Implementation of Blockchain Technology for Image Plagiarism Detection Using DCT, AES128, and SHA-1 Algorithms

Filbert Duran[1*], Leonardo[2], Shickhem[3], Yennimar[4]

[1,2,3,4]Informatics Engineering Department, Universitas Prima Indonesia, Medan, North Sumatera, Indonesia
E-mail: [1*]filbertduran.01@gmail.com, [2]kholenardo100@gmail.com, [3]inggatanpro@gmail.com, [4]yennimar@unprimdn.ac.id

(Received: 4 Feb 2025, revised: 15 Feb 2025, accepted: 17 Feb 2025)

Abstract

Plagiarism encompasses the act of appropriating high-quality user-generated content as if it were one's own intellectual property. Image plagiarism can be conceptualized as a broader category that encompasses the challenges of detecting copied images. Identifying instances of plagiarism is of paramount importance not only for graphic designers, professional photographers, and bloggers but also for publishing entities and legal practitioners seeking to uncover unauthorized reproductions of their creations. In addressing this issue, the implementation of blockchain technology presents a viable solution. Fundamentally more than just a collection of interconnected blocks, blockchain is characterized by the systematic recording of digital signatures or hashes of each block. Blockchain is essentially more than just a collection of interconnected blocks; it is characterized by the systematic recording of a digital signature or hash of each block. To generate the hash, cryptographic methods can be applied. This study aims to develop a web-based application that is adept at detecting image plagiarism through the application of blockchain technology. Images submitted by users will undergo plagiarism detection by an application that uses blockchain methodology. This study applies the DCT method to extract features from images, then uses the AES-128 and SHA-1 methods to generate blockchain. The results of this study are in the form of a website that can be used to detect image plagiarism. From the results of the tests carried out, it was obtained that the combination of the DCT, AES-128 and SHA-1 methods can detect image plagiarism with an accuracy of 100%. This means that the combination of these methods can be applied to carry out the process of detecting image plagiarism with a very high level of accuracy.

Keywords: Blockchain, Plagiarism, Digital Image, DCT, AES-128, SHA-1.

I. INTRODUCTION

Nowadays, nearly all mobile phones are equipped with high-performance cameras capable of producing high-quality images. These captured images can be readily disseminated via social media applications [1]. Every user of the social media application can access this image. With the growing sophistication of digital image manipulation applications, individuals skilled in using these tools can easily modify images obtained from such social media platforms [2]. If an individual redistributes a modified image via social media platforms, this constitutes image plagiarism, as the original image is not their intellectual property [3]. Plagiarism in this case refers to the unethical practice of claiming ownership over high-quality user-generated content [4]. From a technical standpoint, a derivative image file would maintain perceptual similarity to the original while incorporating alterations to its physical attributes. These modifications would be designed to create the perception that the image is not a copy of a plagiarized source [5]. These physical alterations may involve logo embedding, color space conversion, cropping, and others. Plagiarized images can be considered a superset of duplicate image detection problems [6]. Plagiarism detection is not only beneficial for graphic designers, professional photographers, and bloggers but also for publishing institutions and legal professionals seeking to detect unauthorized reproductions of their work [7].To address this issue, the concept of image authentication can be implemented.

Several published signature-based approaches in image authentication include signature-based authentication methods with localization of damaged areas using wavelet transform [8]. However, this method relies heavily on edge detection techniques and is limited to detecting only significant object manipulations. Other research has introduced a hashing technique that incorporates both local and global features. Global features are based on Zernike moments, while local features capture the position and texture information of salient regions in the image [9]. The primary issue with previously

published techniques is the excessively large size of the image signature. Consequently, the signature must be transmitted as a separate file alongside the image itself [10]. This blockchain approach mitigates these vulnerabilities by utilizing a blockchain transaction hash derived from the image signature as the actual payload, and appending an encrypted hash to the JPEG file [11].

In the article entitled "Image Plagiarism Detection System using CBIR (Content-based Image Retrieval)" [12] explains that the detection system begins with the basics consisting of several stages, namely through image processing that functions to solve identification problems such as in the world of forensic medicine and reading weather maps from satellites and the image retrieval process using DBMS (Data Base Management System) where processing text-based images is then displayed again in the form of images. The use of CBIR begins with the stage of extracting image features to then be stored efficiently and the level of similarity is adjusted to the features used. Other research is the Flowmap Image Plagiarism Detection System through the Image Processing Approach. The research was conducted through an image processing approach to solve and find in terms of image detection in the form of a flowmap [13]. The detection system is carried out based on flow diagrams. This detection system is still rarely used and studied. The detection system is carried out by making comparisons in the form of shape, text and orientation. The approach used uses graphs from path diagrams so that it is able to detect objects with the same shape even though the orientation of the graph changes. Another study is the Image Plagiarism Detection System through Perceptual Hash with image-based mitigation of rotational constraints [14] as an alternative to detect image-based plagiarism. The perceptual hash method uses a simplified distance function that produces a perceptual value that can be compared whether it is different from one another. The detection system is also not sensitive to image rotation, when the image is rotated, the system still detects it. The Perceptual Hash is an algorithm that can function to detect the level of similarity between images.

This research aims to develop a web-based application capable of accurately detecting image plagiarism through the implementation of blockchain technology [15]. This application will evaluate user-uploaded images for potential plagiarism, [16] leveraging blockchain mechanisms to detect plagiarism. [17] Based on the above description, this research implements JPEG image authentication using blockchain to detect modified images [18].

II. RESEARCH METHODOLOGY

This research is a quantitative study employing an experimental method. Data was collected from various online sources, subsequently processed, and analyzed to draw conclusions.

A. Blockchain

A blockchain is a continuously growing list of records, called blocks [19], which are linked and secured using cryptography [20]. Each block typically contains a timestamp, a hash of the previous block, and data. For example, a block may contain information about a transaction [21]. The operation of a blockchain is illustrated in Figure 1 [22].

