Singapore Polytechnic
MS0105 PBL AY2017/2018

Lecturer: Dr Loo Wing You

Done by:
Yong Zheng Yu, Javier (1726682)
David Zhu HaoYuan(1703177)
Lim Yong Kang, Bryan (1707139)
Trovald Ong Chu Hao (1703218)

Class:
DISM/FT/1A/21

Mathematics: Problem Based Learning (PBL)
Research Report: Spinning Zebra Encryption (SZE)

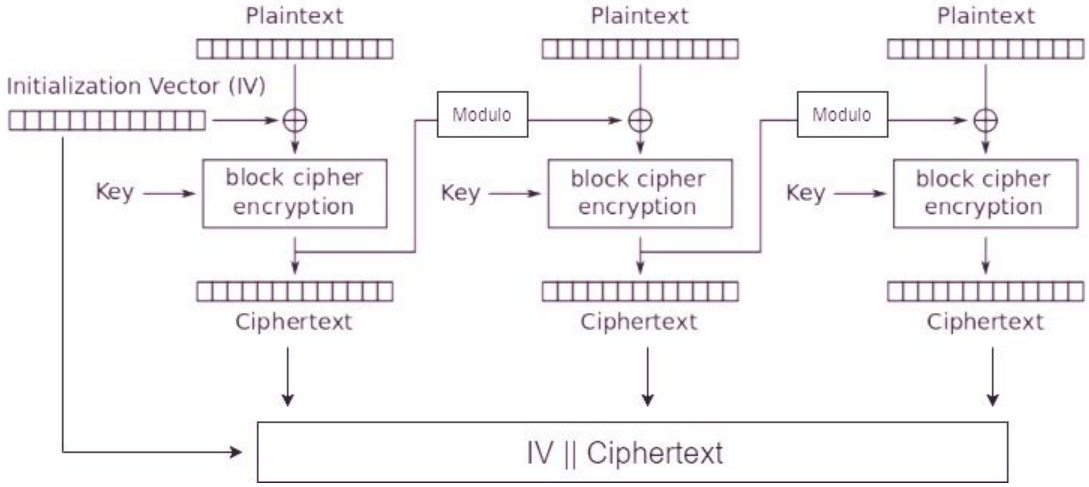
The main aim of this report is to showcase our Symmetric-Key block cipher algorithm.

Our encryption/decryption algorithm revolves around the use of matrix multiplication and XOR.

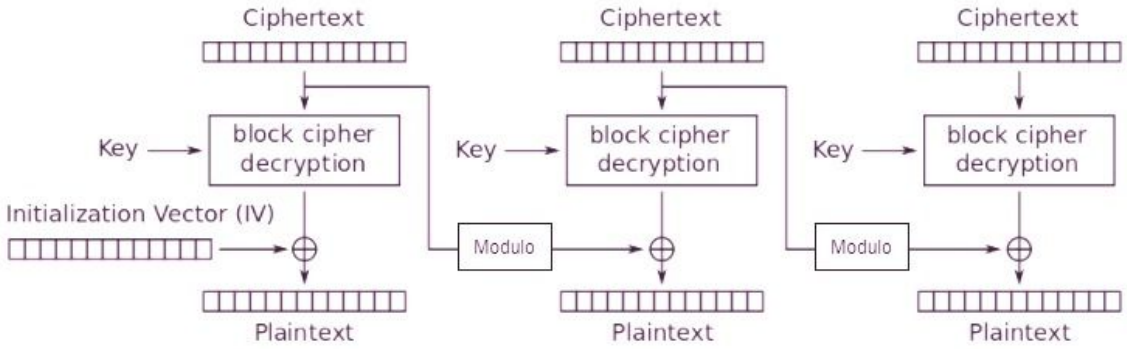
The algorithm has a key size of 213 bits and has $2^{213} = 1.32 \times 10^{64}$ (3 s.f.) combinations, and will take about approximately 4.49×10^{42} years to crack using a supercomputer. (Using 93.01 PFLOPS as computing speed)

This is an overview of the encryption and decryption steps.

