Operation Modes

Eron Romero Argumedo
Erwin Hernandez Garcia
3CV1

September 20, 2016

Contents

1 Theory **1**
1.1 How do permutations work 1
1.2 How does CBC work 1
1.3 How does CTR work 1

2 Functions **1**
2.1 ECB 1
2.2 CBC 3
2.3 CTR 3

3 Screenshots **4**
3.1 Usage 4
3.2 Results 4

List of Figures

3.1 Usage of the program 4
3.2 Original plaintext and the plaintext obtained after encryption and decryption 4
3.3 Ciphertext obtained using the different modes of operation 5

1 Theory

1.1 How do permutations work

A permutation in cryptography consists in a rearrangement of the elements in a specified order, where that order is both the key and the function to encrypt.[1]

1.2 How does CBC work

CBC is an operation mode that uses a initialization vector (IV) and realizes an XOR operation with the plaintext, whose result is then encrypted to obtain the cyphertext. In the following rounds the IV is substituted with the previous ciphertext. For decryption it uses the ciphertext which is decrypted and then an XOR is done with the IV. For the following rounds the IV is replaced with the previous ciphertext.[2]

1.3 How does CTR work

The CTR operation mode consists in calculating an XOR between the plaintext and the encryption of the IV for encryption. For decryption it calculates the XOR between the ciphertext and the encryption of the IV. On each round the IV is incremented by 1.[2]

2 Functions

2.1 ECB

Listing 1: Key generation

```
    for(int i = 0; i < permutation_size; i++)
     aux[i] = i;
    for(int count = 0; count < permutation_size;)
    {
     short aux2 = rand()%permutation_size;
     if (aux[aux2] != -1)
     {
         key[count] = aux[aux2];
         inverse_key[int(aux[aux2])] = count;
         aux[aux2] = -1;
         count++;
     }
    }
```

The key generation creates a permutation and its inverse, then save them in a file with the specified name.

Listing 2: Main function

```
void Process_Request(ifstream& fIn, ofstream& fOut,
```

```
                const char* key,
                const int block_size,
                const bool encrypt,
                const string& operation_mode)
{
    srand(time(nullptr));
    fIn.seekg(0, ios::end);
    int origin_size = int(fIn.tellg()) - 1;
    fIn.seekg(0, ios::beg);
    char* origin = new char[block_size];
    char* result = new char[block_size];
    char* aux = nullptr;
    char* IV = nullptr;
    if(operation_mode != "ECB")
    {
     aux = new char[block_size];
     IV = new char[block_size];
     if(encrypt)
     {
         for(int i = 0; i < block_size; i++)
             IV[i] = i + 'A';
         fOut.write(IV, block_size);
         if(operation_mode == "CBC")
          memcpy(result, IV, block_size);
     }
     else
     {
         fIn.read(IV, block_size);
         origin_size -= block_size;
         if(operation_mode == "CBC")
          memcpy(aux, IV, block_size);
     }
    }
    for(int i = 0; i < origin_size; i += block_size)
    {
     fIn.read(origin, block_size);
     if(fIn.eof())
         for(int j = fIn.gcount()-1; j < block_size; j++)
          origin[j] = 'A';
     if(operation_mode == "ECB")
         permute(origin, result, key, block_size);
     else if(operation_mode == "CBC")
         CBC(origin, aux, result, key, block_size, encrypt);
     else if(operation_mode == "CTR")
         CTR(origin, IV, result, key, block_size, encrypt);
     fOut.write(result, block_size);
    }
```

```
    delete[] origin;
    delete[] result;
    if(aux != nullptr)
    {
     delete[] aux;
     delete[] IV;
    }
}
```

This function is used to partition the plain or cipher text and to call the specified mode of operation.

The IV is created randomly in the case of encryption and extracted from the beginning of the ciphertext if decryption.

2.2 CBC

Listing 3: CBC operation mode

```
void CBC(const char* origin, char* aux, char* result,
      const char* key, const int block_size,
      bool encrypt)
{
    if(encrypt)
    {
     XOR(origin, result, aux, block_size);
     permute(aux, result, key, block_size);
    }
    else
    {
     permute(origin, result, key, block_size);
     XOR(result, aux, result, block_size);
     memcpy(aux, origin, block_size);
    }
}
```

The function to cipher is called each round, if the encryption is solicited, then the result array starts as the IV which is randomly generated, on the other case, the aux array is the one that contains the IV.