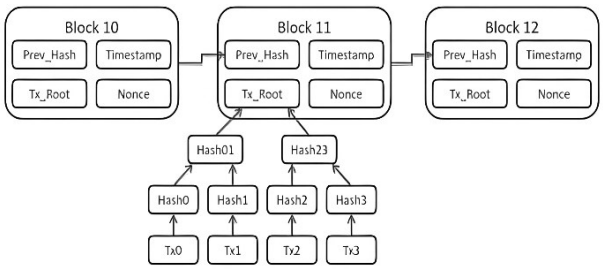

Figure 1. How Blockchain Works

This process is ceaseless. Fundamentally, a blockchain is a consortium of inherently skeptical contributors collaboratively maintaining a database without the reliance on a trusted intermediary. To mitigate chaos within this decentralized framework and establish universal consensus, it is imperative for every blockchain network to enforce a comprehensive set of rules governing all database transactions. These rules are encoded within each blockchain client, which subsequently utilizes them to verify transaction validity and, consequently, determine whether a transaction will be propagated throughout the network [23].

Blockchain should be understood not merely as a chain of interconnected blocks, but rather as a system that records the digital signatures (or hashes) of preceding blocks, as illustrated in Figure 2.

Figure 2. Illustration of Blockchain

As per the analysis presented in Figure 2, [24] it can be inferred that a block within a blockchain comprises three essential elements: the data itself, the hash of the preceding block, and a timestamp denoting the creation of the current block [25].

B. DCT Method

The discrete cosine transform algorithm is outlined as follows:

1. The image is segmented into blocks, each consisting of 8x8 pixels.
2. A value of 128 is subtracted from the original matrix data, as the DCT algorithm operates within a range of -128 to 127, adhering to the principles of digital image processing.
3. Utilizing the Discrete Cosine Transform equation, a matrix D is determined, which will subsequently be used for further quantization (Formula 1).

DOI: 10.34148/teknika.v14i1.1205

$$D = T \cdot M \cdot T^{t} \quad (1)$$

Where :

- Matrix T =

$$T = \begin{pmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1919 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & -0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ -0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & -0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.778 & -0.0975 \end{pmatrix}$$

- Transpose of Matrix T (T^t) =

$$T^t = \begin{pmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1919 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & -0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ -0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & -0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.778 & -0.0975 \end{pmatrix}$$

- Matrix M = a matrix contaning the pixel values of the input image

4. The matrix D now contains the DCT coefficients, wherein data positioned at the top-left corner corresponds to low frequencies of the original dataset, while data at the bottomright corner aligns with high frequencies of the original dataset. Subsequently, a quantization process must be executed at Quality Level 50.

$$Q_{50} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

The quantization matrix formulation is expressed as follows, where rounding denotes the process of adjusting the result of a division to the nearest integer (Formula 2).

$$C_{ij} = round\,\frac{D_{ij}}{Q_{ij}} \quad (2)$$

5. The numerical values are sequentially arranged using a zigzag scanning pattern, representing the final step in the compression procedure (Figure 3).

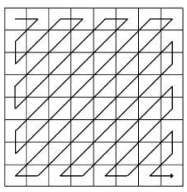

Figure 3. Zigzag Scan Order

C. AES-128 Method

The working procedure of the AES-128 method consists of 3 stages, namely the key formation process, encryption process and decryption process. The working process of the AES-128 method can be seen in the following Figure 4:

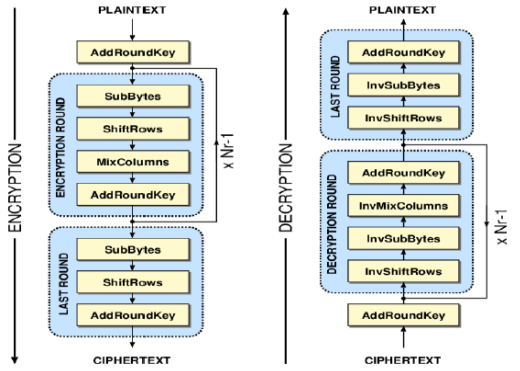

Figure 4. Encryption and Decryption Process in AES-128

The steps taken in key expansion are:

1. Divide the key into blocks where each block consists of 1 word (32 bits). For a 128 bit key size there are 4 words (w[0], ..., w[3]), for a 192 bit key size there are 6 words (w[0], ..., w[5]) and for a 256 bit key size there are 8 words (w[0], ..., w[7]).
2. For the value of i with the range $Nk \leq i < Nb(Nr+1)$, find w[i] with the following conditions:
   - if (i mod **Nk)** $\neq 0$ and (i mod **Nk)** $\neq 4$ then w[*i*]=w[*i*-**Nk**] $\oplus$ w[*i*-1].
   - if (i mod **Nk)** = 0 then w[i] = w[i-Nk] $\oplus$ (SubWord (RotWord(w[i-1])) $\oplus$ Rcon[i/Nk])
3. w[*i*] = w[*i*-**Nk**] $\oplus$ (*SubWord* (*RotWord*(w[*i*-1])) $\oplus$ Rcon[*i*/**Nk**])
   - if (*i* mod **Nk)** = 4 then w[i] = w[i-Nk] $\oplus$ Subword(w[i-1])

Subkey (round key) is obtained with the following conditions:

- Subkey(*i*) = (w[*i*∗4], w[*i*∗4+1], w[*i*∗4+2], w[*i*∗4+3]) ; for $0 \leq i \leq$ **Nr**

In the encryption process, the AES algorithm uses four different transformations, namely:

1. SubBytes()

The SubBytes() transformation is a byte substitution that operates on each byte in the State using a substitution table (S-box) as shown in Table 1.

Table 1. S-Box

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

For example, if s1,1 = {53} then the substitution value is obtained from the intersection of row "5" with column "3" in the S-box so that the result is {ed}.

