

Binary Fields, AES and Operation Modes

Eron Romero Argumedo
Erwin Hernandez Garcia
3CV1

October 17, 2016

Contents

1	Theory	1
1.1	Evariste Galois	1
1.2	Rijndael and AES	1
2	Functions	1
2.1	Section 1	1
2.2	Analysing the algorithm	2
2.3	Modes of operation	3
2.3.1	Key generation	3
2.3.2	ECB	4
2.3.3	CBC	4
2.3.4	CTR	5
3	Screenshots	5
3.1	Inverse	5
3.1.1	Usage	5
3.1.2	Results	6
3.2	Modes of operation	7
3.2.1	Usage	7
3.2.2	Results	8

List of Figures

3.1	Usage of the program	5
3.2	Test with $a(x)=8$ and $m(x)=13$	6
3.3	Test with $a(x)=2$ and $m(x)=b$	6
3.4	Test with $a(x)=1b$ and $m(x)=25$	6
3.5	Test with $a(x)=f0$ and $m(x)=11b$	7
3.6	Usage of the program	7
3.7	Original plaintext	8
3.8	Ciphertext obtained after encrypt with AES	8
3.9	Plaintext obtained after decrypt with AES	9

1 Theory

1.1 Evariste Galois

Evariste Galois was a mathematical genius known nowadays for the development of the group theory.

Many of its constructions (today called Galois group, Galois fields and Galois theory) remain fundamental concepts in modern algebra.[1]

1.2 Rijndael and AES

AES (Advances Encryption Standard) started as an initiative to replace the old Data Encryption Standard (DES) and triple-DES.

Rijndael is an algorithm made by Joan Daemen and Vincent Rijmen. It was selected as a candidate for the AES. On 2 October, 2000, NIST (National Institute of Standards and Technology) officially announced that Rijndael, without modifications, would become the AES. Some factors that gave Rijndael a quick acceptance were that it is royalty-free and that it can be implemented easily on a wide range of platforms. [2]

2 Functions

2.1 Section 1

Listing 1: Obtaining the keys for AES

```
;;Erwin Hernández García
;;Eron Romero Argumedo
;;17 de octubre de 2016
;;Ejercicio 1, sección 1

5
(load "aes32.lisp")
(defun crear-llave (tamaño-llave)
  (let ((array (make-array (/ (nth (1- tamaño-llave)
                                +aes-key-bits+) 8)
                            :element-type 'unsigned-byte
                            :initial-element 0)))
    (do ((i 0 (1+ i)))
        ((eql i (array-dimension array 0)) array)
      (setf (aref array i) (random #xFF))))
10
(defun bin-array->hex-str (bin)
  (let ((hex (make-string (* 2 (length bin)))))
    (dotimes (i (length bin))
      (let ((h (format nil "~2,'0X" (aref bin i))))
        (setf (char hex (* 2 i)) (char h 0))
20        (setf (char hex (1+ (* 2 i))) (char h 1))))
    hex))
```

```

25 (defun obtain-keys ()
    (format t "Bienvenido, Inserte su archivo~%")
    (let* ((file (read))
           (key1 (bin-array->hex-str
                  (aes-key-fkey (aes-expand-key (crear-llave 1))))))
          (key2 (bin-array->hex-str
                  (aes-key-fkey (aes-expand-key (crear-llave 2))))))
          (key3 (bin-array->hex-str
                  (aes-key-fkey (aes-expand-key (crear-llave 3))))))
          (f-stream (open (symbol-name file) :direction :output)))
      (setq function (lambda (limit f-stream string)
                      (do ((i 0 (1+ i)))
                          ((eql i (1+ limit)) (terpri f-stream))
                          (write-line string
                                      f-stream
                                      :start (* i 8)
                                      :end (+ (* i 8) 8))))))
          (write-line "128 bits" f-stream)
          (terpri f-stream)
          (funcall function 10 f-stream key1)
          (write-line "192 bits" f-stream)
          (terpri f-stream)
          (funcall function 12 f-stream key2)
          (write-line "256 bits" f-stream)
          (terpri f-stream)
          (funcall function 14 f-stream key3)
          (close f-stream)))

```

The AES implementation used was changed because the C implementation that was found did not work with variable key size.

