Feb 16, 2026

Chen Hyperchaotic-AES Medical Image Encryption Protocol

DOI
https://dx.doi.org/10.17504/protocols.io.5qpvo12pbg4o/v1

Mutallip Sattar[1]

[1]Xinjiang University of Finance and Economics

Mutallip Sattar

OPEN ACCESS

DOI: https://dx.doi.org/10.17504/protocols.io.5qpvo12pbg4o/v1

Protocol Citation: Mutallip Sattar 2026. Chen Hyperchaotic-AES Medical Image Encryption Protocol. **protocols.io** https://dx.doi.org/10.17504/protocols.io.5qpvo12pbg4o/v1

Protocol status: Working
We use this protocol and it's working

Created: February 14, 2026

Last Modified: February 16, 2026

Protocol Integer ID: 243316

Keywords: aes medical image encryption protocol, aes medical image encryption protocol this protocol, layer encryption scheme for medical image, layer encryption scheme, medical image, complete encryption, chen hyperchaotic, decryption workflow, protocol this protocol, protocol, arnold cat map permutation, combining arnold cat map permutation, chen, chen hyperchaotic system diffusion

Abstract

This protocol describes a three-layer encryption scheme for medical images combining Arnold cat map permutation, Chen hyperchaotic system diffusion, and AES-256 block cipher. The implementation includes synthetic chest X-ray phantom generation, complete encryption/decryption workflows, and validation procedures. All source code is provided as an attachment.

Attachments

ZIP

chen_aes_medical_enc..

12KB

Step 1: Environment Setup and Dependencies

1 Install required Python packages:

pip install numpy==1.24.3 opencv-python==4.8.1 pycryptodome==3.19.0

System Requirements:
- Python 3.9, 3.10, or 3.11 (3.10 recommended)
- 4GB RAM minimum (8GB recommended)
- No GPU required (CPU-only implementation)

Note: This protocol uses pure NumPy implementations to avoid SciPy version conflicts.

Step 2: Generate Synthetic Medical Image

2 Execute the test image generator:

python generate_medical_sample.py

This creates a 512×512 pixel chest X-ray phantom containing:
- Thoracic cage (elliptical boundary)
- Spinal column and rib structures
- Heart shadow (left-central high-density region)
- Pulmonary textures (simulated vasculature)
- Simulated pulmonary nodule (right upper zone)

Output file: realistic_chest_xray.png
Expected runtime: <5 seconds

Step 3: Three-Layer Encryption Process

3 Run the encryption protocol:

python encrypt_medical.py

This implements the three-layer security scheme:

Layer 1 - Arnold Cat Map Permutation:
Scrambles pixel positions using modular matrix transformation
Formula: [x'; y'] = [1 a; b ab+1] × [x; y] mod N

Layer 2 - Chen Hyperchaotic Diffusion:

Generates chaotic sequence via 4D Chen system (RK4 solver)
XOR operation: Ciphertext = Plaintext ⊕ Chaotic_sequence

Layer 3 - AES-256 Block Encryption:
Standard FIPS-197 implementation (ECB mode, PKCS7 padding)

Outputs:
- encrypted_output.png (ciphertext, appears as random noise)
- encryption_params.npz (key material, KEEP SECURE)

Expected runtime: 2-4 minutes (Intel i5 processor)

Step 4: Decryption and Validation

4 Restore the original image:

python decrypt_medical.py

Requirements:
- encrypted_output.png (from Step 3)
- encryption_params.npz (key file from Step 3)

Inverse operations:
1. AES-256 decryption (PyCryptodome)
2. Chen hyperchaotic inverse (XOR is self-inverse)
3. Arnold inverse permutation (inverse matrix)

Validation:
The script automatically calculates pixel difference between original and decrypted images.
Expected result: Total difference = 0 (perfect reconstruction)

Output: decrypted_final.png
Expected runtime: 2-4 minutes