ANNA UNIVERSITY
MADRAS INSTITUTE OF TECHNOLOGY CAMPUS
CHENNAI – 600 044

Creative Inovative Project - Review 3

Sub Code: CS6611

Secure Image Key Generation for Medical image security using WGAN-GP

Presented by

Malarvannan M (2022503011)
Maanasa Prathap Chander (2022503065)
Kalaidharun R (2022503009)

Guided by

Dr.R.Kathiroli
Assistant Professor
Dept of Computer Technology
Anna University, MIT campus

OutLine

OutLine

Problem Statment

- Traditional cryptographic key generation depends on deterministic algorithms which can be suspectible to attacks if the entropy is very less.
- Machine Learning based key generation approaches are currently booming to generate high entropy keys but they suffer from **training instability** and **mode collapse.**
- This highlights the need for the use of a **more stable model** for the training process to overcome **training instability**, **mode collapse** and to **generate a strong quality key**.

Objective

- To develop a secure one-time pad key generation cryptographic system using WGAN-GP and evaluate the quality of it.
- To enhance training stability.
- To provide a more secured cryptosystem.

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
1	DeepKeyGen: A Deep Learning-Based Stream Cipher Generator for Medical Image Encryption and Decryption	IEEE Transactions on Neural Networks and Learning Systems / 2022	DeepKeyGen presented a deep learning-based stream cipher generator that utilized Generative Adversarial Networks (GANs) for the encryption and decryption of medical images.	The process relies heavily on the randomness of deep learning models, which can cause inconsistencies in key generation even under identical conditions and the training process is unstable.
2	DeepEDN: A Deep-Learning-Based Image Encryption and Decryption Network for Internet of Medical Things	IEEE Internet of Things Journal / 2021	DeepEDN is an end-to-end encryption-decryption framework for IoMT. It uses Cycle-GANs to encrypt medical images and an inverse mapping network for loss-minimized decryption. An ROI mining network enables selective retrieval of critical data without full decryption.	The computational complexity is very high as Cycle-GANs and ROI mining require significant processing power

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
3	DeepENC: Deep Learning-Based ROI Selection for Encryption of Medical Images Through Key Generation with Multimodal Information Fusion	IEEE Transactions /2024	DeepENC securely encrypts the Region of Interest (ROI) in medical images using a 2D hybrid chaotic key. It leverages UNet3+ for ROI segmentation and biometric features for key generation, ensuring high security with low computational cost.	There are some potential vulnerabilities in this such as, while selective ROI encryption improves efficiency, the unencrypted background could still leak contextual information about the medical image.

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
4	**Chaotic Medical Image Encryption Method Using Attention Mechanism Fused with ResNet Model**	**Frontiers in Neuroscience / 2023**	**AT-ResNet-CM enhances medical image encryption by integrating an attention mechanism with ResNet for high-dimensional feature extraction. The attention module focuses on critical regions, improving encryption quality, while a chaotic system ensures randomness and security.**	**It requires careful tuning of its attention mechanism and chaotic system. If not configured properly, the encryption quality may drop, potentially affecting security and performance.**

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
5	Deep Learning-Based Encryption Scheme for Medical Images Using Deep Convolutional Generative Adversarial Networks	Nature Scientific Reports / 2024	• Utilizes Deep Convolutional Generative Adversarial Networks (DCGANs) for encrypting medical images. • Implements Virtual Adversarial Training (VAT) to enhance robustness against adversarial attacks.	Mode collapse is a well-known issue in DCGANs where instead of exploring the full distribution of possible transformations, the model starts producing similar outputs for different inputs. This happens because the generator finds a "shortcut" to fool the discriminator without learning a complete mapping from the original image space to the encrypted space.

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
6	**EiMOL: A Secure Medical Image Encryption Algorithm based on Optimization and the Lorenz System**	**ACM Transaction / 2023**	**EiMOL uses the Lorenz chaotic system combined with optimization techniques to generate a random sequence, transforming medical images into encrypted data.**	**EiMOL uses a chaotic system (Lorenz), which could be reverse-engineered by attackers to break the encryption.**
7	**A Faster Privacy-Preserving Medical Image Diagnosis Scheme with Machine Learning**	**Springer / 2024**	**Uses CKKS Homomorphic Encryption for encrypting both medical images and diagnostic parameters.**	**Homomorphic encryption is computationally expensive and requires high processing power.**

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
8	**MedBlindTuner: Towards Privacy-preserving Fine-tuning on Biomedical Images with Transformers and Fully Homomorphic Encryption**	**Springer/ 2024**	**Encrypts medical images before training, allowing ML models to learn without decrypting sensitive data and utilizes DEiT (Data-Efficient Image Transformer) for efficient image classification.**	**FHE operates on encrypted data using techniques like ciphertext addition and multiplication, which are computationally expensive and not natively supported by GPUs/TPUs.**

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
9	CaRENets: Compact and Resource-Efficient CNN for Homomorphic Inference on Encrypted Medical Images	Springer / 2024	CaRENets presents a significant advancement in performing CNN-based inference on encrypted medical images by introducing a compact and resource-efficient approach.	Despite the optimizations, homomorphic encryption inherently introduces substantial computational overhead and processing time is very high for high resolution image
10	RISE: Rubik's Cube and Image Segmentation-Based Secure Medical Image Encryption	Springer / 2024	• The medical image is divided into multiple segments for enhanced security. • Each segment undergoes a scrambling process based on the Rubik's Cube principle to create randomness.	The security of the encryption heavily depends on the initial key and transformation steps, making it vulnerable to brute-force or differential attack

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
11	Quantum image encryption protocol for secure communication in healthcare networks	Springer / 2025	The one-dimensional discrete chaotic map, termed the logistic-sin map is used to generate key and the plain image is split into 2 images and they are converted into gray code and then encrypted	• The final Cipher image is less unpredictable and non-deterministic when compared to WGAN-GP • It has low Scalablity of applications
12	Medical Image Encryption Based on Josephus Scrambling and Dynamic Cross-Diffusion for Patient Privacy Security	IEEE Transactions / 2024	Hyperchaotic system for key generation, Josephus scrambling for pixel position permutation, and dynamic cross-diffusion for pixel value confusion. The encryption key is dynamically updated using the SHA-256 hash of the image, ensuring unique encryption for each image.	• Josephus-based encryption relies on a chaotic system, which is deterministic for the same initial conditions. • WGAN-GP learns to generate diverse keys dynamically, making it harder to predict key patterns.

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
13	Reversible Data Hiding in Encrypted Images With Secret Sharing and Hybrid Coding	IEEE Transactiosn on Circuits and Systems for Video Technology / 2023	• This proposes a method to hide secret data such as text messages, image or binary data into an image and encrypt it securely while ensuring the image can be fully restored back • Uses a combined technique of secret sharing(splitting data into multiple parts) and hybrid coding(to increase the amount of data that can be hidden)	• Usage of secret sharing (CRTSS) and hybrid coding requires significant computational power making it less-efficient for real time applications • Extra storage and bandwidth requirement for storing and sharing multiple shares of the image
14	An algorithm for secure medical image transmission that utilizes Arnold's Cat MaP and encode-decode	Proceedings of the 2024 IEEE Wireless Power Technology Conference and Expo (WPTCE2024) / 2024	Uses a Physics-Informed Neural Network (PINN) to estimate mutual inductance in SS-IPT systems by integrating physics-based equations with neural network learning, leveraging transmitter-side voltage and current data.	The method is validated only through simulations, lacking real-world experimental verification, which may affect accuracy due to noise and hardware tolerances.

Literature Survey

S.NO	Title	Name of the Journal and Published Year	Proposed Methodology	Limitations
15	Efficient Security and Authentication for Edge-Based Internet of Medical Things	IEEE Internt of Things Journal/ 2021	This approach utilizes Left Data Mapping (LDM), Pixel Repetition Method (PRM), RC4 encryption, and checksum computation to ensure data integrity and confidentiality.	• RC4 encryption is weak compared to modern cryptographic standards. • Tamper detection may be less effective under extreme noise or compression. • Scalability issues for real-time, large-scale IoMT applications.