2.2 Analysing the algorithm

Listing 2: Inverse

```

unsigned int polynomialEuclides(unsigned int a, unsigned int m)
{
    unsigned int u, v, g1=0x01, g2=0x00, aux, mask, maskXOR;
    u=a;
    v=m;
5   v=m;
    int j, i=1;
    mask=fMask(m);
    maskXOR=xorMask(m);

10   while ( u != 0x00000001 )
    {

```

```

j=deg(u)-deg(v);

    if ( j<0 )
15  {
    aux = u;
    u = v;
    v = aux;
    aux = g1;
20  g1 = g2;
    g2 = aux;
    j=-j;
    }

25  aux=u ^ mul(j, v, m);
    u=aux;

    if ( deg(u)==deg(m) )
    {
30  u=u&mask;
    u=u^maskXOR;
    }

    i++;
35  aux=g1 ^ mul (j, g2, m);
    g1=aux;
    }

40  return g1;
}

```

The inverse function receives two values, a polynomial $a \in GF(2^t)$ and an irreducible polynomial m of degree t . The output of this function is a^{-1} in the binary field $GF(2^t)$.

2.3 Modes of operation

2.3.1 Key generation

Listing 3: Key generation

```

void generateKey(const string& filename, const int key_size)
{
    byte *key;
    key = new byte[key_size];
5  srand(time(nullptr));
    for(int i = 0; i < key_size; i++)
        key[i] = rand()%0xFF;
    ofstream fOut(filename, ios::binary | ios::trunc);
}

```

```

10      fOut.write((char*)key, key_size);
      fOut.close();
      delete[] key;
  }

```

The key generation creates a 16bytes (128bits) random key, then saves them in a file with the specified name.

2.3.2 ECB

Listing 4: ECB mode

```

    if(operation_mode == "ECB")
        cipher(origin, result, key, block_size, encrypt);

```

Listing 5: AES

```

void cipher(const byte* origin, byte* result, const byte* key,
            const int block_size, bool encrypt)
{
    if(encrypt)
5       AES128_ECB_encrypt(origin, key, result);
    else
        AES128_ECB_decrypt(origin, key, result);
}

```

The plaintext is ciphered independently of the other blocks

2.3.3 CBC

Listing 6: CBC mode

```

void CBC(const byte* origin, byte* aux, byte* result,
          const byte* key, const int block_size,
          bool encrypt)
{
5     if(encrypt)
    {
        XOR_Block(origin, result, aux, block_size);
        cipher(aux, result, key, block_size, encrypt);
    }
10    else
    {
        cipher(origin, result, key, block_size, encrypt);
        XOR_Block(result, aux, result, block_size);
        memcpy(aux, origin, block_size);
15    }
}

```

If the solicited operation is encrypt, then an XOR operation is made with the plaintext and the previous ciphertext, the result is then encrypted. On the other hand, if the decryption is solicited, first the ciphertext is decrypted and then the XOR is made with this result and the previous ciphertext

2.3.4 CTR

Listing 7: CTR mode

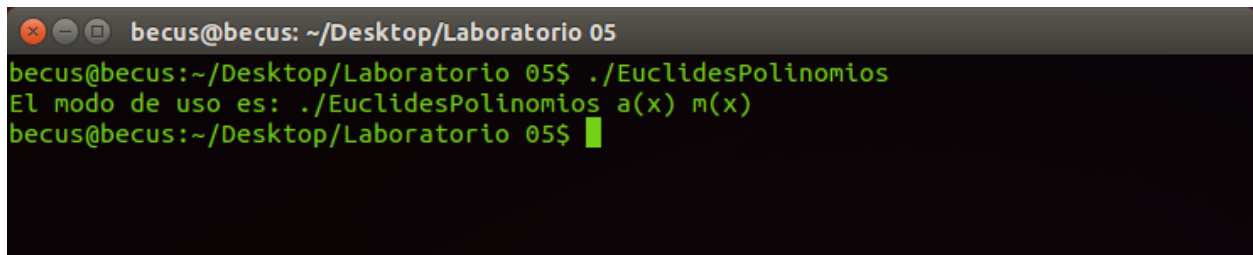
```
void CTR(const byte* origin, byte* IV, byte* result,
         const byte* key, const int block_size,
         bool encrypt)
{
5   cipher(IV, result, key, block_size, true);
   XOR_Block(result, origin, result, block_size);
   add(IV, 1, block_size);
}
```

The IV is encrypted, then an XOR is made with this result and the plaintext, finally the IV is increased by one to use it on the following round

