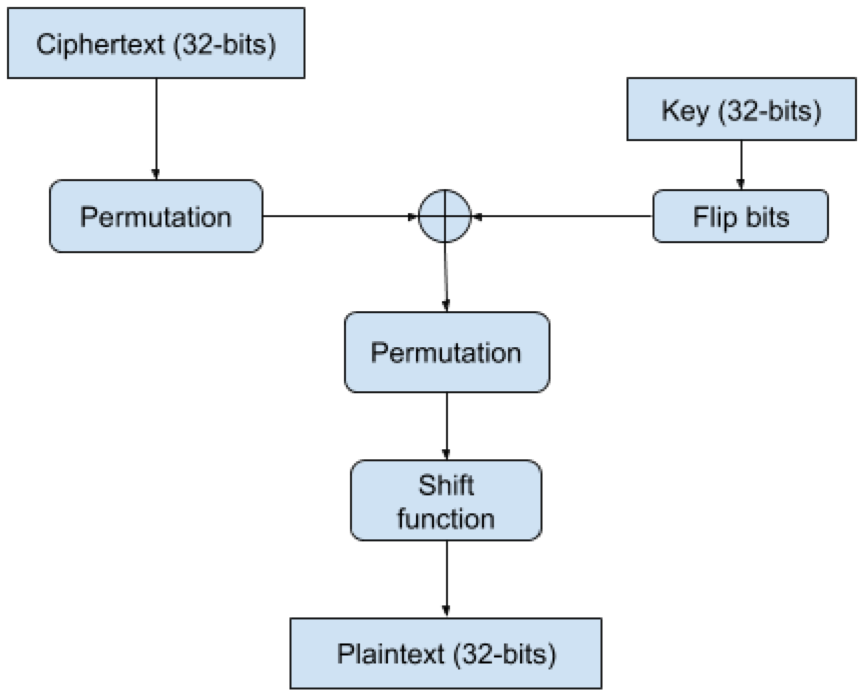
The block diagram for encryption is as follows:



Decryption:

- The algorithm takes ciphertext & key as input.
- The ciphertext is passed through a permutation box for bit-shuffling.
- The bits of the key are flipped and an XOR operation is performed between the resultant of previous operation & the flipped key.
- The result of the previous operation is again passed through a permutation box for bit-shuffling.
- The binary text is now converted into ASCII character set and a type of substitution cipher is used to obtain the original plaintext. (Deciphering is done in reverse, with a left shift.)

The block diagram for decryption is as follows:



What makes the encryption algorithm effective:

- This algorithm passes the text through the permutation box twice. We know that permutation boxes are poly-alphabetic, unlike substitution which is mono-alphabetic.
- As the algorithm uses a combination of permutation box & substitution cipher, the time required for brute force will increase.
- Changing a bit in the plaintext, changes more than one bit in the ciphertext.
- If an attacker somehow does find out the plaintext corresponding to a particular cipher text, brute forcing the key takes a lot of time, making the algorithm very difficult and time consuming to break.

Brute Force:

- For brute force, we used the “Known-Plaintext” technique of cryptanalysis, i.e. we assume the plain text and the cipher text are known.
- Initial thoughts were to generate a list of all possible keys and then use each key to decrypt the ciphertext and then compare them with the plaintext (as it is difficult to say which deciphered version is correct without knowing exactly what we are looking for, like a message in english or any other language etc.).
- If we generate a list of all possible keys, i.e. 2^32 possible combinations (which is 4294967296 keys), it would take a lot of time & disk space for one brute force. This method would be feasible to reduce the time for future brute force attacks as the list of all possible keys can be used as a dictionary.
- For this project, we compare the plaintext with the decrypted text after each iteration of deciphering text with a key and stop upon getting a match.

  Our findings were as follows:
  - It took **32.64 seconds** to encrypt a text file of 10 MB.
  - It took **5613.44 seconds (93.56 minutes)** to brute force the key, assuming we already know the cipher text and plain text (for a 10 MB file). The key we used for the algorithm was “102B9E0F”.

Brute Force on various File types:

- Since the files are opened in binary irrespective of their type, file type has no effect on the time taken to brute force.
- However, the time taken for encryption as well as to brute force varies with the size of the file.

Weakness in the algorithm:

- The algorithm is only 32-bit. If we increase the size of the input block & the key, the strength of the cipher can be increased.
- This algorithm uses only one round of the above mentioned function. More secure algorithms use multiple rounds with multiple keys or modified keys to make the cipher difficult to brute force.
- The encryption uses the P-box twice along with some other functions (as mentioned earlier). A combination of P-box with S-boxes would make it relatively stronger.
- The substitution cipher used shifts the characters by *x*, which is based on the key. If we allow the user to input the value by which we should shift the characters, the attacker will have to try 256 combinations per key, which would significantly increase the time required for brute force.