Encryption



Decryption



For each message, a cryptographically secure pseudo random number generator (CSPRNG) will be used to generate the key values.

Our sample plaintext message is “Valid message here!”.

Encryption

1) Firstly the message that we will be encrypting is “Valid message here!”. This message will be converted into decimal with reference to the ASCII table and by filling the blank space with a padding of 0 which will look like this: $[86, 97, 108, 105, 100, 32, 109, 101,115, 115, 97, 103, 101, 32, 104, 101, 114, 101, 33, 0, 0, 0, 0, 0, 0, 0, 0]$ (Amount of 0s is dependent on how much blank space are there when forming a 3x3 matrix)

2) After the conversion on the padded message we will split the message into blocks each consisting of 9 value

```
Block 1: [  86, 97, 109, 105, 100,  32, 109, 101, 115]
Block 2: [ 115, 97, 103, 101,  32, 104, 101, 114, 101]
Block 3: [  33,  0,   0,   0,   0,   0,   0,   0,   0]
```

3) We generate an Initialization Vector (IV) with a range of 0 to 255, the IV will contain 9 random integers.
This is the IV used in this report: $[46, 199, 0, 187, 104, 160, 73, 92, 159]$

We then generate the XOR key.
XOR key: $[6994, 54643, 45116, 34884, 3486, 41803, 44882, 40658, 26008]$

We create a 3x3 encryption matrix using PRNG with a range of 1 to 80. 80 is set as the limit because the numbers after matrix multiplication has to be kept below 65535 which is the max value for 4 hexadecimal digits. The matrix generated will be checked if it is invertible, if it is not it will be regenerated.

Encryption Matrix
```
[16 36 77]
[16 36 76]
[32 74 50]
```

Turn the XOR Key into 4 hexadecimal digits by padding with 0:
```
[1b52, d573, b03c, 8844, 0d9e, a34b, af52, 9ed2, 6598]
```

Then we flatten the transformation matrix into an array and convert and pad to 2 hexadecimal digits:
```
[10, 24, 4d, 10, 24, 4c, 20, 4a, 32]
```

After which add the two hexadecimal strings together to form the key.
Key: `1b52d573b03c88440d9ea34baf529ed2659810244d10244c204a32`

4) XOR first block and the IV together.
Block 1 $\oplus$ IV results: $[120, 166, 108, 210, 12, 128, 36, 57, 236]$

We represent the result in a 3x3 matrix

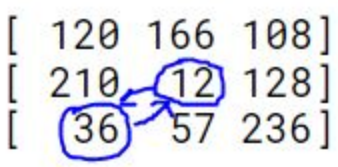
```
[ 120 166 108]
[ 210  12 128]
[  36  57 236]
```

5) After getting the 3x3 matrix, we will shuffle the positions of the values in the 3x3 matrix randomly. This is to allow the value positions to be randomised. The shuffling is done by swapping 2 elements in the block at a time, the values to swap is determined by the bits in the key, the first four bits determines the first character to swap and the next four determines the character to swap it with. The first 2 bits in each 4 bits determine the row of the character and next 2 bits determine the column. However if we get 2 bits that is 00, we will abandon that swap. Since each swap requires 8 bits and our key length is 213 bits, the maximum number of swaps is 26. The shifts will also be different for each block since the first block starts reading the bits from the first bit and the second block starts reading from the second bit and so on, it will continue until 208 bits is read. If the reading of bits reaches the end, it will wrap around and continue from the first bit. We will read the first element as row 0, column 0 which is similar to array indexing.

Key in binary:

```
1101 1010 1001 0110 1010 1011 1001 1101 1000 0001 1110 0100
0100 0010 0010 0000 0110 1100 1111 0101 0001 1010 0101 1101
0111 1010 1001 0100 1111 0110 1001 0011 0010 1100 1100 0000
1000 0001 0010 0010 0110 1000 1000 0001 0010 0010 0110 0001
0000 0010 0101 0001 1001 0
```

Eg. 1101 1010 will swap 36 and 12.



Block 1 After shuffling:

```
[166 120 128]
[ 12 108  36]
[236  57 210]
```

6) The encryption matrix will now be used in matrix multiplication with each block, [block][encryption matrix] = [result].