3 Screenshots

3.1 Inverse

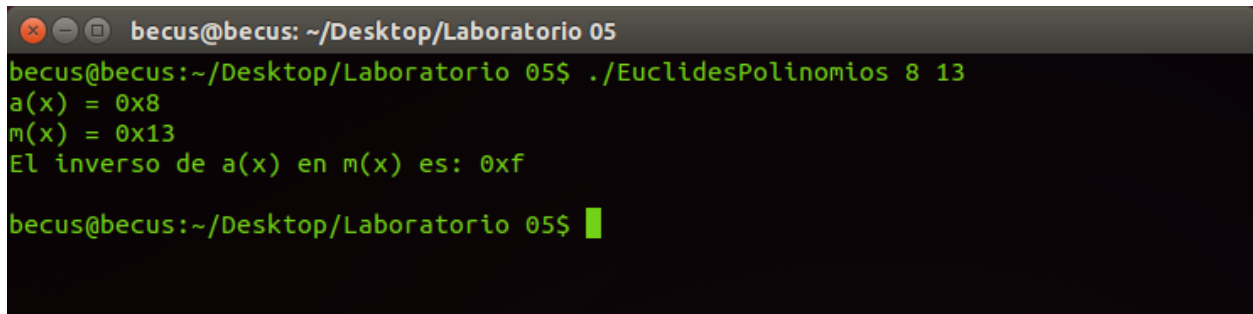
3.1.1 Usage

A terminal window with a dark background and green text. The title bar shows 'becus@becus: ~/Desktop/Laboratorio 05'. The prompt is 'becus@becus:~/Desktop/Laboratorio 05\$'. The user has entered './EuclidesPolinomios', and the output is 'El modo de uso es: ./EuclidesPolinomios a(x) m(x)'. The prompt is now 'becus@becus:~/Desktop/Laboratorio 05\$' followed by a green cursor.

```
becus@becus: ~/Desktop/Laboratorio 05
becus@becus:~/Desktop/Laboratorio 05$ ./EuclidesPolinomios
El modo de uso es: ./EuclidesPolinomios a(x) m(x)
becus@becus:~/Desktop/Laboratorio 05$ █
```

