

Universidad Nacional Autónoma de México
Facultad de Ciencias

Asignatura: Redes de computadoras
Semestre: 2024-1

Profesor: Javier León Cotonieto

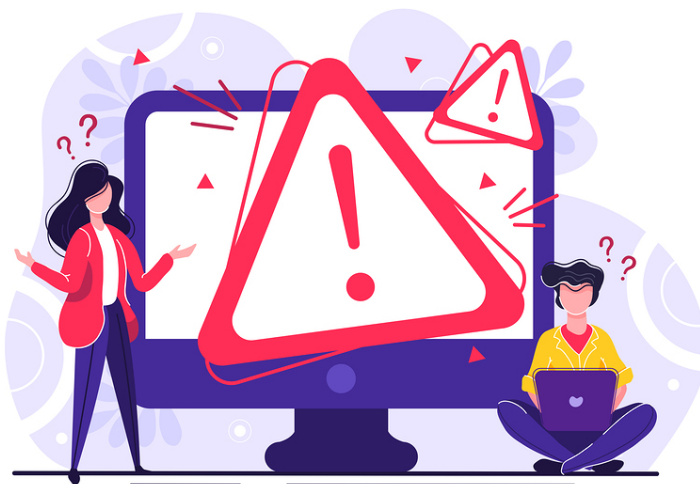
Ayudantes: Magdalena Reyes Granados
Itzel Gómez Muñoz
Sandra Plata Velázquez

***Práctica 4. "Implementación de algoritmos de
detección de errores de capa de enlace."***

Equipo 5

Integrantes:

- **Almanza Torres José Luis**
- **Jimenez Reyes Abraham**
- **Martínez Pardo Esaú**



DETECCIÓN DE ERRORES

PARIDAD

i. Calcula la paridad par de las siguientes cadenas:

a) 1010111

Bit de Paridad	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
1	1	0	1	0	1	1	1

b) 1000010

Bit de Paridad	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	1	0	0	0	0	1	0

c) 1001101 1111000

BP	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	1	0	0	1	1	0	1	1	1	1	1	0	0	0

d) 1111111 0000001

BP	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	1	1	1	1	1	1	1	0	0	0	0	0	0	1

ii. Calcula la paridad impar de las siguientes cadenas:

a) 0101011

Bit de Paridad	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
1	0	1	0	1	0	1	1

b) 0000011

Bit de Paridad	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
1	0	0	0	0	0	1	1

c) 0010111 0011111

BP	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
0	0	0	1	0	1	1	1	0	0	1	1	1	1	1

d) 1111110 0111111

BP	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1
1	1	1	1	1	1	1	0	0	1	1	1	1	1	1

SUMA DE VERIFICACIÓN (CHECKSUM)

Ejercicio Práctico .

Cada equipo tendrá que realizar las siguientes actividades explicando y con capturas de pantalla de lo siguiente:


i. Crear un archivo (del tipo, tamaño y extensión que prefieran).

Creamos un nuevo archivo llamado x.txt.

ii. Investigar cuál es el procedimiento (comando, aplicación, etc..) a realizar para obtener el checksum de su archivo mediante consola/terminal.

En linux (ubuntu)

1. Abrimos una ventana del terminal.
2. Escribimos el comando md5sum [nombre del tipo de archivo con extensión][ruta del archivo].
3. Pulsamos enter y aparecerá.



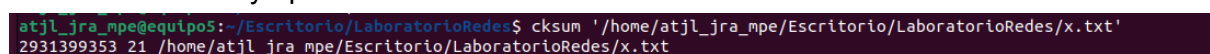
```
abraham@j1me-L: ~/Escritorio/LaboratorioRedes
[\ ATJL_JRA_MPE@j1me-L LaboratorioRedes]$ md5sum '/home/abraham/Escritorio/LaboratorioRedes/x.txt'
28882a66eb73d867b898ed71e3d443ea  /home/abraham/Escritorio/LaboratorioRedes/x.txt
[\ ATJL_JRA_MPE@j1me-L LaboratorioRedes]$
```

iii. Obtener el checksum de su archivo con al menos 2 algoritmos diferentes. NOTA: No se pueden ocupar herramientas online (sitios de internet).

cksum: para comprobar la integridad de archivos de forma doméstica es más que suficiente. Tiene la ventaja de que es un algoritmo con un gran rendimiento y rápido.

