Practice 1

Classical cryptosystems

Vigenere and Affine cipher

Session lab 1 | Monday, february 3

By Joel Harim Hernández Javier

Description

In this session we will work with substitution ciphers. You must write your programs in C/C++. Do the following programming exercises on your own.

Requirements

1. Design a function to encrypt using the Vigenère cipher. The function must receive the plaintext, and the key. The output must be the ciphertext.

2. Design a function to decrypt using the Vigenère cipher. The function must receive the cipher-text, and the key. The output must be the plaintext.

3. Design a function to verify that a candidate key for the affine cipher is a valid key.

4. Design a function that receives a valid key for the affine cipher $K = (a, b)$ and n and calculate $a^{-1} \ mod \ n$.

5. Design a function to encrypt using the affine cipher. The function must receive the plaintext, and a valid key. The output must be the ciphertext.

6. Design a function to decrypt using the affine cipher. The function must receive the ciphertext, and the key. The output must be the plaintext.

Restrictions

- ➔ The alphabet must be chosen by the user. This implies that the language for the plaintext could be other than English.
- ➔ To encrypt and decrypt you must use modular arithmetic.
- ➔ The plaintext must be stored in a textfile and the name of this file must not be fixed.
- ➔ The ciphertext must be stored in a textfile and the name of the file must be the same of the plaintext but adding the extension .vig or .aff depending on the encryption method you used Vigenère or affine cipher respectively.
- ➔ The key can be chosen by the user or can be randomly generated by your program. Your program must offer both options.
- ➔ You could encrypt blank spaces.
- ➔ Your program must work with text files of at least 5Kb.

1.

In `vigenere.hpp`

```
    // Encrypt the given file using the alphabet and the key provided
    inline
    std::string encrypt(std::string& message, std::string& alphabet, std::string
key, bool spaces){
        std::string s;

        if ((s = belongsTo(message, alphabet, spaces)) != ""){
            error("The message provided doesn't belong to given alphabet's Kleene
closure: \"" + s + "\"");
        }

        if ((s = belongsTo(key, alphabet, true)) != ""){
            error("The key provided doesn't belong to given alphabet's Kleene
closure: " + s);
        }

        auto msg = utf8::trim(message);
        auto alph = utf8::trim(alphabet);
        auto k = keyIndexes(utf8::trim(key), alph);

        std::string r = "";
        r.reserve(message.size());

        int j = 0;

        for (int i = 0; i < msg.size(); ++i){
            if (!spaces && msg[i] == " "){
                r += " ";
                continue;
            }

            r += alph[(indexOf(msg[i], alph) + k[j++]) % alph.size()];
            j %= k.size();
        }

        return r;
    }
```

2.

In `vigenere.hpp`

```
    // Decrypt the given file using the alphabet and the key provided
    inline
    std::string decrypt(std::string& message, std::string& alphabet, std::string
key, bool spaces){
        std::string s;

        if ((s = belongsTo(message, alphabet, spaces)) != ""){
            error("The message provided doesn't belong to given alphabet's Kleene
closure: \"" + s + "\"");
        }

        if ((s = belongsTo(key, alphabet, true)) != ""){
            error("The key provided doesn't belong to given alphabet's Kleene
closure: " + s);
        }

        auto msg = utf8::trim(message);
        auto alph = utf8::trim(alphabet);
        auto k = keyIndexes(utf8::trim(key), alph);

        std::string r = "";
        r.reserve(message.size());

        int j = 0;

        for (int i = 0; i < msg.size(); ++i){
            if (!spaces && msg[i] == " "){
                r += " ";
                continue;
            }

            r += alph[mod((indexOf(msg[i], alph) - k[j++]), alph.size())];
            j %= k.size();
        }

        return r;
    }
```

3.

In `affine.hpp`

```
// Check if a key is valid
inline
bool isValidKey(int k, int n){
    return extEuclideanAlg(n, k) < 0 ? false : true;
}
```

4.

