7CM2

Lightweight cryptography II

SELECTED TOPIC IN CRYPTOGRAPHY

FRANCISCO UZIEL CORDOVA PICHARDO

22/06/2023

Para el algoritmo de cifrado que elegiste en la tarea previa, de la lista de finalistas https://csrc.nist.gov/Projects/lightweight-cryptography/finalists,realiza lo siguiente:

1. Busca el código fuente del algoritmo de cifrado en cuestión, compílalo y ejecútalo.
2. Especifica el lenguaje de programación en el que está codificado el algoritmo, explica brevemente los pasos a seguir para compilarlo y ejecutarlo.
3. Ejecútalo al menos 5 veces con distintos tipos de archivo y guarda el resultado del cifrado.

Posteriormente redacta en equipo un pequeño reporte que contenga lo siguiente:

- Nombre completo de cada miembro del equipo
- Fecha de hoy
- Respuesta a cada una de las preguntas anteriores
- Resultados legibles del texto cifrado obtenido.
- Impresiones de pantalla de la ejecución .

Sube el pdf de tu reporte a esta plataforma al concluir la clase.

Código de Sparkle.c por lo tanto esta hecho en C.

```
////////////////////////////////////////////////////////////////////////////
///
// sparkle.c: Optimized C implementation of the SPARKLE permutation
family.  //
// This file is part of the SPARKLE package that was sent to NIST during the
//
// 3rd round of the Lightweight Cryptography (LWC) standardization
project.  //
// Version 1.2.1 (18-Oct-21), see <http://github.com/cryptolu/> for updates.
//
// Authors: The SPARKLE Group (Christof Beierle, Alex Biryukov, Luan Cardoso
//
// dos Santos, Johann Groszschaedl, Amir Moradi, Leo Perrin, Aein
Rezaei     //
// Shahmirzadi, Aleksei Udovenko, Vesselin Velichkov, and Qingju
Wang).      //
// License: GPLv3 (see LICENSE file), other licenses available upon request.
//
// Copyright (C) 2019-2021 University of Luxembourg
<http://www.uni.lu/>.    //
// ------------------------------------------------------------------------
//
```

```
// This program is free software: you can redistribute it and/or modify
it   //
// under the terms of the GNU General Public License as published by
the     //
// Free Software Foundation, either version 3 of the License, or (at
your    //
// option) any later version. This program is distributed in the hope
that   //
// it will be useful, but WITHOUT ANY WARRANTY; without even the
implied     //
// warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See
the  //
// GNU General Public License for more details. You should have received
a   //
// copy of the GNU General Public License along with this program. If
not,   //
// see
<http://www.gnu.org/licenses/>.                                    //
////////////////////////////////////////////////////////////////////////
///


#include <stdio.h>
#include "sparkle.h"


#define ROT(x, n) (((x) >> (n)) | ((x) << (32-(n))))
#define ELL(x) (ROT(((x) ^ ((x) << 16)), 16))


// Round constants
static const uint32_t RCON[MAX_BRANCHES] = {      \
  0xB7E15162, 0xBF715880, 0x38B4DA56, 0x324E7738, \
  0xBB1185EB, 0x4F7C7B57, 0xCFBFA1C8, 0xC2B3293D  \
};


void sparkle(uint32_t *state, int brans, int steps)
{
  int i, j;  // Step and branch counter
  uint32_t rc, tmpx, tmpy, x0, y0;

  for(i = 0; i < steps; i ++) {
    // Add round constant
    state[1] ^= RCON[i%MAX_BRANCHES];
    state[3] ^= i;
```

