Bitácora Técnica del Sistema Esclavo
Descifrado de Imágenes con Caos

Sistema de Cifrado Esclavo - Raspberry Pi 4

13 de febrero de 2026

Índice

1. Introducción

Este documento describe el proceso completo de descifrado de imágenes implementado en el archivo `Esclavo_TLS KEYSTREAM.py`, el cual se ejecuta en una Raspberry Pi 4. El sistema esclavo tiene como función principal recibir información cifrada del sistema maestro y desencriptarla mediante el uso de sistemas caóticos: el mapa logístico y el oscilador de Rössler.

El proceso de descifrado es el inverso del proceso de cifrado realizado por el sistema maestro, utilizando técnicas criptográficas de **confusión** y **difusión**. La comunicación entre maestro y esclavo se realiza de manera segura mediante el protocolo MQTT con TLS como capa adicional de seguridad.

1.1. Contexto del Sistema

- **Plataforma:** Raspberry Pi 4
- **Imagen de prueba:** RGB de 250×250 píxeles
- **Tamaño del vector:** 187,500 elementos (250 × 250 × 3 canales)
- **Sistemas caóticos utilizados:**
  - Mapa logístico para generación de secuencias pseudoaleatorias
  - Oscilador de Rössler para generación del keystream caótico
- **Protocolo de comunicación:** MQTT sobre TLS (puerto 8883)

2. Fase 1: Conexión Segura mediante MQTT con TLS

2.1. Descripción del Proceso

El sistema esclavo inicia estableciendo una conexión segura con el broker MQTT utilizando el protocolo TLS (Transport Layer Security). Esta capa de seguridad garantiza que la comunicación entre el maestro y el esclavo sea confidencial e íntegra.

2.2. Parámetros de Conexión

Cuadro 1: Configuración de la conexión MQTT con TLS

Parámetro	**Valor**
Broker	raspberrypiJED.local
Puerto	8883 (MQTT sobre TLS)
Usuario	usuario1
Certificado CA	/home/tunchi/CifradoSlave/certs/ca.crt
QoS	1 (entrega garantizada)
Topic Keys	chaoskeystream/keys
Topic Data	chaoskeystream/data

2.3. Proceso de Conexión

1. **Configuración del cliente MQTT:** Se crea una instancia del cliente MQTT y se configuran las credenciales de autenticación (usuario y contraseña).
2. **Configuración TLS:** Se establece la capa TLS utilizando el certificado de la autoridad certificadora (CA). Esto permite verificar la identidad del broker y cifrar toda la comunicación.
3. **Establecimiento de conexión:** El cliente se conecta al broker en el puerto 8883, el puerto estándar para MQTT sobre TLS.
4. **Suscripción a topics:** Una vez establecida la conexión, el esclavo se suscribe a dos topics:
   - `chaoskeystream/keys`: Para recibir los parámetros del sistema maestro
   - `chaoskeystream/data`: Para recibir los datos cifrados y las trayectorias del oscilador
5. **Espera activa:** El sistema permanece en espera hasta recibir ambos mensajes (keys y data) del maestro.

2.4. Técnica Criptográfica Aplicada

En esta fase se aplica el principio de **comunicación segura** mediante:

- **Autenticación:** Verificación de la identidad del broker mediante certificados digitales
- **Confidencialidad:** Cifrado de la comunicación mediante TLS
- **Integridad:** Garantía de que los datos no han sido modificados durante la transmisión

3. Fase 2: Recepción y Extracción de Parámetros

3.1. Descripción del Proceso

Una vez establecida la conexión, el sistema esclavo recibe dos tipos de información del maestro:

3.1.1. Parámetros del Sistema (Keys)

Los parámetros recibidos incluyen las condiciones iniciales y constantes de los sistemas caóticos:

Cuadro 2: Parámetros del oscilador de Rössler recibidos

Parámetro	**Símbolo**	**Valor de Ejemplo**
Parámetro a	a	0.2
Parámetro b	b	0.2
Parámetro c	c	5.7