En linux (ubuntu)

1. Abrimos una ventana del terminal.
2. Escribimos el comando cksum [nombre del tipo de archivo con extensión][ruta del archivo].
3. Pulsamos enter y aparecerá.



```
atjl_jra_mpe@equipo5:~/Escritorio/LaboratorioRedes$ cksum '/home/atjl_jra_mpe/Escritorio/LaboratorioRedes/x.txt'
2931399353 21 /home/atjl_jra_mpe/Escritorio/LaboratorioRedes/x.txt
```

sha256sum: SHA-256 es el reemplazo lógico del algoritmo MD5, ya que por el momento no se han descubierto vulnerabilidades.

En linux (ubuntu)

1. Abrimos una ventana del terminal.
2. Escribimos el comando sha256sum [nombre del tipo de archivo con extensión][ruta del archivo].
3. Pulsamos enter y aparecerá.

```
atjl_jra_mpe@equipo5: ~/Escritorio/LaboratorioRedes$ sha256sum '/home/atjl_jra_mpe/Escritorio/LaboratorioRedes/x.txt'
23a02974a3bd4d19b929dd97f77f1aee45646d6014ac467a6bd2b6af4c5762fe /home/atjl_jra_mpe/Escritorio/LaboratorioRedes/x.txt
```

iv. Enviar el archivo a través de internet a otro integrante del equipo, obtener el checksum del archivo recibido y compararlo con el original. NOTA: Realizar este procedimiento en un entorno Windows y Linux.

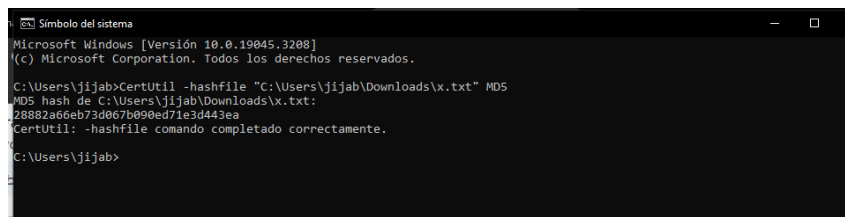
Ubuntu(Linux)



```
abraham@Jime-L: ~/Escritorio/LaboratorioRedes
[ \ ATJL_JRA_MPE@Jime-L LaboratorioRedes$ ]md5sum '/home/abraham/Escritorio/LaboratorioRedes/x.txt'
28882a66eb73d067b090ed71e3d443ea /home/abraham/Escritorio/LaboratorioRedes/x.txt
[ \ ATJL_JRA_MPE@Jime-L LaboratorioRedes$ ][]
```

28882a66eb73d067b090ed71e3d443ea

Windows.



```
Microsoft Windows [Versión 10.0.19045.3208]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\jijab>CertUtil -hashfile "C:\Users\jijab\Downloads\x.txt" MD5
MD5 hash de C:\Users\jijab\Downloads\x.txt:
28882a66eb73d067b090ed71e3d443ea
CertUtil: -hashfile comando completado correctamente.

C:\Users\jijab>
```

28882a66eb73d067b090ed71e3d443ea

v. Investigue y explique al menos 3 ejemplos de la implementación de esta comprobación.

- Protocolos de red: detecta errores de transmisión al momento de enviar paquetes por la red con protocolos como IPv4 o TCP/UDP, porque hay ocasiones en las que los paquetes no se envían de forma íntegra y la suma de verificación ayuda a saber si un paquete se mandó con errores.
- En la tecnología móvil: parte clave de la tecnología GSM a la hora de mejorar las condiciones de transmisión y recepción de datos para evitar un mal servicio de comunicación.
- Archivos ISO: cuando descargas una imagen de un sistema operativo es necesario que se mantenga su integridad, por lo que la empresa que desarrolla el sistema operativo facilita el valor del checksum de la imagen ISO, para que el usuario verifique que el archivo descargado es idéntico al de la empresa.
- Cuentas bancarias: donde ejerce un papel de herramienta de verificación de datos, por ejemplo, en los números de cuenta.
- Direcciones para criptomonedas: para comprobar la dirección de envío de los activos que se desean transferir y que su integridad sea correcta.

