Image Copyright Protection Based on Blockchain and Zero-Watermark

Indian Institute of Technology (ISM) Dhanbad

Guided by

Prof. Sushanta Mukhopadhyay

Department of Computer Science and Engineering

IIT (ISM) Dhanbad

Submitted by:

Gouthami Kethavath (21JE0367)

Kush Milkesh Mistry (21JE0499)

Certificate

This is to certify that a project named "**Image Copyright Protection based on Blockchain and Zero-Watermark"** was designed and developed by **Gouthami Kethavath (21JE0367)** and **Kush Milkesh Mistry (21JE0499)** under the guidance of **Dr. Sushanta Mukhopadhyay**. This study report is submitted for final evaluation of the Final Year BTECH Project during the academic session **2024 to 2025** of the **Indian Institute of Technology (Indian School of Mines), Dhanbad**.

Dr. Sushanta Mukhopadhyay

Associate Professor

Dept. Of Computer Science and Engg.

IIT(ISM) Dhanbad

Image Copyright Protection Based on Blockchain and Zero-Watermark

Abstract

This report introduces a framework for image protection and authentication that combines blockchain technology with zero-watermarking to address the limitations of traditional digital watermarking. While digital watermarking is commonly used for data protection, it often relies on trusted third parties and can lead to irreversible data loss. Zero-watermarking avoids data alteration but depends even more on trusted entities, limiting its use. By incorporating blockchain's decentralized, immutable structure and leveraging smart contracts for automated, transparent processing, this framework eliminates the need for intermediaries. Additionally, using the Inter-Planetary File System (IPFS) solves blockchain's scalability issues, ensuring efficient storage without excessive data expansion. This solution mitigates the drawbacks of both traditional watermarking and zero-watermarking, offering a scalable, secure method for image ownership protection.

In addition to this, an Image Normalization procedure has been integrated with the zero-watermarking, to produce normalized images which are robust against affine geometric attacks. Also, an improved scrambling method called Henon Mapping has been used in the algorithm, for dynamic encryption of image data, providing chaos-based security. The experimental results at the end of the report provide an insight on the robustness of the zero-watermarking algorithm.

Table Of Contents:

1. Introduction

Images serve as powerful carriers of information across the internet, widely used in advertising, media, and social networks. While this accelerates information sharing, it also increases the risk of misappropriation, as images can be easily reproduced and shared. This ease of duplication makes it challenging to prevent misuse, and even when theft is discovered, proving ownership can be difficult. Digital watermarking emerged as a solution, embedding a watermark onto images to verify ownership. However, this method still often requires a trusted third party for notarization, making it vulnerable to issues such as intermediary dependency, unauthorized reselling, and misuse by image buyers. Blockchain technology, particularly through Ethereum and smart contracts, provides a secure and decentralized solution for protecting intellectual property rights in digital images, enabling reliable authentication, ownership verification, and preventing misappropriation without the need for a trusted third party. Its transparent and permanent storage capabilities support digital watermarking and enhance rights protection, simplifying image transactions and deterring theft. While blockchain-based evidence systems exist, few are tailored specifically to image ownership, lacking a comprehensive verification platform. Ethereum's practical application offers an affordable solution for image notarization despite rising costs in blockchain services, with potential for increased stability as global regulatory frameworks evolve. This approach thus fosters cultural industry growth, supports intellectual property protection, and facilitates cultural exchange.

Several key innovations of this project are highlighted below:

- **Trusted Third Party Elimination:** By creating a more flexible and implementable approach, this system removes the need for a trusted third party in digital watermarking. From the start, all ownership changes are securely recorded on the blockchain, allowing for transparent verification and querying of ownership history.
- **Enhanced Zero-Watermarking with Blockchain:** An improved zero-watermarking algorithm incorporating Image Normalization and Henon Mapping is implemented, where the Ethereum blockchain securely stores essential ownership information. Smart contracts (SC) on Ethereum enable a lossless and secure image watermarking process, ensuring both data integrity and security.
- **Comprehensive Security Analysis:** This project tests the robustness of the proposed zero-watermarking algorithm by simulating various types of attacks, such as noise, filtering, and affine geometric distortions.

2. Previous Works

A. Blockchain

Blockchain Technology Overview

- In recent years, blockchain technology has gained prominence due to the rise of digital currencies, driving substantial research and development. Blockchain represents a new model of encryption that combines point-to-point communication, consensus mechanisms, and distributed data storage. At its core, blockchain is a distributed database platform that fosters trust between nodes through consensus algorithms, enabling the secure and decentralized sharing of data.
- Blockchain is an open, programmable platform capable of recording not only financial transactions but also executing and managing valuable records through smart contracts (SCs). Key attributes of blockchain include tamper-proof data storage, traceability, and permanence, making it well-suited for applications that require secure data storage.

Blockchain-Based Storage Frameworks

- Several frameworks have been developed in the recent past to enhance the protection of digital data using blockchain. One such extensible framework ensures evidence integrity and privacy while balancing traceability. Another storage and recovery plan leverages data consensus and smart contracts to enable quick repair of local regenerative code, improving resource efficiency for industrial use.
- Additionally, private and anonymous decentralized systems help address issues like data leakage and identity breaches in crowd-sourcing applications by using blockchain technology.

Blockchain for Digital Image Protection

- Blockchain has also been combined with digital watermarking to protect multimedia transactions. For example, a tracking mechanism using homomorphic encryption has been developed to ensure that media information transactions are secure and leak-proof, with blockchain automating the watermark embedding process.
- In the context of industrial IoT, private blockchain systems have been used to secure image transmissions without third-party involvement, enhancing the security of

image information. Although these systems use cryptographic measures to limit access, they lack the subtlety of watermarking techniques, which are better suited for unobtrusive protection.

- In digital rights management, blockchain combined with watermarking can prevent unauthorized image sharing outside of the blockchain environment, although these solutions still face challenges related to data loss during image protection.