Cuadro 3: Parámetros del mapa logístico recibidos

Parámetro	**Símbolo**	**Valor de Ejemplo**
Tasa de crecimiento	$a_{\log}$	3.9899
Condición inicial	x_0	0.4000001

3.1.2. Datos Cifrados y Trayectorias

Los datos recibidos incluyen:

Cuadro 4: Información recibida del sistema maestro

Dato	**Descripción**
vector_cifrado	Vector de la imagen después de confusión (187,500 elementos)
x_maestro	Trayectoria x(t) del oscilador maestro
y_maestro	Trayectoria y(t) del oscilador maestro
z_maestro	Trayectoria z(t) del oscilador maestro
t_maestro	Vector de tiempos de la simulación
ancho	Ancho de la imagen (250 píxeles)
alto	Alto de la imagen (250 píxeles)
nmax	Tamaño total del vector (187,500)
tiempo_sinc	Tiempo de sincronización (iteraciones)
KEYSTREAM	Longitud del keystream caótico

3.2. Vector Cifrado Recibido

El vector cifrado que recibe el esclavo corresponde a los datos que ya pasaron por el proceso de confusión en el maestro. Este vector contiene los valores que resultan de la suma de tres componentes: la difusión escalada, el vector logístico y la trayectoria x del oscilador de Rössler.

Cuadro 5: Primeros 10 valores del vector cifrado recibido

Índice	**Valor Cifrado**
0	-7.828875
1	-7.281211
2	-8.072610
3	-7.693832
4	-7.270963
5	-7.207665
6	-7.070513
7	-6.642199
8	-7.358779
9	-7.006520

4. Fase 3: Sincronización del Oscilador de Rössler

4.1. Descripción del Proceso

La sincronización es un proceso fundamental para la recuperación del keystream caótico. El esclavo debe sincronizar su oscilador de Rössler con el oscilador maestro utilizando la señal $y(t)$ que recibió del maestro.

4.2. Ecuaciones del Oscilador Esclavo

El oscilador de Rössler en el esclavo se define mediante el siguiente sistema de ecuaciones diferenciales:

$$\frac{dx_s}{dt} = -y_s - z_s \tag{1}$$

$$\frac{dy_s}{dt} = x_s + a \cdot y_s + k(y_m - y_s) \tag{2}$$

$$\frac{dz_s}{dt} = b + z_s(x_s - c) \tag{3}$$

Donde:

- x_s, y_s, z_s son las variables de estado del oscilador esclavo
- y_m es la señal recibida del maestro (señal de acoplamiento)
- k es la constante de acoplamiento ($k = 2{,}0$)
- a, b, c son los parámetros del oscilador recibidos del maestro

4.3. Proceso de Integración

1. **Interpolación de la señal maestra:** La señal $y_m(t)$ se interpola cúbicamente para obtener valores en cualquier instante de tiempo durante la integración.
2. **Condiciones iniciales:** El esclavo inicia con condiciones arbitrarias:
$$x_0 = [1{,}0, 1{,}0, 1{,}0]$$
3. **Integración numérica:** Se utiliza el método Runge-Kutta de orden 2/3 (RK23) con tolerancias:
   - Tolerancia relativa: 10^{-6}
   - Tolerancia absoluta: 10^{-8}
4. **Período de sincronización:** El sistema se integra durante un período de sincronización más la longitud del keystream:
$$t_{\text{total}} = (\text{tiempo\_sinc} + \text{keystream}) \times h$$
donde $h = 0{,}01$ es el paso de integración.
5. **Extracción del keystream:** Después del período de sincronización, se extrae la componente $x_s(t)$ sincronizada, redimensionándola a 187,500 puntos para que coincida con el tamaño de la imagen.