Architecture Diagram

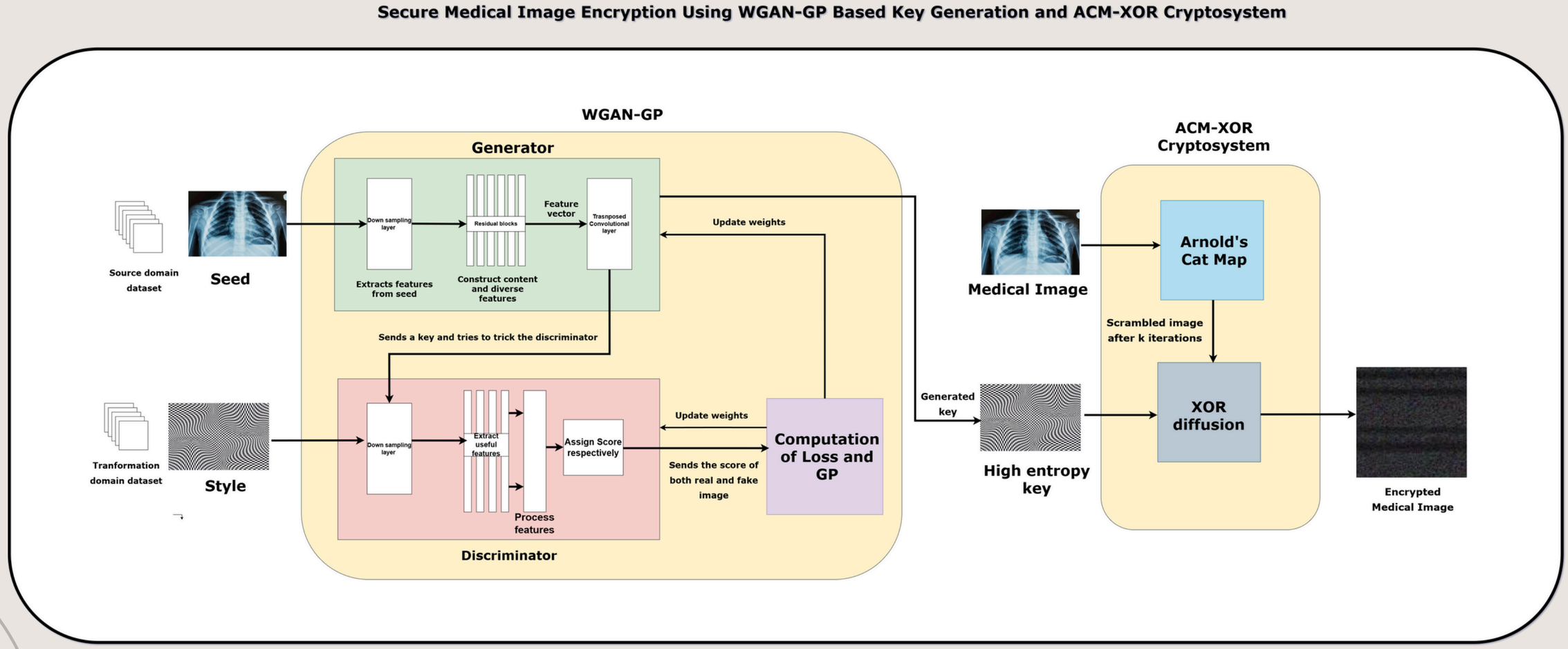

Fig 1. Working of propsed working model

[Created using Draw.io]

Architecture Diagram

Generator

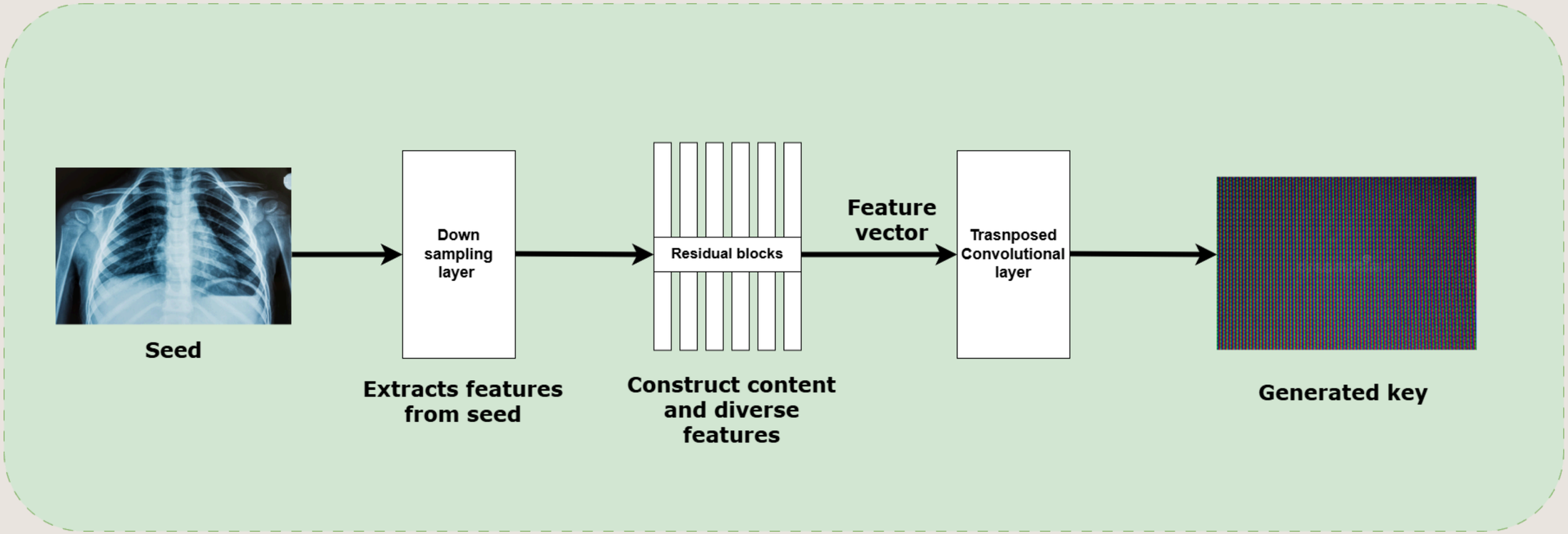

Fig 1.2. How the generator generates a key

[Created using Draw.io]

Architecture Diagram

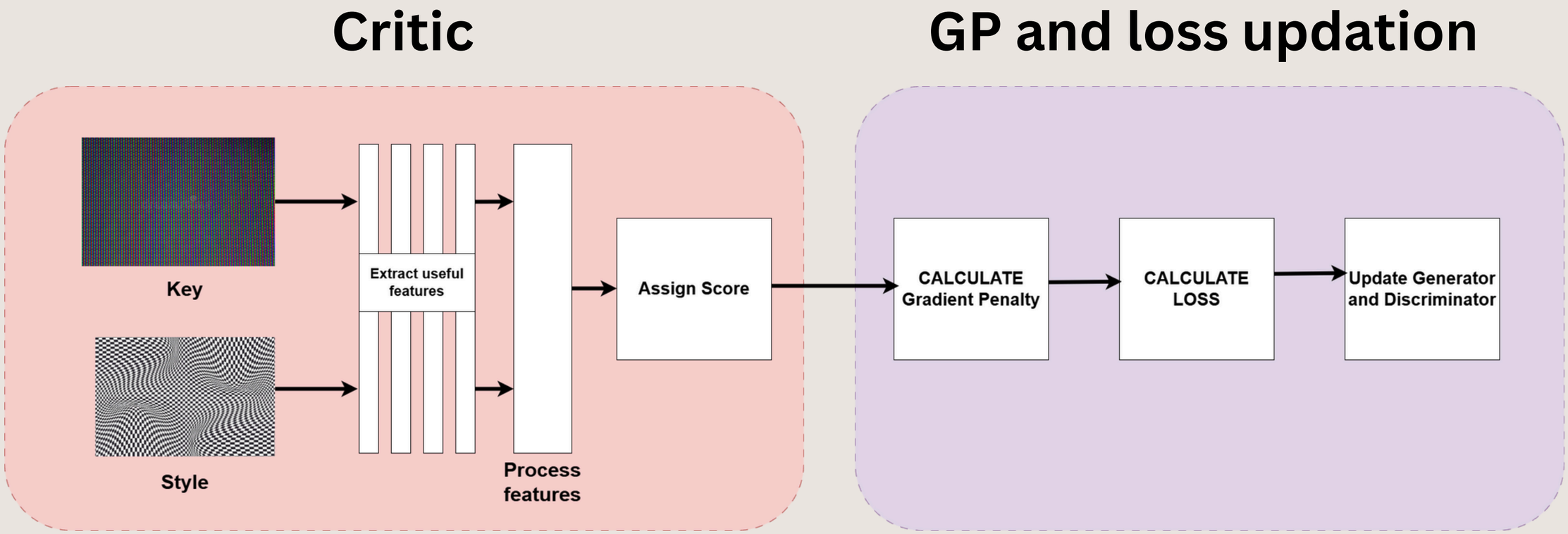

Fig 1.3. How the discriminator assigns score for the key and how the gradient penalty is calculated

[Created using Draw.io]