Block Matrix Encryption Matrix Result

$$\begin{bmatrix}166 & 120 & 128\\ 12 & 108 & 36\\ 236 & 57 & 210\end{bmatrix} \times \begin{bmatrix}16 & 36 & 77\\ 16 & 36 & 76\\ 32 & 74 & 50\end{bmatrix} = \begin{bmatrix}8672 & 19768 & 28302\\ 3072 & 6984 & 10932\\ 11408 & 26088 & 33004\end{bmatrix}$$

7) After the multiplication we represent the values contained inside the matrix into one long string of values.
Result: [4432, 10040, 17751, 5664, 12888, 19577, 10960, 25086, 30544]

We then XOR the value with the generated XOR key with the result which will form the ciphertext for the first block

XOR key: [6994, 54643, 45116, 34884, 3486, 41803, 44882, 40658, 26008]

Ciphertext for first block: [15026, 38987, 57010, 33876, 5874, 34890, 33730, 64314, 58740]

8) We modulo 256 each value of the ciphertext from the previous block (in this case block 1)
Values after modulo: [178, 75, 178, 68, 214, 255, 194, 58, 116]

The values after modulo will be XOR with plaintext of the next block (in this case block 2)

Modulo value $\oplus$ block 2 = [193, 42, 213, 33, 246, 151, 167, 72, 17]
We can represent this in a 3x3 matrix:

```
[193   42  213]
[ 49  210   34]
[167   72   17]
```

Then we shuffle the matrix.
Block 2 After shuffling:

```
[167   42  151]
[ 17  193   72]
[246   33  213]
```

Afterwards, we carry out block cipher encryption (matrix multiplication and XOR with key) on the 3x3 matrix calculated above.

We will perform matrix multiplication on the the XOR result first, [XOR result][encrypted matrix] = [result]

XOR Result Encrypted Matrix Result

$$\begin{bmatrix}167 & 42 & 151\\ 17 & 193 & 72\\ 246 & 33 & 213\end{bmatrix} \times \begin{bmatrix}16 & 36 & 77\\ 16 & 36 & 76\\ 32 & 74 & 50\end{bmatrix} = \begin{bmatrix}8176 & 18698 & 23601\\ 5664 & 12888 & 19577\\ 11280 & 25806 & 32100\end{bmatrix}$$

9) For the second block we will lay out the values contained inside the matrix into a string of values
Result: [3776, 8566, 14434, 10128, 23094, 32695, 6608, 14942, 27717]

We then XOR the value with the generated XOR key with the result which will form the ciphertext for the second block.

XOR key: [6994, 54643, 45116, 34884, 3486, 41803, 44882, 40658, 26008]

Ciphertext for second block: [5570, 62585, 34910, 34631, 22440, 56572, 46722, 42124, 2525]

10) We repeat steps 8 and 9 for block 3.

XOR Result Encrypted Matrix Result

$$\begin{bmatrix} 198 & 100 & 121 \\ 13 & 66 & 50 \\ 131 & 28 & 252 \end{bmatrix} \times \begin{bmatrix} 16 & 36 & 77 \\ 16 & 36 & 76 \\ 32 & 74 & 50 \end{bmatrix} = \begin{bmatrix} 8640 & 19682 & 28896 \\ 2864 & 6544 & 8517 \\ 10608 & 24372 & 24815 \end{bmatrix}$$

Ciphertext for third block: [1874, 60133, 55256, 40276, 8790, 62400, 45730, 55870, 23359]

11) Combined all the blocks of ciphertext into one

Combined ciphertext: [46, 199, 0, 187, 104, 160, 73, 92, 159, 15026, 38987, 57010, 33876, 5874, 34890, 33730, 64314, 58740, 2562, 62027, 62827, 40548, 16326, 61234, 34178, 65324, 4808, 1874, 60133, 55256, 40276, 8790, 62400, 45730, 55870, 23359]

12) Lastly we will convert the the ciphertext into hexadecimal and one more time into base85

