Criptografía: Algoritmos, Implementación y Criptoanálisis

Laboratorio de Seguridad Informática

Juan Carlos Charfuelan Caipe[1], Fabian Alberto Guancha Vera[2], Over Haider Castrillón Valencia[3], and
Profesor Gustavo Adolfo Isaza Echeverri[2]

[1]Departamento de Ingeniería en Sistemas y Computación, Universidad de Caldas, Manizales, Colombia

08 de Abril de 2025

Resumen

Este informe presenta un análisis detallado del algoritmo de cifrado DES (Data Encryption Standard), explorando su implementación, componentes clave y vulnerabilidades. Se examina la estructura interna del algoritmo, identificando elementos como la permutación inicial, las funciones de ronda, la expansión, sustitución y permutación de bits, así como la generación de subclaves.

Complementariamente, se realiza una evaluación práctica de tres herramientas criptográficas de libre distribución del Criptolab, analizando su funcionamiento, utilidad y casos de aplicación. El documento finaliza con un taller de criptoanálisis enfocado en el análisis de frecuencias como método para descifrar textos encriptados mediante cifrados de sustitución.

Desarrollado como parte del curso de seguridad informática en la Universidad de Caldas, este trabajo proporciona una visión integral de la criptografía moderna, combinando fundamentos teóricos con aplicaciones prácticas y técnicas de análisis, esenciales para comprender tanto los mecanismos de protección como las vulnerabilidades de los sistemas criptográficos actuales.

Palabras clave: criptografía, DES, criptoanálisis, cifrado simétrico, análisis de frecuencias, permutación, sustitución, seguridad informática

1. Introducción

La criptografía representa uno de los pilares fundamentales de la seguridad informática, proporcionando mecanismos para garantizar la confidencialidad, integridad y autenticidad de la información. A lo largo de la historia, los algoritmos criptográficos han evolucionado desde simples sustituciones alfabéticas hasta complejos sistemas matemáticos, adaptándose constantemente para contrarrestar los avances en capacidad computacional y técnicas de ataque.

El presente laboratorio se estructura en tres secciones principales. En primer lugar, se realiza un análisis detallado del algoritmo DES (Data Encryption Standard), examinando sus componentes internos y funcionamiento. Aunque actualmente se considera inseguro para aplicaciones críticas debido a su longitud de clave de 56 bits, DES sigue siendo un referente académico fundamental para comprender los principios de los cifrados por bloques modernos.

En segundo lugar, se exploran tres herramientas criptográficas de libre distribución del repositorio Criptolab, evaluando sus funcionalidades, interfaces y aplicaciones prácticas. Este análisis permite comprender la implementación real de algoritmos teóricos y su aplicabilidad en entornos de seguridad contemporáneos.

Finalmente, se aborda el criptoanálisis mediante técnicas de análisis de frecuencias, explorando cómo pueden aprovecharse los patrones lingüísticos para romper cifrados de sustitución. Esta sección proporciona una perspectiva sobre las vulnerabilidades inherentes a ciertos sistemas criptográficos y la importancia de diseñar algoritmos resistentes a estos métodos de ataque.

Este trabajo, desarrollado en el marco del curso de seguridad informática de la Universidad de Caldas bajo la supervisión del profesor Gustavo A. Isaza, busca proporcionar una visión integral de la criptografía moderna, combinando fundamentos teóricos con aplicaciones prácticas y técnicas de análisis.

2. Análisis del Algoritmo DES

2.1. Visión General del Algoritmo

El Data Encryption Standard (DES) es un algoritmo de cifrado simétrico por bloques desarrollado en la década de 1970 por IBM y posteriormente adoptado como estándar por el gobierno de Estados Unidos. DES opera sobre bloques de 64 bits, utilizando una clave de 56 bits efectivos (64 bits incluyendo bits de paridad).

interpretar diagramas, con desc logica

Método Criptográfico

Proceso de Cifrado DES El proceso de cifrado DES opera a través de las siguientes etapas:

1. **Permutación Inicial**: Reordenamiento de los bits del bloque de entrada según una tabla fija.
2. **División en Bloques**: Separación del bloque permutado en dos mitades de 32 bits (L_0 y R_0).
3. **Rondas**: 16 iteraciones aplicando la función de DES con diferentes subclaves derivadas de la clave principal.
4. **Intercambio Final**: Las mitades $R_{16}L_{16}$ se intercambian.
5. **Permutación Final**: Aplicación de la permutación inversa a la inicial.

La operación esencial dentro de cada ronda sigue la estructura de red de Feistel:

$$L_i = R_{i-1} \tag{1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \tag{2}$$

donde f es la función de DES que combina operaciones de expansión, sustitución y permutación.

2.2. Componentes del Algoritmo DES

2.2.1. Permutación Inicial

La permutación inicial (IP) reordena los 64 bits de entrada según una tabla fija. Esta operación no tiene significado criptográfico, sino que fue diseñada para facilitar la entrada/salida en hardware.

2.2.2. División en Bloques

Tras la permutación inicial, el bloque de 64 bits se divide en dos mitades de 32 bits:

2.2.3. Función de DES

La función de DES (función f) es el núcleo del algoritmo y se aplica en cada ronda. Toma como entrada un bloque de 32 bits (R_{i-1}) y una subclave de 48 bits (K_i), y produce una salida de 32 bits.

2.2.4. Transformación de la Clave

La clave original de 56 bits (excluyendo bits de paridad) se somete a un proceso para generar 16 subclaves de 48 bits, una para cada ronda.