Architecture Diagram

Cryptosystem

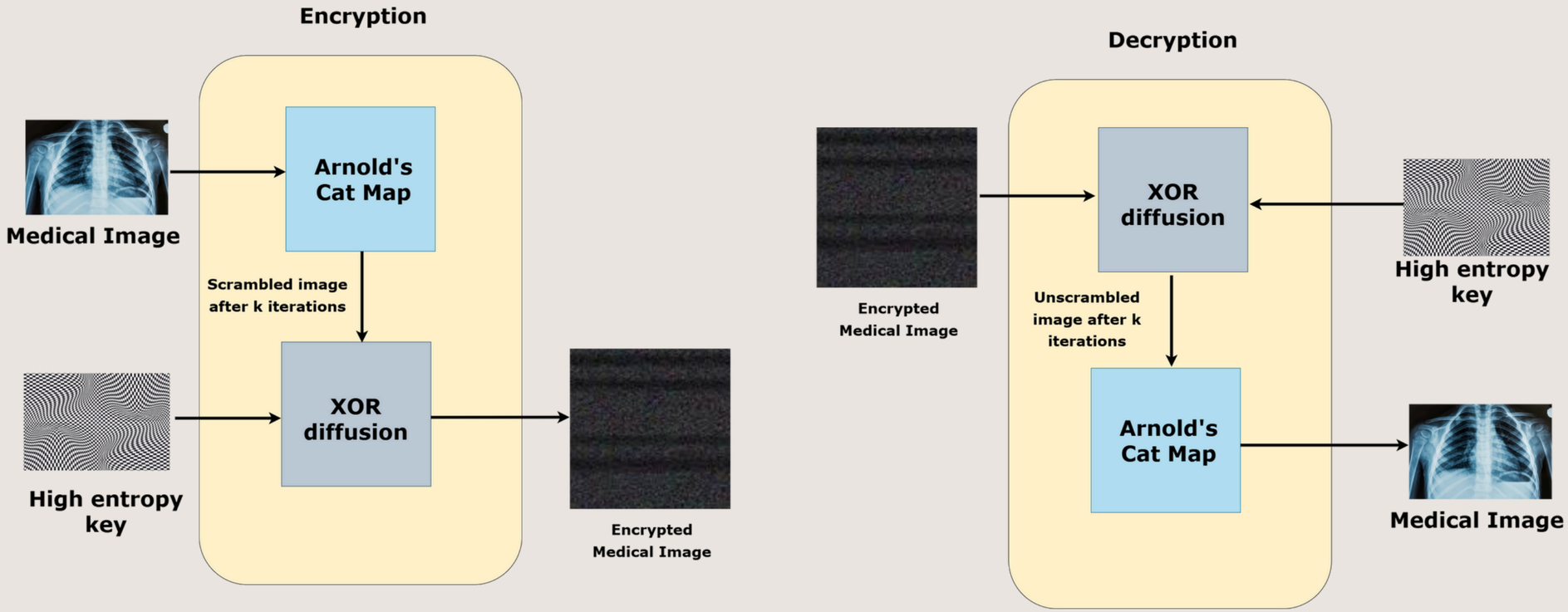

Fig 1.4. How ACM and XOR diffusion is used to encrypt and decrypt the medical image using the high entropy key obtained by WGAN-GP

[Created using Draw.io]

Modules

- **Key Generation :**
  - Here the strong high entropy key is generated using WGAN-GP
  - The WGAN-GP consists of Generator, Discriminator and uses Gradient penalty and the loss is calculated and propagated.
  - This fight will continue till the model stabilizes and finally generates a strong key.
  - **Input:** Seed image and Style image
  - **Expected output:** High Entropy Key
- **Encryption -: 2-layer security (confusion + diffusion)**
  - Arnold’s Cat Map(ACM) - To scramble pixel positions within an image to create a visually distorted version using predefined set of parameters
  - Perform XOR operation between the scrambled image and the WGAN-GP generated key.

Algorithm for Key generation Module : WGAN-GP

Initialization: Randomly initialize the parameters W of the DeepKeyGen, Wn = random [Wn,1,Wn,2, … ,Wn,i]. Define the dimension of KEY as 256 x 256 x 3.

Output: High entropy key

1. **while** Epoch < Epoch_*target* do
2. x = Convert(IMAGE_sourcedomain);
3. y = Convert(IMAGE_transformationdomain) // Convert training images into 256 x 256 x 3 matrices.
4. KEY = G(x) // Forward propagation of generator network G. At the last layer of G, the KEY is generated.
5. Result = D(y) // Forward propagation of discriminator network D. Output the judgment result whether the input is from transformation domain.
6. L = LG + LD / / Calculate the total loss L.
7. Backward(L)// Backward propagation.
8. Wj= Wj-1 - A J(W) // Calculate the gradient that pass back to each layer, and then update the network parameters.
9. **end while**

Algorithm for ACM+XOR Cryptosystem

Encryption :

a. **Input :** Medical Image, key
   i. scrambled_image = ACM(medical_image,k_iterations) // This shuffles the pixel positions of the image up to k iterations
   ii. encrypted_image = XOR(scrambled_image, key)

b. **Output** : ciphered medical image

Decryption :

- **Input :** Encrypted medical Image, key
  - unscrambled_image = Reverse_ACM(encrypted_medical_image,k_iterations) // This unshuffles the pixel positions of the image which was shuffled by the scrambler previously
  - plain_image = XOR(unscrambled_image, key)
- **Output** : deciphered medical image

Metrics

a) NPCR (Sensitivity)

NPCR denotes the pixel change rate, which is used to indicate the ratio of different pixel values at the same location of two images.

$$\mathrm{NPCR} = \frac{\sum_{i=0}^{W} \sum_{j=0}^{H} D(i, j)}{W \times H} \times 100\%.$$

where,

W , H -> Width and Height of the image

D(i, j) -> Pixel Change Indicator Function , where i and j are pixel positions.

D(i,j) is 1 if both pixels are same else it is 0 .

Metrics

b) UACI (Sensitivity)

Measures how much the pixel values change in intensity.Ensures that the differences between the two private keys are not just binary changes but are also spread over a wide range of values.

$$\mathrm{UACI} = \frac{\sum_{i=1}^{W} \sum_{j=1}^{H} |T_1(i, j) - T_2(i, j)|}{255 \times W \times H} \times 100\%.$$

where,

T1(i,j) and T2(i,j) are the pixel values at position (i, j) in the two generated private keys

Metrics

C) Mean Square Error (MSE)

Used to measure the average squared difference between corresponding pixel values of two images. It evaluates how much the encrypted image (ciphertext) deviates from the original image (plaintext)

where,

$$\mathrm{MSE} = \frac{1}{H \times W \times C} \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k=1}^{C} (a(i, j, k) - b(i, j, k))^2$$

$H, W \rightarrow$ Height and width of the image (total pixels).
$C \rightarrow$ Number of color channels (e.g., 3 for RGB).
$a(i, j, k), b(i, j, k) \rightarrow$ Pixel values in plaintext and ciphertext images.

Metrics

D) Structural Similarity Index (SSIM)

Metric that measures the similarity between two images by considering structural information, luminance, and contrast. Used for evaluating structural similarity.

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where,

- $\mu_x, \mu_y \rightarrow$ Mean values (brightness) of images.
- $\sigma_x^2, \sigma_y^2 \rightarrow$ Variance (contrast) of images.
- $\sigma_{xy} \rightarrow$ Covariance (correlation between images).
- $c_1, c_2 \rightarrow$ Small constants for stability.

Metrics

E) Correlation Analysis

Correlation analysis measures the relationship between adjacent pixels in an image. A strong correlation in plaintext images makes them vulnerable to attacks, so encryption aims to reduce this correlation in ciphertext images for higher security.

$$r_{xy} = \frac{\mathrm{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}}$$

where,

- $r_{xy} \rightarrow$ Correlation coefficient (measures similarity between adjacent pixels).
- $\mathrm{cov}(x, y) \rightarrow$ Covariance (measures how two pixel values change together).
- $D(x), D(y) \rightarrow$ Variance of pixel values in the image (measures spread).
- Higher $r_{xy} \rightarrow$ Strong correlation (plaintext images).
- Lower $r_{xy} \rightarrow$ Weak correlation (well-encrypted images).

Test cases

TC1: Ascertain that a high entropy key is conjured.

TC2: Can the encrypted image be restored with the same clarity?

TC3: Is the key forged uniquely with each invocation?

Implementation plan

Phase 1 : Implement the WGAN-GP module.

Phase 2 : Test the quality of the key generated from WGAN-GP module

Phase 3 : Implement ACM + XOR diffusion for image encryption.