2.3 CTR

Listing 4: CTR operation mode

```
void CTR(const char* origin, char* IV, char* result,
      const char* key, const int block_size,
      bool encrypt)
{
```

```
5       permute(IV, result, key, block_size);
        XOR(result, origin, result, block_size);
        add(IV, 1, block_size);
}
```

If the CTR mode is specified, the key used is for encryption regardless the option used. In any other case, for encryption, the second key (inverse permutation) is extracted.

3 Screenshots

3.1 Usage

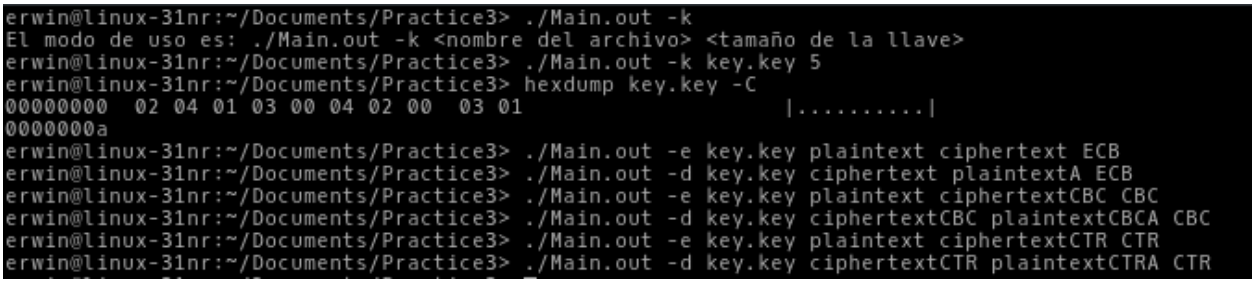