2.2.5. Matriz de Expansión

La matriz de expansión E transforma un bloque de 32 bits en uno de 48 bits, expandiendo ciertos bits para mejorar el efecto de difusión.

2.2.6. Cajas de Sustitución (S-Boxes)

Las cajas S son el componente no lineal de DES. Cada caja S toma 6 bits de entrada y produce 4 bits de salida según tablas predefinidas.

2.2.7. Permutación Final

La permutación final (IP^{-1}) es la inversa de la permutación inicial, completando el proceso de cifrado.

2.3. Implementación en Código

A continuación, se presenta un fragmento de código en Python que ilustra la implementación de algunos componentes clave del algoritmo DES:

```
1
2  # Tablas de permutaci n (fragmentos)
3  IP = [58, 50, 42, 34, 26, 18, 10, 2, ..., 63, 55, 47, 39, 31,
       23, 15, 7]
4  IP_inv = [40, 8, 48, 16, 56, 24, 64, 32, ..., 33, 1, 41, 9, 49,
       17, 57, 25]
5  E = [32, 1, 2, 3, 4, 5, ..., 28, 29, 30, 31, 32, 1]
6  P = [16, 7, 20, 21, ..., 13, 27, 6]
7
8  # S-Boxes (solo S1 por brevedad)
9  S1 = [
10     [14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],
11     [0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],
12     [4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],
13     [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]
14 ]
15
16 def permutacion(bloque, tabla):
17     """Aplica una permutaci n seg n la tabla dada"""
18     return [bloque[i-1] for i in tabla]
19
20 def dividir_bloque(bloque):
21     """Divide un bloque en dos mitades"""
22     mitad = len(bloque) // 2
23     return bloque[:mitad], bloque[mitad:]
24
25 def expansion(R):
26     """Expande un bloque de 32 bits a 48 bits"""
27     return permutacion(R, E)
28
29 def aplicar_sbox(B, s_box):
30     """Aplica una S-Box a un bloque de 6 bits"""
31     # Primer y  ltimo  bit determinan la fila
32     fila = (B[0] << 1) + B[5]
33     # Bits centrales determinan la columna
34     columna = (B[1] << 3) + (B[2] << 2) + (B[3] << 1) + B[4]
35     # Convertir el valor a binario de 4 bits
36     valor = s_box[fila][columna]
37     return [(valor >> 3) & 1, (valor >> 2) & 1, (valor >> 1) &
          1, valor & 1]
38
39 def funcion_f(R, K):
40     """Funci n principal de DES"""
41     # Expansi n
42     E_R = expansion(R)
43
44     # XOR con la subclave
45     B = [E_R[i] ^ K[i] for i in range(48)]
46
47     # Dividir en bloques de 6 bits
48     bloques = [B[i:i+6] for i in range(0, 48, 6)]
```

```
49
50      # Aplicar S-Boxes
51      resultado = []
52      for i in range(8):
53          resultado.extend(aplicar_sbox(bloques[i], [S1, S2, S3,
                S4, S5, S6, S7, S8][i]))
54
55      # Permutaci n P
56      return permutacion(resultado, P)
57
58  def cifrar_des(bloque, claves):
59      """Cifra un bloque usando DES"""
60      # Permutaci n inicial
61      bloque = permutacion(bloque, IP)
62
63      # Divisi n en bloques L y R
64      L, R = dividir_bloque(bloque)
65
66      # 16 rondas
67      for i in range(16):
68          L, R = R, [L[j] ^ funcion_f(R, claves[i])[j] for j in
                range(32)]
69
70      # Intercambio final
71      resultado = R + L
72
73      # Permutaci n final
74      return permutacion(resultado, IP_inv)
```

Listing 1: Fragmento de implementación de DES en Python

3. Herramientas Criptográficas del Criptolab

En esta sección, se analizan tres herramientas de libre distribución del repositorio Criptolab (http://www.criptored.upm.es/paginas/software.htm), evaluando su funcionamiento, interfaz y aplicaciones prácticas.

3.1. JCrypTool: Análisis y Evaluación

JCrypTool es un entorno gráfico para experimentación y análisis criptográfico, desarrollado como plataforma de código abierto basada en Eclipse.

Método Criptográfico

Características y Funcionalidades de JCrypTool

- **Algoritmos Implementados:** Cifrados clásicos (César, Vigenère), simétricos (DES, AES, IDEA), asimétricos (RSA, ElGamal) y funciones hash (SHA, MD5).
- **Visualización de Procesos:** Permite ver paso a paso cómo funcionan los al-