Phase 4 : Test the quality of the encrypted image

Tools and libraries

PyTorch : A machine learning library that includes algorithms like neural networks, support vector machines,

TorchVision : It is a package within Torch that provides image and video processing capabilities

SciPy and Numpy : For mathematical computations and optimizations

Medical Image Dataset :
https://www.kaggle.com/datasets/raddar/tuberculosis-chest-xrays-montgomery
https://www.kaggle.com/masoudnickparvar/brain-tumor-mri-dataset

Transformation domain Dataset:
https://www.kaggle.com/datasets/vishalbakshi/hms-hbac-training-spectrogram-images

Training Environment

- Our WGAN-GP model is trained/being trained in google colab using TPU (Tensor Processing Unit)

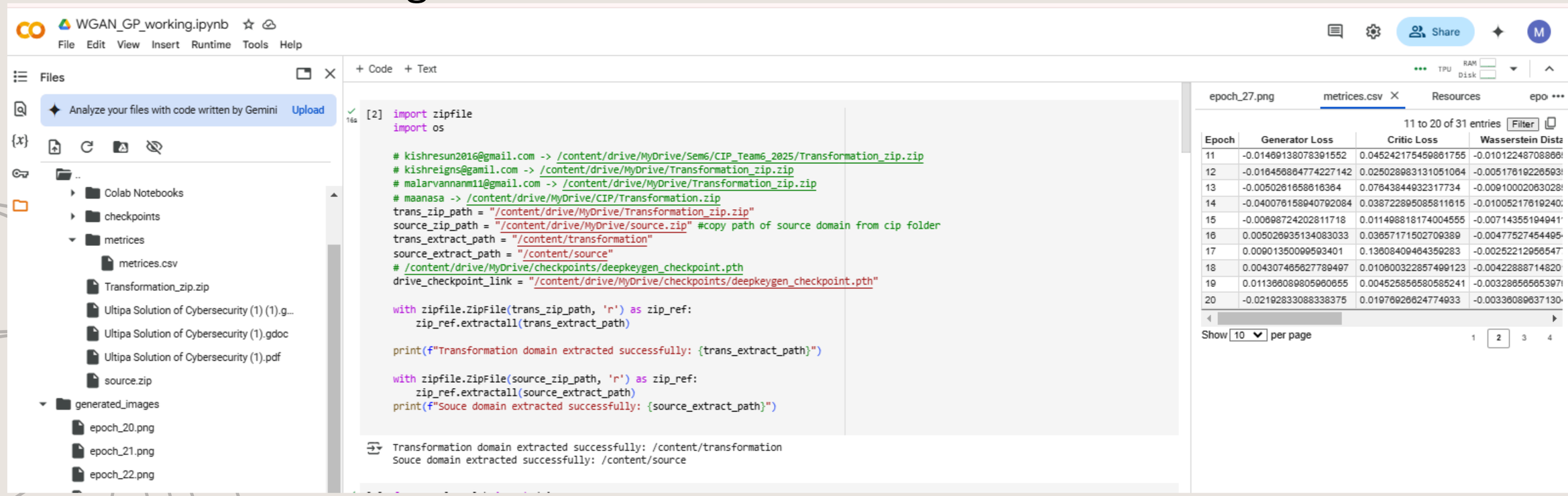

Fig 2. Google colab Environment

Implementation

- train_DeepKeyGen() : Training the model

```
for epoch in range(start_epoch, num_epochs):
    print(f"Epoch: {epoch + 1}/{num_epochs}", flush=True)
    total_gen_correct = 0
    total_critic_correct = 0
    total_samples = 0
    for source_batch, transform_batch in zip(source_loader, transform_loader):
        source_imgs, _ = source_batch  # Extract images, ignore labels
        transform_imgs, _ = transform_batch  # Extract images, ignore labels

        source_imgs = source_imgs.to(device)
        transform_imgs = transform_imgs.to(device)

        for _ in range(critic_iterations):
            fake_imgs = generator(source_imgs).detach()
            real_scores = critic(transform_imgs)
            fake_scores = critic(fake_imgs)

            real_loss = real_scores.mean()
            fake_loss = fake_scores.mean()
            gp = compute_gradient_penalty(critic, transform_imgs, fake_imgs, device)
            critic_loss = fake_loss - real_loss + lambda_gp * gp

            optimizer_d.zero_grad()
            critic_loss.backward()
            xm.optimizer_step(optimizer_d)
            xm.mark_step()  # Free TPU memory

            total_samples += real_scores.size(0) + fake_scores.size(0)
            total_critic_correct += (real_scores > 0).sum().item() + (fake_scores < 0).sum().item()

        fake_imgs = generator(source_imgs)
        fake_scores = critic(fake_imgs)
        generator_loss = -fake_scores.mean()
```

Fig 3. Process in each epoch of the training

Implementation

- Generator() : Creates fake image from real image
- Critic() : Evaluates the authenticity of real and fake images

```python
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(64),
            nn.ReLU(inplace=True),

            nn.Conv2d(64, 128, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(128),
            nn.ReLU(inplace=True),

            nn.Conv2d(128, 256, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(inplace=True),

            nn.Conv2d(256, 512, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(512),
            nn.ReLU(inplace=True),
        )

        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(inplace=True),

            nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(128),
            nn.ReLU(inplace=True),

            nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1, bias=False),
            nn.BatchNorm2d(64),
            nn.ReLU(inplace=True),

            nn.ConvTranspose2d(64, 3, kernel_size=4, stride=2, padding=1, bias=False),
            nn.Tanh()  # Output image [-1,1] range
        )
```

Fig 4. Generator code

```python
# Critic network
class Critic(nn.Module):
    def __init__(self):
        super(Critic, self).__init__()
        self.model = nn.Sequential(
            nn.Conv2d(3, 64, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(64, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2, inplace=True),

            nn.Conv2d(128, 1, 4, 1, 0, bias=False)
        )

    def forward(self, img):
        return self.model(img).view(-1)
```

Fig 5. Critic code

Implementation

- compute_gradient_penalty() : ensures the critic's gradient norms stay close to 1 for stable training

```python
def compute_gradient_penalty(critic, real_samples, fake_samples, device):
    min_batch_size = min(real_samples.size(0), fake_samples.size(0))
    real_samples = real_samples[:min_batch_size]
    fake_samples = fake_samples[:min_batch_size]

    if real_samples.shape != fake_samples.shape:
      print(f"Shape mismatch: real_samples: {real_samples.shape}, fake_samples: {fake_samples.shape}")

    # Create random alpha values for interpolation
    alpha = torch.rand(real_samples.shape[0], 1, 1, 1, device=device)  # Random alpha for each image in the batch

    # Expand alpha to match the spatial dimensions of the image (8, 3, 256, 256)
    alpha = alpha.view(real_samples.shape[0], 1, 1, 1).expand_as(real_samples)


    # Interpolate between real and fake samples
    interpolates = alpha * real_samples + (1 - alpha) * fake_samples
    interpolates.requires_grad_(True)

    # Pass the interpolates through the critic
    critic_interpolates = critic(interpolates)

    # Compute gradients of the critic's output w.r.t. interpolates
    grad_outputs = torch.ones_like(critic_interpolates, device=device)

    gradients = torch.autograd.grad(
        outputs=critic_interpolates,
        inputs=interpolates,
        grad_outputs=grad_outputs,
        create_graph=True,
        retain_graph=True,
        only_inputs=True
    )[0]

    # Flatten the gradients and compute the gradient penalty
    gradients = gradients.view(gradients.shape[0], -1)
    gradient_penalty = ((gradients.norm(2, dim=1) - 1) ** 2).mean()

    return gradient_penalty
```

Fig 6. Gradient Penalty code

Metrics Obtained

- The Metrics such as generator loss, critic loss, wasserstein distance, gp obtained at each each epoch was stored in a csv file

	A	B	C	D	E	F	G	H	I
1	Epoch	Generator Loss	Critic Loss	Wasserstein Distance	Gradient Penalty	D_real	D_fake	Gen_Acc	Critic_Acc
2	1	-0.001852030633	0.06027740985	-0.01053295215	0.009948891568	-0.007995828055	0.002537124092	8.90334902	48.04665811
3	2	0.00583263021	0.2199445218	-0.01126362011	0.0417361781	-0.03350502253	-0.02224140242	8.823508454	48.2052235
4	3	-0.01090332866	0.043957185	-0.009176838212	0.006956068799	0.006615930237	0.01579276845	8.603385365	48.99491144
5	4	-0.01578206941	0.08832772821	-0.01216062787	0.01523341984	0.001653515035	0.01381414291	8.939926484	48.57313873
6	5	-0.02731532231	0.05998094007	-0.006582092494	0.01067976933	0.01902150363	0.02560359612	8.942292848	48.42678782
7	6	-0.0167427361	0.02885365859	-0.01026969682	0.003716792446	0.01181183476	0.02208153158	8.730237972	49.05858625
8	7	-0.03618898615	0.01623897627	-0.00551822409	0.002144150203	0.03978731856	0.04530554265	8.893198876	48.82662842
9	8	-0.01686234213	0.01404315233	-0.009840694256	0.0008404917317	0.003921490163	0.01376218442	8.852746381	48.67743611
10	9	-0.01446935628	0.06727942824	-0.009575522039	0.01154078078	0.00527516054	0.01485068258	8.222560707	50.32836734
11	10	-0.003732479643	0.09879908711	-0.009295710595	0.01790067554	-0.00698995823	0.002305752365	8.424107699	48.76220147
12	11	-0.01469138078	0.04524217546	-0.01012248709	0.007023937535	0.00213349727	0.01225598436	8.322230926	49.14925579
13	12	-0.01645686477	0.02502898313	-0.005176192266	0.003970558289	0.001735451515	0.006911643781	8.315705101	49.4424649
14	13	-0.005026165862	0.07643844932	-0.00910002063	0.01346768625	-0.001358531183	0.007741489448	8.561194019	48.98482874
15	14	-0.04007615894	0.03872289509	-0.01005217619	0.005734143779	0.01791574061	0.0279679168	7.998609505	49.43864412