Figure 3.1: Usage of the program

3.2 Results

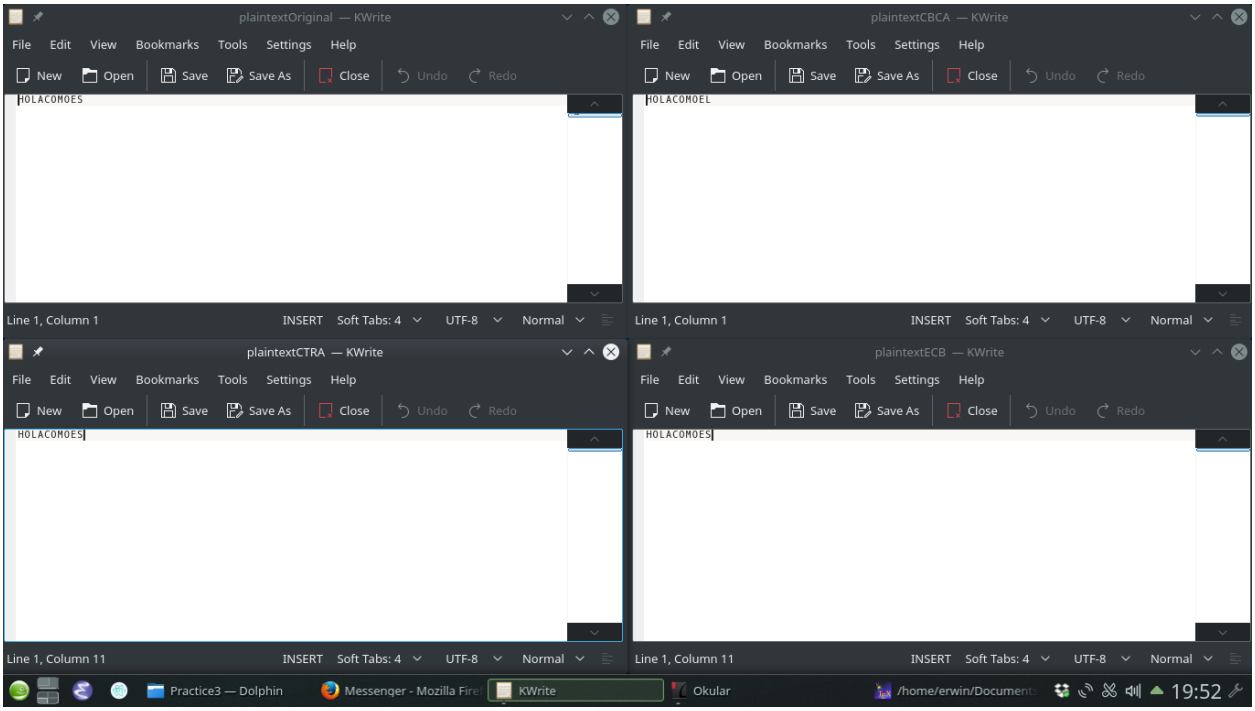

Figure 3.2: Original plaintext and the plaintext obtained after encryption and decryption

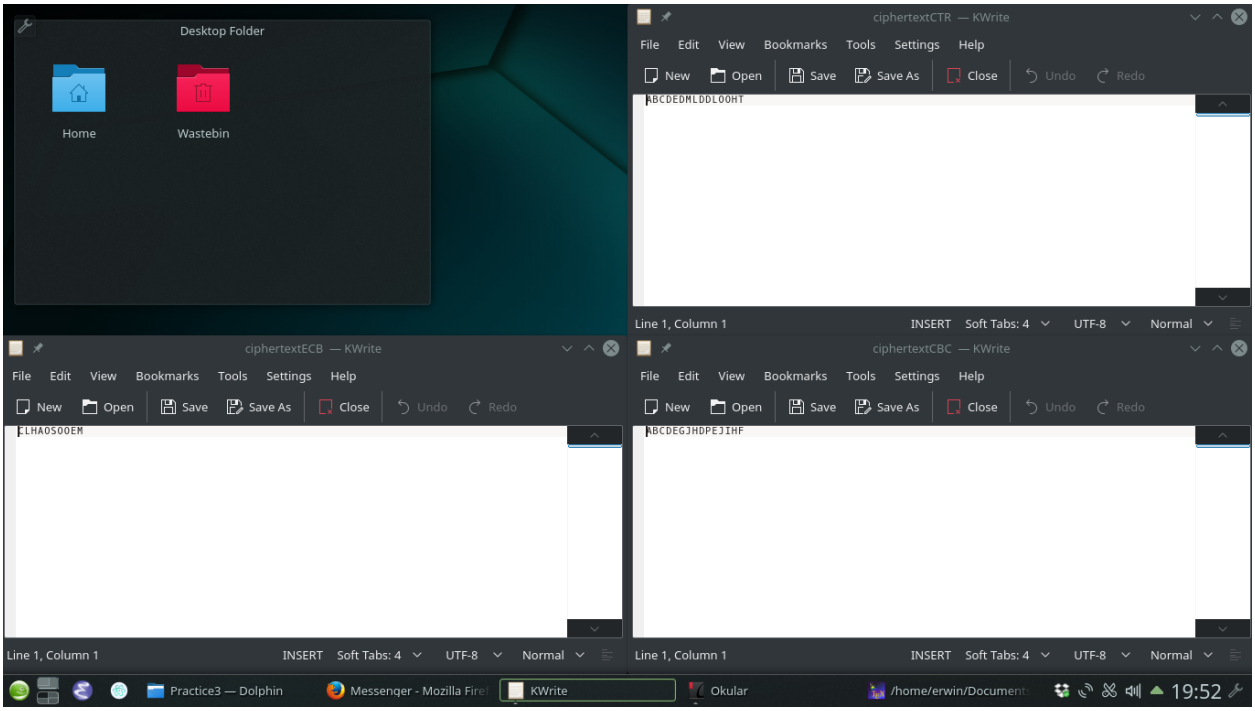

Figure 3.3: Ciphertext obtained using the different modes of operation

References

[1] C. Christensen. Permutation ciphers. [Online]. Available: http://www.nku.edu/~christensen/1402%20permutation%20ciphers.pdf

[2] F. R.-H. Debrup Chakraborty, "Block cipher modes of operation from a hardware implementation perspective."