vi. Realice un programa en Python donde se ejemplifique la implementación del checksum. El programa debe de ser capaz de simular una comunicación Emisor-Receptor, es decir, de calcular el checksum de una cadena de datos y luego verificar si se ha producido un error durante la transmisión calculando nuevamente el checksum y comparándolo con el valor recibido. (25 pts).

Código

```
~/Escritorio/Redes de computadoras/sum.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
fas.txt x README.txt x lectura.py x sum.py x
1 import hashlib
2
3 """
4 Alumnos: Almanza Torres José Luis
5           Jiménez Reyes Abraham
6           Martínez Pardo Esaú
7
8 Practica para calcular el checksum
9
10 @since 15 de septiembre de 2023
11 """
12
13 # Función para calcular el checksum con SHA-256
14 def calcular_checksum_sha256(datos):
15     checksum = hashlib.sha256(datos.encode()).hexdigest()
16     return checksum
17
18 # Función para calcular el checksum con MD5
19 def calcular_checksum_md5(datos):
20     checksum = hashlib.md5(datos.encode()).hexdigest()
21     return checksum
22
23 # Función para calcular el checksum con sha512
24 def calcular_checksum_sha512(datos):
25     checksum = hashlib.sha512(datos.encode()).hexdigest()
26     return checksum
27
28 # Función para convertir un checksum a otro algoritmo
29 def convertir_checksum(checksum, nuevo_algoritmo):
30     if nuevo_algoritmo == "SHA512":
31         if len(checksum) % 2 != 0:
32             checksum = "0" + checksum
33         datos = bytes.fromhex(checksum)
34         nuevo_checksum = hashlib.sha512(datos).hexdigest()
35         return nuevo_checksum
36     else:
```

```
~/Escritorio/Redes de computadoras/sum.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
fas.txt x README.txt x lectura.py x sum.py x
46 def receptor(datos):
47     checksum_sha256 = calcular_checksum_sha256(datos)
48     print(f"Receptor - Datos recibidos: {datos}, Checksum 256 calculado: {checksum_sha256}")
49     return datos, checksum_sha256
50
51 # Leer datos desde un archivo de texto
52 try:
53     with open("cuento.txt", "r") as archivo:
54         datos_a_transmitir = archivo.read()
55 except FileNotFoundError:
56     print("El archivo 'cuento.txt' no se encontró.")
57     exit(1)
58
59 # Calcular el checksum SHA-256 de los datos
60 checksum_recibido = calcular_checksum_sha256(datos_a_transmitir)
61 print(f"Checksum SHA-256 calculado: {checksum_recibido}")
62
63 # Calcular el checksum MD5 de los datos
64 checksum_otro = calcular_checksum_md5(datos_a_transmitir)
65 print(f"Checksum MD5 calculado: {checksum_otro}")
66
67 # Convertir el checksum SHA-256 al nuevo algoritmo (SHA-512)
68 nuevo_checksum_sha512 = convertir_checksum(checksum_recibido, "SHA512")
69 print(f"Checksum SHA-256 convertido a SHA-512: {nuevo_checksum_sha512}")
70
71 # Convertir el checksum MD5 al nuevo algoritmo (SHA-512)
72 nuevo_checksum_md5_sha512 = convertir_checksum(checksum_otro, "SHA512")
73 print(f"Checksum MD5 convertido a SHA-512: {nuevo_checksum_md5_sha512}")
74
75
76 # Comparar los dos nuevos checksums SHA-512
77 if nuevo_checksum_sha512 == nuevo_checksum_md5_sha512:
78     print("Las conversiones y las comparaciones con SHA-512 fueron exitosas. Los checksums coinciden.")
79 else:
80     print("Error en las conversiones o las comparaciones. Los checksums SHA-512 no coinciden.")
81
```