Fig 7. Metrics obtained at each epoch

Analysis

Wasserstein Distance Analysis

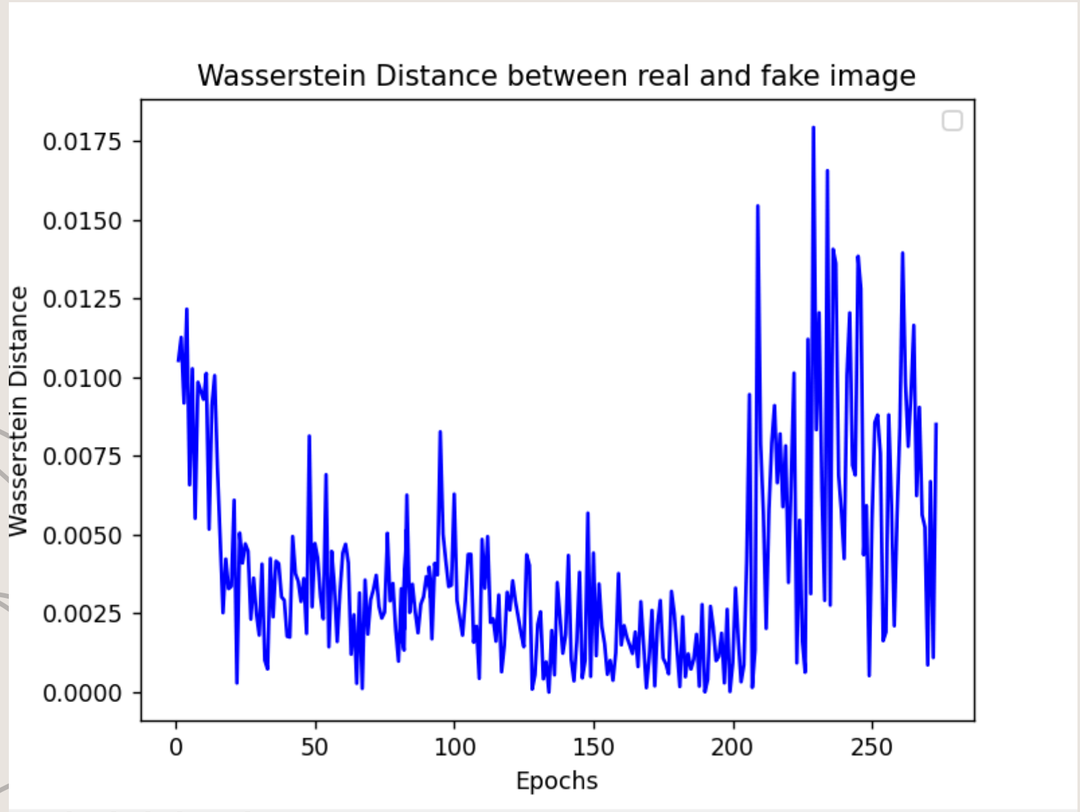

Fig 7.1 Wasserstein distance between the generated fake image and the real image

Score Analysis

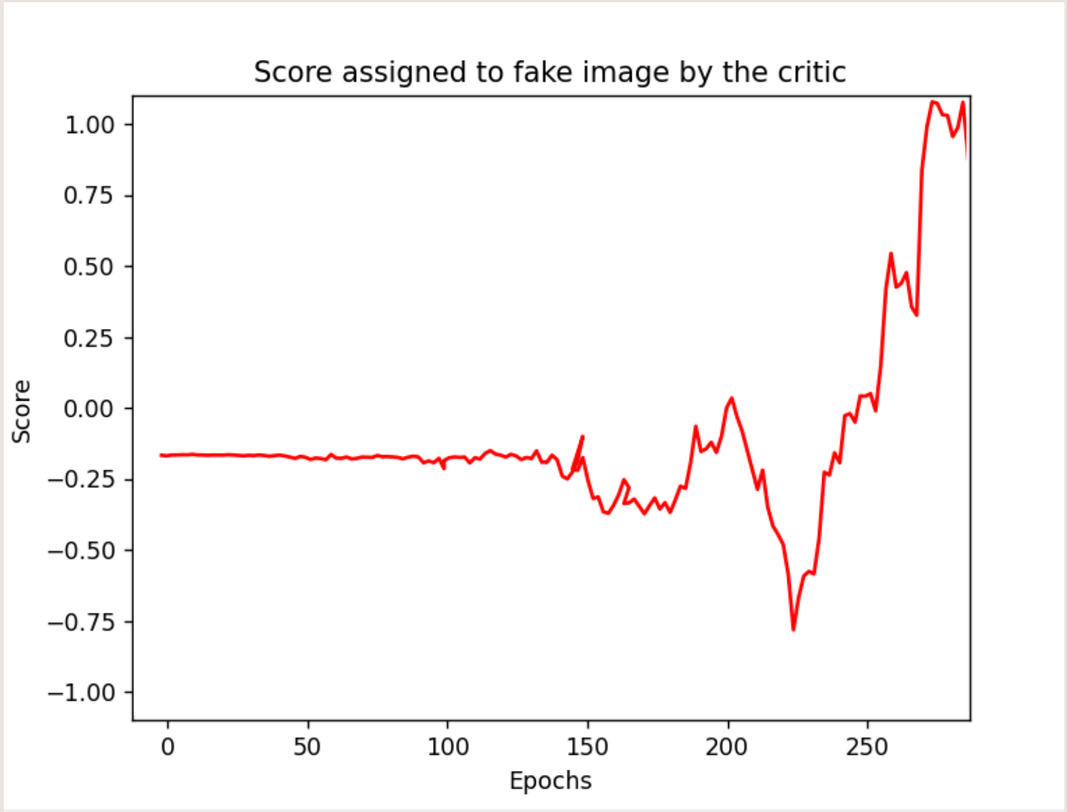

Fig 7.2 Score assigned to fake image by critic

[Graphs plotted using Matplotlib]

Analysis

Loss Analysis

Fig 7.3. Loss assigned to Generator and Critic

Accuracy Analysis

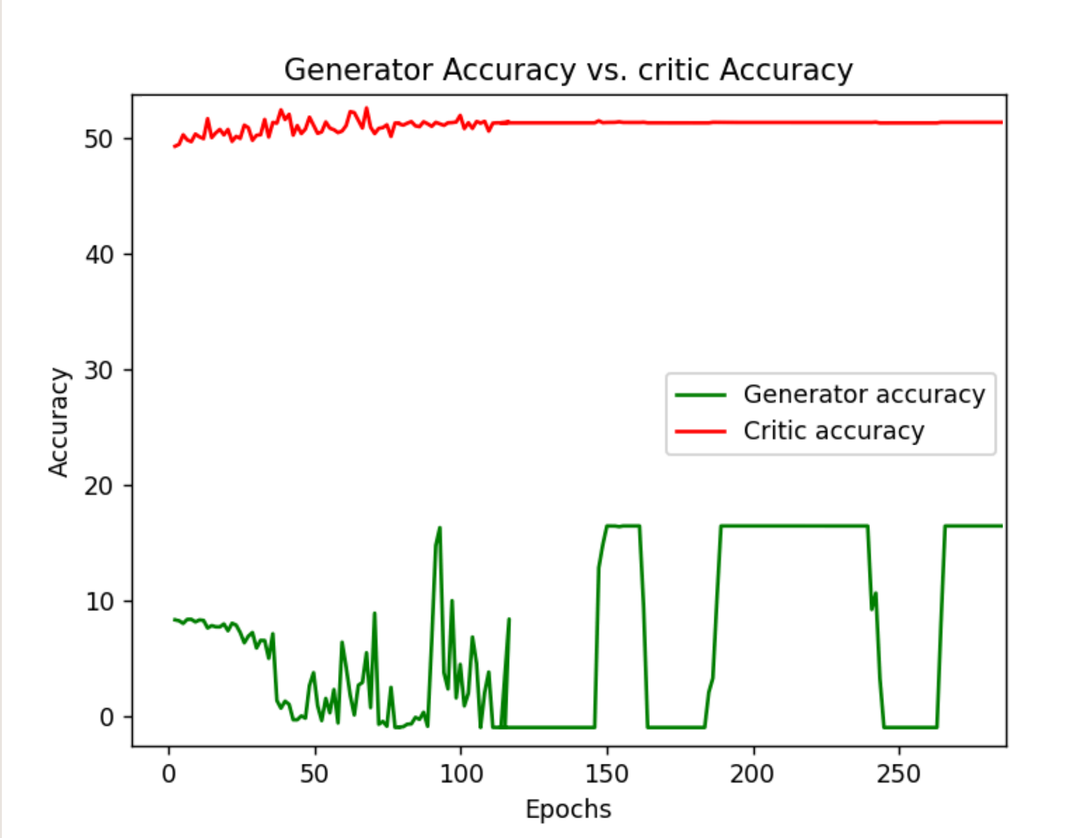

Fig 7.4 Accuracy of the Genearator and Critic at each epoch

[Graphs plotted using Matplotlib]