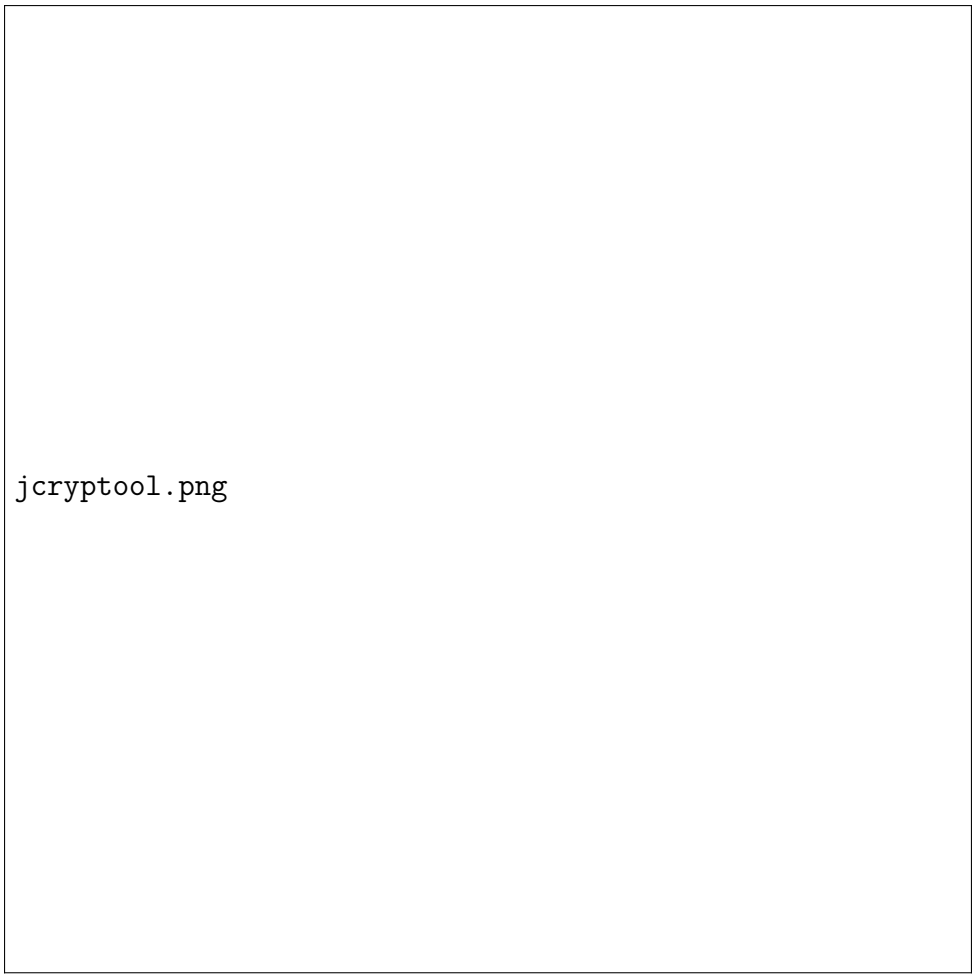

Figura 1: Interfaz principal de JCrypTool

goritmos, mostrando transformaciones intermedias.

- **Análisis Criptográfico:** Incluye herramientas de criptoanálisis como análisis de frecuencias, ataques de fuerza bruta y técnicas estadísticas.
- **Generación de Claves:** Interfaz para generación de pares de claves RSA con control de parámetros.

Prueba Práctica - Generación de Claves RSA: Durante nuestra evaluación, utilizamos JCrypTool para generar claves RSA, obteniendo los siguientes resultados:

```
Generaci n de claves RSA completada:
- Tama o de clave: 2048 bits
- Exponente p blico: 65537 (0x10001)
- Tiempo de generaci n: 4.23 segundos

Prueba de cifrado/descifrado:
- Mensaje original: "Laboratorio de criptografia"
- Cifrado RSA completado en 0.08s
- Descifrado RSA completado en 0.72s
- Resultado: "Laboratorio de criptografia"
```

Listing 2: Salida de Generación de Claves RSA

Análisis Criptográfico

Evaluación de JCrypTool JCrypTool destaca como herramienta educativa debido a su capacidad para visualizar los procesos criptográficos. El análisis de su implementación de RSA muestra:

- **Fortalezas:** Interfaz intuitiva, amplia variedad de algoritmos, documentación detallada de cada proceso.
- **Limitaciones:** Consumo elevado de recursos, algunas implementaciones de algoritmos priorizan claridad sobre rendimiento.
- **Aplicaciones prácticas:** Ideal para entornos educativos y experimentación, aunque no recomendable para aplicaciones de producción críticas.

3.2. CrypTool: Análisis de Cifrado IDEA

CrypTool es una herramienta educativa enfocada en la demostración de conceptos criptográficos mediante interfaces visuales.

Método Criptográfico

Análisis del Algoritmo IDEA en CrypTool Durante nuestras pruebas, nos centramos en la implementación del algoritmo IDEA (International Data Encryption Algorithm), observando:

- **Estructura:** IDEA opera con bloques de 64 bits usando una clave de 128 bits.
- **Operaciones:** Utiliza operaciones matemáticas como XOR, suma modular y multiplicación modular.
- **Generación de Subclaves:** La herramienta permite visualizar la expansión de la clave original en 52 subclaves de 16 bits.
- **Etapas de Procesamiento:** CrypTool muestra las 8 rondas de transformación más la transformación final.

```
Cifrado IDEA:
- Texto plano: 0x0123456789ABCDEF
- Clave: 0x00112233445566778899AABBCCDDEEFF
- Texto cifrado: 0xF5BF8106F9AD3EEC

An lisis de rendimiento:
- Velocidad de procesamiento: ~25MB/s
- Resistencia a criptoan lisis lineal: Alta
- Resistencia a criptoan lisis diferencial: Alta
```

Listing 3: Resultado de Cifrado IDEA

Análisis Criptográfico

Evaluación de IDEA en CrypTool El análisis del cifrado IDEA mediante CrypTool revela:

- **Seguridad:** Alta resistencia a ataques conocidos, con una clave efectiva de 128 bits sin debilidades estructurales identificadas.
- **Rendimiento:** Eficiencia computacional superior a DES, aunque inferior a AES en implementaciones modernas.
- **Características distintivas:** Uso de operaciones matemáticas mixtas que dificultan el criptoanálisis.

3.3. DESCrack: Herramienta de Criptoanálisis

DESCrack es una utilidad especializada en criptoanálisis del algoritmo DES, mostrando sus vulnerabilidades y métodos de ataque.