Terminal

```

abraham@Jlme-L: ~/Escritorio/Redes de computadoras
[\\ ATJL_JRA_MPE@Jlme-L Redes de computadoras]$ python3 sum.py
Checksum SHA-256 calculado: d8d36aceb6df748e72eccd86f99097f99c4bc29408ee404efbaf7c744b73d2c0
Checksum SHA-256 calculado: d8d36aceb6df748e72eccd86f99097f99c4bc29408ee404efbaf7c744b73d2c0
Checksum SHA-256 convertido a SHA-512: 251133bfaada59d44d8b422622ec986d640a5cc281f02e0ab40c9ef645307c79
7bf1a8ae749682bc3cf3349b1ee7770f3a4823f6bac05cb4a3a9c47d40bcd315
Checksum SHA-256 convertido a SHA-512: 251133bfaada59d44d8b422622ec986d640a5cc281f02e0ab40c9ef645307c79
7bf1a8ae749682bc3cf3349b1ee7770f3a4823f6bac05cb4a3a9c47d40bcd315
Las conversiones y las comparaciones con SHA-1 fueron exitosas. Los checksums coinciden.
[\\ ATJL_JRA_MPE@Jlme-L Redes de computadoras]$

```

CÓDIGOS DE REDUNDANCIA CÍCLICA. (CRC)

i. Calcula el CRC con los siguientes datos y realiza la comprobación:

a) Trama: $x^8 + x^7 + x^6 + x^5 + x^2 + 1$, $G(x) = x^4 + x^2 + 1$

$$M = x^8 + x^7 + x^6 + x^5 + x^2 + 1$$

$$G(x) = x^4 + x^2 + 1$$

En este caso, convertimos primero la trama a binario.

x^8	x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0
1	1	1	1	0	0	1	0	1

Ahora si, la trama es: **111100101**

1. Observar el grado del polinomio generador y asignarlo a r.

$$r = 4$$

2. Añadir el número de 0 a la trama de acuerdo con el valor de r

$$x^r M(x) = 1111001010000$$

3. Convertir el polinomio generador a binario.

x^4	x^3	x^2	x^1	x^0
1	0	1	0	1

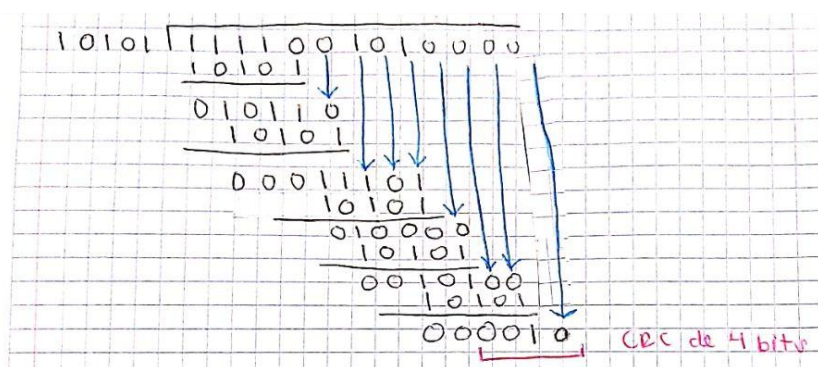
Polinomio generador en binario: **10101**

4. Para obtener CRC, se debe dividir $x^r M(x) / G(x)$

$$\text{CRC} = 1111001010000 / 10101$$

La división se realiza utilizando los valores de la compuerta lógica XOR.

El valor del cociente en la división no es tomado en cuenta.