Ciphertext to hexadecimal:
2ec700bb68a0495c9f3ab2984bdeb2845416f2884a83c2fb3ae5740a02f24bf56b9e643fc6ef328582ff2c12c80752eae5d7d89d542256f3c0b2a2da3e5b3f

Hexadecimal to base 85:
F2?}7XrM`4pE|ObOWv}CR2K4xN`u1tI^}c<0`g1sYo263#_uwPg8wWM$OlsD<=5DqR3cXMz_Oy+K3hK

Decryption

1. Using the Ciphertext above for base 85:
   F2?}7XrM`4pE|ObOWv}CR2K4xN`u1tI^}c<0`g1sYo263#_uwPg8wWM$OlsD<=5Dq
   R3cXMz_Oy+K3hK$_{85}$

   We decode the ciphertext to hexadecimal:
   2ec700bb68a0495c9f3ab2984bdeb2844416d689ff83c2fb3ae57404a29c79ec0d9e
   643fc6ef328342fa1c18fc3a929991c0dc8374140e820e8622c1e60577$_{16}$

2. Parse the key into encryption matrix and XOR key. Get the IV from the first 18 hexadecimal digits of the decoded ciphertext.

   IV = $2ec700bb68a0495c9f$ = $46, 199, 0, 187, 104, 160, 73, 92, 159$

3. Calculate the inverse of the encryption matrix using Gauss-Jordan Elimination.
   - Consider the encryption matrix called A = $\begin{bmatrix}16 & 36 & 77\\16 & 36 & 76\\32 & 74 & 50\end{bmatrix}$
   - Step 1: Augment a 3x3 identity matrix to the right of the encryption matrix, [A|I].

$$[A|I] = \left[\begin{array}{ccc|ccc}16 & 36 & 77 & 1 & 0 & 0\\16 & 32 & 76 & 0 & 1 & 0\\32 & 74 & 50 & 0 & 0 & 1\end{array}\right]$$

   Step 2: Perform row operations on the left side to be equals to the reduced echelon form, simply known as the identity matrix on the right side. It is important to note that we must perform the same row operations on both sides.

   1. We divide the elements in row 1 by 16 $R1 \leftarrow R1/16$

$$\left[\begin{array}{ccc|ccc}1 & \frac{9}{4} & \frac{77}{16} & \frac{1}{16} & 0 & 0\\16 & 36 & 76 & 0 & 1 & 0\\32 & 74 & 50 & 0 & 0 & 1\end{array}\right]$$

   2. Add (-16 x Row 1) to row 2 $R2 \leftarrow R2 + -16(R1)$

$$\left[\begin{array}{ccc|ccc}1 & \frac{9}{4} & \frac{77}{16} & \frac{1}{16} & 0 & 0\\0 & 0 & -1 & -1 & 1 & 0\\32 & 74 & 50 & 0 & 0 & 1\end{array}\right]$$

   3. Add (-32 x Row 1) to row 3 $R3 \leftarrow R3 + -32(R1)$

$$\left[\begin{array}{ccc|ccc}1 & \frac{9}{4} & \frac{77}{16} & \frac{1}{16} & 0 & 0\\0 & 0 & -1 & -1 & 1 & 0\\0 & 2 & -104 & -2 & 0 & 1\end{array}\right]$$

4. Swap row 3 with row 2 $R2 \leftrightarrow R3$

$$\left[\begin{array}{ccc|ccc} 1 & \frac{9}{4} & \frac{77}{16} & \frac{1}{16} & 0 & 0 \\ 0 & 2 & -104 & -2 & 0 & 1 \\ 0 & 0 & -1 & -1 & 1 & 0 \end{array}\right]$$

5. Divide row 2 by 2 $R2 \leftarrow R2/2$

$$\left[\begin{array}{ccc|ccc} 1 & \frac{9}{4} & \frac{77}{16} & \frac{1}{16} & 0 & 0 \\ 0 & 1 & -52 & -1 & 0 & 0.5 \\ 0 & 0 & -1 & -1 & 1 & 0 \end{array}\right]$$

