

# Practice 2

## Permutations

### SDES with CBC and character permutation

#### Session lab 2 | Monday, march 23

By Joel Harim Hernández Javier

#### Description

In this session we will work with permutation ciphers. Please do the following programming exercises.

#### Programming Exercises

1. Design a program in your favorite programming language to encrypt and decrypt using the permutation cipher. Consider the following requirements.
  - a. The key i.e. the permutation must be chosen by the user.
  - b. The inverse permutation must be calculated by your program.
  - c. Your program must work with text files of any size, at least 5Kb.
2. Implement simplified DES, considering the following requirements
  - a. The key must be randomly generated.
  - b. Your program must be able to encrypt and decrypt.
3. Choose at least one mode of operation different from ECB joint with your implementation for simplified DES (CBC chosen) to encrypt and decrypt a file. Please remember that the IV must be randomly chosen.

#### Products

- The most important parts of the source code.
- Include screen capture of your programs showing how they work.
- Please write a small user manual to know how to run your programs.

## Simple char permutation

In permutation.hpp

```
// Permute the vector set with the permutation provided
template<typename T>
inline
auto permute(std::vector<T> const& set, std::vector<int> permutation){

    std::vector<T> r (permutation.size());
    for (int i = 0; i < r.size(); ++i)
        r[i] = set[permutation[i]];

    return r;

}
```

## Inverse of the permutation

In permutation.hpp

```
// Calculate the inverse permutation
inline
auto inversePermutation(std::vector<int> permutation){

    if (!hasInverse(permutation))
        return std::vector<int>(0);

    std::vector<int> r (permutation.size());
    for (int i = 0; i < r.size(); ++i)
        r[permutation[i]] = i;

    return r;

}
```

# Encrypting / Decrypting a std::string

In permutation.hpp

```
// Encrypt / Decrypt the given utf8 string with the permutation.
// Autocomplete with 0's
inline
auto encrypt(std::string input, std::vector<int> perm){

    auto in = utf8::trim(input);

    int d = (perm.size() - (in.size() % perm.size())) % perm.size();

    if (d)
        std::cout << "\033[40;33mWARN\033[0m Added " << d << " 0's to end\n";

    for (int i = 0; i < d; ++i)
        in.push_back("0");

    std::string out = "";
    out.reserve(in.size());

    for (int i = 0; i < in.size(); i += perm.size()){

        std::vector<std::string> aux(perm.size());
        for (int j = 0; j < perm.size(); ++j)
            aux[j] = in[j + i];

        aux = permute(aux, perm);

        for (int j = 0; j < perm.size(); ++j)
            out += aux[j];

    }

    return out;
}
```

## Simple Bit permutation

In sdes.hpp

```
// Permute the unsigned primitive c's bits with the permutation perm and return it
unsigned int permute(unsigned int n, int used, std::vector<char> & perm){

    unsigned int result = 0;
    short current;

    for (int i = 0; i < perm.size(); ++i){

        current = used - 1 - perm[i];

        result <<= 1;

        if (n & (1 << current))
            result |= 1;

    }

    return result;

}
```

## Generate random bits

In helpers.hpp

```
// Generate an array of length `length` fill w/ random bits
inline
unsigned int randomBits(int length){
    unsigned int r = 0;
    for (int i = 0; i < length; ++i){

        r <<= 1;
        r |= rand() % 2;

    }
    return r;
}
```

## Generate keys of SDES

In sdes.hpp

```
// Generate k1 and k2 with permutation p10 and compression p8.  
// Returned within a tuple, it can be used, for example:  
//  
// auto [k1, k2] = generateKeys(...);  
//  
// This syntax is supported in C++17 and beyond.  
inline  
auto generatekeys(unsigned short key, std::vector<char> & p10, std::vector<char>  
& p8){  
  
    unsigned char k1, k2;  
  
    key = bitperm::permute(key, 10, p10);  
    key = circularLeftShift(key, 1);  
    k1 = bitperm::permute(key, 10, p8);  
  
    key = circularLeftShift(key, 2);  
    k2 = bitperm::permute(key, 10, p8);  
  
    return std::tuple(k1, k2);  
  
}
```