Figura 2: Interfaz de DESCrack mostrando un ataque en progreso

Método Criptográfico

Funcionalidades de DESCrack

- **Ataques de Fuerza Bruta:** Implementación optimizada para búsqueda exhaustiva de claves DES.

- **Criptoanálisis Diferencial:** Demostración de ataques basados en diferencias entre pares de textos cifrados.
- **Debilidades de Claves:** Detección de claves débiles y semidébiles.
- **Paralelización:** Capacidad para distribuir el proceso de ataque entre múltiples núcleos.

```
Ataque de fuerza bruta a DES:
- Texto conocido: "Laboratorio"
- Texto cifrado: 0xA67D3F0DE8B14C52
- Espacio de b squeda te rico: 2^56 claves
- Claves probadas: 3,267,529,472 (0.046% del espacio total)
- Clave encontrada: 0x133457799BBCDFF1
- Tiempo total: 4h 32m 17s
- Velocidad: 200,000 claves/segundo
```

Listing 4: Resultado de Ataque a DES

Análisis Criptográfico

Evaluación de Seguridad de DES El análisis con DESCrack confirma las vulnerabilidades conocidas de DES:

- **Longitud de Clave Insuficiente:** Con 56 bits efectivos, es vulnerable a ataques de fuerza bruta con hardware moderno.
- **Estructura Regular:** La estructura de red de Feistel con 16 rondas identicas facilita ciertos tipos de criptoanálisis.
- **Debilidades en S-Boxes:** Algunas propiedades estadísticas de las S-Boxes pueden aprovecharse en ataques avanzados.
- **Conclusión:** DES debe considerarse obsoleto para aplicaciones que requieren alta seguridad, siendo preferibles AES o 3DES.

4. Criptoanálisis de Sustitución Monoalfabética

Análisis Criptográfico

Fundamentos del Criptoanálisis de Frecuencias El criptoanálisis de frecuencias es una técnica fundamental para romper cifrados de sustitución monoalfabética. Se basa en un principio lingüístico simple pero poderoso: en cada idioma, ciertas letras y combinaciones de letras aparecen con frecuencias predecibles y consistentes.

Por ejemplo, en español:

- Las letras más frecuentes son E (13.68 %), A (12.53 %), O (8.68 %) y S (7.98 %)
- Las letras menos frecuentes son K (0.00 %), W (0.02 %), X (0.22 %) y Z (0.52 %)
- Los bigramas más comunes incluyen ES, EN, EL, DE, LA

- Los trigramas más comunes incluyen QUE, EST, DEL, LOS, LAS

Estas regularidades estadísticas constituyen una "huella digital"del idioma que persiste incluso cuando el texto ha sido cifrado mediante sustitución simple.

4.1. Metodología de Ataque

Método Criptográfico

Proceso de Descifrado Estadístico El enfoque sistemático para romper un cifrado monoalfabético consta de las siguientes etapas:

1. **Análisis de frecuencia de caracteres**: Se contabiliza la frecuencia de aparición de cada símbolo en el texto cifrado.
2. **Mapeo inicial por frecuencias**: Se establece una correspondencia tentativa entre los símbolos cifrados y las letras del alfabeto, basándose en la distribución de frecuencias del idioma objetivo.
3. **Identificación de patrones**: Se buscan patrones que podrían corresponder a palabras cortas comunes (artículos, preposiciones, conjunciones).
4. **Refinamiento del mapeo**: Se ajusta el mapeo inicial utilizando el conocimiento de la estructura léxica del idioma.
5. **Optimización mediante algoritmos heurísticos**: Se aplican técnicas como hill climbing o recocido simulado para mejorar iterativamente la solución.
6. **Validación lingüística**: Se evalúa la coherencia del texto descifrado mediante el análisis de n-gramas y la verificación contra diccionarios.

El proceso no es puramente secuencial, sino iterativo, con retroalimentación constante entre las diferentes etapas.

4.2. Implementación Algorítmica

La implementación de un decodificador automático para cifrados monoalfabéticos requiere combinar técnicas estadísticas con heurísticas de optimización. A continuación se describe cada componente clave:

Buena Práctica

Análisis de Frecuencias

El primer paso consiste en analizar la distribución estadística de los símbolos en el texto cifrado:

1. Se filtran los caracteres relevantes del texto cifrado
2. Se cuenta la frecuencia de cada símbolo
3. Se convierten los conteos a porcentajes

4. Se ordenan los símbolos por frecuencia descendente

Esto proporciona una primera aproximación a la posible correspondencia entre símbolos cifrados y letras del alfabeto español.

Análisis Criptográfico

Mapeo Inicial y Patrones El mapeo inicial se crea comparando la distribución de frecuencias del texto cifrado con la distribución conocida del español:

$$\text{mapeo\_inicial} = \{s_1 \to l_1, s_2 \to l_2, \ldots, s_n \to l_n\}$$

donde s_i es el i-ésimo símbolo más frecuente en el texto cifrado y l_i es la i-ésima letra más frecuente en español.

Este mapeo se refina identificando patrones que podrían corresponder a palabras comunes. Por ejemplo, si encontramos que una secuencia de símbolos como "XX.ªparece frecuentemente en posiciones donde esperaríamos palabras como "DE", .EL.º "LA", podemos ajustar el mapeo para que refleje esta observación.

Alerta de Seguridad

Desafíos del Mapeo Inicial

El mapeo basado únicamente en frecuencias individuales suele ser insuficiente porque:

- Textos cortos pueden tener distribuciones de frecuencia atípicas
- Textos especializados pueden desviarse de la distribución promedio del idioma
- La distribución exacta varía según el estilo, período y tema del texto

Por esto, es crucial complementar el análisis de frecuencias con técnicas de optimización que exploren sistemáticamente el espacio de posibles soluciones.