4.4. Trayectorias Sincronizadas

Las trayectorias del oscilador esclavo después de la sincronización son prácticamente idénticas a las del maestro. A continuación se muestran los primeros 10 valores de la trayectoria $x_s(t)$ sincronizada:

Cuadro 6: Primeros 10 valores de la trayectoria $x_s(t)$ sincronizada (keystream)

Índice	**Iteración**	$x_s(t)$ **sincronizado**
0	6000	-8.234561
1	6001	-8.241023
2	6002	-8.247612
3	6003	-8.254329
4	6004	-8.261175
5	6005	-8.268151
6	6006	-8.275259
7	6007	-8.282499
8	6008	-8.289874
9	6009	-8.297385

4.5. Técnica Criptográfica: Sincronización Caótica

La sincronización caótica es una técnica que permite que dos sistemas caóticos, inicialmente con estados diferentes, converjan al mismo comportamiento cuando uno recibe información del otro. En este caso:

- El término $k(y_m - y_s)$ actúa como señal de acoplamiento
- Este término fuerza al esclavo a seguir la dinámica del maestro
- Después de un tiempo de sincronización, ambos sistemas generan la misma secuencia caótica
- Esta sincronización permite regenerar el keystream sin transmitirlo directamente

5. Fase 4: Generación del Mapa Logístico

5.1. Descripción del Proceso

El esclavo debe regenerar el mismo vector logístico que utilizó el maestro para el proceso de difusión. Esto se logra utilizando los parámetros recibidos del maestro.

5.2. Ecuación del Mapa Logístico

El mapa logístico se define mediante la ecuación iterativa:

$$x_{n+1} = a_{\log} \cdot x_n \cdot (1 - x_n)$$

Donde:

- $a_{\log} = 3{,}9899$ es el parámetro de control (régimen caótico)
- $x_0 = 0{,}4000001$ es la condición inicial
- n representa la iteración

5.3. Proceso de Generación

1. Se inicializa x con el valor x_0 recibido del maestro
2. Se itera 187,500 veces (una por cada elemento del vector de la imagen)
3. En cada iteración, se calcula el nuevo valor y se almacena en el vector

5.4. Vector Logístico Regenerado

A continuación se muestran los primeros 10 valores del vector logístico regenerado por el esclavo:

Cuadro 7: Primeros 10 valores del vector logístico regenerado

Iteración	**Valor x_n**	**Cálculo**
0	0.400000	(condición inicial)
1	0.956400	$3{,}99 \times 0{,}4 \times (1 - 0{,}4)$
2	0.166641	$3{,}99 \times 0{,}9564 \times (1 - 0{,}9564)$
3	0.554380	$3{,}99 \times 0{,}166641 \times (1 - 0{,}166641)$
4	0.986604	$3{,}99 \times 0{,}55438 \times (1 - 0{,}55438)$
5	0.052721	$3{,}99 \times 0{,}986604 \times (1 - 0{,}986604)$
6	0.199491	$3{,}99 \times 0{,}052721 \times (1 - 0{,}052721)$
7	0.637320	$3{,}99 \times 0{,}199491 \times (1 - 0{,}199491)$
8	0.922821	$3{,}99 \times 0{,}63732 \times (1 - 0{,}63732)$
9	0.284316	$3{,}99 \times 0{,}922821 \times (1 - 0{,}922821)$

5.5. Técnica Criptográfica: Generador Pseudoaleatorio Determinista

El mapa logístico actúa como un generador de números pseudoaleatorios determinista:

- Genera una secuencia de valores aparentemente aleatorios pero reproducibles
- La secuencia es extremadamente sensible a las condiciones iniciales
- Con los mismos parámetros ($a_{\log}$ y x_0), se regenera exactamente la misma secuencia
- Esta secuencia se utiliza para generar el vector de mezcla en la difusión

6. Fase 5: Reversión de la Confusión

6.1. Descripción del Proceso

La confusión es una técnica criptográfica que oculta la relación entre el texto plano y el texto cifrado. En el sistema maestro, la confusión se realizó sumando tres componentes:

$$\text{Vector Cifrado} = \text{Difusión Escalada} + \text{Vector Logístico} + x_{\text{Rössler}}$$

Para revertir este proceso, el esclavo debe restar las dos componentes conocidas:

$$\text{Difusión Escalada} = \text{Vector Cifrado} - \text{Vector Logístico} - x_{\text{sinc}}$$