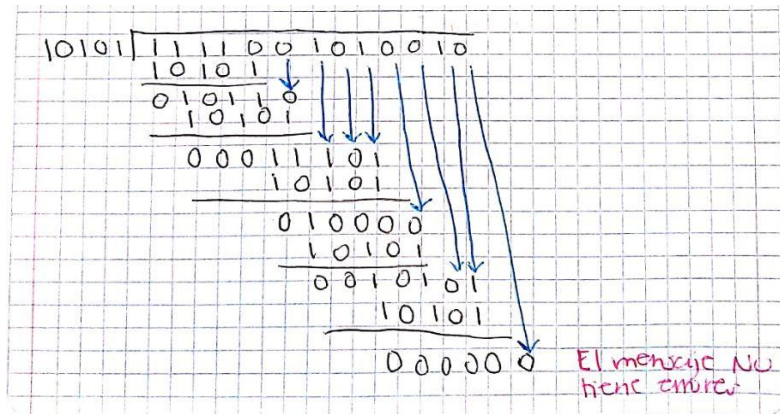
XOR		
Bit 1	Bit 2	Salida
0	0	0
0	1	1
1	0	1
1	1	0

5. El mensaje que se envía al receptor T se obtiene de unir la trama con el residuo de la división (CRC)

T = 1111001010010

6. Para comprobar que el mensaje no tuvo errores se realiza la división de T/G(x) y el residuo obtenido debe ser 0.

1111001010010 / 10101



XOR		
Bit 1	Bit 2	Salida
0	0	0
0	1	1
1	0	1
1	1	0

b) Trama: 110101111, $G(x) = x^4 + x + 1$

M = 110101111

$G(x) = x^4 + x + 1$

1. Observar el grado del polinomio generador y asignarlo a r.
r = 4

2. Añadir el número de 0 a la trama de acuerdo con el valor de r
 $x^r M(x) = 1101011110000$

3. Convertir el polinomio generador a binario.

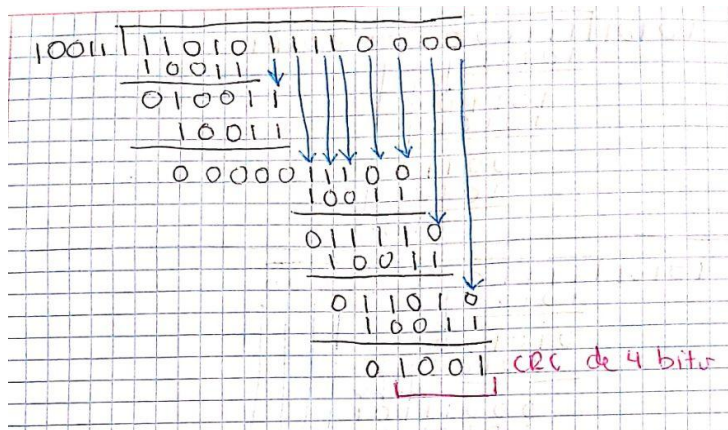
x^4	x^3	x^2	x^1	x^0
1	0	0	1	1

Polinomio generador en binario: 10011

4. Para obtener CRC, se debe dividir $x^r M(x) / G(x)$

CRC = 1101011110000 / 10011

La división se realiza utilizando los valores de la compuerta lógica XOR.
El valor del cociente en la división no es tomado en cuenta.



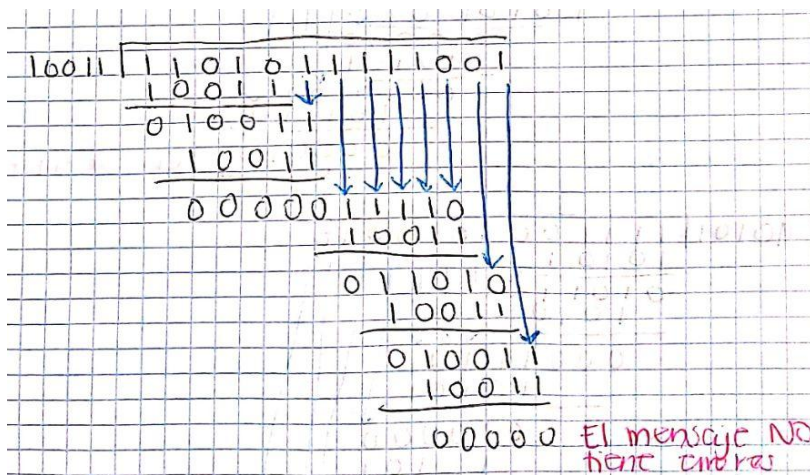
XOR		
Bit 1	Bit 2	Salida
0	0	0
0	1	1
1	0	1
1	1	0