B. Zero-Watermarking

Traditional and Zero-Watermarking Techniques

- Traditional digital watermarking techniques embed watermark information into the original image, which inevitably alters the image data. Zero-watermarking has emerged as a solution to this issue by constructing the watermark without modifying the original image. One approach uses high-order cumulants to build a zero-watermark that demonstrates robust resistance to noise filtering and minor rotation attacks.
- Building on this, a scheme that combines zero-watermarking with DWT decomposition has been proposed. This method segments the image into blocks and uses DWT decomposition, resulting in improved resistance to shearing, noise, and compression attacks.

Robustness to Geometric Distortions

- Many watermarking schemes struggle with robustness when subject to affine transformations, resulting in vulnerability to geometric distortions. An algorithm that uses image normalization techniques was developed to address this, converting images into a standardized format. This standardization enables the watermark to better resist affine transform attacks and enhances overall robustness.
- By applying an XOR operation between binary feature images and watermarks generated through chaotic sequences, i.e. Henon scrambling, the scheme creates zero-watermarked images that offer resilience against various attacks.

3.Fundamental Understanding

A. Zero-Watermarking Prerequisites

Zero-watermarking is a robust technique for protecting digital images without altering them visibly. Here, we discuss three key prerequisites: Henon Mapping, Discrete Wavelet Transform (DWT), and Singular Value Decomposition (SVD).

1. Henon Mapping

Henon Mapping is a chaotic function used to randomly (but deterministically) scramble pixels. This ensures the scrambling is different for each image and each pair of keys, known as chaos sensitivity. It prevents attackers from learning a fixed scrambling pattern (unlike Cat Map).

Parameters:

- a, b: The chaos key used to generate pixel positions. This pair of keys is generated uniquely during runtime.
- x, y: This pair is used as seed for chaos. It is initialized to (0.1, 0.1)
- No. Of iterations: No. Of rounds performed to generate mappings (more means better scrambling).
- N : Order of the image.

Henon Map Equations:

$$x_{n+1} = y_n + 1 - a \cdot x_n^2$$

$$y_{n+1} = b \cdot x_n$$

- Starting from initial seed conditions:- x = 0.1, y = 0.1,
- We compute (x, y) points for n^2 × (no. Of iterations) times.
- Then, we convert (x, y) to valid pixel positions: (i, j), by scaling to 10^6 and taking modulo n.
- Use these to create a mapping from original pixel positions to new positions.
- We flatten the image to 1D, reassign values using the new order, then reshape the scrambled image array back to 2D.

2. Discrete Wavelet Transform (DWT)

The DWT is derived from the continuous wavelet transform and is useful for analyzing image frequency content. The discrete wavelet function of an arbitrary function f(t) can be expressed as:

$$\Phi_{j,k}(t) = 2^{-j}\Phi(2^{-j}t - k)$$

Variables:

- j and k: Scaling and translation parameters, respectively.

The DWT of the function f(t) is defined as:

$$F_{j,k} = \int f(t)\overline{\Phi_{j,k}}(t)dt$$

Upon performing wavelet decomposition, an image is split into four frequency bands: low-frequency, vertical, horizontal, and diagonal bands. A secondary decomposition of the low-frequency band further refines these into more bands. The low-frequency sub band, LL2, typically captures the majority of the image's energy and remains stable against external interference, making it ideal for zero-watermarking.

3. Singular Value Decomposition (SVD)

SVD is a powerful method for decomposing matrices, particularly useful for images. For an image matrix A of size m*n, the decomposition is expressed as:

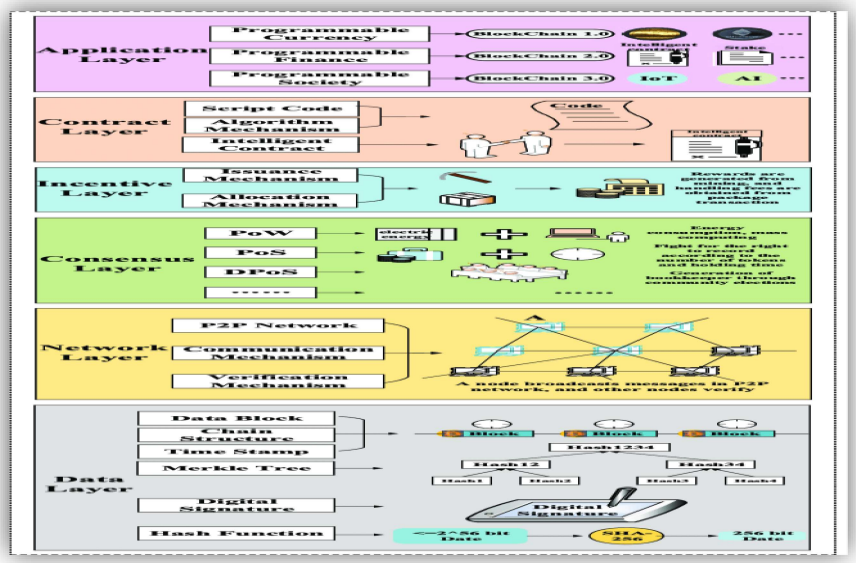
$$A = USV^T$$

Variables:

- U: Left singular value matrix of size m× m.
- V: Right singular value matrix of size n× n.
- S: Diagonal matrix of singular values.