In `affine.hpp`

```
// Calculate the modular inverse of b mod a
// Return -1 if a, b aren't coprimes.
// O(log(min(a, b)))
inline
int extEuclideanAlg(int a, int b){
    // int s = 0, s0 = 1;
    int t = 1, t0 = 0;
    int r = b, r0 = a;
    int aux, q = 0;

    while (r) {
        q = r0 / r;
        aux = r;
        r = r0 - q * r;
        r0 = aux;
        aux = t;
        t = t0 - q * t;
        t0 = aux;
    }

    // Bezeout coefficients: s, t | a*s + b*t = gcd(a,b)
    // Thus, t0 is the b^-1 mod a if gcd(a,b) = 1
    // Source and explanation:
    // https://es.wikipedia.org/wiki/Algoritmo_de_Euclides#
       Algoritmo_de_Euclides_extendido

    return r0 != 1 ? -1 : mod(t0, a);
}
```

5.

In `affine.hpp`

```
    // Encrypt the given file using the alphabet and the key provided
    inline
    std::string encrypt(std::string& message, std::string& alphabet, std::tuple<int,
int> key, bool spaces){
        if (std::string s; (s = belongsTo(message, alphabet, spaces)) != ""){
            error("The message provided doesn't belong to given alphabet's Kleene
closure: \"" + s + "\"");
        }

        std::string r = "";
        r.reserve(message.size());

        auto msg = utf8::trim(message);
        auto alph = utf8::trim(alphabet);
        auto [a, b] = key;

        if (!isValidKey(a, alph.size())){
            std::string m = "Invalid key, the a isn't coprime of the alphabet size:
gcd(";
            m += std::to_string(a);
            m.append(", ");
            m += std::to_string(alph.size());
            m.append(") != 1");
            error(m);
        }

        for (int i = 0; i < msg.size(); ++i){
            if (!spaces && msg[i] == " "){
                r += " ";
                continue;
            }

            r += alph[(a * indexOf(msg[i], alph) + b) % alph.size()];
        }

        return r;
    }
```

6.

In `affine.hpp`

```
    // decrypt the given file using the alphabet and the key provided
    inline
    std::string decrypt(std::string& message, std::string& alphabet, std::tuple<int,
int> key, bool spaces){
        if (std::string s; (s = belongsTo(message, alphabet, spaces)) != ""){
            error("The message provided doesn't belong to given alphabet's Kleene
closure: \"" + s + "\"");
        }

        std::string r = "";
        r.reserve(message.size());

        auto msg = utf8::trim(message);
        auto alph = utf8::trim(alphabet);
        auto [a, b] = key;
        auto inverseA = extEuclideanAlg(alph.size(), a);

        for (int i = 0; i < msg.size(); ++i){
            if (!spaces && msg[i] == " "){
                r += " ";
                continue;
            }

            r += alph[mod(inverseA * (indexOf(msg[i], alph) - b), alph.size())];
        }

        return r;
    }
```

Design

Each program (vigenere, affine) was build with under linux command usage in mind. In order to simplify the user experience, a .json file must be provided to use each one.

Also, take advantage of the utf8 infrastructure build inside the UNIX systems, like Linux, allowing the user to use the alphabet of their choice.

Vigenere:

```
Cryptography/P01/vigenere
→ ./vigenere -h

Encryp/Decrypt utf8 messages using Vigenere cipher

JSON file structure:
    * alphabet                The alphabet to use
    * key                     The key to use to encrypt/decrypt
    + messageFile             Path to the clear message file
    + encryptedMessageFile    Path to the encrypted message file
      spaces                  Forces to encrypt using spaces

* required.
+ Either is required

Usage:
  ./vigenere [OPTION...]

  -e, --encrypt [JSON file]  Encrypt using the given options in the .json
                             file
  -d, --decrypt [JSON file]  Decrypt using the given options in the .json
                             file
  -h, --help                 Print help
```

Vigenere .json file example

```
{
    "alphabet": "abcdefghijklmnñopqrstuvwxyz",
    "key": "poder",
    "messageFile": "message.txt",
    "encryptedMessageFile": "message.vig",
    "spaces": false
}
```

Affine:

```
Cryptography/P01/affine
→ ./affine -h

Encryp/Decrypt utf8 messages using Affine cipher

JSON file structure:
    * alphabet                The alphabet to use
    * key                     The key to use to encrypt/decrypt
        + a                   The a value
          b                   The b value
        + random              Or if you want use a random key.
    + messageFile             Path to the clear message file
    + encryptedMessageFile    Path to the encrypted message file
      spaces                  Forces to encrypt using spaces

* required.
+ Either is required

Usage:
  ./vigenere [OPTION...]

  -e, --encrypt [JSON file]  Encrypt using the given options in the .json
                             file
  -d, --decrypt [JSON file]  Decrypt using the given options in the .json
                             file
  -h, --help                 Print help
```

