VE475 Intro to Cryptography Homework 1

Taoyue Xia, 518370910087

2020/05/20

1 Ex1

1. As Alice use the Caesar cipher, we can use the equation $x - \kappa \ mod \ 26$ to find the answer. After running the algorithm in python, the word "river" is the only word which can be recognized.
   So we know that Bob should meet Alice at the river, and the parameter κ is 13.

2. Let the key matrix K be a 2×2 matrix $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$, then use matrix inversion to calculate K.

$$\begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix} \cdot K = \begin{pmatrix} 4 & 11 \\ 13 & 8 \end{pmatrix} \ mod 26$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 4 & 11 \\ 13 & 8 \end{pmatrix} \ mod 26$$

   For the plain text matrix A $\begin{pmatrix} 3 & 14 \\ 13 & 19 \end{pmatrix}$, it has $det(A) = -125$ and $gcd(-125, 26) = 1$

So $A^{-1} = \frac{1}{-125}\begin{pmatrix} 19 & -14 \\ -13 & 3 \end{pmatrix}$ And it is calculated that 21 is the inverse of -125 modulo 26, such that we get:

$$A^{-1} = \begin{pmatrix} 399 & -294 \\ -273 & 63 \end{pmatrix} = \begin{pmatrix} 9 & 18 \\ 13 & 11 \end{pmatrix} \, mod \, 26$$

Then we put A^{-1} into the initial equation:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 9 & 18 \\ 13 & 11 \end{pmatrix} \cdot \begin{pmatrix} 4 & 11 \\ 13 & 8 \end{pmatrix} \, mod \, 26$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 270 & 243 \\ 195 & 231 \end{pmatrix} = \begin{pmatrix} 10 & 9 \\ 13 & 23 \end{pmatrix} \, mod \, 26$$

So we get the answer that the encryption matrix is $K = \begin{pmatrix} 10 & 9 \\ 13 & 23 \end{pmatrix}$.

3. As $n|ab$, we know that $gcd(ab, n) = n$. While $gcd(a, n) = 1$, we can deduce that $gcd(b, n) = n \ \Rightarrow \ n|b$, and the proof is done.

4.

$$\begin{aligned} 30030 \ \div \ 257 &= 116...218 \\ 257 \ \div \ 218 &= 1...39 \\ 218 \ \div \ 39 &= 5...23 \\ 39 \ \div \ 23 &= 1...16 \\ 23 \ \div \ 16 &= 1...7 \\ 16 \ \div \ 7 &= 2...2 \\ 7 \ \div \ 2 &= 3...1 \\ 2 \ \div \ 1 &= 2 \end{aligned}$$

According to the computation above, we find that $gcd(30030, 257) = 1$, which indicate that 30030 and 257 are relatively prime.

5. Set the OTP as k, and the first message as m1, the second message as m2, the first time ciphertext as c1, the second-time ciphertext as c2.

$$c1 = m1 \oplus k,\ c2 = m2 \oplus k$$
$$\Rightarrow c1 \oplus c2 = (m1 \oplus k) \oplus (m2 \oplus k) = m1 \oplus m2$$

Then attackers can use KPA to get the real key by multiple attacking, and finally get the key.

6. To be secure, it should has greater or equal to 2^{128} operations. Then the size n can be caluclated as following (choose the base of $log()$ function as 2):

$$\begin{aligned} 2^{O(\sqrt{nlog(n)})} &\geq 2^{128} \\ nlog(n) &\geq 128^2 \\ n &\geq 1546.43 \end{aligned}$$

Then choose n to be 1547, so the size of the graph should be at least 1547.

2 Ex2

1. The Vigenère cipher is partly like the Caesar cipher, it uses a random key to decide how much it will change from plain text to ciphertext. Then it will look up in a alphabet table to finally determine the ciper-text. What's more, if the key's length is shorter than the plain text, it will be repeated.

   For example, if the plain text is "goodmorning" and the key is "beautiful", then we extend the key to be "beautifulbe", then match the ciphertext for "g" and "b", "o" and "e", etc. Finally, after looking up in the alphabet table, we will finally get the ciphertext.

2. a) As Bob send the same letter several hundred times, after being encrypted, the ciphertext will look like a paragraph fulfilled with a repeated six-letter word, because the key is six-letter long.

   b) Because Bob send the same letter a lot of times, the ciphertext is periodic with length six, so the key is six-letter long.

   c) Eve just need to add six numbers to the corresponding number of the ciphertext, and try for 26 times, then she will find the final answer.