5. El mensaje que se envía al receptor T se obtiene de unir la trama con el residuo de la división (CRC)

T = 1101011111001

6. Para comprobar que el mensaje no tuvo errores se realiza la división de T/G(x) y el residuo obtenido debe ser 0.

1101011111001 / 10011



XOR		
Bit 1	Bit 2	Salida
0	0	0
0	1	1
1	0	1
1	1	0

Nota: Para la entrega de los ejercicios deben colocar todo el procedimiento, se pueden realizar a mano y escanear o tomarle foto y anexarla en el reporte.

ii. Realiza un programa en Python donde se calcule el CRC (Valor 15pts).

a. La trama se debe mostrar en binario y $G(x)$ deberá mostrar tanto en polinomio como en binario.

b. Se debe mostrar el valor de r e indicar cuantos 0 se deben agregar.

c. Antes de realizar la división mostrar el valor de $x^r M(x)$

d. Calcular el valor de CRC.

e. Mostrar el valor que tiene T .

Nota: Para la evidencia en el reporte usen la información de los ejercicios anteriores (Trama y $G(x)$)

```
La trama es: 111100101
```

```
El polinomio generador es:  $1x^4 + 1x^2 + 1x^0 \Rightarrow 10101$ 
```

```
Se agregarán 4 bits 0 a la cadena.
```

```
El valor de  $x^r M(x)$  es: 1111001010000
```

```
===== Procedimiento de División =====
```

```
Divisor: 10101, dividendo: 11110
```

```
XOR: 01011
```

```
Var actual nueva : 10110
```

```
Divisor: 10101, dividendo: 10110
```

```
XOR: 00011
```

```
Var actual nueva : 11101
```

```
Divisor: 10101, dividendo: 11101
```

```
XOR: 01000
```

```
Var actual nueva : 10000
```

```
Divisor: 10101, dividendo: 10000
```

```
XOR: 00101
```

```
Var actual nueva : 10100
```

```
Divisor: 10101, dividendo: 10100
```

```
XOR: 00001
```

```
El valor de crc es: 0010
```

```
El valor de T es: 1111001010010
```

Por favor, ingresa una opción de un numero: 2

La trama es: 110101111

El polinomio generador es: $1x^4 + 1x^1 + 1x^0 \Rightarrow 10011$

Se agregarán 4 bits 0 a la cadena.

