Universidad de Antioquia

Informática 2

Desafío 1

Estudiantes: Alexa Carolina Gamboa Mier - Manuela Arboleda Montoya

El Desafío 1 plantea un escenario de ingeniería inversa y criptoanálisis donde se debe recuperar información oculta en archivos que han sido sometidos a procesos desconocidos de compresión y cifrado. El objetivo principal es aplicar lógica, deducción y conocimientos avanzados de C/C++ (punteros, memoria dinámica, operaciones a nivel de bits) para revertir estos procesos y obtener el mensaje original legible.

La estrategia se basó en la modularidad del código y en la aplicación secuencial de hipótesis, según las técnicas de compresión y cifrado más comunes para datos binarios.

Se identificaron dos posibles esquemas de compresión que el sistema debe poder manejar:

1. RLE: Se implementó una rutina en funciones2.cpp para revertir la compresión basada en la secuencia (Control, conteo, valor), que es un patrón frecuente.
2. LZ78: Se incluyó un módulo en funciones3.cpp para validar y simular el procesamiento de este esquema, asegurando la cobertura del requisito.

El núcleo del desafío reside en el descifrado. La hipótesis más probable para un cifrado de bajo nivel es la combinación de operaciones: XOR de clave de 1 byte y rotación de bits.

El modulo funciones4.cpp implementa una búsqueda probando las combinaciones posibles.

Las decisiones fueron tomadas para cumplir con los objetivos de demostrar destrezas en el manejo de bajo nivel y eficiencia

COMPONENTE	TIPO DE DATO/ESTRUCTURA	JUSTIFICACION
Lectura de Archivos	Punteros	Se utiliza char* para manejar los datos binarios como bytes sin interpretar, fread y fseek que permiten lectura directa y eficiente del tamaño exacto
Memoria Dinámica	Malloc y realloc	Usado en leerArchivo y en descomprimir_rle_terna para asignar y redimensionar el buffer de salida de forma eficiente y solo según la necesidad
Operaciones Criptograficas	unsigned char	Se usa unsigned char para garantizar que los 256 valores de un byte se manejen como

		valores binarios puros (0 a 255), evitando problemas de signo. Los operadores <<, >>.
Metrica de Legibilidad	Funcion cp	Se implemento un algoritmo de puntuación estricta que premia la aparición de carecteres imprimibles comunes en el español para diferenciar el mensaje correcto entre los 4096 intentos fallidos

El archivo main.cpp inicia el sistema. La función procesarArchivo() en funciones1.cpp actúa como el controlador, ejecutando secuencialmente:

1. Carga: Obtiene el arrayComprimido usando leerArchivo().
2. Descompresión RLE: Genera el arrayDescomprimido usando descomprimir_rle_terna().
3. Análisis: Llama dos veces a la función desencriptarArray() en fucniones4.cpp, para probar claves y ordenes sobre:
    - El array Original/Comprimido
    - El array Descomprimido RLE

De esta manera, el sistema prueba la clave en dos puntos distintos del proceso de transformación del archivo.

Este proyecto es una prueba de las habilidades de análisis de problemas y programación avanzada en C++. Se logró la recuperación completa de la información mediante una arquitectura que gestiona la lectura binaria, revierte la compresión de RLE y aplica un algoritmo de análisis de fuerza bruta eficiente. El resultado final valida la capacidad del equipo para enfrentar escenarios complejos y utilizando las herramientas de programación para aclarar la lógica de cifrado y compresión aplicada