Ex3

```cpp
#include <iostream>
#include <cstdlib>
#include <gmpxx.h>
#include <gmp.h>
using namespace std;

unsigned long int extended(mpz_t a,mpz_t b,mpz_t& x,mpz_t& y) {
    mpz_t r, s, t;
    mpz_init(r);
    mpz_init_set_str(s, "1", 10);
    mpz_init(t);
    mpz_set_ui(x, 1);
    mpz_set_ui(y, 0);

    //gmp_printf("%Zd %Zd %Zd %Zd %Zd %Zd %Zd\n", a, b, x, y, m, n, t);
    mpz_t temp;
    mpz_init(temp);
    while(mpz_cmp_ui(b, 0) != 0){
        mpz_set(t, r);
        mpz_fdiv_q(temp, a, b);
        mpz_mul(temp, temp, r);
        mpz_sub(r, x, temp);
        mpz_set(x, t);

        mpz_set(t, s);
        mpz_fdiv_q(temp, a, b);
        mpz_mul(temp, temp, s);
        mpz_sub(s, y, temp);
        mpz_set(y, t);

        mpz_set(t, b);
        mpz_fdiv_r(b, a, b);
        mpz_set(a, t);
    }

    unsigned long int answer = mpz_get_ui(a);

    mpz_clear(r);
    mpz_clear(s);
    mpz_clear(t);
    mpz_clear(temp);
    return answer;
}

int main() {
    mpz_t a, b, c;
    string s1 = "", s2 = "";

    srand(time(0));
    for(int i = 0; i < 4096; i++) {
        if(rand() % 2 == 1){
            s1 += "1";
```

```
        }
        else s1 += "0";
    }
    const char *s11 = s1.c_str();
    srand(rand() % 10);
    for(int i = 0; i < 4096; i++) {
        if(rand() % 2 == 0){
            s2 += "0";
        }
        else s2 += "1";
    }

    const char* s22 = s2.c_str();
    mpz_init_set_str(a, s11, 2);
    mpz_init_set_str(b, s22, 2);
    mpz_init(c);

    gmp_printf("%Zd\n", a);
    gmp_printf("%Zd\n", b);
    mpz_gcd(c, a, b);   # use the gcd function in GMP to calculate
    gmp_printf("%Zd\n", c);

    mpz_t x, y;
    mpz_init(x);
    mpz_init(y);

    # use the Extended Euclidean Algorithm to calculate gcd
    unsigned long int gcd = extended(a, b, x, y);

    mpz_clear(a);
    mpz_clear(b);
    mpz_clear(c);
    mpz_clear(x);
    mpz_clear(y);
    cout << gcd << endl;
    //std::cout << s11 << endl;
    //cout << s22 << std::endl;
    return 0;
}