El valor de $x^M(x)$ es: 1101011110000

===== Procedimiento de División =====

Divisor: 10011, dividendo: 11010

XOR: 01001

Var actual nueva : 10011

Divisor: 10011, dividendo: 10011

XOR: 00000

Var actual nueva : 11100

Divisor: 10011, dividendo: 11100

XOR: 01111

Var actual nueva : 11110

Divisor: 10011, dividendo: 11110

XOR: 01101

El valor de crc es: 1010

El valor de T es: 1101011111010

```
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda ← → 🔍 Buscar
Pruebaaaaa.py X
C:\Users> Luis > Desktop > Pruebaaaaa.py > polinomio_a_binario
1 """
2 Implementación del algoritmo de comprobación de redundancia cíclica (CRC).
3 Equipo 5:
4 - José Luis Almanza Torres
5 - Abraham Jimenez Reyes
6 - Esai Martínez Pardo
7 """
8
9 # Función para transformar números en superíndices
10 def convertir_a_superindice(numero):
11     numeros_normales = "0123456789"
12     numeros_superindice = "0123456789"
13     tabla_de_conversion = numero.maketrans(numeros_normales, numeros_superindice)
14     return numero.translate(tabla_de_conversion)
15
16
17 # Veamos que aquí hacemos la representación del polinomio, para poder verificar que el procedimiento hecho
18 # en el ensayo concuerda con el del código
19 # Ejercicio a
20 trama_ejercicio_a = [1, 1, 1, 1, 0, 0, 1, 0, 1] # Trama del ejercicio a
21 polinomio_generador_ejercicio_a = [1, 0, 1, 0, 1] # Polinomio generador del ejercicio a
22
23 # Ejercicio b
24 trama_ejercicio_b = '110101111' # Trama del ejercicio b
25 polinomio_generador_ejercicio_b = [1, 0, 0, 1, 1] # Polinomio generador del ejercicio b
26
27
28
29 # Función que obtiene una representación legible de un polinomio
30 def obtener_polinomio_legible(polinomio):
31     grados = [convertir_a_superindice(str(i)) for i in range(len(polinomio) - 1, -1, -1)]
32     terminos = [f'coeficiente x {grado}' for coeficiente, grado in zip(polinomio, grados) if coeficiente != 0]
33     polinomio_legible = ' + '.join(terminos)
34     return polinomio_legible
35
36
37 # Función que convierte de un polinomio a un binario.
38 def polinomio_a_binario(polinomio):
39     binario = ''.join(map(str, polinomio))
40     return binario
```

```
Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda ← → 🔍 Buscar
Explorador (Ctrl+Mayús+E)
C:\Users> Luis > Desktop > Pruebaaaaa.py > polinomio_a_binario
38 def polinomio_a_binario(polinomio):
39     binario = ''.join(map(str, polinomio))
40     return binario
41
42
43 # Obtener el grado del polinomio generador
44 def polinomio_grado(polinomio):
45     return len(polinomio) - 1
46
47 # Función que nos ayuda a calcular el xrx
48 def obtener_xrx(trama, polinomio):
49     grado = polinomio_grado(polinomio) # Obtener el grado del polinomio generador
50
51     polinomio_generador_binario = polinomio_a_binario(polinomio)
52     trama_binaria = polinomio_a_binario(trama)
53     # Mostrar información relevante
54     print(f"La trama es: {trama_binaria}\n")
55     print(f"El polinomio generador es: {obtener_polinomio_legible(polinomio)} => {polinomio_generador_binario}\n")
56     print(f"Se agregarán {grado} bits 0 a la cadena.\n")
57     # Calcular x^n * M(x)
58     x_r_mult_message = trama_binaria + '0' * grado
59     # Mostrar el valor de x^n * M(x)
60     print(f"El valor de x(convertir_a_superindice("r"))M(x) es: {x_r_mult_message}\n")
61
62     return x_r_mult_message
63
64
65
66 def xor(a, b):
67     """
68     Realiza una operación XOR a nivel de bits entre dos cadenas binarias a y b.
69
70     Args:
71     a (str): Cadena binaria 1.
72     b (str): Cadena binaria 2.
73
74     Returns:
75     str: Resultado de la operación XOR entre a y b.
76     """
```

```
64
65
66 def xor(a, b):
67     """
68     Realiza una operación XOR a nivel de bits entre dos cadenas binarias a y b.
69
70     Args:
71     a (str): Cadena binaria 1.
72     b (str): Cadena binaria 2.
73
74     Returns:
75     str: Resultado de la operación XOR entre a y b.
76     """
77     if len(a) != len(b):
78         raise ValueError("Las cadenas deben tener la misma longitud para realizar XOR.")
79
80     result = ""
81     for i in range(len(a)):
82         if a[i] == b[i]:
83             result += '0'
84         else:
85             result += '1'
86     return result
87
88
89
90
91 def crc_div(trama, gen):
92     """
93     Realiza la división de trama entre el polinomio generador binario.
94
95     Args:
96     trama (str): Cadena binaria de la trama.
97     gen (str): Polinomio generador binario.
98
99     Returns:
100     str: Valor del CRC binario.
101     """
102
103     grado = polinomio_grado(gen)
```

```
149
150 def obten_t(trama, polinomio):
151     trama = polinomio_a_binario(trama)
152     crc = crc_div(trama, polinomio)
153     t = trama + crc
154     print("El valor de T es: " + t)
155     return
156
157
158 print("\n--- Programa para el cálculo de CRC ---\n")
159 print("Menú:\n")
160 print("Opciones disponibles:\n")
161 print("1. Calcular CRC del ejercicio 1")
162 print("2. Calcular CRC del ejercicio 2")
163 print("3. Ingresar trama y polinomio generador personalizados para su calculo")
164 print("=====")
165 print("\n")
166
167 opcion = input("Por favor, ingresa una opción de un numero: ")
168 print("\n")
169
170 if opcion == "1":
171     obten_t(trama_ejercicio_a, polinomio_generador_ejercicio_a)
172     exit()
173
174 elif opcion == "2":
175     obten_t(trama_ejercicio_b, polinomio_generador_ejercicio_b)
176     exit()
177
178 elif opcion == "3":
179     trama = input("Ingresa la trama en binario: ")
180     grado = int(input("Ingresa el grado del polinomio generador: "))
181     polinomio = []
182     literal = 0
183     auxGrado = grado
184     print("Ingresa '1' si existe la literal o '0' si no existe. Comienza desde el término de mayor grado hasta el término independiente.")
185     # Ingresar el polinomio generador literal a literal
186     for i in range(grado + 1):
187         literal = int(input("Ingresa el coeficiente del término x" + str(auxGrado) + ": "))
```