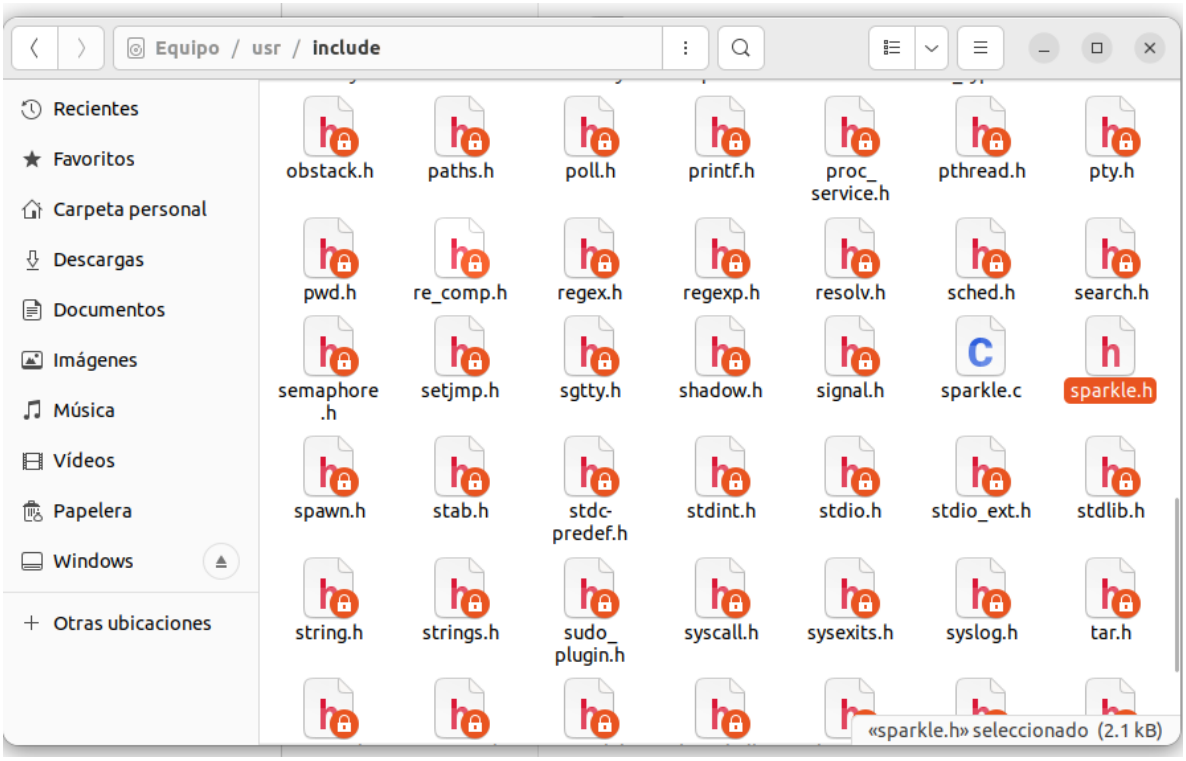
```
    // ARXBOX layer
    for(j = 0; j < 2*brans; j += 2) {
      rc = RCON[j>>1];
      state[j] += ROT(state[j+1], 31);
      state[j+1] ^= ROT(state[j], 24);
      state[j] ^= rc;
      state[j] += ROT(state[j+1], 17);
      state[j+1] ^= ROT(state[j], 17);
      state[j] ^= rc;
      state[j] += state[j+1];
      state[j+1] ^= ROT(state[j], 31);
      state[j] ^= rc;
      state[j] += ROT(state[j+1], 24);
      state[j+1] ^= ROT(state[j], 16);
      state[j] ^= rc;
    }
    // Linear layer
    tmpx = x0 = state[0];
    tmpy = y0 = state[1];
    for(j = 2; j < brans; j += 2) {
      tmpx ^= state[j];
      tmpy ^= state[j+1];
    }
    tmpx = ELL(tmpx);
    tmpy = ELL(tmpy);
    for (j = 2; j < brans; j += 2) {
      state[j-2] = state[j+brans] ^ state[j] ^ tmpy;
      state[j+brans] = state[j];
      state[j-1] = state[j+brans+1] ^ state[j+1] ^ tmpx;
      state[j+brans+1] = state[j+1];
    }
    state[brans-2] = state[brans] ^ x0 ^ tmpy;
    state[brans] = x0;
    state[brans-1] = state[brans+1] ^ y0 ^ tmpx;
    state[brans+1] = y0;
  }
}


void sparkle_inv(uint32_t *state, int brans, int steps)
{
  int i, j;  // Step and branch counter
  uint32_t rc, tmpx, tmpy, xb1, yb1;

  for(i = steps-1; i >= 0; i --) {
```

```
    // Linear layer
    tmpx = tmpy = 0;
    xb1 = state[brans-2];
    yb1 = state[brans-1];
    for (j = brans-2; j > 0; j -= 2) {
      tmpx ^= (state[j] = state[j+brans]);
      state[j+brans] = state[j-2];
      tmpy ^= (state[j+1] = state[j+brans+1]);
      state[j+brans+1] = state[j-1];
    }
    tmpx ^= (state[0] = state[brans]);
    state[brans] = xb1;
    tmpy ^= (state[1] = state[brans+1]);
    state[brans+1] = yb1;
    tmpx = ELL(tmpx);
    tmpy = ELL(tmpy);
    for(j = brans-2; j >= 0; j -= 2) {
      state[j+brans] ^= (tmpy ^ state[j]);
      state[j+brans+1] ^= (tmpx ^ state[j+1]);
    }
    // ARXBOX layer
    for(j = 0; j < 2*brans; j += 2) {
      rc = RCON[j>>1];
      state[j] ^= rc;
      state[j+1] ^= ROT(state[j], 16);
      state[j] -= ROT(state[j+1], 24);
      state[j] ^= rc;
      state[j+1] ^= ROT(state[j], 31);
      state[j] -= state[j+1];
      state[j] ^= rc;
      state[j+1] ^= ROT(state[j], 17);
      state[j] -= ROT(state[j+1], 17);
      state[j] ^= rc;
      state[j+1] ^= ROT(state[j], 24);
      state[j] -= ROT(state[j+1], 31);
    }
    // Add round constant
    state[1] ^= RCON[i%MAX_BRANCHES];
    state[3] ^= i;
  }
}


void clear_state(uint32_t *state, int brans)
{
```

```
  int i;

  for (i = 0; i < 2*brans; i ++) {
    state[i] = 0;
  }
}


void print_state(const uint32_t *state, int brans)
{
  uint8_t *sbytes = (uint8_t *) state;
  int i, j;

  for (i = 0; i < brans; i ++) {
    j = 8*i;
    printf("(%02x%02x%02x%02x %02x%02x%02x%02x)",       \
    sbytes[j],   sbytes[j+1], sbytes[j+2], sbytes[j+3], \
    sbytes[j+4], sbytes[j+5], sbytes[j+6], sbytes[j+7]);
    if (i < brans-1) printf(" ");
  }
  printf("\n");
}


void test_sparkle(int brans, int steps)
{
  uint32_t state[2*MAX_BRANCHES] = { 0 };

  printf("input:\n");
  print_state(state, brans);
  sparkle(state, brans, steps);
  printf("sparkle:\n");
  print_state(state, brans);
  sparkle_inv(state, brans, steps);
  printf("sparkle inv:\n");
  print_state(state, brans);
  printf("\n");
}
```