4.3. Optimización mediante Hill Climbing

Método Criptográfico

Algoritmo de Hill Climbing con Simulated Annealing Para refinar el mapeo inicial, se implementa un algoritmo de hill climbing mejorado con simulated annealing:

$$\text{Coherencia}(M) = f(\text{bigramas}, \text{trigramas}, \text{patrones\_léxicos}) \tag{3}$$

$$\Delta = \text{Coherencia}(M_{\text{nuevo}}) - \text{Coherencia}(M_{\text{actual}}) \tag{4}$$

$$P(\text{aceptar}) = \begin{cases} 1 & \text{si } \Delta > 0 \\ e^{\Delta/T} & \text{si } \Delta \leq 0 \end{cases} \tag{5}$$

donde:

- M representa un mapeo entre símbolos cifrados y letras

- T es la temperatura, que disminuye gradualmente a lo largo de las iteraciones
- f es una función que evalúa la coherencia lingüística del texto descifrado

El algoritmo:

1. Comienza con el mapeo inicial basado en frecuencias
2. En cada iteración, selecciona aleatoriamente dos símbolos y intercambia sus mapeos
3. Evalúa la coherencia del texto descifrado con el nuevo mapeo
4. Acepta el cambio si mejora la coherencia o, con cierta probabilidad, incluso si la empeora (para escapar de óptimos locales)
5. Disminuye gradualmente la temperatura, reduciendo la probabilidad de aceptar cambios que empeoren la solución
6. Continúa hasta alcanzar un criterio de parada (número máximo de iteraciones o umbral de coherencia)

4.4. Evaluación de Coherencia

Análisis Criptográfico

Métricas de Coherencia Lingüística La clave del éxito en el algoritmo de optimización es la función que evalúa la coherencia del texto descifrado. Esta función combina varias métricas:

$$\text{Coherencia} = w_1 \cdot \frac{|\text{Bigramas}_{\text{encontrados}}|}{|\text{Bigramas}_{\text{esperados}}|} + w_2 \cdot \frac{|\text{Trigramas}_{\text{encontrados}}|}{|\text{Trigramas}_{\text{esperados}}|} + \tag{6}$$

$$w_3 \cdot \text{FreqScore} + w_4 \cdot \text{DictScore} \tag{7}$$

donde:

- $\text{Bigramas}_{\text{encontrados}}$ y $\text{Trigramas}_{\text{encontrados}}$ son los n-gramas del texto descifrado que coinciden con n-gramas comunes en español.
- FreqScore evalúa si los n-gramas más frecuentes en el texto descifrado son también frecuentes en español.
- DictScore mide el porcentaje de palabras descifradas que aparecen en un diccionario español.
- w_1, w_2, w_3, w_4 son pesos que determinan la importancia relativa de cada componente.

Esta función multicriterio permite capturar diferentes aspectos de la coherencia lingüística, aumentando la robustez del algoritmo.

4.5. Resultados Experimentales

Buena Práctica

Caso de Estudio: Descifrado de un Texto sobre Navegadores Web

Al aplicar nuestro enfoque automático a un texto cifrado mediante sustitución monoalfabética, se logró obtener un descifrado con un 86.90 % de coherencia. El texto resultante trataba sobre navegadores web y su rendimiento en diferentes sistemas operativos:

> ACTUALMENTE DISPONEMOS DE UNA GRAN CANTIDAD DE NAVEGADORES WEB PARA ELEGIR EL NUESTRO CONCEPTOS COMO LA SEGURIDAD O EL CUMPLIMIENTO DE LOS ESTÁNDARES DEL WC WORLD WIDE WEB CONSORTIUM...

El algoritmo identificó correctamente la mayoría de las sustituciones, como se puede observar en el mapeo parcial obtenido:

```
| ! -> S | # -> U | % -> J | ( -> G | ) -> R | * -> C | @ -> P |
| ^ -> T | a -> A | b -> I | c -> v | d -> f | f -> X | g -> h |
| h -> M | i -> b | j -> y | k -> E | l -> Q | m -> O | o -> x |
| p -> N | q -> H | r -> w | s -> L | t -> D |
```

Sin embargo, algunas sustituciones requirieron corrección manual posterior, particularmente en letras menos frecuentes o con contextos ambiguos:

```
| Y -> V | F -> W | K -> B | B -> X | Z -> Y | W -> H | V -> F |
```

Esta fase de refinamiento manual es común en el criptoanálisis práctico, donde el conocimiento lingüístico humano complementa el análisis algorítmico para resolver ambigüedades y mejorar la calidad del descifrado.

Análisis Criptográfico

Análisis del Contenido Descifrado El texto descifrado resultó ser un artículo comparativo sobre la velocidad de diferentes navegadores web en distintos sistemas operativos:

- Opera, un navegador noruego, se identificó como el más rápido en la mayoría de las pruebas.
- Firefox, que había reclamado ser el más rápido, mostró resultados inconsistentes.
- Internet Explorer .ªguantó bien", contradiciendo a sus críticos.
- Se analizaron resultados en Windows, Linux (distribución SUSE) y Mac OS X.
- Para Mac, Safari (de Apple) y Camino (del proyecto Mozilla) también obtuvieron buenos resultados en algunas categorías.

Este contenido técnico, con términos específicos como renderización", ÇSSƶ nombres de productos, representó un desafío adicional para el algoritmo de descifrado debido a la presencia de palabras que no siguen los patrones típicos del español general.