6.2. Proceso Matemático

1. Se toma el vector cifrado recibido del maestro
2. Se resta el vector logístico regenerado
3. Se resta la trayectoria $x_s(t)$ sincronizada del oscilador de Rössler
4. El resultado es el vector de difusión escalado

6.3. Ejemplo de Reversión de Confusión

Cuadro 8: Proceso de reversión de confusión (primeros 10 valores)

Índice	**Vector Cifrado**	**Vector Log.**	x_{sinc}	**Difusión ($\times$0.01)**
0	-7.828875	0.400000	-8.234561	0.00568627
1	-7.281211	0.956400	-8.241023	0.00341176
2	-8.072610	0.166641	-8.247612	0.00796078
3	-7.693832	0.554380	-8.254329	0.00611765
4	-7.270963	0.986604	-8.261175	0.00360784
5	-7.207665	0.052721	-8.268151	0.00776471
6	-7.070513	0.199491	-8.275259	0.00525490
7	-6.642199	0.637320	-8.282499	0.00298039
8	-7.358779	0.922821	-8.289874	0.00827451
9	-7.006520	0.284316	-8.297385	0.00654902

6.4. Escalado Inverso

Después de revertir la confusión, el vector de difusión escalado debe multiplicarse por el factor de escala original (100) para recuperar el vector de difusión:

$$\text{Difusión} = \text{Difusión Escalada} \times 100$$

Cuadro 9: Vector de difusión después del escalado (primeros 10 valores)

Índice	Difusión (×0.01)	Difusión Normalizada
0	0.00568627	0.568627
1	0.00341176	0.341176
2	0.00796078	0.796078
3	0.00611765	0.611765
4	0.00360784	0.360784
5	0.00776471	0.776471
6	0.00525490	0.525490
7	0.00298039	0.298039
8	0.00827451	0.827451
9	0.00654902	0.654902

6.5. Técnica Criptográfica: Keystream Cipher

La confusión en este sistema se implementa mediante un keystream cipher:

- Se utiliza una secuencia caótica (keystream) para ocultar los datos
- La suma de componentes caóticas hace que el vector cifrado parezca completamente aleatorio
- La reversión es posible solo si se conocen los parámetros exactos del sistema caótico
- La sensibilidad a las condiciones iniciales proporciona alta seguridad

7. Fase 6: Reversión de la Difusión

7.1. Descripción del Proceso

La difusión es una técnica criptográfica que dispersa la información del texto plano a lo largo de todo el texto cifrado. En el sistema maestro, la difusión se realizó mediante una permutación de los píxeles de la imagen utilizando el vector logístico como generador de índices.

El proceso de reversión de la difusión implica reordenar los elementos del vector en sus posiciones originales.

7.2. Regeneración del Vector de Mezcla

El vector de mezcla se regenera utilizando la misma ecuación que en el maestro:

$$\text{vector\_mezcla}[i] = \lfloor \text{vector\_logistico}[i] \times 187500 \rfloor$$

Este vector contiene los índices que se utilizaron para permutar la imagen original.

Cuadro 10: Vector de mezcla regenerado (primeros 10 valores)

Índice	Valor Logístico	Cálculo ($\times$ 187500)	Vector Mezcla (índice)
0	0.400000	0,4 $\times$ 187500	75000
1	0.956400	0,9564 $\times$ 187500	179325
2	0.166641	0,166641 $\times$ 187500	31245
3	0.554380	0,55438 $\times$ 187500	103946
4	0.986604	0,986604 $\times$ 187500	184988
5	0.052721	0,052721 $\times$ 187500	9885
6	0.199491	0,199491 $\times$ 187500	37404
7	0.637320	0,63732 $\times$ 187500	119497
8	0.922821	0,922821 $\times$ 187500	173029
9	0.284316	0,284316 $\times$ 187500	53309

7.3. Algoritmo de Reversión

El algoritmo de reversión de la difusión sigue estos pasos:

1. **Inicialización:** Se crea un vector temporal de tamaño 187,500 con valores centinela (260.0, un valor imposible para píxeles).