NOTA: las evidencias con respecto a los códigos de programación a entregar serán las siguientes:

- *En su reporte anexarán una captura de pantalla de su código completo con los elementos necesarios para evitar el plagio.
- * Capturas de pantalla de la ejecución del código mediante terminal, así como el resultado.
- *No es necesario que el programa cuente con una interfaz gráfica, si la quieren agregar es opcional.
- *En la plataforma deberán de anexar el código fuente o si su programa tiene interfaz gráfica, el ejecutable.
- *Si al momento de calificar el código se sospecha de plagio, el equipo será acreedor de una sanción. Dependiendo de la gravedad del asunto será desde penalización en la calificación como la anulación de la misma.

CONCLUSIONES

José Luis: En el desarrollo de esta práctica, pude apreciar cómo la detección de errores desempeña un papel muy importante en la transmisión y recepción de datos en sistemas digitales. Aprendí que existen varios algoritmos, como el CRC, el checksum y la paridad, que se utilizan para garantizar la integridad de los datos durante su transmisión. Estos algoritmos varían en complejidad, y cada uno tiene su lugar en diferentes aplicaciones. Además, me di cuenta de que la detección de errores no solo es importante en el ámbito académico, sino que también tiene un impacto significativo en nuestra vida cotidiana, desde la descarga de archivos hasta las transacciones bancarias. En última instancia, esta práctica destacó la importancia de la fiabilidad de la transmisión de datos en el mundo digital actual.

Abraham: Como lo vimos en clase y con esta práctica con suma de verificación es el método más común para detección de errores. Los checksums implican la generación de un valor resumen (checksum) a partir de los datos transmitidos y su verificación en el receptor. Si el checksum calculado no coincide con el checksum recibido, se asume que hay un error. También aprendí que tenemos distintos algoritmos que desconocía para el checksum y con la terminal podemos obtener el checksum de un archivo con solo un comando.

Esaú: En esta práctica aprendimos el uso e implementación de distintos algoritmos para la detección de errores en la capa de enlace, como el de paridad, que es una técnica de

errores simples, el Checksum, que es un método menos robusto y común entre estos algoritmos y el CRC para detectar errores en redes digitales y en dispositivos de almacenamiento, en este el procedimiento si fue un poco más laborioso al hacer la división binaria y la comprobación de que el mensaje no tuvo errores. También vimos usos comunes del checksum, esto nos permitió observar que algoritmos como estos, están presentes en nuestra vida cotidiana sin darnos cuenta, cómo al descargar una imagen, enviar un mensaje o hacer una transferencia bancaria. La detección de errores es importante pues nos permite detectar los cambios en la transmisión de datos y controlar el flujo de dichos datos.

REFERENCIAS

- Martín, I. (2023). Checksum: el pequeño algoritmo con grandes funciones de seguridad | roams. Recuperado el 17 de septiembre de 2023, de <https://finanzas.roams.es/academia/criptomonedas/checksum/>
- Voz Idea. (2017). Usar la terminal Linux para verificar la integridad de archivos descargados | Voz Idea. Recuperado el 17 de septiembre de 2023, de <https://www.vozidea.com/verificar-integridad-de-archivos-en-linux>