Affine .json file example:

```
{
    "alphabet": "abcdefghijklmnñopqrstuvwxyz",
    "messageFile": "message.txt",
    "encryptedMessageFile": "message2.aff",
    "spaces": false,
    "key": {
        "a": 19,
        "b": 7,
        "random": true
    }
}
```

Usage

Vigenere:

Small file:

```
hola mundo
```

.json file

```
{
   "alphabet": "abcdefghijklmnñopqrstuvwxyz",
   "key": "poder",
   "messageFile": "message.txt",
   "encryptedMessageFile": "message.vig",
   "spaces": false
}
```

Use:

Output:

```
wdñe dkbgs
```

Large file:

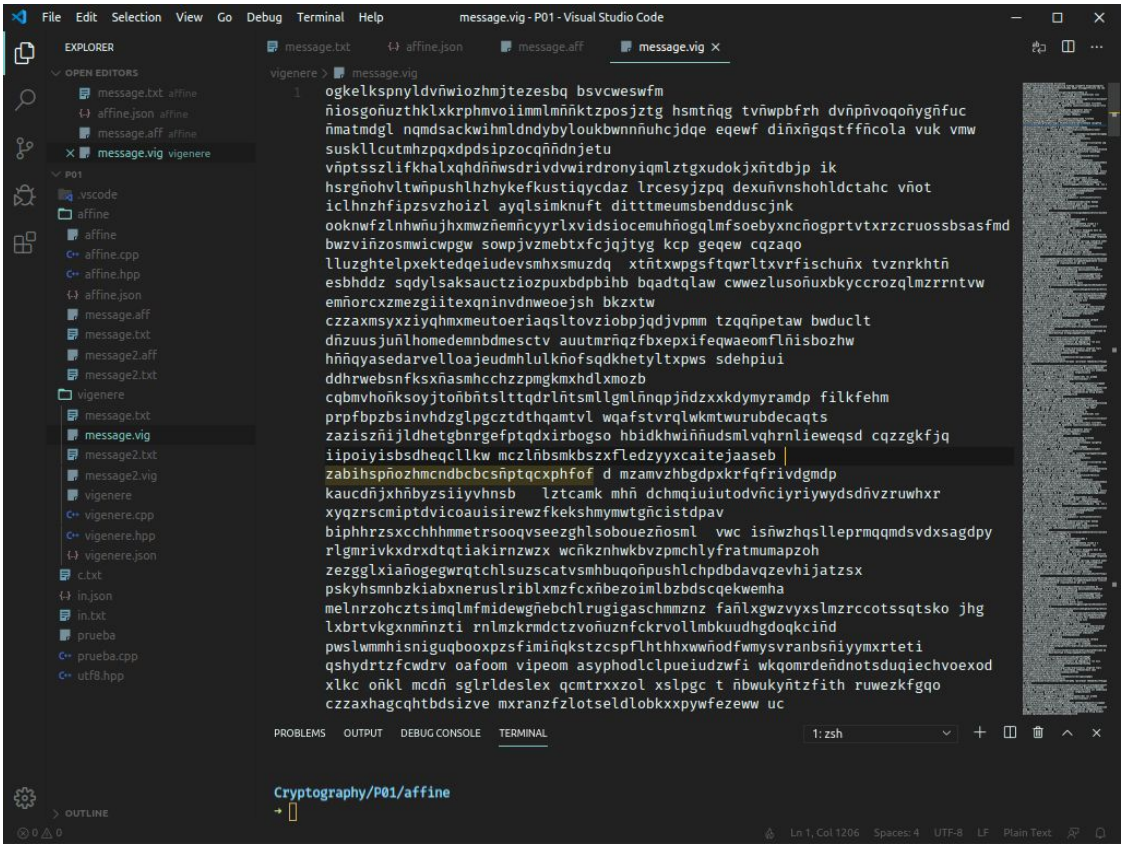
```
lorem ipsum dolor sit amet consectetur adipiscing elit sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua ut enim ad minim veniam quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat duis aute
irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur excepteur sint occaecat cupidatat non proident sunt in culpa qui officia
deserunt mollit anim id est laborum
lorem ipsum dolor sit amet consectetur adipiscing elit sed do eiusmod tempor ...
```