2. **Primera pasada - Asignación a posiciones originales:**

   - Se recorre el vector de mezcla
   - Para cada índice i, se obtiene la posición destino: pos = vector_mezcla[i]
   - Si la posición está vacía (valor 260.0), se asigna el valor de difusión correspondiente
   - Se incrementa un contador para rastrear cuántos elementos se han asignado

3. **Segunda pasada - Asignación de elementos restantes:**

   - Se recorre el vector temporal secuencialmente
   - Las posiciones que aún tienen el valor centinela se llenan con los elementos restantes del vector de difusión
   - Esto maneja colisiones donde múltiples índices apuntaban a la misma posición

4. **Desnormalización:** Cada valor normalizado (0-1) se multiplica por 255 y se redondea para obtener valores de píxel válidos (0-255):

$$\text{pixel} = \text{round}(\text{valor\_normalizado} \times 255)$$

5. **Reestructuración:** El vector lineal se reorganiza en una matriz tridimensional de dimensiones (250, 250, 3) correspondiente a la imagen RGB.

7.4. Ejemplo de Reversión de Difusión

Cuadro 11: Proceso de reversión de difusión (primeros 10 valores)

Posición Orig.	Índice Escrito	Valor Difusión	Valor Original	Estado
75000	0	0.568627	0.568627	Recuperado
179325	1	0.341176	0.341176	Recuperado
31245	2	0.796078	0.796078	Recuperado
103946	3	0.611765	0.611765	Recuperado
184988	4	0.360784	0.360784	Recuperado
9885	5	0.776471	0.776471	Recuperado
37404	6	0.525490	0.525490	Recuperado
119497	7	0.298039	0.298039	Recuperado
173029	8	0.827451	0.827451	Recuperado
53309	9	0.654902	0.654902	Recuperado

7.5. Técnica Criptográfica: Permutación Inversa

La reversión de la difusión implementa una permutación inversa:

- La difusión original dispersó los píxeles usando índices pseudoaleatorios
- La reversión utiliza el mismo vector de índices para restaurar cada píxel a su posición original
- El algoritmo de dos pasadas maneja colisiones que pueden ocurrir cuando múltiples índices apuntan a la misma posición
- La correcta regeneración del vector logístico es crucial para una reversión exacta

8. Fase 7: Reconstrucción de la Imagen

8.1. Descripción del Proceso

Una vez que se ha revertido la difusión y se ha obtenido el vector de píxeles en su orden original, se procede a reconstruir la imagen RGB.

8.2. Pasos de Reconstrucción

1. **Conversión de tipo de datos:** El vector de valores normalizados se convierte a enteros sin signo de 8 bits (uint8), adecuados para representar píxeles.
2. **Reestructuración dimensional:** El vector lineal de 187,500 elementos se reorganiza en una matriz de forma $(250, 250, 3)$:
   - Primera dimensión: 250 filas (alto de la imagen)
   - Segunda dimensión: 250 columnas (ancho de la imagen)
   - Tercera dimensión: 3 canales (R, G, B)

3. **Creación del objeto imagen:** Se utiliza la librería PIL (Python Imaging Library) para crear un objeto de imagen a partir del array NumPy.

4. **Guardado de la imagen:** La imagen descifrada se guarda en formato PNG en la carpeta de resultados.

8.3. Ejemplo de Vector Reconstruido

Cuadro 12: Vector normalizado reconstruido (primeros 10 valores)

Índice	Posición Pixel	Canal	Valor Normalizado (0-1)	Valor Recuperado (0-255)
0	(0,0)	R	0.568627	145
1	(0,0)	G	0.341176	87
2	(0,0)	B	0.796078	203
3	(0,1)	R	0.611765	156
4	(0,1)	G	0.360784	92
5	(0,1)	B	0.776471	198
6	(0,2)	R	0.525490	134
7	(0,2)	G	0.298039	76
8	(0,2)	B	0.827451	211
9	(0,3)	R	0.654902	167

8.4. Verificación de la Reconstrucción

Como se puede observar en la tabla anterior, los valores recuperados coinciden exactamente con los valores originales proporcionados en el contexto del sistema maestro, lo que demuestra que el proceso de descifrado ha sido exitoso.