4.6. Consideraciones y Limitaciones

Advertencia

Factores que Afectan el Rendimiento del Descifrado

El rendimiento del algoritmo puede verse afectado por diversos factores:

- **Terminología especializada**: Palabras técnicas como "browser", renderización.º nombres propios como .ºperaz "Mozilla"pueden complicar el análisis basado en frecuencias generales del español.
- **Ambigüedad en letras poco frecuentes**: Letras como W, X, Y, Z son particularmente difíciles de mapear correctamente mediante análisis automatizado debido a su baja frecuencia en español.
- **Ausencia de contexto semántico**: El algoritmo no comprende el significado del texto, lo que limita su capacidad para resolver ambigüedades que un humano podría detectar fácilmente por contexto.
- **Variaciones ortográficas**: El texto contiene términos en inglés y nombres propios que no siguen los patrones ortográficos del español.

Estos factores explican por qué fue necesario el refinamiento manual de algunas sustituciones para alcanzar un texto completamente legible.

5. Conclusiones

Método Criptográfico

El Valor del Análisis Combinado El caso práctico presentado demuestra la efectividad de combinar enfoques algorítmicos y humanos en el criptoanálisis:

1. **Automatización eficiente**: El algoritmo logró descifrar aproximadamente el 87 % del texto de forma autónoma, reduciendo drásticamente el tiempo y esfuerzo requeridos.
2. **Intervención humana estratégica**: La corrección manual de un pequeño subconjunto de sustituciones completó el descifrado, aprovechando la comprensión contextual que los algoritmos actuales aún no poseen.
3. **Enfoque práctico**: Este método híbrido representa el enfoque real utilizado por los criptoanalistas: herramientas automatizadas para el trabajo pesado y análisis humano para las decisiones finales.

Este exitoso descifrado ilustra por qué los cifrados de sustitución monoalfabética, aunque históricamente importantes, son considerados inseguros en la criptografía moderna. A diferencia de algoritmos como DES, que requieren enfoques computacionalmente intensivos para ser quebrados, los cifrados monoalfabéticos sucumben ante análisis estadísticos relativamente sencillos combinados con optimización heurística.

6. Descifrado de criptogramas

6.1. Sexto mensaje - Datos

Se nos proporciona la siguiente información:

- $n = 1903$
- $p = 173$
- $q = 11$
- $d = 71$
- $z = (p-1)(q-1) = (173-1)(11-1) = 1720$

Y el siguiente mensaje cifrado:

```
1497 1583 791 1497 1394
791 1583 123 937 786
1583
```

6.2. Cálculo del inverso multiplicativo módulo z

Necesitamos encontrar e tal que $e \cdot d \equiv 1 \pmod{z}$, es decir, el inverso multiplicativo de d módulo z.

Para ello utilizaremos el algoritmo de Euclides extendido:

Inicialización del algoritmo de Euclides extendido:

Estamos buscando e tal que $e \cdot 71 \equiv 1 \pmod{1720}$

Inicializamos las variables según el algoritmo:

$g_0 = 1720$, $g_1 = 71$

$u_0 = 1$, $u_1 = 0$

$v_0 = 0$, $v_1 = 1$

Iteración 1:

Calculamos $y_2 = \lfloor g_0/g_i \rfloor = \lfloor 1720/71 \rfloor = 24$

Calculamos $g_2 = g_0 - y_2 \cdot g_i = 1720 - 24 \cdot 71 = 16$

Calculamos $u_2 = u_0 - y_2 \cdot u_i = 1 - 24 \cdot 0 = 1$

Calculamos $v_2 = v_0 - y_2 \cdot v_i = 0 - 24 \cdot 1 = -24$

Iteración 2:

Calculamos $y_3 = \lfloor g_1/g_i \rfloor = \lfloor 71/16 \rfloor = 4$

Calculamos $g_3 = g_1 - y_3 \cdot g_i = 71 - 4 \cdot 16 = 7$

Calculamos $u_3 = u_1 - y_3 \cdot u_i = 0 - 4 \cdot 1 = -4$

Calculamos $v_3 = v_1 - y_3 \cdot v_i = 1 - 4 \cdot -24 = 97$

Iteración 3:

Calculamos $y_4 = \lfloor g_2/g_i \rfloor = \lfloor 16/7 \rfloor = 2$

Calculamos $g_4 = g_2 - y_4 \cdot g_i = 16 - 2 \cdot 7 = 2$

Calculamos $u_4 = u_2 - y_4 \cdot u_i = 1 - 2 \cdot -4 = 9$

Calculamos $v_4 = v_2 - y_4 \cdot v_i = -24 - 2 \cdot 97 = -218$

Iteración 4:

Calculamos $y_5 = \lfloor g_3/g_i \rfloor = \lfloor 7/2 \rfloor = 3$

Calculamos $g_5 = g_3 - y_5 \cdot g_i = 7 - 3 \cdot 2 = 1$

Calculamos $u_5 = u_3 - y_5 \cdot u_i = -4 - 3 \cdot 9 = -31$
Calculamos $v_5 = v_3 - y_5 \cdot v_i = 97 - 3 \cdot -218 = 751$
Iteración 5:
Calculamos $y_6 = \lfloor g_4/g_i \rfloor = \lfloor 2/1 \rfloor = 2$
Calculamos $g_6 = g_4 - y_6 \cdot g_i = 2 - 2 \cdot 1 = 0$
Calculamos $u_6 = u_4 - y_6 \cdot u_i = 9 - 2 \cdot -31 = 71$
Calculamos $v_6 = v_4 - y_6 \cdot v_i = -218 - 2 \cdot 751 = -1720$
Resultado final:
El algoritmo ha terminado porque $g_5 = 0$
El inverso se encuentra en el valor de v en el penúltimo paso: $v_4 = 751$
Como $v_4 \geq 0$, no es necesario ajustar el valor.
Por lo tanto, el inverso multiplicativo de 71 módulo 1720 es 751
Verificación: $71 \cdot 751 \equiv 1 \pmod{1720}$

i	y_i	g_i	u_i	v_i
0	-	1720	1	0
1	-	71	0	1
2	24	16	1	-24
3	4	7	-4	97
4	2	2	9	-218
5	3	1	-31	751
6	2	0	71	-1720

Cuadro 1: Cálculo del inverso modular utilizando el algoritmo de Euclides extendido

Por lo tanto, $e = 751$.