## Round of SDES implementation

In sdes.hpp

```
// Implementation of a generic sdes round.  
inline  
auto round(unsigned char input, std::vector<char> & exp, unsigned char key,  
std::vector<std::vector<char>> s0, std::vector<std::vector<char>> s1,  
std::vector<char> & p4){  
  
    auto e = key ^ bitperm::permute(input, 4, exp);  
  
    int i = (e & 0x80) >> 6 | (e & 0x10) >> 4;  
    int j = (e & 0x40) >> 5 | (e & 0x20) >> 5;  
  
    int x = (e & 0x08) >> 2 | (e & 0x01);
```

```

    int y = (e & 0x04) >> 1 | (e & 0x02) >> 1;

    auto aux = s0[i][j] << 2 | s1[x][y];

    aux = bitperm::permute(aux, 4, p4);
    aux &= 0xF;

    aux ^= input >> 4;

    return aux << 4 | input & 0xF;

}

```

## The main process of encryption / decryption of SDES

In sdes.hpp

```

// Encrypt / Decrypt, since, internally, are the same algorithm.
inline
auto crypt(unsigned char input, std::vector<char> & ip, std::vector<char> & exp,
std::vector<std::vector<char>> s0, std::vector<std::vector<char>> s1,
std::vector<char> & p4, unsigned short key, std::vector<char> & p10,
std::vector<char> & p8, bool encrypt){

    auto [k1, k2] = generatekeys(key, p10, p8);
    auto iip = bitperm::inversePermutation(ip);

    input = bitperm::permute(input, 8, ip);

    input = round(input, exp, encrypt ? k1 : k2, s0, s1, p4);
    input = ((input << 4) & 0xF0) | ((input >> 4) & 0x0F);
    input = round(input, exp, encrypt ? k2 : k1, s0, s1, p4);

    return bitperm::permute(input, 8, iip);

}

```

## CBC implementation – Encryption

In sdes.hpp

```
inline
auto encryptCBC(std::string & input, unsigned char iv, std::vector<char> & ip,
std::vector<char> & exp, std::vector<std::vector<char>> s0,
std::vector<std::vector<char>> s1, std::vector<char> & p4, unsigned short key,
std::vector<char> & p10, std::vector<char> & p8){

    std::string output = "";
    output.reserve(input.size());

    unsigned char prev = iv;

    for (int i = 0; i < input.size(); ++i){

        prev ^= input[i];
        prev = sdes::encrypt(prev, ip, exp, s0, s1, p4, key, p10, p8);
        output += prev;

    }

    return output;

}
```

## CBC implementation – Decryption

In sdes.hpp

```
inline
auto decryptCBC(std::string & input, unsigned char iv, std::vector<char> & ip,
std::vector<char> & exp, std::vector<std::vector<char>> s0,
std::vector<std::vector<char>> s1, std::vector<char> & p4, unsigned short key,
std::vector<char> & p10, std::vector<char> & p8){

    std::string output = "";
    output.reserve(input.size());

    unsigned char prev = iv;

    for (int i = 0; i < input.size(); ++i){
```

```
        output += prev ^ sdes::decrypt(input[i], ip, exp, s0, s1, p4, key, p10,  
p8);  
        prev = input[i];  
    }  
  
    return output;  
  
}
```



## Usage

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.**

## Permutation:

```
Cryptography/P02/permutation/build on master [!]  
→ ./permutation --help  
Encrypt / Decrypt files using permutation  
Usage:  
  permute [OPTION...]  
  
  -e, --encrypt [JSON FILE]    Encrypt using given options in .json file.  
  -d, --decrypt [JSON FILE]    Decrypt using given options in .json file.  
      --json [all | example | desc]  
                                View a .json example and/or its description.  
  -h, --help                    Print this help.
```

permutation.json file example

```
{  
  "permutation": [3, 2, 1, 0],  
  "encrypt": {  
    "inputFile": "in.txt",  
    "outputFile": "out.txt"  
  },  
  "decrypt": {  
    "inputFile": "out.txt",  
    "outputFile": "in.txt"  
  }  
}
```

Text:

Esto no es un poema, esto es un mensaje a ti, a la persona que ya no me piensa, a la persona que ya no me sueña, a la persona que ya no me ama. Estoy aquí, recordándote aún, pues en mi corazón tus recuerdos siguen vívidos como el amanecer.