Quick Demo

- Code to get a key (image) by giving a medical image as input

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = torch.load('./deepkeygen_checkpoint.pth', map_location=torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu'))
generator = Generator()
generator.load_state_dict(model['generator_state_dict'])
generator.eval()

image_path = './mri_images/Tr-gl_0098.jpg'
image_path2 = './mri_images/Tr-gl_0099.jpg'

start_time = time.time()
medical_image = Image.open(image_path).convert("RGB")
medical_image = transform(medical_image)
medical_image = medical_image.unsqueeze(0)

generator.to(device)

medical_image = medical_image.to(device)
key = generator(medical_image).detach()
key_image = transforms.ToPILImage()(key.squeeze(0))

end_time = time.time()

os.system('cls' if os.name == 'nt' else 'clear')
print(f"Time took to generate to tranform key from medical image : { (end_time - start_time):.4f} sec ")
key_image.show()
Utils.save_image_key_pair(medical_image,key_image)

Utils.generate_histogram(key_image)
```

Fig 8. Implementaion of code which deals with the key generation with medical image as input

Output

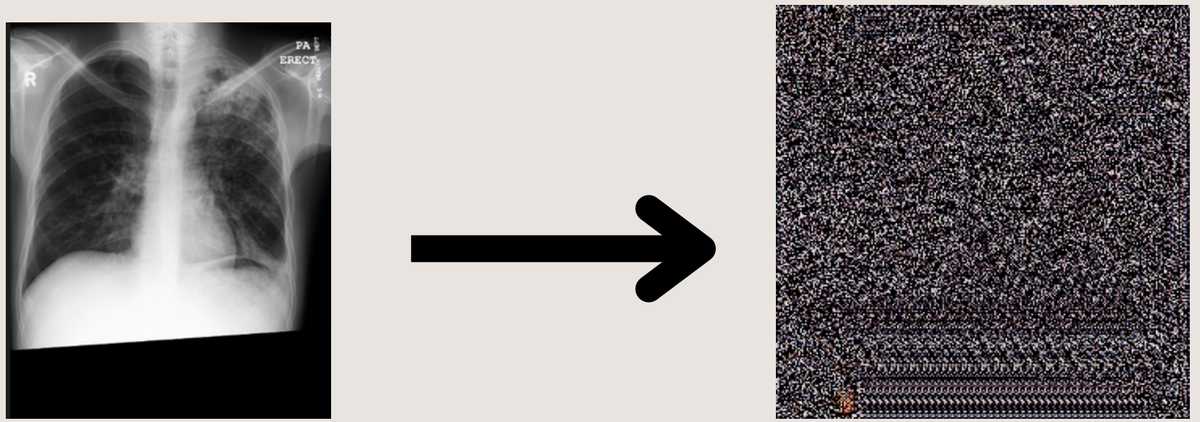

Fig 9.1. Input image and generated key

```
Time took to generate to tranform key from medical image : 0.5492 sec
12 ./image-key/image_key_pair_12
```

Fig 9.2. Time taken to generate the key

Output

- Frequency of the each gray scale value (0-255) in the key

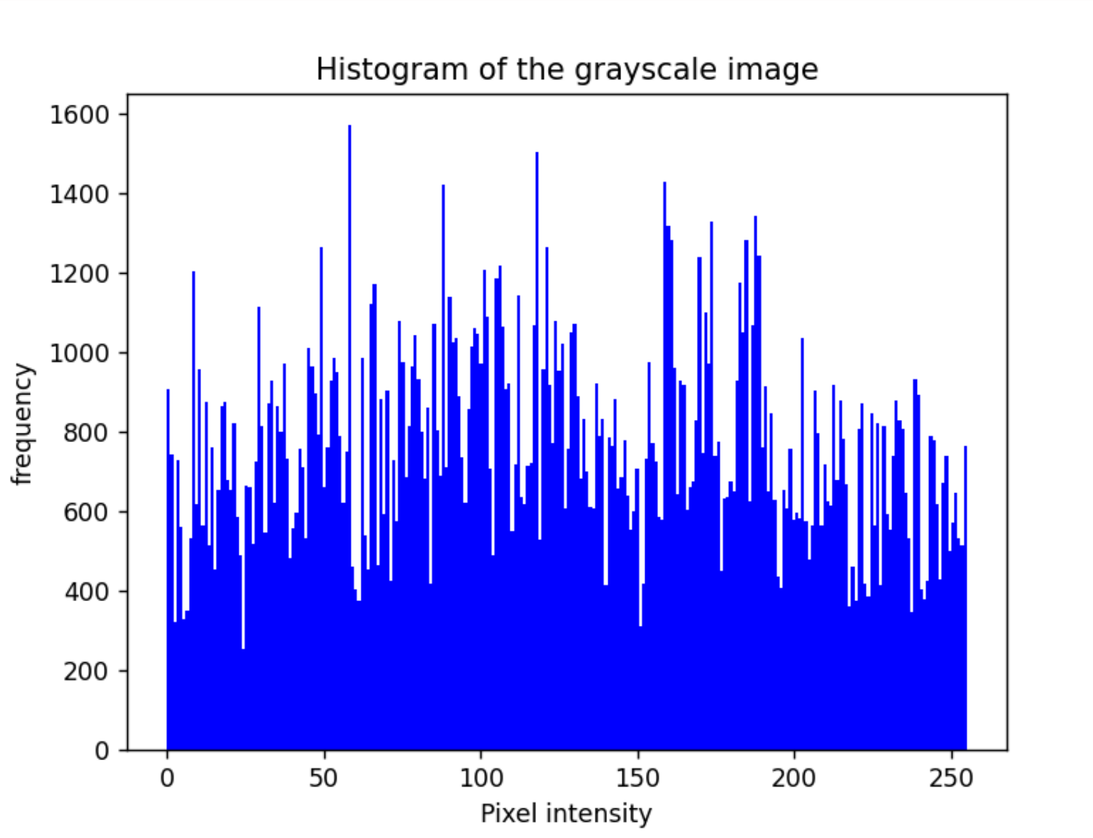

Fig 9.3. Gray Scale Histogram Analysis

[Graphs plotted using Matplotlib]

Output

- Number of Pixel Rate and Unified Averaged Changed Intensity between the 2 generated private keys and Entropy

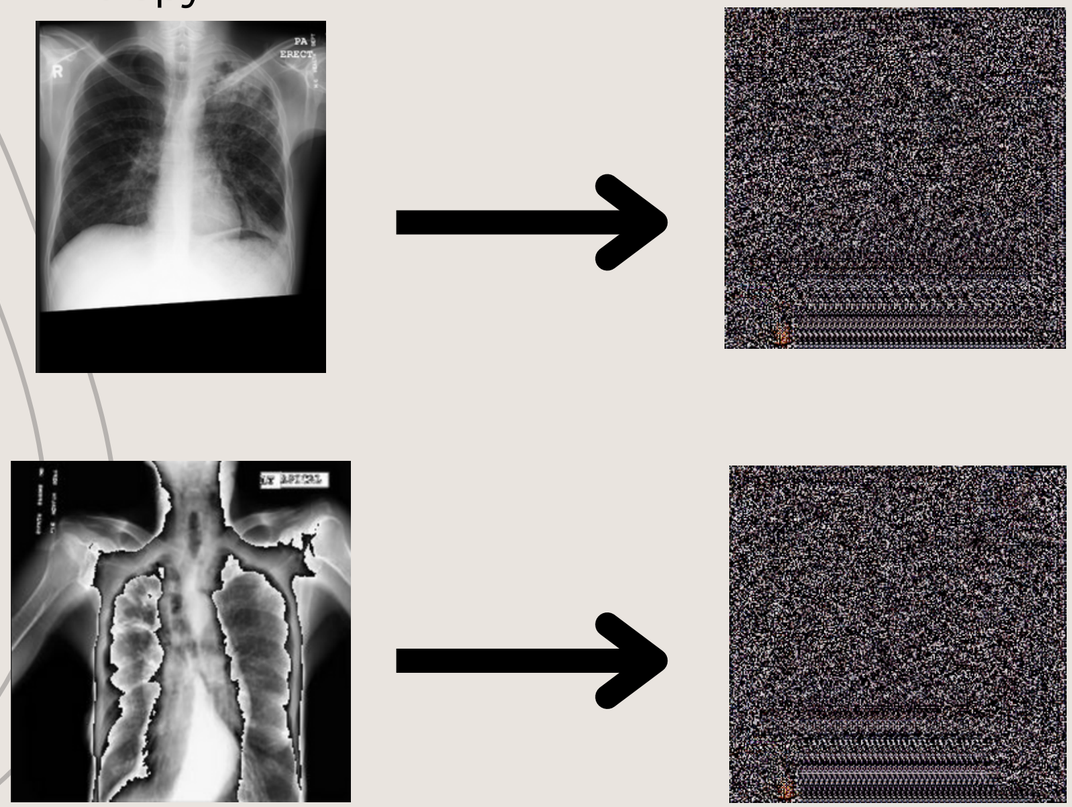

Fig 10.1. Image-Key pair generated using two differnt Iamges