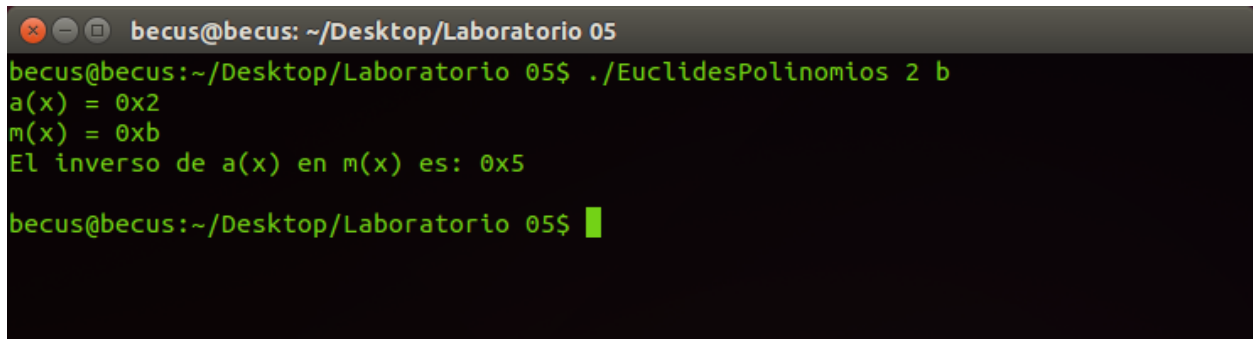
Figure 3.1: Usage of the program

3.1.2 Results



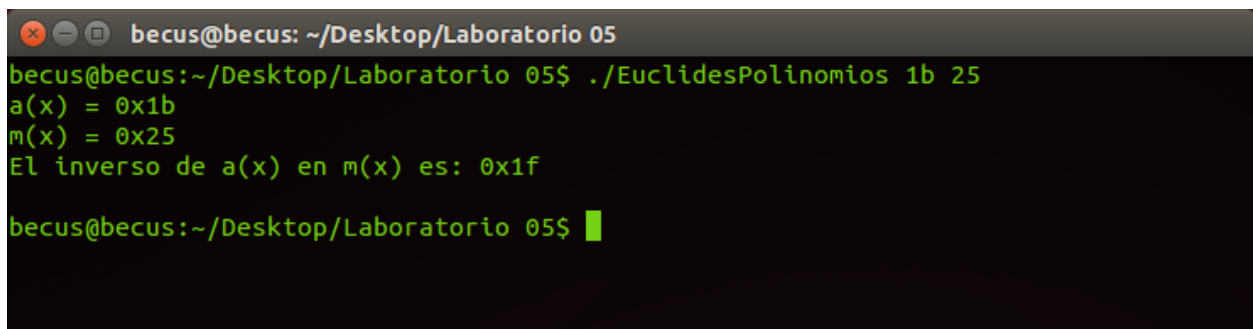
```
becus@becus: ~/Desktop/Laboratorio 05
becus@becus:~/Desktop/Laboratorio 05$ ./EuclidesPolinomios 8 13
a(x) = 0x8
m(x) = 0x13
El inverso de a(x) en m(x) es: 0xf
becus@becus:~/Desktop/Laboratorio 05$
```

Figure 3.2: Test with $a(x)=8$ and $m(x)=13$



```
becus@becus: ~/Desktop/Laboratorio 05
becus@becus:~/Desktop/Laboratorio 05$ ./EuclidesPolinomios 2 b
a(x) = 0x2
m(x) = 0xb
El inverso de a(x) en m(x) es: 0x5
becus@becus:~/Desktop/Laboratorio 05$
```

Figure 3.3: Test with $a(x)=2$ and $m(x)=b$



```
becus@becus: ~/Desktop/Laboratorio 05
becus@becus:~/Desktop/Laboratorio 05$ ./EuclidesPolinomios 1b 25
a(x) = 0x1b
m(x) = 0x25
El inverso de a(x) en m(x) es: 0x1f
becus@becus:~/Desktop/Laboratorio 05$
```

Figure 3.4: Test with $a(x)=1b$ and $m(x)=25$


```
becus@becus: ~/Desktop/Laboratorio 05
becus@becus:~/Desktop/Laboratorio 05$ ./EuclidesPolinomios f0 11b
a(x) = 0xf0
m(x) = 0x11b
El inverso de a(x) en m(x) es: 0x5b
becus@becus:~/Desktop/Laboratorio 05$ █
```

Figure 3.5: Test with $a(x)=f0$ and $m(x)=11b$

3.2 Modes of operation

3.2.1 Usage

```
HernándezGarcíaErwin_Section2_Code: bash — Konsole
File Edit View Bookmarks Settings Help
./Main.out
Se debe especificar un modo de uso
./Main.out -k llaves.key 128
hexdump llaves.key -C
00000000 df 7a fc 99 e4 e5 a5 af 34 a0 50 38 f3 32 a2 2f |.z.....4.P8.2./|
00000010

ECB
Encryption
./Main.out -e
El modo de uso es: ./Main.out -e <llave> <texto plano> <texto cifrado> <modo de operación>
./Main.out llaves.key plaintext.txt ciphertextECB.txt ECB
Decryption
./Main.out -d
El modo de uso es: ./Main.out -d <llave> <texto cifrado> <texto plano> <modo de operación>
./Main.out llaves.key ciphertextECB.txt plaintextECB.txt ECB

CBC
Encryption
./Main.out -e
El modo de uso es: ./Main.out -e <llave> <texto plano> <texto cifrado> <modo de operación>
./Main.out llaves.key plaintext.txt ciphertextCBC.txt CBC
Decryption
./Main.out -d
El modo de uso es: ./Main.out -d <llave> <texto cifrado> <texto plano> <modo de operación>
./Main.out llaves.key ciphertextCBC.txt plaintextCBC.txt CBC

CTR
Encryption
./Main.out -e
El modo de uso es: ./Main.out -e <llaves> <texto plano> <texto cifrado> <modo de operación>
./Main.out llaves.key plaintext.txt ciphertextCTR.txt CTR
Decryption
./Main.out -d
El modo de uso es: ./Main.out -d <llaves> <texto cifrado> <texto plano> <modo de operación>
./Main.out llaves.key ciphertextCTR.txt plaintextCTR.txt CTR
```