6. Divide row 3 by -1 $R3 \leftarrow R3/-1$

$$\left[\begin{array}{ccc|ccc} 1 & \frac{9}{4} & \frac{77}{16} & \frac{1}{16} & 0 & 0 \\ 0 & 1 & -52 & -1 & 0 & 0.5 \\ 0 & 0 & 1 & 1 & -1 & 0 \end{array}\right]$$

7. Add (52 x row 3) to row 2 $R2 \leftarrow R2 + 52(R3)$

$$\left[\begin{array}{ccc|ccc} 1 & \frac{9}{4} & \frac{77}{16} & \frac{1}{16} & 0 & 0 \\ 0 & 1 & 0 & 51 & -52 & 0.5 \\ 0 & 0 & 1 & 1 & -1 & 0 \end{array}\right]$$

8. Add (-77/16 x row 3) to row 1 $R1 \leftarrow R1 + -77/16(R3)$

$$\left[\begin{array}{ccc|ccc} 1 & \frac{9}{4} & 0 & -\frac{19}{4} & \frac{77}{16} & 0 \\ 0 & 1 & 0 & 51 & -52 & 0.5 \\ 0 & 0 & 1 & 1 & -1 & 0 \end{array}\right]$$

9. Add (-9/4 x row 2) to row 1 $R1 \leftarrow R1 + -9/4(R2)$

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -\frac{239}{2} & \frac{1949}{16} & -\frac{9}{8} \\ 0 & 1 & 0 & 51 & -52 & 0.5 \\ 0 & 0 & 1 & 1 & -1 & 0 \end{array}\right] = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -119.5 & 121.8125 & -1.125 \\ 0 & 1 & 0 & 51 & -52 & 0.5 \\ 0 & 0 & 1 & 1 & -1 & 0 \end{array}\right]$$

   - This result on the right side is the inverse of the encryption matrix.

The inverse will be used later to reverse the matrix multiplication in encryption.

4. Separate message from ciphertext, split the message to four hexadecimals, convert to decimal and store the values into an array.

   Split by 4 hexadecimals:
   [3ab2,984b,deb2,8444,16d6,89ff,83c2,fb3a,e574,04a2,9c79,ec0d,9e64,3fc6,ef32,8342,fa1c,18fc,3a92,9991,c0dc,8374,140e,820e,8622,c1e6,0577]

   Convert values to decimal and store values into an array:
   [15026, 38987, 57010, 33860, 5846, 35327, 33730, 64314, 58740, 1186, 40057, 60429, 40548, 16326, 61234, 33602, 64028, 6396, 14994, 39313, 49372, 33652, 5134, 33294, 34338, 49638, 1399]

We can separate and determine each ciphertext block by dividing the decimal array into nine decimal values:
Block 1:[15026, 38987, 57010, 33860, 5846, 35327, 33730, 64314, 58740]
Block 2:[1186, 40057, 60429, 40548, 16326, 61234, 33602, 64028, 6396]
Block 3:[14994, 39313, 49372, 33652, 5134, 33294, 34338, 49638, 1399]

5. Afterwards, we can get the plaintext for each block (starting from first) by XORing each value in the array with our key first. We use the same key for each block.

   Our XOR key: [6994, 54643, 45116, 34884, 3486, 41803, 44882, 40658, 26008]

   Result:
   [8672, 19768, 28302, 3072, 6984, 10932, 11408, 26088, 33004]

   We can express the result in a 3x3 matrix as follows:

$$\begin{bmatrix} 8672 & 19768 & 28302 \\ 3072 & 6984 & 10932 \\ 11408 & 26088 & 33004 \end{bmatrix}$$

6. Next, we perform matrix multiplication on the 3x3 resultant matrix above with our inverse matrix.
   The result will be:

$$\begin{bmatrix} 8672 & 19768 & 28302 \\ 3072 & 6984 & 10932 \\ 11408 & 26088 & 33004 \end{bmatrix} \times \begin{bmatrix} -119.5 & 121.8125 & -1.125 \\ 51 & -52 & 0.5 \\ 1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 166 & 120 & 128 \\ 12 & 108 & 36 \\ 236 & 57 & 210 \end{bmatrix}$$

7. Likewise, we will unshuffle the values in the 3x3 resultant matrix positions. So that the value positions will be randomised. This is the same process as what we did after we perform matrix multiplication for encryption.

