**Problem1.**

The encryption method I want to talk about is Twofish, which is initially designed by by Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. It’s a symmetric key block cipher **whose block size is 128 bits and key size can be 128 bits, 192 bits or 256 bits**. Twofish is among the five AES finallist (the other five candidates are Rijndael, Serpent, RC6 and MARS) and is a very secure encryption protocol, since its encryption is theoretically safe from brute force attacks. One of the main reasons it loses to AES as the standard encryption method is that on most software platforms, its encryption speed for the 128-bit-size key is relatively slower than AES. There’re many products/softwares that use Twofish including GnuPG, KeePass, TrueCrypt, PGP and PasswordSafe.

One advantage of Twofish is its flexibility. It allows the users to customized their options based on the need. You can choose to take longer time to set up the key so that the encryption runs faster. This works well for encrypting large amounts of plaintext with the same key. You can also choose to set up the key faster, as a result of which the encryption becomes slower. This makes sense for encrypting many short blocks with changing keys.

As with DES, **Twofish is a Feistel network**, which means that in each round, half of the text block go through an f function and then do an XOR operation with the other half of the text block.

I will divide the two fish encryption process into three steps: Input Whitening, 16-rounds and Output Whitening. For each round of the 16-rounds, there are four steps: S-boxes, MDS Matrices, PHT and Addition mod 232. Now let’s look into each process for a 128-bit key.

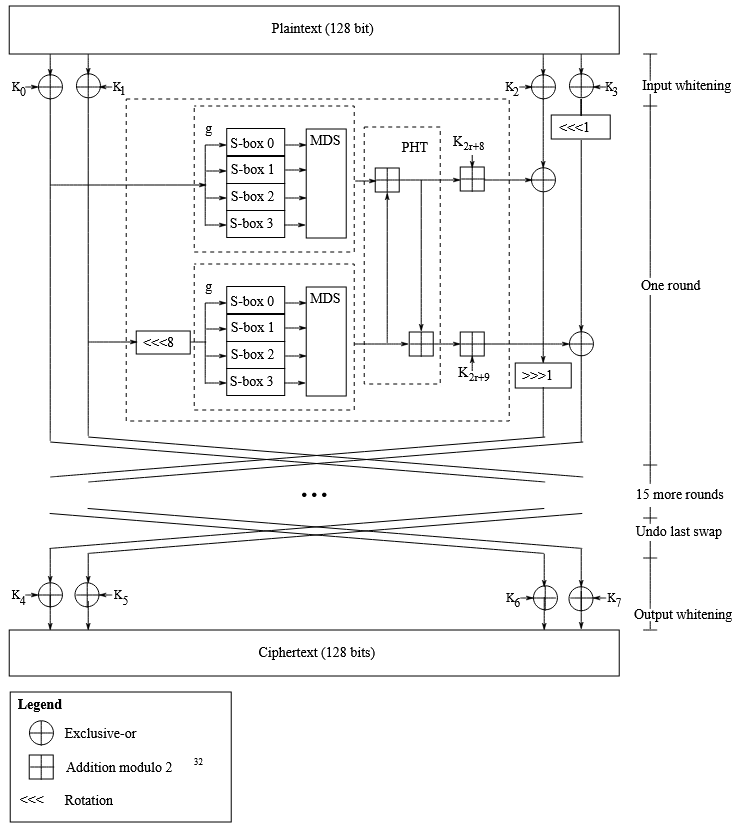
**Input Whitening:**

The 128-bit plaintext is divided into four 32-bit text and it will do XOR operation with a 128-bit key divided in the same way. We let each four parts of them be P0, P1, P2, P3 and K0, K1, K2, K3. After the XOR operation, the input becomes R0, R1, R2 and R3.

**16 rounds:**

During first round, R0, R1 will go in the f function to generate F0 and F1. F0 will XOR R2 and the result will be rotated right by one bit to generate C2. F1 will XOR R3 and the result will be rotated left by one bit to generate C3. The text at this point beomes R0, R1, C2, C3. Then we swap them to be C2, C3, R0, R1. This is the result of the first round. This result will go to the next round. Repeat this for the remain 15 rounds.

A image from wikipedia is helpful here(the notation in the following image is different from my text):



**The s-box is used to provide confusion.**

**The MDS matrix was carefully chosen to provide good diffusion for the plaintext.**

**The PHT and key addition provide diffusion between the subblocks and the key.**

Output Whitening:

The output whitening is similar to the input whitening, which is very easy, I will skip it here.

**Problem2.**

For Monoalphabetic cipher, letters in plaintext are mapped to letters in the ciphertext in a one-to-one relationship. So the letter distribution of the ciphertext is closer to the general English Letter Frequency. For Polyalphabetic cipher, letters in plaintext are mapped to letters in the ciphertext in a one-to-many relationship. So the letter distribution is more evenly distributed.

Thus, we can calculate the prequency of the letters in the ciphertext to determine the encryption is monoalphabetic or polyalphabetic.

To be more mathematically concise, for a text in ciphertext, the probability of “drawing” two same letters is: I = .082 .082 + .015 .015 + .028 .028 +… + .001 .001 = 0.0656010

As the distribution in a polyalphabetic is more uniform, the probability of “drawing” two same letters is: I = 1/26 \* 1/ 26 \*…\* 1/26 = 0.038

So we can calculate the prequency of each letter in the cipher text and calculate the value of I and see if I is closer to 0.0656010 or 0.038.

**Problem3.**

Step1:

I calculated the length of the key to be 11.

Step2:

calculated possible keys(part of the full keys) if the crib “think” is at different positions. (refer to the text file)

Step3:

Observe the patterns. The last key is most likely since the decripted text is more text-like. There’s a possible text parts: “AREWO\*\*\*\*\*\*TROUB”. I guess it to be AREWORTHTHETROUBLE.

From this Text, we can get the key to be “**PHILOSOPHER**”

Step4: Use this key to decrypt the cipherText and adding punctuation:

**I am myself inclined to think that deciphering is an affair of time, ingenuity and patience, and that very few ciphers are worth the trouble of unravelling them. One of the most singular characteristics of the art of deciphering is the strong conviction possessed by every person, even moderately acquainted with it, that he is able to construct a cipher which nobody else can decipher. I have also observed that the cleverer the person, the more intimate is his conviction. In my earliest study of the subject I shared in this belief, and maintained it for many years.**

Author of the passage: **Charles Babbage**

**Problem4.**

**=RSA=**

**Yu, Xilong:**

**56763037168054771025680132955599390663234435882793342069286521279333167943551**

**e: 641261**

**cm: 408994303563360699050726896700925651775**

**cs: 184343845647752000462226058066905613970**

**Problem5.**

**=MITM=**

**Yu, Xilong**

**DESkey: 271 315 152 49**

**Shift: 11 5 0 1 (correspondingly)**

**Problem6.**

**=ElGamal=**

**Yu, Xilong**

**y1 1051147401**

**y2 89256289134980**