.json file

```
{
   "alphabet": "abcdefghijklmnñopqrstuvwxyz",
   "key": "esta llave es muy pequeña comparada con la longitud del mensaje",
   "messageFile": "message.txt",
   "encryptedMessageFile": "message.vig",
   "spaces": true
}
```

Use:

Output:



Affine:

Small file:

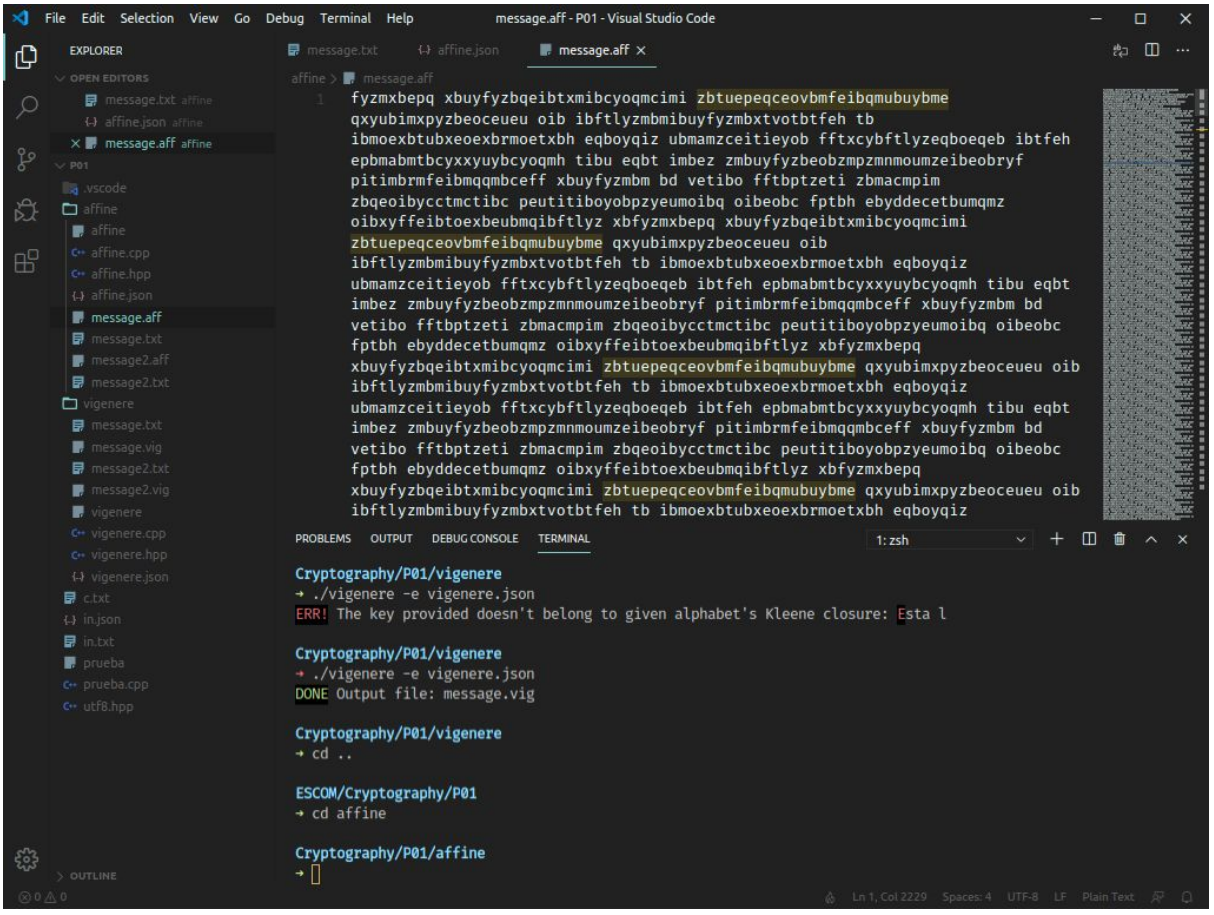
```
lorem ipsum dolor sit amet consectetur adipiscing elit sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua ut enim ad minim veniam quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat duis aute
irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
pariatur excepteur sint occaecat cupidatat non proident sunt in culpa qui officia
deserunt mollit anim id est laborum
lorem ipsum dolor sit amet consectetur adipiscing elit sed do eiusmod tempor ...
```