2. ShiftRows()
   The ShiftRows() transformation is performed on the last 3 rows by rotating the shift with different shift values depending on the row. The first row (r = 0) does not perform the ShiftRows() operation. For example, ShiftRows 1 byte for input hexadecimal value '09 83 00 C7' is '83 00 C7 09'
3. MixColumns()
   The MixColumns() transformation operates on the State column by column, each column being multiplied by a specified matrix.
4. AddRoundKey()
   In the AddRoundKey() transformation, a subkey is added to the State by XOR operation. Each subkey consists of Nb words from the subkey set. Subkeys are added to the State

At the beginning of encryption, the input in the form of plaintext is entered into the State, in the initial round the AddRoundKey(State, SubKey(0)) transformation is performed, after the initial round the process goes to the round function as many as Nr-1 rounds ($1 \leq round < Nr$), where in this round function the transformations are carried out in sequence, namely SubBytes*(), ShiftRows(), MixColumns(), and AddRoundKey(). After that the process will go to the last round (final round) where in this last round the transformations of SubBytes(), ShiftRows() and AddRoundKey() are performed, in this last round after the AddRoundKey() transformation it will produce the final State which is the output called ciphertext.

The decryption process in the AES algorithm is the reverse of the encryption process. The transformations used in the decryption process are:

1. InvShiftRows()
   This transformation is the opposite of the ShiftRows() transformation in the encryption process. The InvShiftRows() transformation is performed on the last 3 rows by rotating the shift with different shift values depending on the row. The first row (r = 0) does not perform the ShiftRows() operation. For example, InvShiftRows 1 byte for input hexadecimal value '83 00 C7 09' is '09 83 00 C7'.
2. InvSubBytes()

Table 2. Inverse S-box

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

InvSubBytes() is the inverse of the SubBytes() transformation. The InvSubBytes() transformation uses S-boxes as shown in Table 2.

3. InvMixColumns()
   The InvMixColumns() transformation is the opposite of the MixColumns() transformation, the InvMixColumns transformation operates on the State column by column, each column is multiplied by a predetermined matrix.
4. AddRoundKey()
   The AddRoundKey() transformation in the encryption and decryption processes is the same because it only involves the XOR operation.

The process of decrypting ciphertext into plaintext is done in a way that is not much different from the encryption process, the difference is only in the sequence of transformations used. In the decryption process, the input in the form of ciphertext is entered into the State, in the initial round the AddRoundKey(State, Subkey(Nr)) transformation is carried out, after the initial round the process goes to the round function as many as Nr-1 rounds ($1 \leq round < Nr$), where in this round function the transformations are carried out in sequence, namely InvShiftRows(), InvSubBytes(), AddRoundKey(), and InvMixColumns(). After that the process will go to the last round (final round) where in this last round the transformations of InvShiftRows(), InvSubBytes(), and AddRoundKey() are carried out, in this last round after the AddRoundKey() transformation it will produce the final State which is the output called plaintext.

D. SHA-1 Algorithm

The SHA-1 algorithm can be summarized as follows:

a. The calculation uses two buffers where each buffer consists of five 32-bit words and a sequence of 80 also 32-bit words. The first five words in the word buffer are named A, B, C, D, E while the second five words are named H_0, H_1, H_2, H_3, and H_4. Then the 80 consecutive words are named W_0, W_1, …, W_{79} and in this calculation also uses a temporary variable, TEMP.
b. Perform message filling, M and then parse the message into N 512 bit message blocks, $M^{(1)}$, $M^{(2)}$, …, $M^{(n)}$. The method: The first 32 bits of the message block are shown to $M_0^{(i)}$, then the next 32 bits are $M_1^{(i)}$ and so on until $M_{15}^{(i)}$.
c. Initialize Hash Value (in hex form):
   H_0 = 67452301 H_3 = 10325476
   H_1 = EFCDAB89 H_4 = C3D2E1F0
   H_2 = 98BADCFE
d. Perform the process M_1, M_2, …, M_n by dividing Mi into 16 words W_0, W_1, ….W_{15} where W_0 is the left most.
e. Compute : For t = 16 to 79
   $W_t = S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$
f. Initialize 5 variables A, B, C, D, and E with Hash values:
   $A = H_0$; $B = H_1$; $C = H_2$; $D = H_3$; $E = H_4$.
g. Compute : For t = 0 to 79
   $TEMP = S^5(A) + f_t(B,C,D) + E + Wt + Kt$
   $E = D$; $D = C$; $C = S^{30}(B)$; $B = A$; $A = TEMP$.
h. Compute hash value:
   $H_0 = H_0 + A$; $H_1 = H_1 + B$;
   $H_2 = H_2 + C$; $H_3 = H_3 + D$; $H_4 = H_4 + E$.
   The result of the 160-bit message digest of the message, M is: H_0 H_1 H_2 H_3 H_4.

DOI: 10.34148/teknika.v14i1.1205

III. RESULT AND DISCUSSION

A. Result

1. Blockchain Creation Process

A flowchart, as presented in Figure 5, provides a visual representation of the operational workflow involved in implementing blockchain technology for image plagiarism detection.

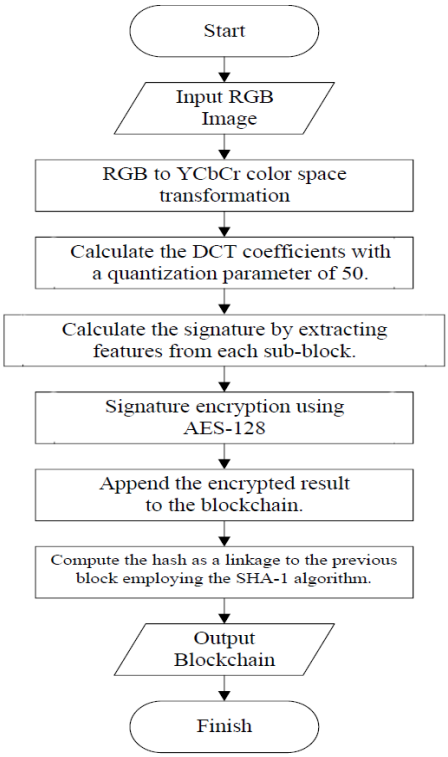

Figure 5. Flowchart of Blockchain Generation Process

Based on the flowchart in Figure 5, the work steps of the blockchain creation process are as follows:

- Input RGB image.
- Convert RGB image to YCbCr color space.
- The process of converting image pixel colors from RGB color space to YCbCr color space. Since the value used in the next calculation is only the Y element value, only the Y element value will be calculated.
- Calculate the DCT coefficient with a value of q = 50.