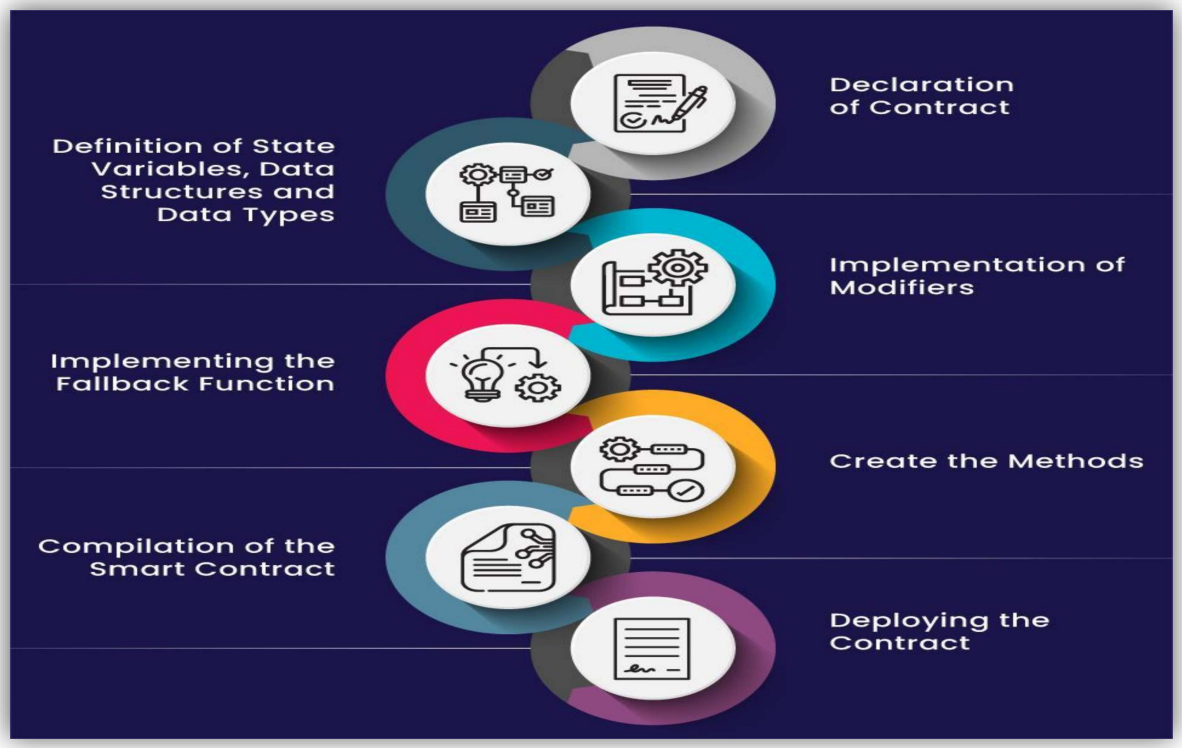
The singular values of an image are robust and do not change significantly with image processing operations. This stability enhances the robustness of the zero-watermarking algorithm, ensuring the watermark remains resilient against various types of attacks.

B. Blockchain and Ethereum



Blockchain technology is structured into six layers: data, networking, consensus, incentives, contracts, and applications. The data layer serves as the foundation, organizing blocks and their connections, while the networking layer facilitates the distribution and verification of information across the decentralized network. The consensus layer ensures agreement among participants, often paired with the incentives layer to motivate network maintenance. The contract layer includes smart contracts-self-executing agreements encoded in the blockchain, while the application layer hosts various decentralized applications (d Apps), showcasing the versatility and adaptability of blockchain solutions.

Ethereum, introduced by Vitalik Buterin in 2013, is a public blockchain platform known for its capability to support smart contracts (SCs). It features two account types: Externally Owned Accounts (EOAs), which are controlled by users via private keys, and Contract Accounts (CAs), which hold and execute smart contract code. The Ethereum Virtual Machine (EVM) serves as the runtime environment for these contracts, enabling secure execution in a sandboxed environment. The EVM uses a gas pricing mechanism to manage transaction costs, ensuring efficient resource utilization and protecting the network from abuse.



Ethereum Virtual Machine (EVM): The EVM is a critical component of Ethereum, functioning as a separate sandbox where smart contracts execute. It provides an environment that isolates contract code from the external network, ensuring security and reliability.

- **Gas Pricing Mechanism:** The EVM employs a gas pricing mechanism to manage the costs associated with executing smart contracts. Every operation consumes a certain amount of gas, which must be paid for with Ether. If a transaction runs out of gas before completion, it is rolled back, preserving the integrity of the blockchain.
- **State Management:** The EVM acts as a state machine, continuously monitoring changes in state as transactions occur. Each new transaction triggers an update to the system state, ensuring that all nodes maintain a consistent view of the blockchain.
- **Contract Creation Process**: Creating a new contract involves a fixed set of parameters, which initiate a transaction to establish the contract and its associated states.

IPFS:-The Interplanetary File System (IPFS) is a decentralized protocol designed for permanent file storage and sharing. Instead of relying on traditional location-based addressing, IPFS uses unique hash values generated from file content, which helps eliminate duplicates and supports version control. Operating on a peer-to-peer network, IPFS connects various nodes, allowing them to share data and maintain a hash table for file queries. By integrating with File Coin, a cryptocurrency that incentivizes data storage, IPFS creates a decentralized economy for file management, while the Interplanetary Naming System (IPNS) maps hash values to more user-friendly names for easier access.

4.System Architecture

Process Overview:

The process of interaction and transfer of watermark and image data, between the Image Owner and the Image user through Blockchain, is given below:-

1. **IO (Image Owner) initializes** the Depository contract and sets parameters, then **deploys the smart contract (SC)** on Ethereum.
2. **IU (Image User) registers** and sends a request to IO.
3. **IO verifies IU's identity** and gives restricted query access.
4. IO shares contract information with IU, which IU uses to deploy and connect the Validation Contract with the Depository Contract.
5. IO **applies the zero-watermark algorithm** to create zero-watermark and stores the original image, the logo image, and the zero-watermark on IPFS.
6. **IO stores the IPFS location** on Ethereum and records it, along with the scrambling parameters in the Depository Contract.
7. **IU retrieves** the IPFS address by calling the Validation Contract to search and access the images.
8. IU **downloads images from IPFS** and uses the scrambling parameters to recover the logo image.

Five Stages:

The above process can be divided into five stages:-

1. **Initialization Stage**: IO initializes and deploys the Depository SC on Ethereum.

2. **Request Stage**: IU registers with IO, which verifies and authorizes IU's Ethereum address.
3. **Watermark Generation Stage**: IO creates the zero-watermark and stores the watermark data along with the images on IPFS, recording the location on Ethereum.
4. **Retrieval Stage**: IU retrieves the location and scrambling parameters through the Validation SC.
5. **Use Stage**: IU verifies ownership using the scrambling parameters, finalizing the retrieval of the image and watermark.

5.Implementation

In this section, we implement a zero-watermarking algorithm which is tailored for blockchain use. Some notations which have been used in the algorithm are given in the table below:-

TABLE I
SOME NOTATIONS USED BELOW

Notations	Description
IO	Image Owner is a person or an organization who has images to protect
IU	Image Users want to confirm image ownership or obtain image usage rights
Key	Scrambling parameters
F	The original image
F'	Scrambled image
W	Watermark figure (In this case, ownership information)
W'	Scrambled watermark figure
ω	Zero watermark
LOCATION_ADDR	Location of files in IPFS
TXH	Transaction Hash
RESULT	A set of search results returned by image depository contract

A. Zero-Watermark Algorithm Design

In this framework, a robust zero-watermarking algorithm is introduced. To enhance robustness, the algorithm incorporates an image normalization procedure, discrete wavelet transform, and singular value decomposition.

1. **Image Normalization**: This technique is used to standardize the image, making it resistant to geometric distortions like translation, rotation and scaling.

The normalization procedure consists of the following steps for a given image $f(x,y)$

1. Center the image $f(x,y)$. .

   This is achieved by performing a translation affine transform, setting

   $\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and the translation vector $\mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}$ with

$$d_1 = \frac{m_{10}}{m_{00}}, \quad d_2 = \frac{m_{01}}{m_{00}}$$

   in the equation of affine transform, which converts f(x,y) to g(x,y), such that g(x,y) = f(x_a , y_a) , where:-

$$\begin{pmatrix} x_a \\ y_a \end{pmatrix} = \mathbf{A} \cdot \begin{pmatrix} x \\ y \end{pmatrix} - \mathbf{d}.$$

   Here, m_{10} , m_{01} , and m_{00} are first order geometric moment in x-direction, first order geometric moment in y-direction, and zeroth order moment respectively. This step aims to achieve translation invariance. Let the resulting centered image be denoted by $f_1(x,y)$

2. Apply a shearing transform to $f_1(x,y)$ in the x-direction with d = 0 and matrix

$$\mathbf{A}_x = \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix},$$

   so that the resulting image, denoted by $f_2(x,y) \triangleq A_x[f_1(x,y)]$ achieves $\mu_{30}^{(2)} = 0$ where the superscript is used to denote $f_2(x,y)$.

   It is achieved by calculating β from the cubic equation (where $\mu_{pq}^{(1)}$ is a central moment of $f_1(x,y)$)

$$\mu_{30}^{(1)} + 3\beta\mu_{21}^{(1)} + 3\beta^2\mu_{12}^{(1)} + \beta^3\mu_{03}^{(1)} = 0.$$

   Here, there are two possibilities :- either one root is real, and other two complex, or all the three roots are real. In the first case, we set β to be the real root, and in the second case, we set β as the median of the three roots.

3. Apply a shearing transform to $f_2(x,y)$ in the y-direction with d = 0 and matrix

$$\mathbf{A}_y = \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix}$$

so that the resulting image, denoted by $f_3(x,y) \triangleq A_y[f_2(x,y)]$

achieves $\mu_{11}^{(3)} = 0$

This is achieved by setting $\mu_{11}^{(3)} = 0$ in the equation

$$\mu_{11}^{(3)} = \gamma\mu_{20}^{(2)} + \mu_{11}^{(2)}$$,thus obtaining γ

4. Scale $f_3(x,y)$ in both x and y directions with

$$\mathbf{A}_s = \begin{pmatrix} \alpha & 0 \\ 0 & \delta \end{pmatrix},$$

so that the resulting image, denoted by $f_4(x,y) \triangleq A_s[f_3(x,y)]$

achieves a) a prescribed standard size

and b) $\mu_{50}^{(4)} > 0$ and $\mu_{05}^{(4)} > 0.$

The magnitudes of scaling parameters α and δ are determined by resizing the Image to a prescribed standard size in both x and y directions.

Their signs are determined such that both

$\mu_{50}^{(4)}$ and $\mu_{05}^{(4)}$ are positive,

which can be changed by flipping either horizontally or vertically. The resulting image $f_4(x,y)$ is the final normalized image.

The normalization procedure effectively removes any distortions, such as translation, Rotation, scaling, shearing, or a combination of these, produced by an affine attack. Thus, even when the image is attacked, the normalized image produced is the same as that produced by the original image.

2. **Wavelet Transform and SVD**: By focusing on low-frequency LL2 coefficients obtained by discrete wavelet transform and applying singular value decomposition (SVD), the algorithm gains stability and resilience against noise, filtering and compression attacks.

The algorithm comprises two main parts: **zero-watermark generation** and **zero-watermark verification**.

1) Zero-Watermark Generation Process

The generation process creates a unique zero -watermark using the following steps:

- **Step 1**: Normalize the original image to an image of a standard size 512 x 512.
- **Step 2**: Extract a square region of the normalized image, centered on its centroid, of size 128 x 128. This is the effective region used for extracting wavelet coefficients.
- **Step 3**: Apply a 2-level discrete wavelet transform to the effective region. Then, divide the resulting low-frequency LL2 sub band into 2 x 2 subblocks, yielding a total of K = 256 subblocks, calculated as:-

  $$K = \frac{N^2}{(2^l \cdot n)^2}$$

  Where l = 2 (i.e. level of decomposition), N = 128 (i.e. order of the effective region), and n = 2 (i.e. order of subblock)
- **Step 4**: Perform SVD on each subblock and record the maximum singular value for each, forming a 16 x 16 matrix B.
- **Step 5**: Generate a feature image F by encoding the parity of each element's highest bit in B (1 for odd, 0 for even).
- **Step 6**: Generate a unique pair of keys (a,b), and apply Henon scrambling to the logo image and the feature image, using the unique key pair.
- **Step 7**: Generate the final binary zero-watermark image v by XORing the scrambled logo image W0 and scrambled feature image F0.

2) Zero-Watermark Validation Process

The validation process verifies image ownership by comparing the original and recovered logo images.

- **Step 1**: Normalize the original image to an image of standard size 512 x 512.
- **Step 2**: Extract a square region of the normalized image, centered on its centroid, of size 128 x 128. This is the effective region used for extracting wavelet coefficients.
- **Step 3**: Apply a 2-level discrete wavelet transform to the effective region. Then, divide the resulting low-frequency LL2 sub band into 2 x 2 subblocks, yielding a total of 256 subblocks.
- **Step 4**: Perform SVD on each subblock, and use the maximum singular values to form a 16 x 16 matrix B.
- **Step 5**: Generate a feature image F by encoding the parity of the highest bit for each element in B (1 for odd, 0 for even).

- **Step 6**: Use the saved key pair (a,b) to scramble F using Henon Mapping.
- **Step 7**: Perform XOR on the scrambled feature image F0 and the stored watermark image v to retrieve the scrambled logo image W0.
- **Step 8**: Perform inverse Henon scrambling to recover the logo image. Compare the recovered logo image with the original logo image, using the average Normalized Correlation value (NC), calculated as:-

$$NC(F,\widehat{F}) = \frac{\sum_i \sum_j F(i,j)\widehat{F}(i,j)}{\sqrt{\sum_i \sum_j F(i,j)^2}\sqrt{\sum_i \sum_j \widehat{F}(i,j)^2}}$$

Where F and F –cap represent the original and the recovered logo image respectively, and the value of NC is a decimal number less than 1. The larger the value, the better the restored watermark effect will be.

- If NC value exceeds a set threshold = 0.8, copyright authentication is successful; otherwise, it fails.

B. Smart Contract Design

The contract system is designed to support only the **Image Owner (IO)** and authorized **Image Users (IU)**. The contract enforces confidentiality of the entities involved and the security of the image and watermark data.

1. **Image Depository Contract**: Created by the IO during initialization, this contract manages image storage and access. Key elements include:
    a. **DATA_OWNER**: Stores the IO's Ethereum address.
    b. **AUTH_USER**: A mapping that keeps track of authorized users.(address → bool)
    c. **KEYS**: A mapping that stores watermark-related data(string -> Key)
    d. **STRUCT KEY**: A structure that holds:
        - **Scrambling parameters:** Used in the watermarking algorithm for scrambling. (i.e., Henon mapping key pair).
        - **IPFS address:** The location of the watermark data stored on IPFS.
        - **User information:** Identifiers like the user's email or ID.

2. **Image Validation Contract**: This contract allows IU to search the Image Depository Contract privately, and it stores the returned search results in a secure structure accessible only to the IU. Key components include:-
    a. **CONTRACT_ADDR:** Address of the deployed Image Depository Contract.
    b. **repository:** An instance of the Image Depository Contract for interaction.
    c. **OWNER:** Stores the IU's Ethereum address.
    d. **RESULTS:** A mapping to store retrieved watermark data (string → KeyInfo).
    e. **STRUCT KEYINFO:-** Holds the "scramblingParameters" and "ipfsAddress".

Algorithm 1: Main interfaces of the image depository contract.

address payable DATA_OWNER;

mapping(address => bool) public auth_user;

mapping(string => key) keys;

// function addUser

Input: address of user;

Output: result;

1: require(DATA_OWNER == msg.sender);

2: auth_user[user] = true;

3: return result;

// function removeUser

Input: address of user;

Output: result;

1: require(DATA_OWNER == msg.sender);

2: delete auth_user[user];

3: return result;

// function addKey

Input: Scrambling parameters, IPFS address, user information;

Output: result;

1: require(DATA_OWNER == msg.sender);

2: keys[information] = k1;

3: return result;

// function deleteKey

Input: user information;

Output: result;

1: require(DATA_OWNER == msg.sender);

2: delete keys[information];

3: return result;

// function Search

Input: user information;

Output: Scrambling parameters, IPFS address;

1: require(msg.sender == DATA_OWNER || auth_user[msg.sender] == true);

2: string memory IPFS_addr = keys[information].IPFS_address;

3: return public view Scrambling parameters, IPFS address;

// function Withdraw

Input: NONE;

Output: result;

1: external modifier onlyOwner() {

2: require(msg.sender == DATA_OWNER);

3: }

4: DATA_OWNER.transfer(address(this).balance);

5: return result;

Details of the Interface functions of Image Depository Contract:-

1. Add User:- Allows the IO to add an IU to the list of authorized users.

- IO calls this function, passing the IU's Ethereum address.
- This function ensures that only the IO can add users, by checking the address of the caller of the function and matching with the address of IO, stored in "DATA_OWNER".
- The IU's address is added to the mapping "auth_user" with the value "true".

2. Remove User:- Allows the IO to revoke an IU's authorization.

- **IO calls this function**, passing the IU's Ethereum address.
- The IU's address is removed from "auth_user".

3. Add Key:- Stores the watermark data and keys associated with a particular image.

- **IO calls this function**, providing:
    - **"userInfo":** Identifier (e.g., image ID or user email).
    - **"scramblingParameters":** Parameters used in the zero-watermarking algorithm (Cat mapping keys).
    - **"ipfsAddress":** IPFS hash where the watermark data is stored.

4. Delete Key:- Removes watermark data associated with "userInfo".

- **IO calls this function**, specifying the “userInfo” to delete.
- The entry in “keys” mapping is deleted.

5. Search:- Allows authorized users to retrieve watermark data.

- Either IO or IU calls this function, providing “userInfo”.
- This function ensures only the IO or authorized IUs can access the data, by checking whether the address of the function caller matches with the IO Address, or an authorized IU Address.
- Returns the “scramblingParameters” and “ipfsAddress” associated with “userInfo”.

Algorithm 2: Main interfaces of the image validation contract.

```
// Main Interfaces for Image Depository Contract Interaction

 address public CONTRACT_ADDR = 0xE26BeDB0ea15b34e720805ee982B647a428671c6;

 // Assign Depository contract address to variable CONTRACT_ADDR

Repository repository = Repository(CONTRACT_ADDR);

// function deposit
// Input: value;
// Output: result;
function deposit(uint256 value) public returns (bool result) {
   address payable addr = payable(address(this));
   addr.transfer(value);
```

```
    return result;
}

// function Search
// Input: user information;
// Output: none;
function search(string memory information) public {
    require(msg.sender == OWNER);
    RESULTs[information] = repository.search(information);
}

// function getResult
// Input: user information;
// Output: Scrambling parameters, IPFS address;
function getResult(string memory information) public view returns (ScramblingParams
params, string memory IPFS_address) {
    require(msg.sender == OWNER);
    string memory IPFS_addr = RESULTs[information].IPFS_address;
    return (params, IPFS_addr);
}
```

Details of the Interface Functions of the Image Validation Contract:-

1. Constructor:- Initializes the contract with the address of the Image Depository Contract. It establishes the connection to the Image Depository Contract for data retrieval.

- IU deploys the contract, providing the Image Depository Contract's address.
- Sets “OWNER” to the IU's address.

2. Search:- Retrieves watermark data from the Image Depository Contract.

- **IU calls this function**, providing “userInfo”.
- This function ensures only the IU can access the watermark data, by checking the address of the function caller and matching it with the IU address, stored in “OWNER”.
- This search function in the Image Validation Contract calls the search function of the Image Depository Contract using the “repository” variable, and passing the “userInfo” value to it, to get the data.
- Stores the data in “RESULTS”.

3. Get Result:- Allows the IU to access the retrieved watermark data.

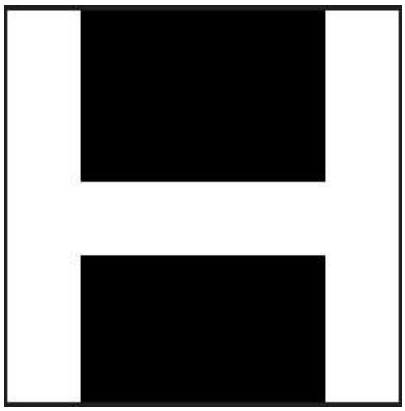
- **IU calls this function**, providing “userInfo”.
- Like the search function of the Image Validation Contract, this function ensures only the IU can access the watermark data.
- Returns the “scramblingParameters” and “ipfsAddress” from “RESULTS”.

6.Experimental Results and Analysis

Input Images for the Zero-Watermarking Algorithm:-

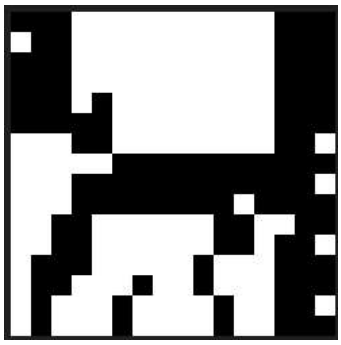
1. **Original Image**:- A PNG grayscale image of dimensions 720 x 576 pixels.

2. **Logo Image:-** A PNG binary image of dimensions 16 x 16 pixels.



Output Zero-Watermark Image by the Zero-Watermarking Algorithm:-

Zero-Watermark Image:- A binary image of dimensions 16 x 16 pixels.

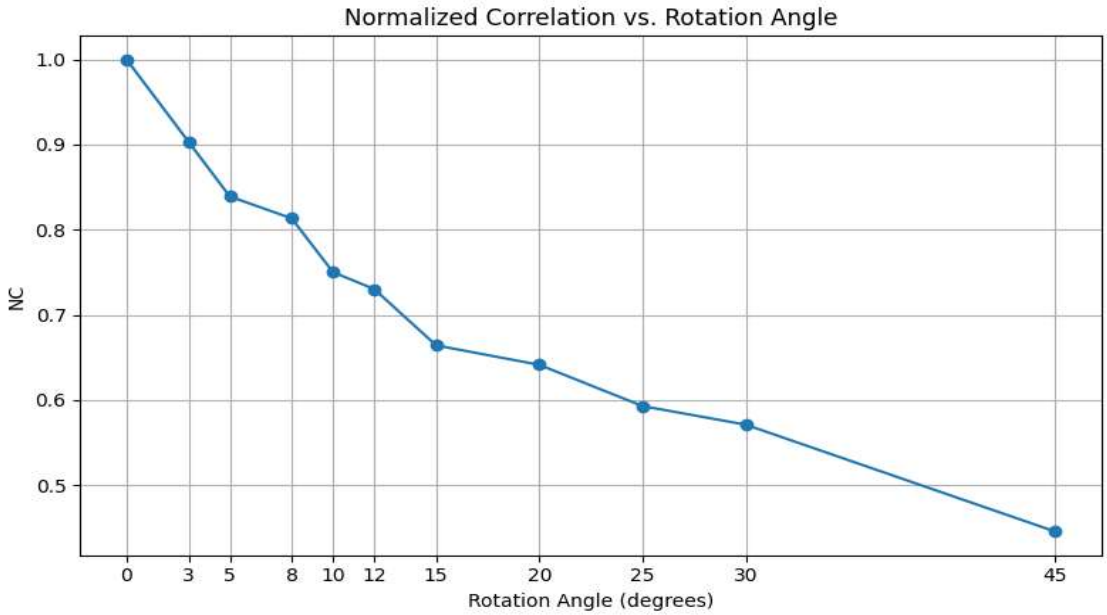


Performance Test:-

We tested the robustness of the Zero-Watermarking Algorithm by carrying out various attacks on the original image, and calculated the NC value of the original and recovered logo images for test results. The attacks performed and the results obtained are as follows:-

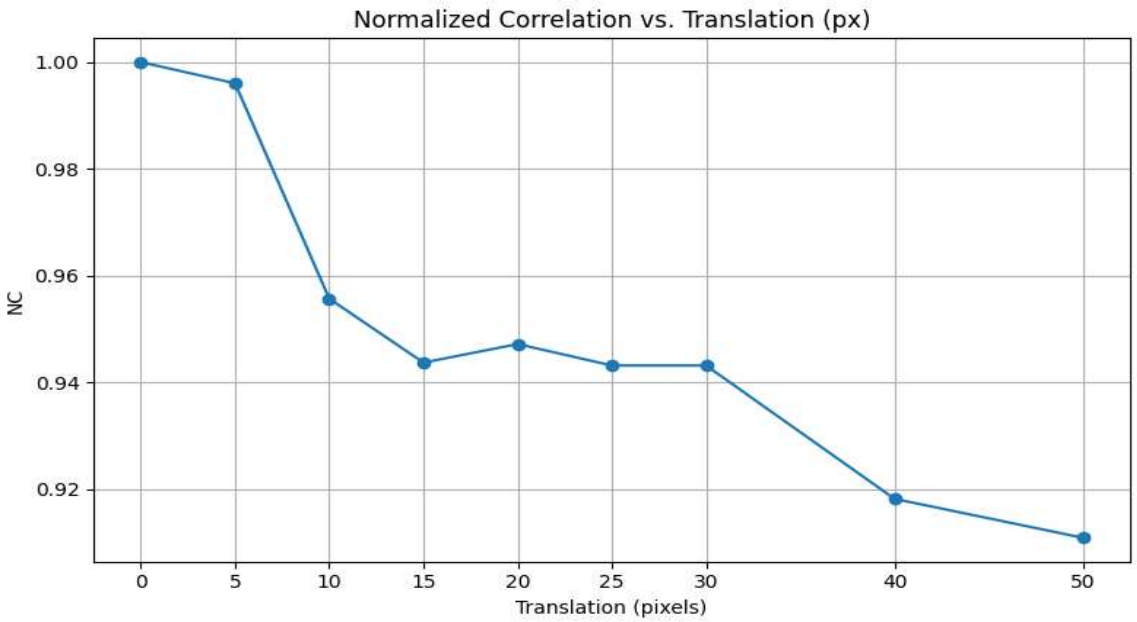
1. **Rotation Attack:-** We performed anticlockwise rotation attack with various angles. The test results show that the algorithm is robust against smaller angles. As the angle increases, the NC value decreases, thus the robustness decreases.

   For angles upto 8 degrees, the NC value is greater than 0.8, and upto 25 degrees, the NC value is greater than 0.6. For angles 45 degrees or higher, the NC value is less than 0.5.



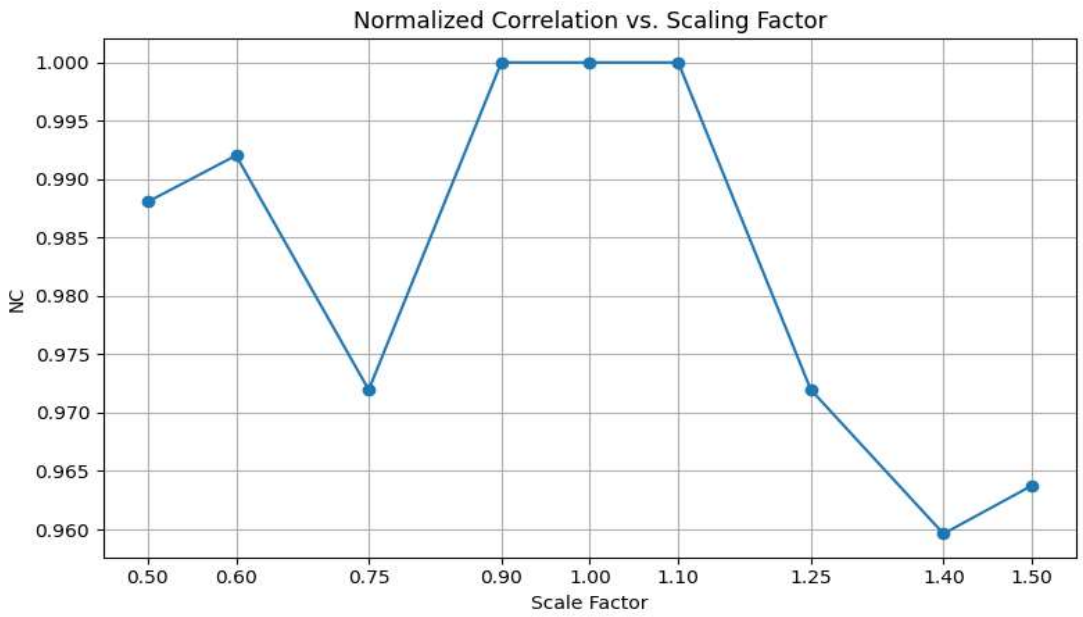
2. **<u>Translation Attack</u>** :- We performed lower-right translation attack, with various translation pixel values, keeping the translation along the x and y directions same.

With increase in translation pixels (from 0 to 50), the NC value decreases, but it is not seriously affected and is greater than 0.9.