```

Above is the code for the realization of the Extended Euclidean Algorithm and the comparison between it and the gcd function inside the GMP library. The comparison is shown in the pictures below.

```
The first number:
392845262937047083353428506678705740589494530697745804469728648264490869670579532087515936548812903332205331077
435927601486488568123229445510066606622977850634058756511743199224980809196056395141115005526545609449233130564
397520670410423043428754797550181122639020285045743612137057451028764194692702890040314533733594541279892683384
392447818233435642675349696919586652954979314918632920369208772254854286506807269483624287269099246758006623794
817990683713731445961159510291196328894923410682250635606190261506152527662772441950583171971917267963780023 92
466374518078698289819551572430778440996493386070314250275883365343139288280179898636618976097871415919329546489
081895966615501506275520003776267971563535544940103435085655737898470124266119844894271820659685471162425807823
186020370921108289812024472751814163677175398145571078051955387037908164742106094579408517650971799300251172802
931326236691944725813951790088113783608532265611341124330617923546577189980702716530132843913903195015391892477
575151403207397773916360861644276372952551230134289016691701454584285828905019802387631391848746496751415026933
282229706122423917498542206351122140745396318372172777848152596916100730678549773815281105484872475735092427264
03992711624
The second Number:
869922893253455259261548234202348236930062507401147479017003721460808949776109465983757548752529414038716926532
095091595833632657098974490848488905107079380444227192031330856836688074186800978281126826693298930932408315049
674303780736033952378890573689453900138102355107936472796681333576811905522711370588575475799209452876442578691
797075656379643040704779464045037871587825342849878771189286177373463346494457888284133016549319146421356664997
539339722869249208878390912423151382562309741148381289216527200927762650122173381048412032319703636450763047773
424676480213993714417794391485205929501454978500337302985780792547989076964881784520168152534000621117116911271
178721177683887843642396013008622229768804044185051110045131407844880904938215309538679731145365521496356288343
294439687185267655245919254467782274965912244995360770578127547652784867364053542261830973353614727760969496917
307927665400540254965649644865909082989252312679342706066500766764086808033849242193008516501550872051264787535
754789222524723908637068807665247422404134539969233815016114029230814447494099051153207797408389737260173406932
558992234349240332114431833604481000138843258884162843978129111993677295349894498104140278836981702013410672492
057953603027
gcd using mpz_gcd():
1
gcd using Extended Euclidean Algorithm:
1
```

```
The first number:
998848986348987673750344403165846825269684498944587069307075435376535552947658849953670424592242425170614504421
313971397258952141168930124702093915338193544120081983999610561301403305042283915661224006678215057974591492088
159784788510480802667594416848197844922439284678186583654150405894002909728440160258047691585068094979637861987
679879657365083047382422095607262531363686817384596865047899381098927491215745883157467721294265797683129935615
888522612490005753386329818987267787493955646202517863843698812123079258360076922315451931364680437559755029 32
812951777682738879135214874059410326396532292388046108405416763372893534026391999168910922462733784594483427857
693442529366893351727681445909747496447161315552948398931108026771963847168901914166204925183160991321420790971
912312254130026534591293606787856157157801863293205184549982195818782196551444228305950618628148528194308185564
814537175659022814272074226624034537667195380193292631455303324373638464880802712709845868544097071504548017925
275633719847102044137053388953456869738738379885174949786081576041832699296986267942134142934130954941130065749
225839507004690611528921353566932571215182740394683643590632978640454451530177953607620628211372606114227964096
067387764933
The second Number:
869922893253455259261548234202348236930062507401147479017003721460808949776109465983757548752529414038716926532
095091595833632657098974490848488905107079380444227192031330856836688074186800978281126826693298930932408315049
674303780736033952378890573689453900138102355107936472796681333576811905522711370588575475799209452876442578691
797075656379643040704779464045037871587825342849878771189286177373463346494457888284133016549319146421356664997
539339722869249208878390912423151382562309741148381289216527200927762650122173381048412032319703636450763047773
424676480213993714417794391485205929501454978500337302985780792547989076964881784520168152534000621117116911271
178721177683887843642396013008622229768804044185051110045131407844880904938215309538679731145365521496356288343
294439687185267655245919254467782274965912244995360770578127547652784867364053542261830973353614727760969496917
307927665400540254965649644865909082989252312679342706066500766764086808033849242193008516501550872051264787535
754789222524723908637068807665247422404134539969233815016114029230814447494099051153207797408389737260173406932
558992234349240332114431833604481000138843258884162843978129111993677295349894498104140278836981702013410672492
057953603027
gcd using mpz_gcd():
7
gcd using Extended Euclidean Algorithm:
7
```

```
The first number:
678882466271504447176007628592781721993307024010813093877940862786235778823730799474868533210850514962129008904
529971134078044721752213301620058450410241560818768105831878320734915344817222014968592704652624119262585100422
876052604104119582361836234067829094267804142159664621840749021187029134392681294631225829810174553884248998783
995183414433868784966636590968105546270412087772889756346228113554488324313555193325837767657366505673516417997
132183575461038712132638474198807391266336336986120354608767658208038247121816754226716405630063793983275540780
691186910159661881677426486561150309910032412369726736551000995310227354278263225984847525961142671278613337925
186972610032707828254800993654499936510147218544277320233635140202373167950739635713703277885913255962339940294
885585370586581065256003382020183619916968074167572837455610087589019270134833347460342904002832663079755747718
172637820762465513210938012036301344628919474400494213721107834710527606485532358924555753648559278626294589711
120796902718527318752864290067896753061637197253786864780537225902732200218343221348578626965418784860140203193
082056156006515615513083002811782612084489317078391512093128794324656414163951178389145014483705774457075779059
406298228583
The second Number:
869922893253455259261548234202348236930062507401147479017003721460808949776109465983757548752529414038716926532
095091595833632657098974490848488905107079380444227192031330856836688074186800978281126826693298930932408315049
674303780736033952378890573689453900138102355107936472796681333576811905522711370588575475799209452876442578691
797075656379643040704779464045037871587825342849878771189286177373463346494457888284133016549319146421356664997
539339722869249208878390912423151382562309741148381289216527200927762650122173381048412032319703636450763047773
424676480213993714417794391485205929501454978500337302985780792547989076964881784520168152534000621117116911271
178721177683887843642396013008622229768804044185051110045131407844880904938215309538679731145365521496356288343
294439687185267655245919254467782274965912244995360770578127547652784867364053542261830973353614727760969496917
307927665400540254965649644865909082989252312679342706066500766764086808033849242193008516501550872051264787535
754789222524723908637068807665247422404134539969233815016114029230814447494099051153207797408389737260173406932
558992234349240332114431833604481000138843258884162843978129111993677295349894498104140278836981702013410672492
057953603027
gcd using mpz_gcd():
3
gcd using Extended Euclidean Algorithm:
3
```

It is obvious that the value is always the same, which means that the Extended Euclidean Algorithm we implement has the same effect as the gcd function the GMP library has.