Instrucciones para su compilación

1. Primer paso importar en la carpeta include en el caso de Linux la cabecera sparkle.h

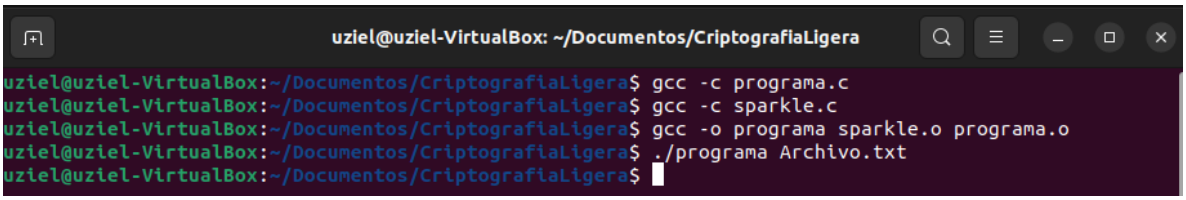


2. Crear el archivo de nuestro programa incluyendo sparkle.h

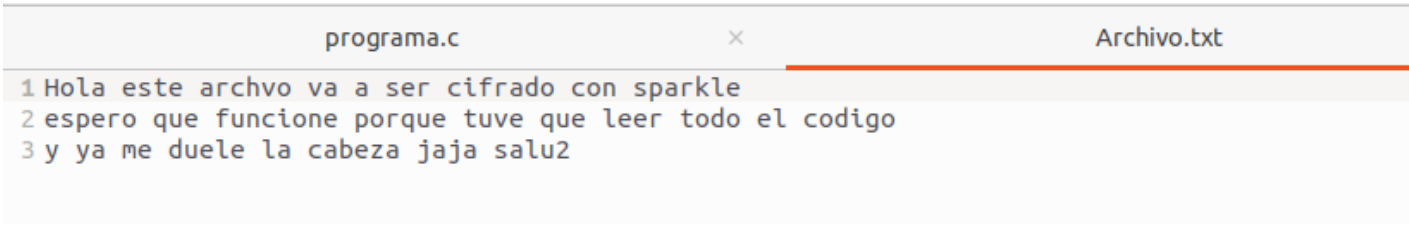
```c
#include <stdio.h>
#include "sparkle.h"
#define BUFFER_SIZE 1024
uint8_t input_buffer[BUFFER_SIZE];
uint8_t encrypted_buffer[BUFFER_SIZE];
uint8_t decrypted_buffer[BUFFER_SIZE];
int main(int argc, char *argv[]){
    FILE* input_file = fopen(argv[1], "r");
    FILE* output_file = fopen("archivo_output.bin", "wb");
    //cifrado
    size_t bytes_read = fread(input_buffer, sizeof(uint8_t), BUFFER_SIZE,
input_file);
    int brains = 4;
    int steps = 10;
    sparkle((uint32_t*)input_buffer, brains, steps);
    fwrite(input_buffer, sizeof(uint8_t), bytes_read, output_file);
    fclose(input_file);
    fclose(output_file);
    FILE* encrypted_file = fopen("archivo_output.bin", "rb");
    FILE* decrypted_file = fopen("archivo_decifrado.txt", "w");
```

```
    //descifrado
    bytes_read = fread(encrypted_buffer, sizeof(uint8_t), BUFFER_SIZE,
encrypted_file);
    sparkle_inv((uint32_t*)encrypted_buffer, brains, steps);
    fwrite(encrypted_buffer, sizeof(uint8_t), bytes_read, decrypted_file);
    fclose(encrypted_file);
    fclose(decrypted_file);
    return 0;
}
```

3. Para compilar en Ubuntu usamos el comando gcc donde compilamos por separado el programa.c y sparkle.c
4. Después compilamos sparkle.o y programa.o juntos para crear el archivo programa
5. Por ultimo ejecutamos el programa con ./programa [Nombre del archivo y extensión]



Archivos en la carpeta



archivo_decifrado.txt

```
1 Hola este archvo va a ser cifrado con sparkle
2 espero que funcione porque tuve que leer todo el codigo
3 y ya me duele la cabeza jaja salu2
```