8. Lastly, we perform another XOR operation. The XOR operation will be between the IV and the result from matrix multiplication result above.

   The result will be:
   [86, 97, 108, 105, 100, 32, 109, 101, 115] = Block 1 decrypted

9. We repeat steps 5 to 7 for each cipher block. For the other blocks block in step 7, we substitute the IV with the modulo 256 value of the cipher values of the previous block to be the IV values instead.

10. We concat all the decrypted values together and convert each element in the resultant array to their ascii character and remove the padding as follow: 86, 97, 108, 105, 100, 32, 109, 101,115, 115, 97, 103, 101, 115, 32, 104, 101, 114, 101, 33 = Valid message here!

Research write-up

Cryptography in everyday life

Cryptography refer to methods of storing and transmitting data in a particular form so that only those for whom that is intended can read and process it.
Cryptography is something that we encounter probably everyday without even noticing its presence. It plays a role in ensuring the security of all digitalise services that we use on a daily basis such as cash withdrawal from an ATM, email and file storage.

Why Cryptography is so important

The purpose of cryptography is to protect the transmitted information from being read and understood by anyone except the intended receiver. Cryptography can be used to implement confidentiality and integrity. Firstly, cryptography protects the confidentiality of information. Even when the transmission medium is compromised, the encrypted message is rendered useless when an unauthorized personnel tries to access without proper key for decryption. Secondly, hashing algorithms (e.g MD5, sha-256) checks if messages have been altered by comparing hash checksums, ensuring data integrity.

Real life cryptography failures

In 2017, it was announced that Secure Hashing Algorithm 1 (SHA-1) would be unsafe to be used due to a hashing collision. A team of five Google researchers discovered this vulnerability when two pdf files of different content resulted in the same SHA-1 digest, this is also known as a collision where two different pieces of data hash to the same digest. A collision should never occur in a hashing algorithm because this removes the integrity of a message, as digests are compared to check if the contents of a message has been modified. Even though SHA-1 is deprecated, many applications still use SHA-1 such as file deduplication systems to HTTPS certificates used to protect online banking and other websites. This allow attackers to alter the contents of data, while others might think the altered copy is the original copy since both digests are the same.

How mathematics is used in cryptography

Math is the basis for cryptography, each algorithm comprise of several mathematical functions. An example is the RSA algorithm. The sender will have to use the recipient's public key to encrypt, and the recipient will have to use his private key to decrypt. The public key contains the modulus n and public exponent e. n is calculated from the multiplication of two random prime numbers called p and q, while e is a random number which is a coprime of the lowest common multiple of p-1 and q-1. A coprime simply means no common factors other than 1. The public key is then used in the encryption function as such, m^n mode, where m is the padded plaintext message.

References

Marc Stevens. 2017. Announcing the first SHA1 collision. [ONLINE] Available at:https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html

John Leyden. 2017. 'First ever' SHA-1 hash collision calculated. All it took were five clever brains... and 6,610 years of processor time. [ONLINE] Available at:https://www.theregister.co.uk/2017/02/23/google_first_sha1_collision/

Mark Gordon. 2014. *What is cryptography and why is it important?*. [ONLINE] Available at: https://www.quora.com/What-is-cryptography-and-why-is-it-important

Pseudo Random Number Generator (PRNG). [ONLINE] Available at: https://www.geeksforgeeks.org/pseudo-random-number-generator-prng/.

RSA (cryptosystem). [ONLINE] Available at:https://en.wikipedia.org/wiki/RSA_(cryptosystem)

Base 85 encoding. [ONLINE] Available at: http://www.tenminutetutor.com/data-formats/binary-encoding/ascii85-encoding/.

Linear congruential generator. [ONLINE] Available at:https://en.wikipedia.org/wiki/Pseudorandom_number_generator

Gaussian Elimination. [ONLINE] Available at:https://en.wikipedia.org/wiki/Gaussian_elimination

Block_cipher_mode_of_operation. 2017. *Block cipher mode of operation*. [ONLINE] Available at: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation.