Xi Liu, Assignment 1

1
key = 8
plaintext = attack with full force as soon as the sun rises

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char * shift_dec(int key, char * cipher)
{
    int n = strlen(cipher);
    char * ret = (char *)malloc((n + 1) * sizeof(char));
    for(int i = 0; i < n; ++i)
    {
        if('a' <= cipher[i] && cipher[i] <= 'z')
            ret[i] = ((cipher[i] - 'a' - key) % 26 + 26) % 26 + 'a';
        else if('A' <= cipher[i] && cipher[i] <= 'Z')
            ret[i] = ((cipher[i] - 'A' - key) % 26 + 26) % 26 + 'a';
        else
            ret[i] = cipher[i];
    }
    ret[n] = '\0';
    return ret;
}

int main()
{
    const char * s = "ibbiks eqbp nctt nwzkm ia awwv ia bpm acv zqama";
    for(int key = 1; key < 10; ++key)
    {
        char * ret = shift_dec(key, (char *)s);
        printf("key = %d, %s\n", key, ret);
        free(ret);
    }
}
```

2

not secure

suppose the keys are a, b, c, using those keys to encrypt 3 times is equivalent to encrypt 1 time using 1 key $a + b + c$, which can be decrypted by trying keys from 0 to 26 until the correct plain text is found

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char * shift_dec(int key, char * cipher); /* defined in question 1 */

char * shift_enc(int key, char * message)
{
    int n = strlen(message);
    char * ret = (char *)malloc((n + 1) * sizeof(char));
    for(int i = 0; i < n; ++i)
    {
        if('a' <= message[i] && message[i] <= 'z')
            ret[i] = (message[i] - 'a' + key) % 26 + 'a';
        else if('A' <= message[i] && message[i] <= 'Z')
            ret[i] = (message[i] - 'A' + key) % 26 + 'A';
        else
            ret[i] = message[i];
    }
    ret[n] = '\0';
    return ret;
}

int main()
{
    srand(time(0));
    int n = 3;
    int keys[n];
    for(int i = 0; i < n; ++i)
    {
        keys[i] = rand() % 26;
        printf("key[%d] = %d\n", i, keys[i]);
    }

    const char * message = "abc";
    char * ptr = shift_enc(*keys, (char *)message);
    for(int i = 1; i < n; ++i)
    {
        char * prev = ptr;
        ptr = shift_enc(keys[i], ptr);
        free(prev);
    }
    printf("encrypted ciphertext = %s\n", ptr);

    for(int key = 0; key < 26; ++key)
    {
        char * ret = shift_dec(key, (char *)ptr);
        printf("key = %d, %s\n", key, ret);
        free(ret);
    }
}
```

3

let n be the message length

then 1 way is to try all of 26^n possible permutations, but this is not practical when n is large

cannot use frequency analysis here since each shift only appears once, since key length = message length = n

4
7 multiplications are used
the numbers that I had to multiply have 2 digits

$$41 = 32 + 8 + 1 = 2^5 + 2^3 + 2^0$$

$$\begin{aligned}
3292213^2 \mod 100 &= 13^2 \mod 100 = 69\\
3292213^4 \mod 100 &= (3292213^2 \mod 100)^2 = 69^2 \mod 100 = 61\\
3292213^8 \mod 100 &= (3292213^4 \mod 100)^2 = 61^2 \mod 100 = 21\\
3292213^{16} \mod 100 &= (3292213^8 \mod 100)^2 = 21^2 \mod 100 = 41\\
3292213^{32} \mod 100 &= (3292213^{16} \mod 100)^2 = 41^2 \mod 100 = 81\\
3292213^{41} \mod 100 &= (3292213^{32} \mod 100)(3292213^8 \mod 100)(3292213^1 \mod 100)\\
&= (81)(21)(13) \mod 100\\
&= ((81)(21) \mod 100) * 13) \mod 100 = 1 * 13 \mod 100 = 13
\end{aligned}$$

5

$$
\begin{aligned}
y &= g^x \mod N \\
6 &= 5^x \mod 7
\end{aligned}
$$

use trial and error

$\boxed{\text{x = 3}}$

$5^3 = 125 = 17 * 7 + 6$

$5^3 \mod 7 = (17 * 7 + 6) \mod 7 = 6$

```
#include <stdio.h>
#include <math.h>

int main()
{
    for(int i = 0; i < 10; ++i)
    {
        float res = pow(5, i);
        printf("5^%d = %f, 5^%d %% 7 = %d\n", i, res, i, (int)res % 7);
    }
}
```