CSE 539 Applied Cryptography Project-1

CHIRAG BHANSALI (1215185491)
SHIVANK TIWARI (1217351382)

Describes your algorithm design and why you made those choices.

For this project, we modified the Vignere cipher. Firstly, we are using the hexadecimal (0-9, A-F), instead of the English alphabets. We made this choice to reduce the memory space it would take to store the matrix of 26*26. Instead, we reduced the size of the matrix to 16*16. Secondly, we changed the original hexadecimal mapping to the different number values, so instead of having the 0 mappings to 0, 1 mapping to 1 and so on, we are mapping the values starting from 5, going till 15 and then starting from 0 again. This would help in adding an extra level of complexity since it would be completely mapped to a different character than the expected one. Lastly, we have used 3 PBoxes for *diffusion* and make the intermediate cipher-text even more than obscure.

Hexadecimal	Mapped Number
0	5
1	6
2	7
3	8
4	9
5	10
6	11
7	12
8	13
9	14
A	15

B	0
C	1
D	2
E	3
F	4

Algorithm for encryption :

1. Obtain input file in binary.
2. Convert binary to hexadecimal.
3. Make the key size equal to the size of the input hexadecimal, by repeating the key the said number of times.
4. Use the modified Vignere cipher and use the above table to compute the sum of the message's and key's converted hexadecimal to mapped value for each individual character independently. Take modulo 16 for the sum and substitute the value obtained with the corresponding hexadecimal value from the table above.
5. Add padding to the intermediate ciphertext to make it a multiple of 32 (useful when doing the permutation step below)
6. Take blocks of 32bits and Input the intermediate ciphertext block into the PBox-1.
7. Input the output of PBox-1 into the PBox-2.
8. Input the output of PBox-2 into the PBox-3.
9. Repeat the steps 5 to 7 for each block of 32 bits / 8 hexadecimal digits.

The resultant concentrated output value of PBox-3 is our final cipher-text.

Algorithm for decryption:

1. Use the cipher-text and divide it into blocks of 32 bits/ 8 hexadecimal digits.
2. Insert the block in the reverse PBox-3.
3. Input the output of reverse PBox-3 into reverse PBox-2
4. Input the output of reverse PBox-2 into reverse PBox-1
5. Repeat steps 2 to 4 for each block and concentrate them.
6. Remove the padding from the total concentrated output of the reverse PBox-1, since we already know the length of the plaintext, make the intermediate ciphertext's length the same as the plaintext one.
7. Take Key = 0x0.

8. Make the key equivalent to the size of the intermediate cipher-text, by repeating it the required number of times.
9. Use the modified Vignere cipher and compute the difference of the ciphertext's and key's converted hexadecimal to mapped value from the table for each character separately and independent of other characters. Take modulo 16 of the value and substitute it with the corresponding hexadecimal value.
10. Compare the value obtained from step 7 with the known plaintext and if they don't match, add 0x1 to the key and repeat steps 6-8 until the converted string and plaintext match.

Below is the character shuffling for the PBox's. The second one is shuffled characters. First is the place where the respective character will move to in an 8 character long array.

PBOX 1:

0	1	2	3	4	5	6	7

Converts to

2	3	1	7	6	5	4	0

PBOX 2:

0	1	2	3	4	5	6	7

Converts to

7	2	3	5	1	6	0	4

PBOX3:

0	1	2	3	4	5	6	7

Converts to

5	4	6	0	2	3	7	1

Reverse PBOX 3:

0	1	2	3	4	5	6	7

Converts to

3	7	4	5	1	0	2	6

Reverse PBOX 2:

0	1	2	3	4	5	6	7

Converts to

4	6	1	2	7	3	5	0

Reverse PBOX 1:

0	1	2	3	4	5	6	7

Converts to

7	2	0	1	6	5	4	3

Explain what makes your encryption algorithm effective.

Firstly by using the vignere cipher, which is a polyalphabetic cipher, we obtain different values for the same character because of the shifts, thus the algorithm isn't susceptible to frequency analysis.

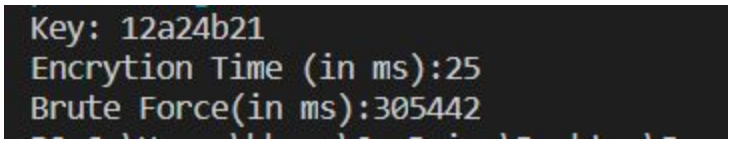
Since one of the biggest issues with the Vignere cipher is the repeating key to make it match the length of the plaintext, we mitigated the issue with the help of the PBox, which helps in diffusion, making it difficult to do key length prediction or frequency analysis.

We aren't storing the entire viphere cipher table for our program, instead, we are calculating the value sum and taking the modulo character-wise, thus reducing the memory space.

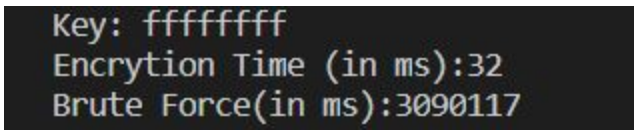
We changed the hexadecimal mapping to a decimal number, which makes it a bit difficult to go with the conventional method, to reverse the map and find the solution.

Documents the time it takes for you to brute force the key.

For key : 12a24b21, it took 5.0907 minutes.



For key: ffffffff, it took 51.50 minutes



Explain how the file type relates to the amount of time it takes to brute force that file.

Keeping in mind the design of the algorithm, the amount of time it took to brute force key with respect to different file types pretty much remained the same. Once the binary format is obtained, time taken remains the same irrespective of file type, information type, etc. In a nutshell, the process can be said to have two parts. First being, conversion of information to bits. Second, being the whole encryption process. Time can vary only due to the first part i.e. file conversion to binary, but if provided number of bits obtained are same, time taken for the second stage remains the same be it from notepad, pdf, word, excel etc or images in JPG, BMP format. The first stage takes a couple of hundreds of milliseconds. So this contributes insignificantly (1000 ms makes 1 second and considering this alongside minutes won't affect the results). Even during the brute force part, we matched hexadecimal

of plain text and hexadecimal of obtained cipher through a particular key. This also reduced the overhead of repetitive conversion of hexadecimal to plain text.

What weaknesses did you find? How did they result in faster breaks of the encryption?

1. The key is small, known and of a fixed size. This makes the brute force process easier.
2. The number of characters in the vignere cipher is less, instead of 26 for the English language, the modified one, has only 16. Thus, reducing the number of ways in which plaintext could be encoded to 16^k.