Figure 3.6: Usage of the program

3.2.2 Results

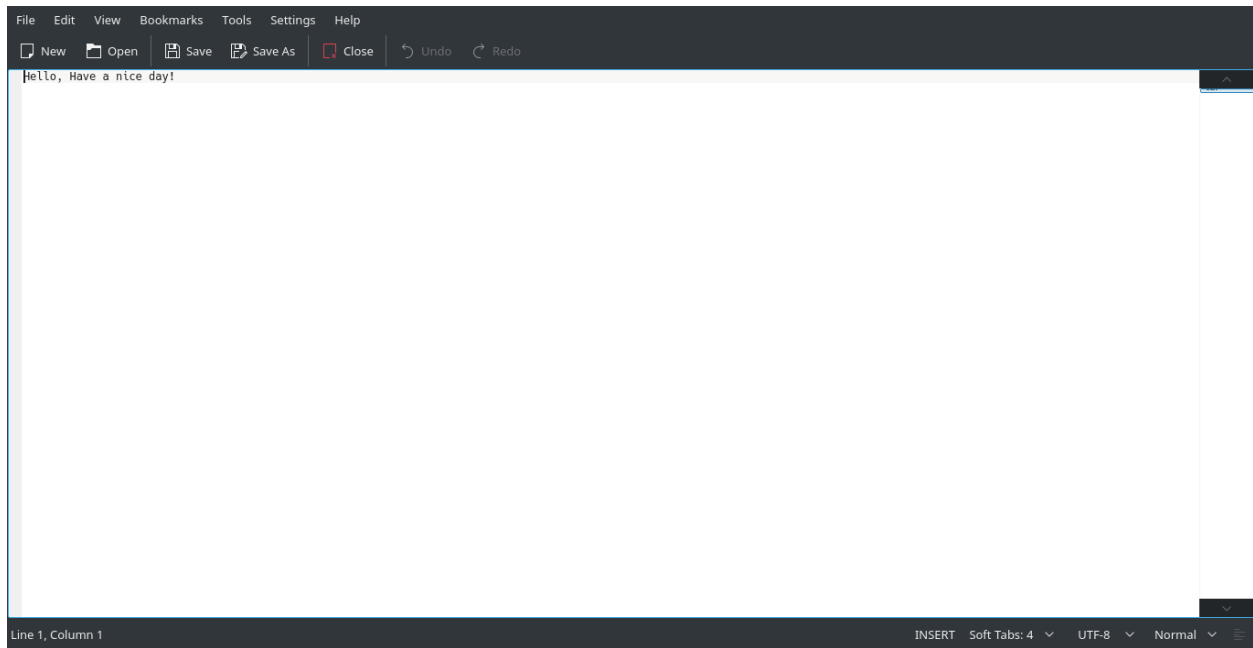


Figure 3.7: Original plaintext

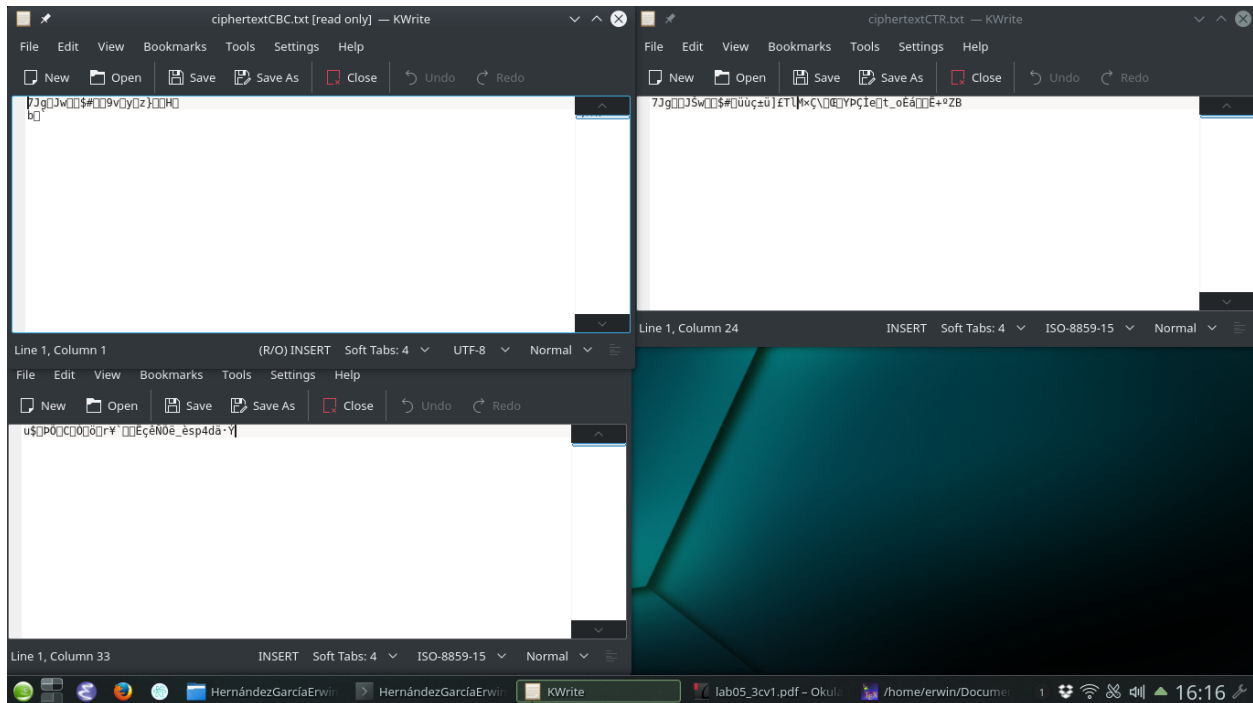


Figure 3.8: Ciphertext obtained after encrypt with AES

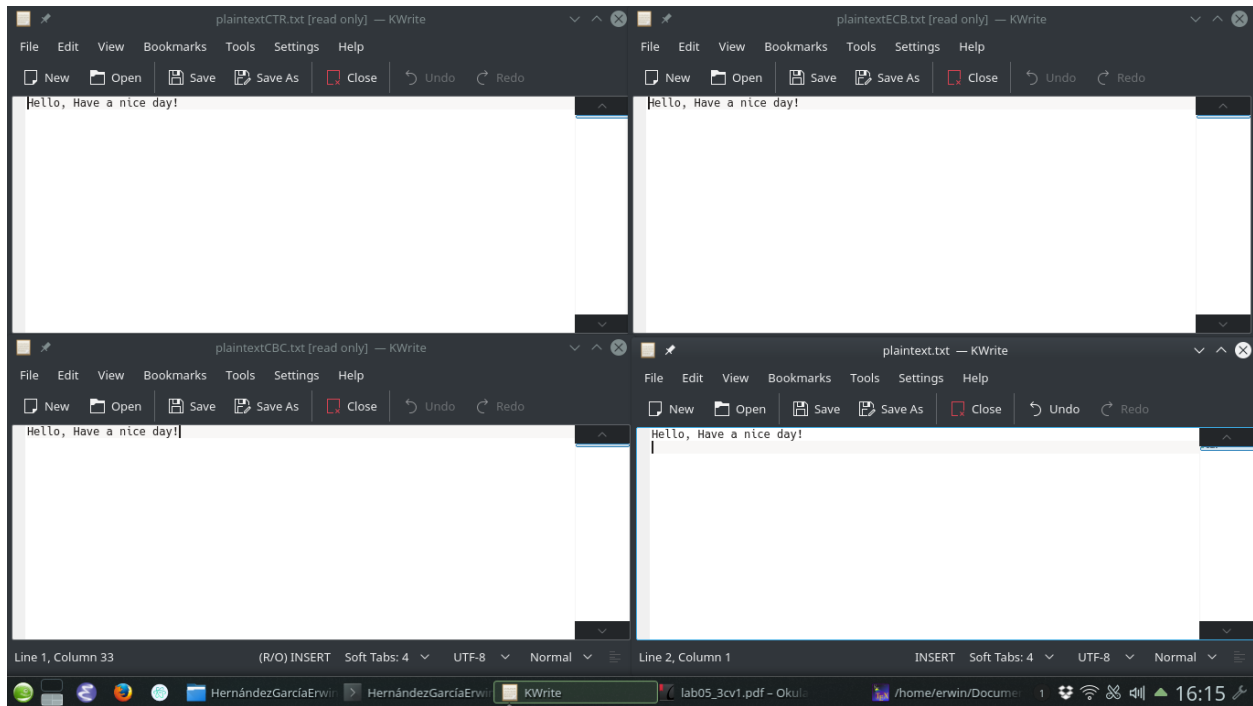


Figure 3.9: Plaintext obtained after decrypt with AES

References

- [1] La trágica historia de evariste galois. [Online]. Available: <http://www.uv.es/~jagular/historias/galois.html>
- [2] V. R. Joan Daemen. The design of rijndael. [Online]. Available: http://jda.noekeon.org/JDA_VRI_Rijndael.2002.pdf