For example, a 16 x 16 pixel RGB color image is selected. The process of converting image pixel colors from RGB color space in table 3 to YCbCr color space. For example, the value of pixel (0, 0) is (R,G,B) = (105,177,65). Since the value used in the next calculation is only the Y element value, only the Y element value will be calculated, as seen in the following calculation details:

Y = 0.2990 * 105 + 0.5870 * 177 + 0.1140 * 65

Y = 48

The calculation process will be carried out for each pixel.

Then, the process is continued by calculating the DCT coefficient with a value of q = 50. The operational process of calculating DCT coefficients can be illustrated in a flowchart as shown in Figure 6.

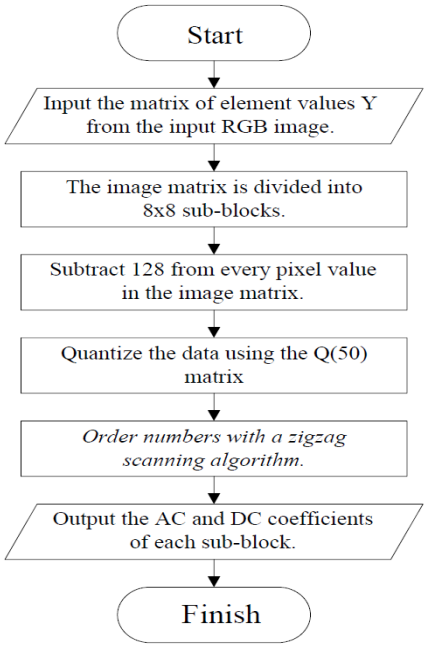

Figure 5. Flowchart of the Discrete Cosine Transform Coefficient Calculation Process

The input matrix of elemen value Y from the input RGB image will be divided into 8x8 sub-blocks. After that, the value 128 reduces the image matrix data, , for example if the first pixel value of first subblock is 48, then output value is 128 – 48 = 80. Then, the T matrix is multiplied with the image matrix data, the result of the multiplication with the T matrix is then multiplied again with the T^t matrix. Then continued with the quantization process with the Q50 matrix, resulting in the following Table 3:

Table 3. Output Quantization Process

Block 1

21	-2	-4	-4	0	1	0	0
12	8	6	5	-2	1	0	0
-6	-2	-3	-2	0	0	0	0
3	-1	1	2	-1	0	0	0
8	1	-1	0	0	0	0	0
-5	0	-1	0	1	0	0	-2
1	1	0	0	-1	0	0	0
0	0	0	0	0	0	0	0

Finally, the numbers are read using zig-zag scanning on each block, resulting in the following sequence of blocks:

Block 1 : 21, -2, 12, -6, 8, -4, -4, 6, -2, 3, 8, -1, -3, 5, 0, 1, -2, -2, 1, 1, -5, 1, 0, -1, 2, 0, 1, 0, 0, 0, 0, -1, 0, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2, 0, 0 ,0

The process above will be done for all sub-block.

- Calculate the signature by extracting features from each subblock. The process of signature calculation with feature extraction can be visualized in a flowchart as shown in Figure 7.

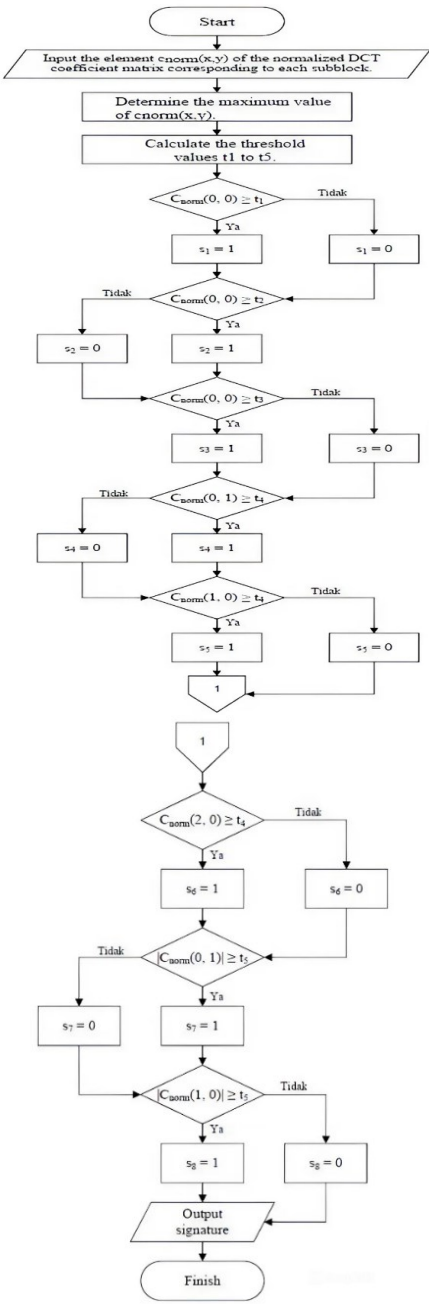

Figure 6. Signature Generation Flowchart with Feature Extraction

The working process of the signature calculation by extracting the characteristics of each subblock can be seen in the following calculation details:

1) Determine $c_{norm}(0,0)_{max}$ value.
   Subblock 1: $c_{norm}(0,0) = 21$
   Subblock 2: $c_{norm}(0,0) = -10$
   Subblock 3: $c_{norm}(0,0) = 26$
   Subblock 4: $c_{norm}(0,0) = 2$
   $c_{norm}(0,0)_{max} = 26$
2) Compute *threshold* value $t_1 \ldots t_5$.
   $t_1 = C_{norm}(0,0)_{max}/4 = 26/4 = 6.5$
   $t_2 = C_{norm}(0,0)_{max}/2 = 26/2 = 13$
   $t_3 = 3 * C_{norm}(0,0)_{max}/4 = 3 * 26/4 = 19.5$
   $t_4 = 0$
   $t_5 = 3$
3) Compute *signature* value for every subblock:
   Subblock 1:
   $c_{norm}(0,0) = 21$
   $c_{norm}(0,0) \geq t_1 \rightarrow 21 \geq 6.5 \rightarrow s_1 = 1$
   $c_{norm}(0,0) \geq t_2 \rightarrow 21 \geq 13 \rightarrow s_2 = 1$
   $c_{norm}(0,0) \geq t_3 \rightarrow 21 \geq 19.5 \rightarrow s_3 = 1$
   $c_{norm}(0,1) \geq t_4 \rightarrow -2 \geq 0 \rightarrow s_4 = 0$
   $c_{norm}(1,0) \geq t_4 \rightarrow 12 \geq 0 \rightarrow s_5 = 1$
   $c_{norm}(2,0) \geq t_4 \rightarrow -6 \geq 0 \rightarrow s_6 = 0$
   $|c_{norm}(0,1)| \geq t_5 \rightarrow 2 \geq 3 \rightarrow s_7 = 0$
   $|c_{norm}(1,0)| \geq t_5 \rightarrow 12 \geq 3 \rightarrow s_8 = 1$
   Signature = 11101001
   The process will be done for all sub block.
4) *Output signature* = $(11101001\ 00010101\ 11111101\ 00011011)_2 = (E9\ 15\ FD\ 1B)_{16}$

- Signature encryption using the AES-128 method.

To perform the encryption process with the AES-128 method, a key formation process is required first. The key formation process in the AES-128 method can be described in flowchart as shown in Figure 8.

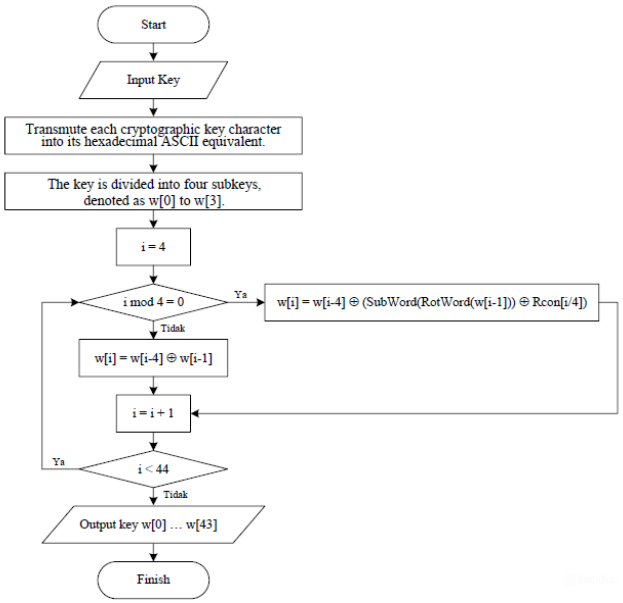

Figure 7. Key Generation Process Using AES 128-bit Method

The working steps of the key formation process using the AES method (all numbers in the calculation are in hexadecimal format) are as follows:

1. Key input
   Key = TUGASAKHIRAES128
2. Convert each key character to Hexadecimal ASCII Code. Key after being converted to hexadecimal notation (symbolized by 'X')
   X = 5455474153414B484952414553313238

3. The key is divided into 4 subkeys w[0] … w[3]
   w[0] = 54554741 w[1] = 53414B48
   w[2] = 49524145 w[3] = 53313238
4. i = 4.
5. i mod 4 = 0, then:
   w[4] = w[0] XOR (SubWord(RotWord(w[3])) XOR Rcon[1])
   w[4] = 54554741 XOR (SubWord(RotWord(53313238)) XOR 01000000)
   RotWord = Do left rotation 1 byte.
   RotWord(53313238) = 31323853
   SubWord = subtitution using S-Box
   SubWord(31323853) = C72307ED
   w[4] = 54554741 XOR (C72307ED XOR 01000000)
   w[4] = 927640AC
6. i = i + 1 = 4 + 1 = 5
   i mod 4 != 0, maka:
   w[5] = w[1] w[4] = 53414B48 XOR 927640AC
   w[5] = C1370BE4
7. The above process will be carried out until W[43].
8. Key output:
   Subkey[0] = w[0] w[1] w[2] w[3]
   = 5455474153414B484952414553313238
   …
   Subkey[10] = w[40] w[41] w[42] w[43]
   = 6AC27574D8492A9E7D9EA864A971A469

The signature encryption process with the AES-128 method can be described in the form of a flowchart as shown in Figure 9.

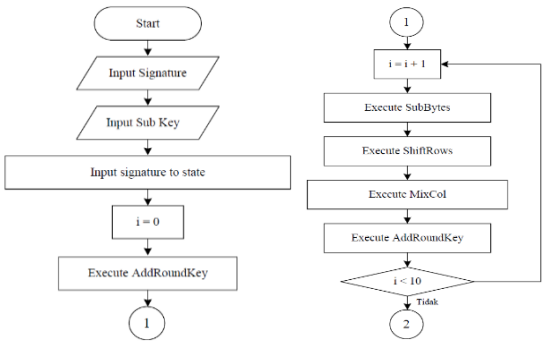

Figure 8. The 128-bit Encryption Process

The working steps of the signature encryption process using the AES-128 method are as follows:

1. Input signature = $(E9\ 15\ FD\ 1B)_{16}$.
2. Input subkey, as shown in step (8) above.
3. Insert signature into state.
   E9 15 FD 1B (hexadecimal)
   State
   E9 00 00 00
   15 00 00 00
   FD 00 00 00
   1B 00 00 00
4. i = 0.
5. Perform the AddRoundkey process on State with Subkey[0].
6. i = i + 1 = 0 + 1 = 1.
7. Perform the SubBytes process
8. Perform the ShiftRows process
9. Perform the MixCol process
10. Perform the AddRoundKey process
11. Repeat step (6) to step (10) until i = 9. AddRoundKey process for subkey[9] are shown as below:
12. i = i + 1 = 9 + 1 = 10.
13. Perform SubBytes process.
14. Perform the ShiftRows process.
15. Perform the AddRoundKey process
16. *Ciphertext* = **FF65247E8EA110530B57C597C4279DF0**

- Paste the encryption result into the blockchain.
  Blockchain:
  Transaction ID = 1
  Ciphertext = FF65247E8EA110530B57C597C4279DF0
  Timestamp = 26 November 2024 17:38
- Calculate the hash value as the block relationship with the SHA-1 function.
  Input string = 1,
  FF65247E8EA110530B57C597C4279DF0, 26 November 2024 17:38
  SHA-1 hash value =
  6E04E269B6AC7FA525A7BA0BD6D4D04E9EAA513A
- Output the blockchain.

2. Image Authentication Process for Plagiarism Detection

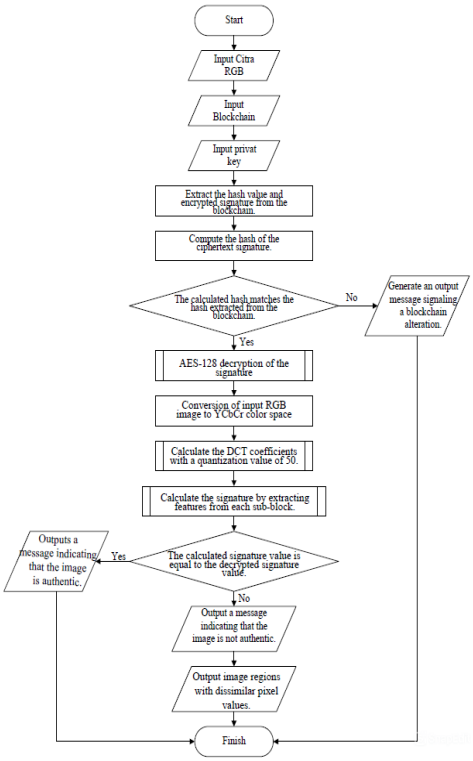

Figure 9. Process Flow Diagram for Image Authentication

Suppose the RGB image is replaced by pixel (0,0) = {105,177,65} and pixel (0,1) = {94,167,55} to pixel (0,0) = {88,100,60} and pixel (0,1) = {60,65,20}.

1. Input blockchain, as shown in Figure 10.
2. Input the private key, namely TUGASAKHIRAES128.
3. Extract the hash value and encrypted signature from the blockchain.
4. Hash value = 6E04E269B6AC7FA525A7BA0BD6D4D04E9EAA513A
5. Encrypted signature value = FF65247E8EA110530B57C597C4279DF0
6. Calculate the hash value of the encrypted signature.
7. SHA-1 hash value = 6E04E269B6AC7FA525A7BA0BD6D4D04E9EAA513A
8. Because the SHA-1 hash value in step (7) is the same as step (4), this means that there has been no data replacement on the blockchain so the process continues.
9. Decrypt the signature using the AES 128 method.
   The process of decrypting the signature using the AES 128 method can be visualized in a flowchart as depicted in Figure 11.

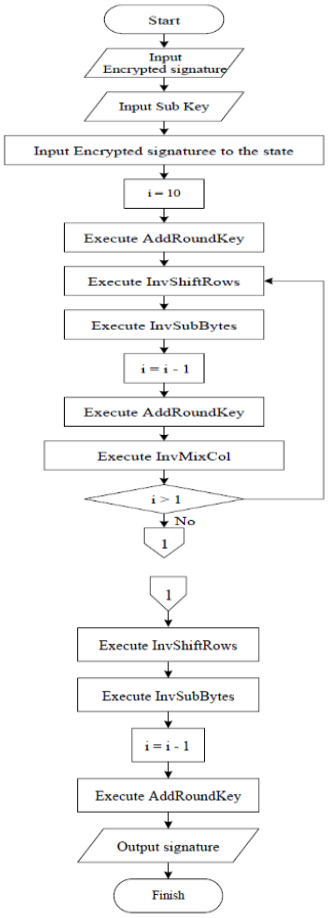

Figure 10. The Process of Decrypting a Signature Using The AES 128 Algorithm

The working steps of the encrypted signature decryption process using the AES-128 method are as follows:

1. Encrypted signature input = $(FF65247E8EA110530B57C597C4279DF0)_{16}$.
2. Input the sub key, as seen in step of the key formation process above.
3. Insert the encrypted signature into the state.
4. i = 10.
5. Do AddRoundKey.
6. Do InvShiftRows.
7. Do InvSubBytes.
8. i = i – 1 = 10 – 1 = 9.
9. Do AddRoundKey
10. Do InvMixCol.
11. Do InvShiftRows.
12. Do InvSubBytes.
13. Repeat step (8) to (12) until i = 1.
14. i = i – 1 = 1 – 1 = 0.
15. Do AddRoundKey.
    Signature = E9 15 FD 1B

After that, the process is continued by converting input RGB image to YCbCr color space. The calculation process will be performed for each pixel. Then, the Y value elements of the 16 x 16 input image will be divided into 4 subblocks measuring 8 x 8. After that, the value 128 reduces the image matrix data, for example if the first pixel value of first subblock is 92, then output value is 128 – 92 = 36. Then, the T matrix is multiplied with the image matrix data, the result of the multiplication with the T matrix is then multiplied again with the T^t matrix. Then continued with the quantization process with the Q50 matrix, resulting in the following Table 4:

Table 4. Output Quantization Process Block 1

16	1	-4	-4	0	1	0	0
12	8	6	5	-2	1	0	0
-6	-2	-3	-2	0	0	0	0
3	-1	1	2	-1	0	0	0
8	1	-1	0	0	0	0	0
-5	0	-1	0	1	0	0	-2
1	1	0	0	-1	0	0	0
0	0	0	0	0	0	0	0

Finally, the numbers are read using a zig-zag scanning method on each block, resulting in the following block sequence:

Block 1 : 16, 1, 12, -6, 8, -4, -4, 6, -2, 3, 8, -1, -3, 5, 0, 1, -2, -2, 1, 1, -5, 1, 0, -1, 2, 0, 1, 0, 0, 0, 0, -1, 0, -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2, 0, 0 ,0

Then, the process is continued by computing the signature by extracting features from each subblock. The working process of the signature calculation by extracting the characteristics of each subblock can be seen in the following calculation details:

1. Determine $c_{norm}(0,0)_{max}$ value.
2. Compute threshold value $t_1 \ldots t_5$.
3. Compute signature value for every subblock.
4. Output signature = $(11011001\ 00010101\ 11111101\ 00011011)_2 = (D9\ 15\ FD\ 1B)_{16}$

DOI: 10.34148/teknika.v14i1.1205

Because the signature value in step (4) is not the same as the signature in AES-128 decryption process, the output message states that the image is NOT authentic.

Original signature (from blockchain decryption) = **E9** 15 FD 1B. Whereas, signature of the input image feature extraction results = **D9** 15 FD 1B. From the comparison results, it is known that the difference lies in subblock 1, meaning that plagiarism occurred in subblock 1.

B. Discussion

The testing will be conducted using a variety of images. The original images that will be inputted in Table 5.

Table 5. Data Training

Data Training	Image
1	
2	
3	
4	
5	

For evaluation, the testing images were compared to original image (data training image) by using mean squared error (MSE) method. If the MSE value = 0, it means that the testing image was original image. Whereas, the testing image was plagiarism. A comprehensive tabulation of the experimental findings is provided in Table 6.

Table 6. Experimental Results

No.	Testing Image	System Result		Actual		Details
		Plagiarism Percentage	System Results	MSE Value	Actual Data	
1		0	Original	0	Original	SUCCEED
2		88.14%	Plagiarism	13.935	Plagiarism	SUCCEED
3		0	Original	0	Original	SUCCEED
4		94.11%	Plagiarism	6.983	Plagiarism	SUCCEED
5		0	Original	0	Original	SUCCEED
6		98.05%	Plagiarism	2.048	Plagiarism	SUCCEED
7		0	Original	0	Original	SUCCEED
8		93.24%	Plagiarism	7.762	Plagiarism	SUCCEED
9		0	Original	0	Original	SUCCEED
10		89.44%	Plagiarism	11.618	Plagiarism	SUCCEED

The testing process will be carried out using the confusion matrix method, which could be detailed in Table 7:

Table 7. Data Analysis Results

		Actual Data	
		Original	**Plagiarism**
System Result	**Original**	TP = 5	FP = 0
	Plagiarism	FN = 0	TN = 5

Based on the test that show in Tabel 7, the following information is obtained:

- Accuracy: (TP+TN)/(TP+FP+FN+TN)
  = (5+5)/(5+0+0+5) = 10/10 * 100 % = 100 %.
- Error: (FP+FN)/ (TP+FP+FN+TN)
  = (0+0) /(5+0+0+5) = 0/10 * 100 % = 0 %.

IV. CONCLUSION

A comprehensive evaluation has demonstrated the promising potential of blockchain technology in image plagiarism detection systems. By integrating unique hash values of each image into the blockchain, any alterations or manipulations to the image can be accurately and transparently tracked. The inherent consensus mechanism of blockchain ensures data integrity and prevents forgery.

According to the experimental result which has been done using the software developed, It can be seen that the combination of DCT, AES-128 and SHA-1 methods into blockchain techniques is able to detect plagiarism with an accuracy rate of 100%. The developed system is also able to detect a large percentage of plagiarism from input images.

Nevertheless, this research has also identified several technical challenges that require further attention. One such challenge is related to system performance, particularly in terms of processing speed for large-scale images. Additionally, the security of private keys used to generate hash values is a significant concern.

To address these challenges, several recommendations can be made. Firstly, optimization of hashing algorithms and more efficient blockchain implementations are necessary. Secondly, the utilization of distributed cloud computing can accelerate the verification process. Thirdly, the adoption of multi-signature mechanisms can enhance the security of private key storage.

Overall, this research has successfully demonstrated the significant potential of blockchain technology in addressing

image plagiarism. However, further research is required to refine the system and overcome the remaining challenges.

REFERENCES

[1] S. Anwar, A. Ullah, Á. Rocha, and M. J. Sousa, *Lecture Notes in Networks and Systems 614 Proceedings of International Conference on Information Technology and Applications ICITA 2022*. 2022.

[2] M. F. Sidiq, A. I. Basuki, D. Rosiyadi, I. Setiawan, Y. H. Siregar, and S. Sriyadi, "Secure protection for covid-19 infographic using blockchain and discrete cosine transform-singular value decomposition (DCT-SVD) watermarking," *J. Infotel*, vol. 14, no. 2, pp. 93–100, 2022, doi: 10.20895/infotel.v14i2.749.

[3] R. Uki Indriani and M. Hardjianto, "3 rd Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI) 30 Agustus 2023-Jakarta," 2023.

[4] P. T. A. Muzizat, "Algoritma AES128-CBC (Advanced Encryption Standard) Untuk Enkripsi Dan Dekripsi Berkas Dokumen," vol. 7, no. 1, pp. 166–176, 2025.

[5] M. G. Royhan and D. K. Ngabekti, "Problematika Desain Komunikasi Visual dan Plagiarisme dalam Dunia Desain Grafis," *CITRAWIRA J. Advert. Vis. Commun.*, vol. 2, no. 1, pp. 86–95, Jun. 2021, doi: 10.33153/citrawira.v2i1.3671.

[6] R. Maulana Rachman, A. Sobandi, and A. Wahyudin, "Penggunaan Aplikasi Pendeteksi Plagiarisme Image Sebagai Fasilitas Pendukung Otomatisasi Perkantoran," 2022. [Online]. Available: http://ejournal.upi.edu/index.php/manajerial/

[7] A. Chaudhuri, M. S. Bhatia, Y. Kayikci, K. J. Fernandes, and S. Fosso-Wamba, "Improving social sustainability and reducing supply chain risks through blockchain implementation: role of outcome and behavioural mechanisms," *Ann. Oper. Res.*, vol. 327, no. 1, pp. 401–433, 2023, doi: 10.1007/s10479-021-04307-6.

[8] I. S. Farahat, W. Aladrousy, M. Elhoseny, S. Elmougy, and A. E. Tolba, "Secure Medical Blockchain Model," *Inf.*, vol. 14, no. 2, pp. 1–15, 2023, doi: 10.3390/info14020080.

[9] A. R. Kairaldeen, N. F. Abdullah, A. Abu-Samah, and R. Nordin, "Peer-to-Peer User Identity Verification Time Optimization in IoT Blockchain Network," *Sensors*, vol. 23, no. 4, 2023, doi: 10.3390/s23042106.

[10] H. Guo and X. Yu, "A survey on blockchain technology and its security," *Blockchain Res. Appl.*, vol. 3, no. 2, p. 100067, 2022, doi: 10.1016/j.bcra.2022.100067.

[11] A. K. Peepliwal *et al.*, "A Prototype Model of Zero Trust Architecture Blockchain with EigenTrust-Based Practical Byzantine Fault Tolerance Protocol to Manage Decentralized Clinical Trials," *Blockchain Res. Appl.*, vol. 5, no. 4, p. 100232, 2024, doi: 10.1016/j.bcra.2024.100232.

[12] J. Swati and P. Nitin, "Securing Decentralized Storage in Blockchain: A Hybrid Cryptographic Framework," *Cybern. Inf. Technol.*, vol. 24, no. 2, pp. 16–31, 2024, doi: 10.2478/cait-2024-0013.

[13] O. Itohan Oriekhoe, G. Bolawale Omotoye, O. Peter Oyeyemi, S. Tubokirifuruar Tula, A. Ifesinachi Daraojimba, and A. Adefemi, "Blockchain in Supply Chain Management: a Systematic Review: Evaluating the Implementation, Challenges, and Future Prospects of Blockchain Technology in Supply Chains," *Eng. Sci. Technol. J.*, vol. 5, no. 1, pp. 128–151, 2024, doi: 10.51594/estj/v5i1.732.

[14] Y. A. Prabowo, W. S. Pambudi, and I. R. Imaduddin, "Identification of the Flip Folder Folding Machine Using Artificial Neuro Network Method with NARX (Nonlinear Auto Regressive Exogenous) Structure," *Inf. J. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 5, no. 2, pp. 74–79, 2020, doi: 10.25139/inform.v5i2.2743.

[15] M. C. Kriswantoro, A. Sudarsono, and M. Yuliana, "Secret Key Establishment Using Modified Quantization Log For Vehicular Ad-Hoc Network," *Inf. J. Ilm. Bid. Teknol. Inf. dan Komun.*, vol. 6, no. 2, pp. 103–109, 2021, doi: 10.25139/inform.v6i2.4037.

[16] T. Wira and E. Suryawijaya, "Memperkuat Keamanan Data melalui Teknologi Blockchain: Mengeksplorasi Implementasi Sukses dalam Transformasi Digital di Indonesia Strengthening Data Security through Blockchain Technology: Exploring Successful Implementations in Digital Transformation in Indonesia," vol. 2, no. 1, pp. 55–67, 2023, doi: 10.21787/jskp.2.2023.55-67.

[17] Z. Munawar, N. Indah Putri, I. Iswanto, and D. Widhiantoro, "Analisis Keamanan Pada Teknologi Blockchain," *Infotronik J. Teknol. Inf. dan Elektron.*, vol. 8, no. 2, p. 67, Dec. 2023, doi: 10.32897/infotronik.2023.8.2.2062.

[18] Pedro et al, "Title Page (with Author Details) Title: Insights into Blockchain Implementation in Construction: Models for 2," 2021.

[19] R. Fotohi and F. Shams Aliee, "Securing communication between things using blockchain technology based on authentication and SHA-256 to improving scalability in large-scale IoT," *Comput. Networks*, vol. 197, no. December 2020, 2021, doi: 10.1016/j.comnet.2021.108331.

[20] C. E. B. Santos, L. M. D. d. Silva, M. F. Torquato, S. N. Silva, and M. A. C. Fernandes, "SHA-256 Hardware Proposal for IoT Devices in the Blockchain Context," *Sensors*, vol. 24, no. 12, pp. 1–25, 2024, doi: 10.3390/s24123908.

[21] A. K. Yadav and V. P. Vishwakarma, "An integrated blockchain and fractional DCT based highly secured framework for storage and retrieval of retinal images," *Ain Shams Eng. J.*, vol. 15, no. 11, p. 103047, 2024, doi: 10.1016/j.asej.2024.103047.

[22] A. H. Jasim and A. H. Kashmar, "An Evaluation of RSA and a Modified SHA-3 for a New Design of Blockchain Technology," *EAI/Springer Innov. Commun. Comput.*, vol. Part F632, pp. 477–489, 2023, doi: 10.1007/978-3-031-23602-0_28.

[23] I. F. Azmi and A. A. Nugroho, "Anti-corruption system

4.0: The adoption of blockchain technology in the public sector," *Integritas J. Antikorupsi*, vol. 9, no. 1, pp. 93–108, Jun. 2023, doi: 10.32697/integritas.v9i1.985.

[24] S. Hafeez, M. A. Shawky, M. Al-Quraan, L. Mohjazi, M. A. Imran, and Y. Sun, "BETA-UAV: Blockchain-based Efficient Authentication for Secure UAV Communication," pp. 5–9, 2024, [Online]. Available: http://arxiv.org/abs/2402.15817

[25] A. Garba *et al.*, "A digital rights management system based on a scalable blockchain," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 5, pp. 2665–2680, 2021, doi: 10.1007/s12083-020-01023-z.