```
al Studio/Shared/Python39_64/python.exe" "g:/My Drive/Sem6
thon_files/Utils.py"
NPCR: 77.49%
UACI: 30.47%
Entropy of key : 7.876251163718249
```

Fig 10.2. Number of Pixel Change Rate, Unified Averaged Changed Intensity between two generated private keys and Entropy of the key1

Output

- Difference between two generated private key

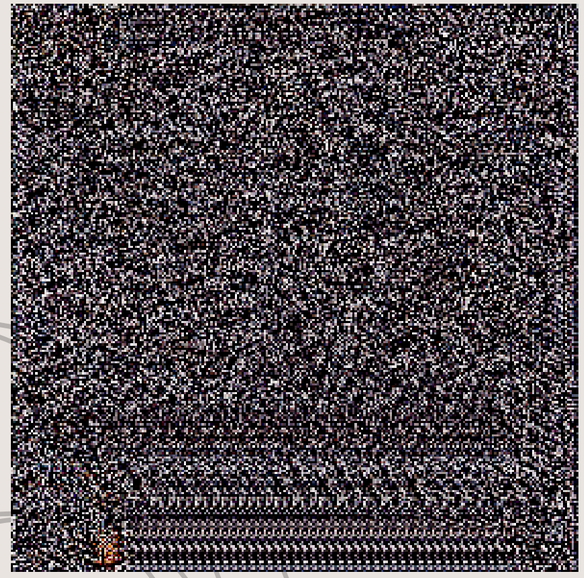

Fig 11.1. Generated private key 1

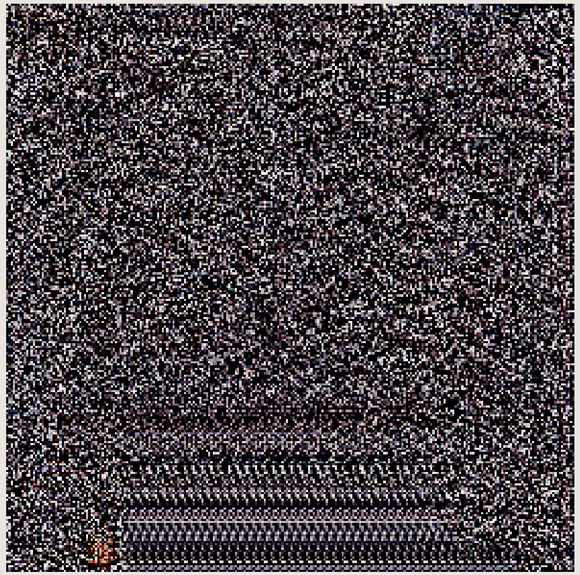

Fig 11.2. Generated private key 2

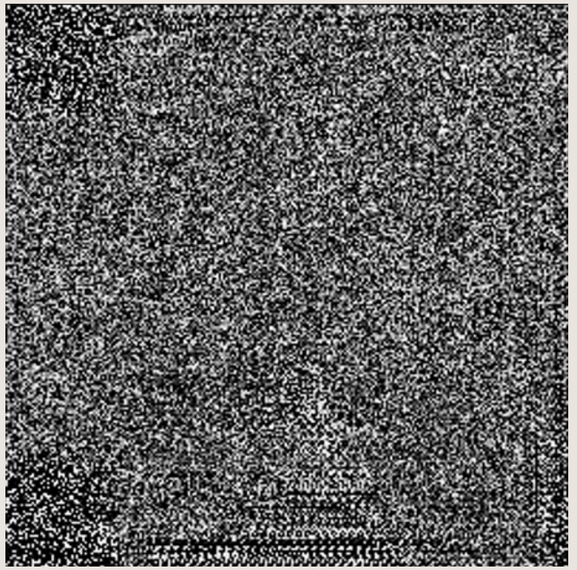

Fig 11.3. Difference between two keys

Fig 11. Visually represented difference between the two generated private keys

Comparison

- Comparion of the intermediate image generated at each level of 2 plain images

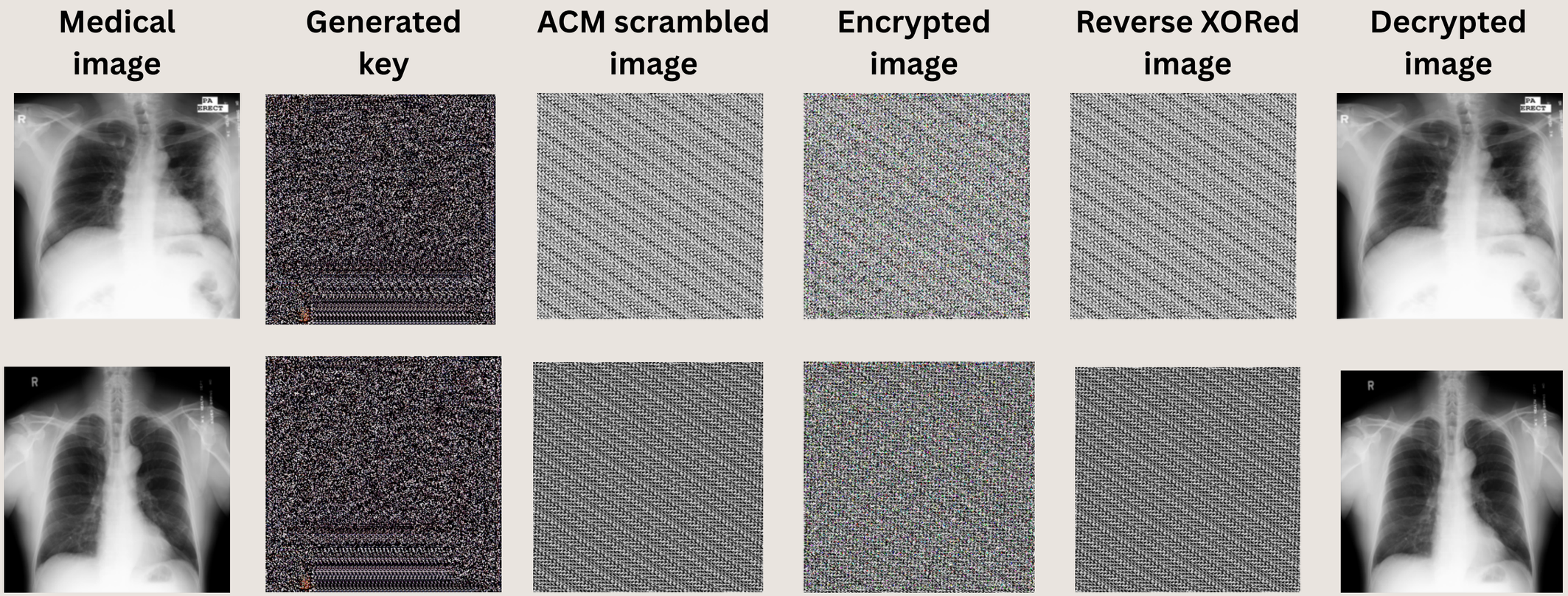

Fig 12. Output at each process of the encyption and decryption

Comparison

- **One-time-pad key assurance**
  - Four medical images are taken and the key is genearted using them.
  - They generated key is then used to decrypt each and every other encrypted image.
  - From the fig 13 the one time pad property of our key is verifed and thus the increasing the level of security.

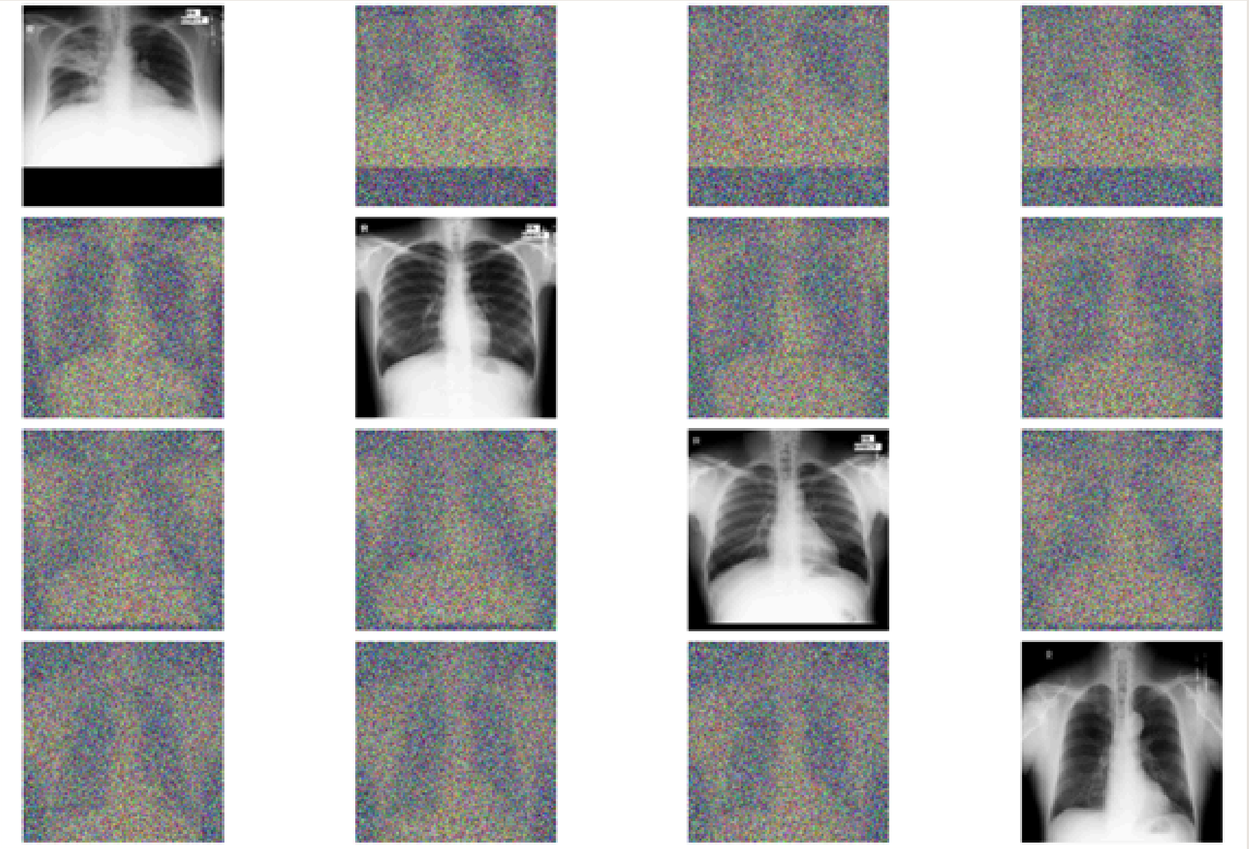

Fig 13. Demonstartion of the one time pad nature of the generated key

Comparison

- **Comparion of MSE and SSIM**

IMAGE ID	1	2	3	4	5	6	7	8
MSE	11017.56	10032.44	10224.44	10295.42	10432.93	10770.87	10391.61	10139.78
SSIM	0.0018	0.0017	0.0024	0.0019	0.0031	0.0023	0.0012	0.0019

Fig 13.1. Evaluation of MSE and SSIM of Montgomery chest Xray dataset using GAN and xor diffusion

Image ID	1	2	3	4	5	6	7	8
MSE	10781.14	15500.66	13775.3	11765.37	12449.99	15506.03	12596.16	11821.42
SSIM	0.010772344	0.006812764	0.005893204	0.007179169	0.007337721	0.005954015	0.006821211	0.010181674

Fig 13.2. Evaluation of MSE and SSIM Montgomery chest Xray dataset using WGAN-GP with ACM and xor diffusion

Comparison

- **Correlation between adjacent pixels :**

IMAGE ID	1	2	3	4
Horizontal	0.9636	0.9054	0.9421	0.9093
Horizontal*	0.0383	0.0280	0.0511	0.0395
Vertical	0.7077	0.9927	0.7917	0.8387
Vertical*	0.2259	0.1344	0.1785	0.1637
Diagonal	0.8477	0.8105	0.6266	0.8407
Diagonal*	0.1158	0.0380	0.0453	0.1242

Fig 13.3. Evaluation of corelation between adjacent pixels in Montgomery chest Xray dataset using GAN and xor diffusion

Image ID	1	2	3	4
Horizontal	0.995331033	0.997238166	0.995811247	0.996506074
Horizontal*	0.087681	-0.074636448	-0.110081893	0.110988083
Vertical	0.993644687	0.988450449	0.987106105	0.997980917
Vertical*	0.048936614	-0.105503275	-0.182751675	-0.079815119
Diagonal	0.990103407	0.98661297	0.983969796	0.995008
Diagonal*	0.080361195	0.054705851	-0.023182686	-0.021987201

Fig 13.4. Evaluation of corelation between adjacent pixels in Montgomery chest Xray dataset using WGAN-GP with ACM and xor diffusion

Comparison

- **Comparison between existing key generation algorithms**

Methods/Metrics	Non-overlapping Template Matching	Binary Matrix Rank	Maurer's Universal Statistical	Random Excursions Variant
Chaotic System	0	0.0504	0.2098	**0.1316**
LCG	0	0.9799	0.1966	0
Mersenne Twister	0	0.8106	0.1764	0
RSA	**0.998**	5.22e-88	0	0
Our Proposed Model	0.5671	**0.9902**	**0.67**	0.0517

- It can be concluded that the private key generated by the proposed model achieves a better performance in the term of randomness.

Conclusion

- The proposed WGAN-GP key generator is designed to automatically generate the private key by directly learning the desirable style.
- The propsed model uses the images in the transformation domain as the desired style of the private key.