Use:

```
Cryptography/P02/permutation/build on master [!]  
→ ./permutation --encrypt permutation.json  
WARN Added 1 0's to end  
DONE Output file: out.txt  
  
Cryptography/P02/permutation/build on master [!]  
→
```

Output:

otsE on u seop n,ametse se o nu snem ejait a a ,p alosreq any euon a em neip ,asal arep  
anoseuq ay m onus e,añel a ep anosruq aay e on a em .amotsEqa y ,íuocernádretohúa up  
,e seim nroc nózasut cer dreus soeugiív nodivoc se omma lcena0.re

### SDES:

```
Cryptography/P02/sdes/build on master [!]  
→ ./sdes --help  
Encrypt / Decrypt files using Simplified DES and CBC mode (for now)  
  
Usage:  
sdes [OPTION...]  
  
-e, --encrypt [JSON FILE]    Encrypt using given options in .json file.  
-d, --decrypt [JSON FILE]    Decrypt using given options in .json file.  
    --json [all | example | desc]  
                                View a .json example and/or its description.  
-h, --help                    Print this help.
```

sdes.json file example:

```
{  
  "iv": "01100101",  
  "encrypt": {  
    "inputFile": "message.txt",  
    "outputFile": "message.sdes"  
  },  
  "decrypt": {  
    "inputFile": "message.sdes",  
    "outputFile": "message.txt"  
  },  
}
```

```

"keyConfig": {
  "random": false,
  "key": "1010000010",
  "p10": [2, 4, 1, 6, 3, 9, 0, 8, 7, 5],
  "p8": [5, 2, 6, 3, 7, 4, 9, 8]
},
"cryptConfig": {
  "initialPermutation": [1, 5, 2, 0, 3, 7, 4, 6],
  "expansion": [3, 0, 1, 2, 1, 2, 3, 0],
  "p4": [1, 3, 2, 0],
  "s0": [
    [1, 0, 3, 2],
    [3, 2, 1, 0],
    [0, 2, 1, 3],
    [3, 1, 3, 2]
  ],
  "s1": [
    [0, 1, 2, 3],
    [2, 0, 1, 3],
    [3, 0, 1, 0],
    [2, 1, 0, 3]
  ]
}
}

```

Text:

```

File Edit Selection View Go Run Terminal Help message.txt - P02 - Visual Studio C...
permutation.json message.txt x sdes.json message.sdes
sdes > build > message.txt
1 Esto no es un poema, esto es un mensaje a ti, a la persona
que ya no me piensa, a la persona que ya no me sueña, a la
persona que ya no me ama. Estoy aquí, recordándote aún,
pues en mi corazón tus recuerdos siguen vívidos como el
amanecer.

```

Use:

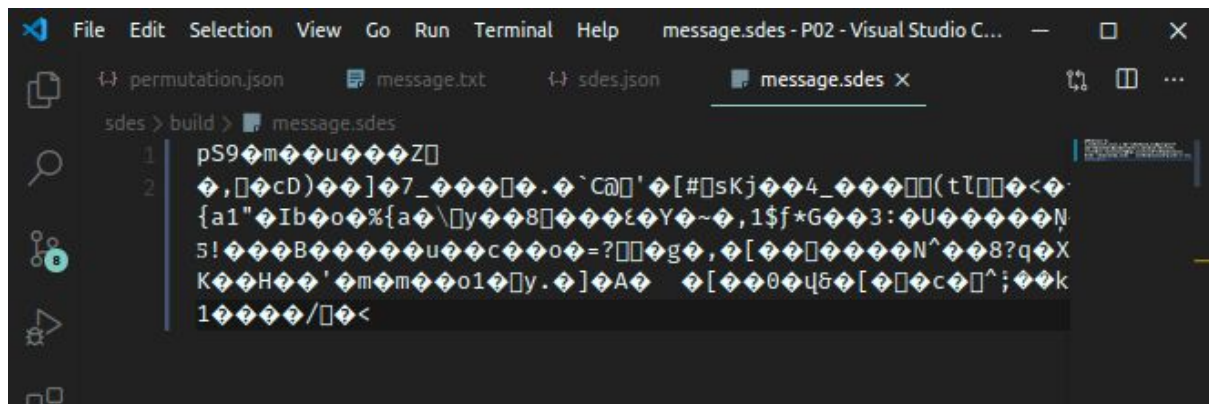
```

Cryptography/P02/sdes/build on master [!]
→ ./sdes --encrypt sdes.json
WARN Using the random IV: 1 1 1 0 1 0 0 0
DONE Output file: message.sdes

Cryptography/P02/sdes/build on master [!]
→

```

Output:



```
sdes > build > message.sdes
1 pS9m??u???Z
2 ,[]cD)??]7_???.?`C@'[##sKj??4_???(t[]?<?
{a1"?Ib?o?%{a?\[]y??8[]??Y?~?,1$f*G??3:U??N
5!??B??u??c??o=?[]g?,?[??N^??8?q?X
K??H??'m?m??o1[]y.?)?A? [??0?q?[]c?^;?k
1??/?<
```

The complete source code can be found at my personal Github:

**[github.com/JoelHernandez343/Cryptography](https://github.com/JoelHernandez343/Cryptography)**

And in the .zip file.