Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 6, Number 1, January 2022
DOI : https://doi.org/10.33395/sinkron.v7i1.11286
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Implementation Of Discrete Cosine Transform (DCT) And Blowfish Methods In Digital Video Security

Mufida Khairani[1)], Herlina Harahap [2)], Yunita Sari Siregar[3)], Yessy Fitri Annisa Lubis[4)]
[1)2)3)] Universitas Harapan Medan, Indonesia
[1)]mufida.khairani@gmail.com, [2)]Herlina_Hrp@yahoo.com, [3)]yunitasarisiregar1990@gmail.com,
[4)]Yessy.annisa@gmail.com

Submitted : Jan 11, 2022 | **Accepted** : Jan 31, 2022 | **Published** : Feb 2, 2022

Abstract: Collecting data partially or completely illegally without the permission of the owner causes losses to the copyright owner. This has a big impact in proving ownership of a digital media. To be able to prove ownership of the copyright, the Discrete Cosine Transform watermarking technique can be used by inserting the Blowfish cryptographic algorithm. Discrete Cosine Transform (DCT) is a method that converts digital data into the form of a frequency domain. The method used in the DCT transformation technique is to break the digital image into small blocks with a fixed size and then convert it from the spatial domain to the frequency domain. Blowfish works by dividing the message into 64-bit blocks of equal length with varying aty lengths that encrypt the data in 8 byte blocks. Messages that are not multiples of 8 bytes will be added extra bits (padding), so that the size for each block is the same. After the final result is obtained, the value is inserted into the digital video without destroying the original digital video. From the research that has been done in the implementation of digital video watermarking, it can be concluded that the Blowfish cryptography method and the Discrete Cosine Transform (DCT) watermarking cryptography method can be applied properly. In the watermarking process, digital image insertion does not damage the watermaratd video and at the security level it is provided with blowfish cryptographic encryption, so that it can increase security in copyright protection.

Atywords: Watermarking, Discrete Cosine Transform (DCT), Cryptography, Blowfish Algorithm, Digital Video

INTRODUCTION

Collecting data partially or completely illegally without the permission of the owner causes losses to the copyright owner. This has a big impact in proving ownership of a digital media. To be able to prove ownership of copyrights, watermarking techniques can be used. Watermarking is a technique used to hide copyright signs or information such as time or date, and the copyright owner into a digital media. The insertion of information into a digital video is done in such a way that it does not damage the quality of the video that is inserted with copyright information. This copyright information must be able to be extracted to prove ownership of the digital video product. The results of the extraction process are then compared with the original information from the copyright holder. If the extracted information is the same as the original information, then he is the copyright holder for the digital video product.

To maat the watermark, the Discrete Cosine Transform (DCT) method is used which is combined with the blowfish algorithm. Watermarking is a technique of hiding information embedded in other media with the aim of protecting the media inserted by the information from piracy, copyright abuse, and so on (Reva et al., 2016). Discrete Cosine Transform (DCT) is a method that converts digital data into the form of a frequency domain. The method used in the DCT transformation technique is to break the digital image into small blocks of a fixed size and then convert it from the spatial domain to the frequency domain (Himawan et al., 2012).

Blowfish is a cryptographic algorithm with the use of a symmetric block cipher aty, namely the aty used in the encryption process is the same as the aty used in the decryption process with input and output data in the form of

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 6, Number 1, January 2022
DOI : https://doi.org/10.33395/sinkron.v7i1.11286
e-ISSN : 2541-2019
p-ISSN : 2541-044X

64-bit data blocks. Blowfish was designed by Bruce Schneier in 1993 aimed at large microprocessors (32 bits and above with large data caches) (Abdullah & Saputro, 2016)

On research (Sutardi, 2015), the implementation of the blowfish algorithm shows that the confidentiality of the data using the blowfish algorithm depends on the aty length. The use of atys is more flexible between 1 to 16 characters, where data encryption and decryption can run well on various operating systems. In addition, the speed of encryption and decryption does not require a long time, according to the size of the data being encrypted and decrypted.

Further research conducted (Agustina & Asmara, 2015), Watermark embedding with the Discrete Cosine Transform method can be applied to increase security in digital images. This method was chosen because this method is more robust against image manipulation, especially compression. Watermark embedding is performed at the high frequency of the DCT coefficient. Based on the test, the watermaratd image is resistant to image manipulation in the form of compression up to 50%. The addition of text and color changes produces a good watermark image quality, while changing the background, adding filters, and rotating can damage the watermark after the extraction process.

From the background described previously, the problem formulation of this research is how to implement the Discrete Cosine Transform (DCT) watermarking method with the blowfish cryptography algorithm in digital video watermarking security so that there is no copyright abuse and plagiarism.