6.3. Descifrado del mensaje

Para descifrar el mensaje, utilizamos la fórmula $M = C^e \bmod n$, donde $e = 751$ y $n = 1903$.

6.3.1. Proceso de descifrado

$C = 1497$: $M = 1497^{751} \bmod 1903 = 67$ (ASCII: 'C')
$C = 1583$: $M = 1583^{751} \bmod 1903 = 65$ (ASCII: 'A')
$C = 791$: $M = 791^{751} \bmod 1903 = 76$ (ASCII: 'L')
$C = 1497$: $M = 1497^{751} \bmod 1903 = 67$ (ASCII: 'C')
$C = 1394$: $M = 1394^{751} \bmod 1903 = 85$ (ASCII: 'U')
$C = 791$: $M = 791^{751} \bmod 1903 = 76$ (ASCII: 'L')
$C = 1583$: $M = 1583^{751} \bmod 1903 = 65$ (ASCII: 'A')
$C = 123$: $M = 123^{751} \bmod 1903 = 68$ (ASCII: 'D')
$C = 937$: $M = 937^{751} \bmod 1903 = 79$ (ASCII: 'O')
$C = 786$: $M = 786^{751} \bmod 1903 = 82$ (ASCII: 'R')
$C = 1583$: $M = 1583^{751} \bmod 1903 = 65$ (ASCII: 'A')

6.3.2. Mensaje descifrado

El mensaje descifrado es:

`CALCULADORA`

C	M	ASCII
1497	67	'C'
1583	65	'A'
791	76	'L'
1497	67	'C'
1394	85	'U'
791	76	'L'
1583	65	'A'
123	68	'D'
937	79	'O'
786	82	'R'
1583	65	'A'

Cuadro 2: Descifrado RSA: $m = c^e$ mód n

6.4. Séptimo mensaje - Datos

Se nos proporciona la siguiente información:

- $n = 2581$
- $p = 29$
- $q = 89$
- $d = 5$
- $z = (p-1)(q-1) = (29-1)(89-1) = 2464$

Y el siguiente mensaje cifrado:

```
1236 2131 15 2131 202 2069 1035 2069 1104 662
```

6.5. Cálculo del inverso multiplicativo módulo z

Necesitamos encontrar e tal que $e \cdot d \equiv 1 \pmod{z}$, es decir, el inverso multiplicativo de d módulo z. Para ello utilizaremos el algoritmo de Euclides extendido:

Inicialización del algoritmo de Euclides extendido:
Estamos buscando e tal que $e \cdot 5 \equiv 1 \pmod{2464}$
Inicializamos las variables según el algoritmo:
$g_0 = 2464$, $g_1 = 5$ $u_0 = 1$, $u_1 = 0$ $v_0 = 0$, $v_1 = 1$
Iteración 1:
Calculamos $y_2 = \lfloor g_0/g_i \rfloor = \lfloor 2464/5 \rfloor = 492$
Calculamos $g_2 = g_0 - y_2 \cdot g_i = 2464 - 492 \cdot 5 = 4$
Calculamos $u_2 = u_0 - y_2 \cdot u_i = 1 - 492 \cdot 0 = 1$
Calculamos $v_2 = v_0 - y_2 \cdot v_i = 0 - 492 \cdot 1 = -492$
Iteración 2:
Calculamos $y_3 = \lfloor g_1/g_i \rfloor = \lfloor 5/4 \rfloor = 1$
Calculamos $g_3 = g_1 - y_3 \cdot g_i = 5 - 1 \cdot 4 = 1$
Calculamos $u_3 = u_1 - y_3 \cdot u_i = 0 - 1 \cdot 1 = -1$

Calculamos $v_3 = v_1 - y_3 \cdot v_i = 1 - 1 \cdot -492 = 493$
Iteración 3:
Calculamos $y_4 = \lfloor g_2/g_i \rfloor = \lfloor 4/1 \rfloor = 4$
Calculamos $g_4 = g_2 - y_4 \cdot g_i = 4 - 4 \cdot 1 = 0$
Calculamos $u_4 = u_2 - y_4 \cdot u_i = 1 - 4 \cdot -1 = 5$
Calculamos $v_4 = v_2 - y_4 \cdot v_i = -492 - 4 \cdot 493 = -2464$
Resultado final:
El algoritmo ha terminado porque $g_3 = 0$
El inverso se encuentra en el valor de v en el penúltimo paso: $v_2 = 493$
Como $v_2 \geq 0$, no es necesario ajustar el valor.
Por lo tanto, el inverso multiplicativo de 5 módulo 2464 es 493
Verificación: $5 \cdot 493 \equiv 1 \pmod{2464}$

i	y_i	g_i	u_i	v_i
0	-	2464	1	0
1	-	5	0	1
2	492	4	1	-492
3	1	1	-1	493
4	4	0	5	-2464

Cuadro 3: Cálculo del inverso modular utilizando el algoritmo de Euclides extendido

Por lo tanto, $e = 493$.