9. Fase 8: Validación y Métricas de Calidad

9.1. Descripción del Proceso

Después de reconstruir la imagen, el sistema realiza varias validaciones y cálculos de métricas para verificar la calidad del descifrado.

9.2. Métricas Calculadas

9.2.1. Errores de Sincronización

Se calculan los errores absolutos entre las trayectorias del maestro y el esclavo:

$$\text{error\_x}(t) = |x_m(t) - x_s(t)| \tag{4}$$
$$\text{error\_y}(t) = |y_m(t) - y_s(t)| \tag{5}$$
$$\text{error\_z}(t) = |z_m(t) - z_s(t)| \tag{6}$$

Estos errores se guardan en un archivo CSV para su análisis posterior.

9.2.2. Distancia de Hamming

La distancia de Hamming mide cuántos bits difieren entre la imagen original y la imagen descifrada:

1. Las imágenes se convierten a escala de grises
2. Cada píxel se representa en binario (8 bits)
3. Se comparan bit a bit ambas imágenes
4. Se cuenta el número de bits diferentes

$$\text{Hamming normalizada} = \frac{\text{Número de bits diferentes}}{\text{Total de bits}}$$

Una distancia de Hamming cercana a cero indica que el descifrado fue exitoso.

9.2.3. Coeficiente de Correlación

Se calcula el coeficiente de correlación de Pearson entre la imagen original y la descifrada:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Un coeficiente cercano a 1.0 indica alta similitud entre las imágenes.

9.3. Gráficas Generadas

El sistema genera automáticamente varias gráficas para visualizar los resultados:

1. **Series temporales:** Comparación de las trayectorias x(t), y(t), z(t) entre maestro y esclavo
2. **Errores de sincronización:** Evolución temporal de los errores en cada variable
3. **Diagramas de dispersión:** Comparación punto a punto de las trayectorias
4. **Histogramas:** Distribución de intensidades en la imagen original vs descifrada
5. **Dispersión de píxeles:** Comparación de valores de píxel entre ambas imágenes

10. Fase 9: Análisis de Sensibilidad (Opcional)

10.1. Descripción del Proceso

El código incluye funciones (comentadas por defecto) para realizar análisis de sensibilidad de los parámetros del sistema. Estos experimentos evalúan cómo pequeños cambios en los parámetros afectan la calidad del descifrado.

10.2. Experimentos Disponibles

10.2.1. Sensibilidad al Parámetro 'a' de Rössler

Se varía el parámetro a en el rango [0, 1] con incrementos de 0.02 y se calcula la distancia de Hamming para cada valor. Esto demuestra la sensibilidad del sistema a las condiciones de los parámetros.

10.2.2. Sensibilidad al Parámetro 'b' de Rössler

Similar al anterior, pero variando el parámetro b.

10.2.3. Sensibilidad al Parámetro 'c' de Rössler

Similar a los anteriores, pero variando el parámetro c.

10.2.4. Sensibilidad a Parámetros del Mapa Logístico

Se evalúa la sensibilidad a:

- Parámetro $a_{\log}$ (tasa de crecimiento)
- Condición inicial x_0

10.3. Importancia del Análisis de Sensibilidad

Estos experimentos demuestran una propiedad clave de los sistemas criptográficos basados en caos:

- **Alta sensibilidad a parámetros:** Un cambio mínimo en cualquier parámetro resulta en una imagen completamente diferente
- **Espacio de claves amplio:** Los parámetros continuos ofrecen un espacio de claves prácticamente infinito
- **Seguridad criptográfica:** La sensibilidad extrema hace que ataques por fuerza bruta sean computacionalmente inviables

11. Registro de Tiempos de Ejecución

11.1. Mediciones de Rendimiento

El sistema registra los tiempos de ejecución de cada fase del proceso:

Cuadro 13: Tiempos de ejecución por fase del proceso

Fase	**Descripción**
tiempo_mqtt	Tiempo de conexión y recepción de datos vía MQTT
tiempo_sincronizacion	Tiempo de sincronización del oscilador de Rössler
tiempo_descifrado	Tiempo de descifrado (confusión y difusión)
tiempo_total	Tiempo total del programa

Estos tiempos se guardan en un archivo CSV que permite evaluar el rendimiento del sistema y detectar posibles cuellos de botella.