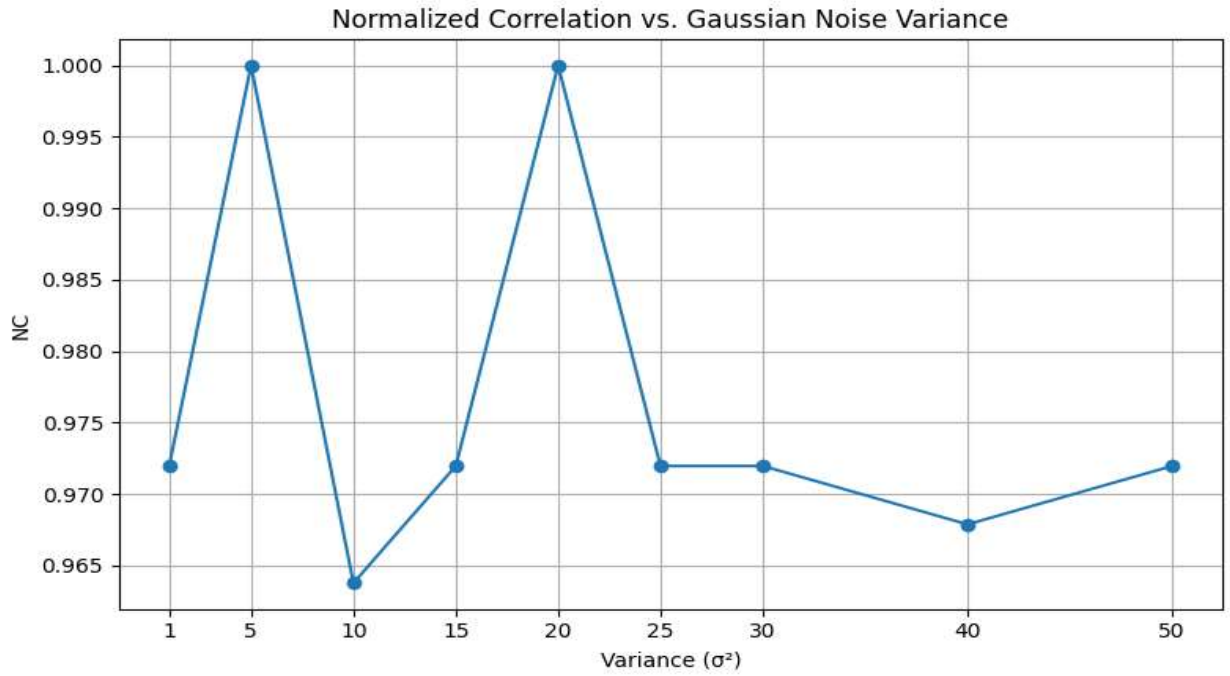


3. **<u>Scaling attack</u>** :- We performed scaling attack with scaling factors between 0.5 and 1.5 (same for both x and y directions).

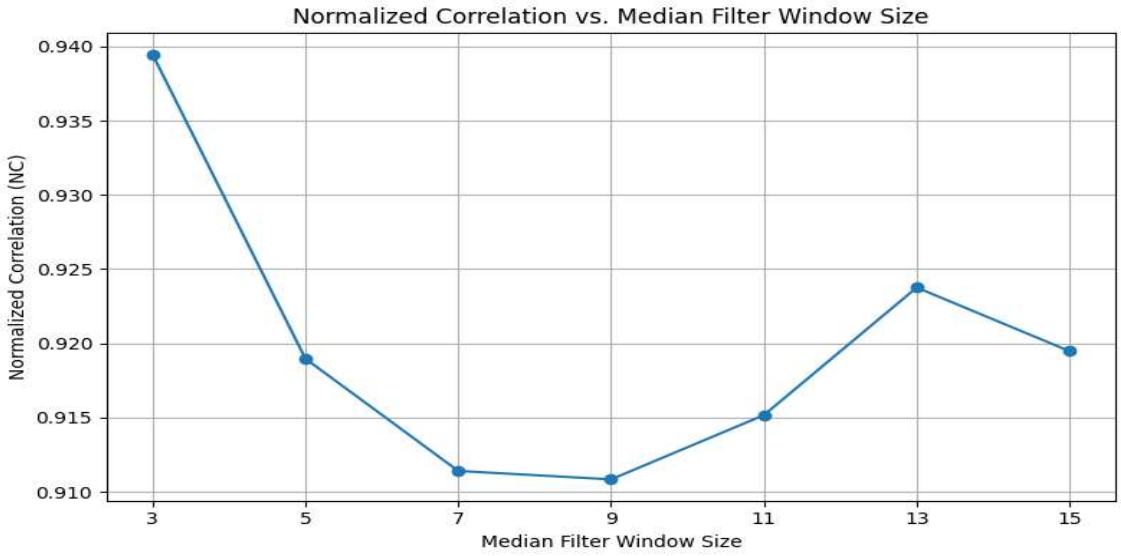
   For scaling factors near to 1, like 0.9 and 1.1, the NC value remains 1, and it decreases for scaling factors below 0.9 and above 1.1. But overall, the NC value is not affected much, and it remains greater than 0.96.



4. **Gaussian Noise Attack** :- We performed gaussian noise attack with mean = 0 and variance values from 1 to 50. The test results show that the image is only slightly affected by the gaussian noise, and for most of the variances, the NC values remain in between 0.96 and 0.975.



5. **Median Filtering Attack**:- We performed median filtering attack with window sizes from 3x3 to 15x15. Here also, with increase in the window size, the NC values decrease, but it remains greater than 0.91.



7.Conclusion

Images, as a primary information carrier, still rely on digital watermarking for copyright protection. However, the credibility of third parties in digital watermarking technology remains a significant challenge due to cost. The second generation of blockchain technology offers a feasible solution. Here a blockchain framework to address the limitations of traditional digital watermarking and solve the trusted third-party problem in image copyright protection.

The presented framework integrates a zero-watermarking algorithm, IPFS distributed storage, and the Ethereum blockchain. Within this framework, smart contracts replace third parties, ensuring reliability in processing and results. IPFS and smart contracts also facilitate keyword-based image management, addressing scalability and cost issues associated with traditional blockchain-based solutions. The zero-watermarking algorithm developed for this system demonstrates clear advantages in robustness and efficiency, as shown by experimental results.

Project Link

https://github.com/Kush27110405/Final_Year_Project

8.References

1. https://drive.google.com/file/d/13cyyhTW3t6SkKZaOEY5Gxzeljg6NQnvk/view?usp=sharing

2. https://research.cbs.dk/en/publications/a-digital-rights-management-system-based-on-a-scalable-blockchain

3.https://www.researchgate.net/publication/224171150_Digital_Image_Watermarking_Using_Discrete_Wavelet_Transform_and_Singular_Value_Decomposition_IEEE_Transactions_on_Instrumentation_and_Measurement_59_3060-3063

4.https://www.researchgate.net/publication/356141081_Robust_Zero_Watermarking_Algorithm_for_Medical_Images_Based_on_Zernike-DCT

5.https://www.researchgate.net/publication/384044771_EtherWatch_A_Framework_for_Detecting_Suspicious_Ethereum_Accounts_and_Their_Activities

6. https://www.researchgate.net/publication/331142719_Blockchain-Enabled_Smart_Contracts_Architecture_Applications_and_Future_Trends

7.https://www.researchgate.net/publication/369869912_Ethereum_Introduction_Expectation_and_Implementation

8. https://ieeexplore.ieee.org/document/1532313