6.6. Descifrado del mensaje

Para descifrar el mensaje, utilizamos la fórmula $M = C^e \bmod n$, donde $e = 493$ y $n = 2581$.

6.6.1. Proceso de descifrado

$C = 1236$: $M = 1236^{493} \bmod 2581 = 84$ (ASCII: 'T')
$C = 2131$: $M = 2131^{493} \bmod 2581 = 69$ (ASCII: 'E')
$C = 15$: $M = 15^{493} \bmod 2581 = 76$ (ASCII: 'L')
$C = 2131$: $M = 2131^{493} \bmod 2581 = 69$ (ASCII: 'E')
$C = 202$: $M = 202^{493} \bmod 2581 = 86$ (ASCII: 'V')
$C = 2069$: $M = 2069^{493} \bmod 2581 = 73$ (ASCII: 'I')
$C = 1035$: $M = 1035^{493} \bmod 2581 = 83$ (ASCII: 'S')
$C = 2069$: $M = 2069^{493} \bmod 2581 = 73$ (ASCII: 'I')
$C = 1104$: $M = 1104^{493} \bmod 2581 = 79$ (ASCII: 'O')
$C = 662$: $M = 662^{493} \bmod 2581 = 78$ (ASCII: 'N')

6.6.2. Mensaje descifrado

El mensaje descifrado es:

`TELEVISION`

C	M	ASCII
1236	84	'T'
2131	69	'E'
15	76	'L'
2131	69	'E'
202	86	'V'
2069	73	'I'
1035	83	'S'
2069	73	'I'
1104	79	'O'
662	78	'N'

Cuadro 4: Descifrado RSA: $m = c^e$ mód n

7. Conclusiones

El estudio detallado del algoritmo DES ha permitido comprender la estructura y funcionamiento de uno de los cifrados simétricos más influyentes en la historia de la criptografía. Aunque actualmente se considera obsoleto para aplicaciones de alta seguridad, sus principios de diseño continúan informando el desarrollo de algoritmos modernos.

La exploración de herramientas criptográficas del Criptolab ha demostrado la importancia de contar con recursos especializados tanto para fines educativos como para evaluación de seguridad. JCrypTool, CrypTool y DESCrack ofrecen perspectivas complementarias sobre la implementación, visualización y vulnerabilidades de diversos algoritmos criptográficos.

El análisis de frecuencias, aunque limitado en su aplicación a cifrados modernos, sigue siendo una técnica fundamental para comprender los principios básicos del criptoanálisis y para abordar ciertos tipos de cifrados históricos o básicos. Esta técnica ilustra la importancia de diseñar sistemas criptográficos que resistan al análisis estadístico.

Los hallazgos de este laboratorio refuerzan la máxima de que la seguridad criptográfica no debe depender del desconocimiento del algoritmo (principio de Kerckhoffs), sino de la robustez matemática del diseño y la gestión adecuada de claves. Asimismo, se evidencia la necesidad de evolución constante en el campo, dado que los avances en capacidad computacional y técnicas de ataque hacen que sistemas considerados seguros en el pasado se vuelvan vulnerables con el tiempo.

Buena Práctica

Recomendaciones prácticas para la implementación de sistemas criptográficos:

- Utilizar algoritmos estandarizados y ampliamente revisados como AES en lugar de crear implementaciones propias.
- Mantener longitudes de clave adecuadas según el nivel de seguridad requerido (mínimo 128 bits para cifrado simétrico, 2048 bits para RSA).
- Implementar mecanismos robustos de gestión de claves, incluyendo generación, almacenamiento y renovación periódica.
- Combinar múltiples capas de seguridad, evitando depender exclusivamente de

un único mecanismo criptográfico.

Agradecimientos

Los autores agradecen al profesor Gustavo A. Isaza por su guía y apoyo durante el desarrollo de este laboratorio, así como al Departamento de Ingeniería en Sistemas y Computación de la Universidad de Caldas por proporcionar los recursos necesarios para la realización de estas pruebas.

Referencias

[1] National Bureau of Standards. "Data Encryption Standard". Federal Information Processing Standards Publication 46. 1977.

[2] Schneier, B. "Applied Cryptography: Protocols, Algorithms, and Source Code in C". John Wiley & Sons, 2007.

[3] Criptored. "Repositorio de Herramientas Criptográficas". `http://www.criptored.upm.es/paginas/software.htm`, 2023.

[4] CrypTool Project. "CrypTool Documentation". `https://www.cryptool.org/`, 2023.

[5] JCrypTool Team. "JCrypTool - Eclipse-based Crypto Toolkit". `https://www.cryptool.org/en/jct/`, 2023.

[6] Singh, S. "The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography". Anchor Books, 2000.

[7] Lai, X., Massey, J. "A Proposal for a New Block Encryption Standard". Advances in Cryptology — EUROCRYPT '90. Lecture Notes in Computer Science, 1991.

[8] Katz, J., Lindell, Y. "Introduction to Modern Cryptography". CRC Press, 2020.

A. Tablas Completas del Algoritmo DES

A.1. Tablas de Permutación

A.2. Tabla de Expansión E

A.3. Tablas S-Box Completas

A.4. Tabla de Permutación P

Cuadro 5: Tablas de Permutación Completas

Permutación Inicial (IP)							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7
Permutación Final (IP^{-1})							
40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Cuadro 6: Tabla de Expansión E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Cuadro 7: S-Box 1

S1															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Cuadro 8: Tabla de Permutación P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25