.json file

```
{
   "alphabet": "abcdefghijklmnñopqrstuvwxyz",
   "messageFile": "message.txt",
   "encryptedMessageFile": "message2.aff",
   "spaces": false,
   "key": {
       "random": true
   }
}
```

Use:

Output:


File Edit Selection View Go Debug Terminal Help
message.aff - P01 - Visual Studio Code
EXPLORER
OPEN EDITORS
message.txt affine
affine.json affine
message.aff affine
P01
.vscode
affine
affine
affine.cpp
affine.hpp
affine.json
message.aff
message.txt
message2.aff
message2.txt
vigenere
message.txt
message.vig
message2.txt
message2.vig
vigenere
vigenere.cpp
vigenere.hpp
vigenere.json
c.txt
in.json
in.txt
prueba
prueba.cpp
utf8.hpp
OUTLINE
message.txt
affine.json
message.aff
affine > message.aff
fyzmxbepq xbuyfyzbqeibtxmibcyoqmcimi zbtuepeqceovbmfeibqmubuybme
qxyubimxpyzbeoceueu oib ibftlyzmbmibuyfyzmbxtvotbtfeh tb
ibmoexbtubxeoexbrmoetxbh eqboyqiz ubmamzceitieyob fftxcybftlyzeqboeqeb ibtfeh
epbmabmtbcyxxyuybcyoqmh tibu eqbt imbez zmbuyfyzbeobzmpzmnmoumzeibeobryf
pitimbrmfeibmqqmbceff xbuyfyzmbm bd vetibo fftbptzeti zbmacmpim
zbqeoibycctmctibc peutitiboyobpzyeumoibq oibeobc fptbh ebyddecetbumqmz
oibxyffeibtoexbeubmqibftlyz xbfyzmxbepq xbuyfyzbqeibtxmibcyoqmcimi
zbtuepeqceovbmfeibqmubuybme qxyubimxpyzbeoceueu oib
ibftlyzmbmibuyfyzmbxtvotbtfeh tb ibmoexbtubxeoexbrmoetxbh eqboyqiz
ubmamzceitieyob fftxcybftlyzeqboeqeb ibtfeh epbmabmtbcyxxyuybcyoqmh tibu eqbt
imbez zmbuyfyzbeobzmpzmnmoumzeibeobryf pitimbrmfeibmqqmbceff xbuyfyzmbm bd
vetibo fftbptzeti zbmacmpim zbqeoibycctmctibc peutitiboyobpzyeumoibq oibeobc
fptbh ebyddecetbumqmz oibxyffeibtoexbeubmqibftlyz xbfyzmxbepq
xbuyfyzbqeibtxmibcyoqmcimi zbtuepeqceovbmfeibqmubuybme qxyubimxpyzbeoceueu oib
ibftlyzmbmibuyfyzmbxtvotbtfeh tb ibmoexbtubxeoexbrmoetxbh eqboyqiz
ubmamzceitieyob fftxcybftlyzeqboeqeb ibtfeh epbmabmtbcyxxyuybcyoqmh tibu eqbt
imbez zmbuyfyzbeobzmpzmnmoumzeibeobryf pitimbrmfeibmqqmbceff xbuyfyzmbm bd
vetibo fftbptzeti zbmacmpim zbqeoibycctmctibc peutitiboyobpzyeumoibq oibeobc
fptbh ebyddecetbumqmz oibxyffeibtoexbeubmqibftlyz xbfyzmxbepq
xbuyfyzbqeibtxmibcyoqmcimi zbtuepeqceovbmfeibqmubuybme qxyubimxpyzbeoceueu oib
ibftlyzmbmibuyfyzmbxtvotbtfeh tb ibmoexbtubxeoexbrmoetxbh eqboyqiz
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: zsh
Cryptography/P01/vigenere
→ ./vigenere -e vigenere.json
ERR! The key provided doesn't belong to given alphabet's Kleene closure: Esta l
Cryptography/P01/vigenere
→ ./vigenere -e vigenere.json
DONE Output file: message.vig
Cryptography/P01/vigenere
→ cd ..
ESCOM/Cryptography/P01
→ cd affine
Cryptography/P01/affine
→
Ln 1, Col 2229 Spaces: 4 UTF-8 LF Plain Text

The complete source code can be found at my personal Github:
github.com/JoelHernandez343