- It adopts the learning network to translate the initial image in the source domain into the transformation domain.
- The generated private key is used to encrypt/decrypt medical images by employing ACM + XOR diffusion.
- While evaluating and comparing the metrics with the other key generation process, it is seen this our propsed model performs well than the other.
- Thus, it achieves a high level of security and it achieves good performance

What’s next ?

- We will be running further epochs to get a even more robust model.
- We are preparing a conference paper to present our project.
- We are planning to wrap up all process before April 30.

References

- [1] P. Tang, X. Wang, B. Shi, X. Bai, W. Liu, and Z. Tu, “Deep FisherNet for imageclassification,” IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 7, pp. 2244–2250, Jul. 2019.

- [2] G. Li and Y. Yu, “Contrast-oriented deep neural networks for salient object detection,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 12, pp. 6038–6051, Dec. 2018.

- [3] Y. Ding, C. Luo, C. Li, T. Lian, and Z. G. Qin, “High-order correlation detecting in features for diagnosis of Alzheimer’s disease and mild cog- nitive impairment,” Biomed. Signal Process. Control, vol. 53, Aug. 2019, Art. no. 101564.

- [4] H. Chen, Z. Qin, Y. Ding, L. Tian, and Z. Qin, “Brain tumor seg- mentation with deep convolutional symmetric neural network,” Neuro- computing, vol. 392, pp. 305–313, Jun. 2020, doi: 10.1016/j.neucom. 2019.01.111.

- [5] S. Jaeger, S. Candemir, S. Antani, Y. X. Wang, P. X. Lu, and G. Thoma, “Two public chest X-ray datasets for computer-aided screening of pulmonary diseases,” Quant. Imag. Med. Surg., vol. 4, p. 475, Dec. 2014. algorithm with NW algorithm for key generation,” J. Med. Syst., vol. 42, no. 1, p. 17, Jan. 2018.

References

- [6] Ultrasound Nerve Segmentation. Accessed: Jan. 9. [Online]. Available: https://www.kaggle.com/c/ultrasound-nerve-segmentation/data/
- [7] B. H. Menze et al., “The multimodal brain tumor image segmentation benchmark (BRATS),” IEEE Trans. Med. Imag., vol. 34, no. 10, pp. 1993–2024, Oct. 2015.
- [8] S. Bakas et al., “Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features,” Sci. Data, vol. 4, no. 1, Dec. 2017, Art. no. 170117.
- [9] S. R. Moosavi, E. Nigussie, M. Levorato, S. Virtanen, and J. Isoaho, “Low-latency approach for secure ECG feature based cryptographic key generation,” IEEE Access, vol. 6, pp. 428–442, 2018.
- [10] T. Liu, J. Tian, Y. Gui, Y. Liu, and P. Liu, “SEDEA: State estimation- based dynamic encryption and authentication in smart grid,” IEEE Access, vol. 5, pp. 15682–15693, 2017.

References

- [11] S. Kalsi, H. Kaur, and V. Chang, “DNA cryptography and deep learning using genetic algorithm with NW algorithm for key generation,” J. Med. Syst., vol. 42, no. 1, p. 17, Jan. 2018.

- [12] Y. Ding, C. Luo, C. Li, T. Lian, and Z. G. Qin, “High-order correlation detecting in features for diagnosis of Alzheimer’s disease and mild cog nitive impairment,” Biomed. Signal Process. Control, vol. 53, Aug. 2019, Art. no. 101564.

- [13] H. Chen, Z. Qin, Y. Ding, L. Tian, and Z. Qin, “Brain tumor seg mentation with deep convolutional symmetric neural network,” Neuro computing, vol. 392, pp. 305–313, Jun. 2020, doi: 10.1016/j.neucom. 2019.01.111.

- [14] S. Jaeger, S. Candemir, S. Antani, Y. X. Wang, P. X. Lu, and G. Thoma, “Two public chest X-ray datasets for computer-aided screening of pulmonary diseases,” Quant. Imag. Med. Surg., vol. 4, p. 475, Dec. 2014.

- [15] Ultrasound Nerve Segmentation. Accessed: Jan. 9. [Online]. Available: https://www.kaggle.com/c/ultrasound-nerve-segmentation/data/

Thank You