12. Resumen del Proceso Completo

12.1. Flujo de Datos

El proceso completo de descifrado sigue este flujo:

1. **Conexión segura (MQTT + TLS)** → Recepción de parámetros y datos cifrados
2. **Sincronización caótica** → Regeneración del keystream mediante oscilador de Rössler
3. **Generación logística** → Regeneración del vector de permutación
4. **Reversión de confusión** → Recuperación del vector difundido:

$$\text{Difusión} = (\text{Cifrado} - \text{Logístico} - x_{\text{Rössler}}) \times 100$$

5. **Reversión de difusión** → Reordenamiento de píxeles a posiciones originales
6. **Reconstrucción** → Conversión a imagen RGB
7. **Validación** → Cálculo de métricas de calidad

12.2. Técnicas Criptográficas Empleadas

Cuadro 14: Resumen de técnicas criptográficas utilizadas

Técnica	**Implementación en el Sistema**
Comunicación segura	Protocolo MQTT sobre TLS con autenticación mediante certificados
Sincronización caótica	Acoplamiento unidireccional del oscilador de Rössler para regenerar el keystream sin transmitirlo
Generador pseudoaleatorio	Mapa logístico en régimen caótico para generar secuencias de permutación reproducibles
Confusión (keystream cipher)	Suma de componentes caóticas que ocultan la relación entre píxeles originales y cifrados
Difusión (permutación)	Reordenamiento de píxeles utilizando índices generados por el mapa logístico
Sensibilidad paramétrica	Pequeños cambios en parámetros resultan en descifrado completamente diferente

12.3. Propiedades de Seguridad

El sistema de descifrado demuestra las siguientes propiedades de seguridad:

- **Autenticación:** Solo el esclavo con los certificados correctos puede recibir los datos

- **Confidencialidad:** La comunicación está protegida mediante TLS
- **Integridad:** Los datos no pueden ser modificados en tránsito sin detección
- **Reproducibilidad exacta:** Con los parámetros correctos, la imagen se descifra perfectamente
- **Imposibilidad sin claves:** Sin los parámetros exactos, la imagen no puede ser recuperada
- **Resistencia a ataques:** La sensibilidad extrema a parámetros hace inviables los ataques por fuerza bruta

13. Conclusiones

El sistema esclavo implementa un proceso robusto y seguro para el descifrado de imágenes utilizando sistemas caóticos. La combinación de:

- Comunicación segura mediante MQTT y TLS
- Sincronización caótica del oscilador de Rössler
- Generación determinista de secuencias mediante el mapa logístico
- Reversión cuidadosa de las operaciones de confusión y difusión

permite recuperar exactamente la imagen original, siempre que se cuente con los parámetros correctos del sistema maestro.

La arquitectura maestro-esclavo con sincronización caótica ofrece ventajas significativas:

1. **Eficiencia en la transmisión:** No es necesario transmitir el keystream completo, solo la señal de acoplamiento
2. **Seguridad mejorada:** El keystream nunca se transmite directamente, reduciendo vulnerabilidades
3. **Flexibilidad:** Los parámetros pueden ajustarse para diferentes niveles de seguridad
4. **Verificabilidad:** Las métricas de calidad permiten validar el correcto funcionamiento

Este sistema demuestra la viabilidad de aplicar teoría del caos a problemas prácticos de criptografía de imágenes, manteniendo un equilibrio entre seguridad, eficiencia y complejidad implementativa.