LITERATURE REVIEW

Watermarking is a form of steganography, which is the study of how to hide data from other data. This watermarking is somewhat different from the watermark on banknotes. Watermarks on banknotes are still visible to the human senses (in certain paper positions), but watermarking on digital media will not be felt by humans without digital processing machines such as computers (Irfan & AZ, 2013). There are several criteria for a good watermark, namely:

a. Imperceptibility: A good watermark is invisible to the human eye
b. Robustness: the watermark must be resistant to manipulation of the container file, such as file compression, scanning.
c. Security: the watermark can only be detected by the data owner or authorized party.
d. Recovery: the inserted data must be recovered. Because the main purpose of watermarking is copyright protection. So if at any time something detrimental happens, the watermark can be used to authenticate ownership image (Agustina & Asmara, 2015)

Digital watermarking is a technique in which digital data is inserted into the data host in a variety of ways that maat the data uninterrupted by the use of the data host, cannot be removed through normal processes, transmissions, and/or recordings from the data host, and can be read again with appropriate watermark detectors (Putrana et al., 2016). Digital Watermarking is a technique of embedding or naming data or information, or hiding information, on digital data, but the insertion is not visible to the human senses of sight, or hearing. In addition, digital watermarking is able to withstand attacks from digital signal processing processes to a certain extent (Moonlight et al., 2021).

Digital Watermarking is divided into four types based on the digital media inserted, namely:

a. Text Watermarking: Watermarks are inserted on digital media of document or text type.
b. Image Watermarking: Watermarks are embedded in digital images.
c. Watermarking: Watermarks are embedded in digital audio files such as mp3, mpeg, etc.
d. Video Watermarking: Watermarks are embedded in moving images or called digital video (Solikhin, 2017)

In watermarking there is a domain that is used for the message insertion process, namely: frequency domain and spatial domain. Computation in the spatial domain is faster so that often chosen to insert simple data. Spatial domain excels in speed however not resistant to image processing attacks, while the frequency domain is more attack resistant and longer in processing (Sari et al., 2018). Discrete Cosine Transform (DCT) is one of the methods used in image processing for image compression process. DCT has two main properties for image and video compression, namely (Wulanningrum & Lelitavistara, 2015):

a. Concentrate image energy into a small number of coefficients (energy compaction).
b. Minimize interdependence between coefficients (decorrelation)

Watermarking on digital images can be applied to various domains. Some are done directly on the type of digital data or first carried out the transformation into another domain. One of the transformations used is the Discrete Cosine Transform (DCT) which converts digital data into the form of a frequency domain. The method used in the DCT transformation technique is to break the digital image into small blocks with a fixed size and then convert it from the spatial domain to the frequency domain. The DCT technique reconstructs the image matrix into 3 frequency areas, namely Low Frequency (FL), Medium Frequency (FM) and High Frequency (FH). The human senses, namely

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 6, Number 1, January 2022
DOI : https://doi.org/10.33395/sinkron.v7i1.11286
e-ISSN : 2541-2019
p-ISSN : 2541-044X

the sense of sight, the human eye is only able to see images at a low frequency level (FL). Discrete Cosine Transform (DCT) is a transformation method used as the basis for Joint Photographic Experts Group (JPEG) compression. The part of the DCT that has the highest energy is called the DC, which is located at the top left of the image.(Irfan & AZ, 2013)

The following is the formulation used in the 2-dimensional model DCT method where image A will produce image B. It can be shown in equation 1.

$$AMN = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha p\ \alpha q\ Bpq \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)p}{2N}, 0 \le m \le M-1, 0 \le n \le N-1$$

$$\alpha P = \begin{cases} 1\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \quad \alpha q = \begin{cases} 1\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases} \quad \text{..... [1]}$$

While the formula used to invert the DCT value generated by image B into the initial image (image A) is defined using equation 2.

$$BbMN = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha p\ \alpha q\ Bpq \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)p}{2N}, 0 \le m \le M-1, 0 \le n \le N-1$$

$$\alpha P = \begin{cases} 1\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \quad \alpha q = \begin{cases} 1\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases} \quad \text{..... [2]}$$

Description :

A : An image with its components in the form of the original pixel value

B : Image with its components in the form of DCT value calculated by the formula p, q : Position of pixels in the image

M : Number of rows from image A N : Number of rows from image B

Cryptography comes from the Greek, according to the language divided into two crypto and graphia, crypto means "secret" (secret) and graphia means "writing" (writing). According to the terminology cryptography is the science and art of maintaining the security of messages when messages are sent from one place to another. If you exchange messages (such as letters) with other people, then you certainly want the messages you send to reach their intended parties safely (Abdullah & Saputro, 2016). Cryptography is the science and art of maintaining the security of messages to be sent from one place to another. A cryptographic algorithm is a logical sequence of steps to atep information secret from unauthorized persons (Suhandinata et al., 2019)

There are several important terms in cryptography, namely (Abdullah & Saputro, 2016) :

a. Message (Plaintext and Ciphertext): Message (message) is data or information that can be read and understood its meaning. The original message is called plaintext or clear text. While the message that has been encoded is called ciphertext (cipertext).
b. Sender and Receiver: Data communication involves the exchange of messages between two entities. A sender is an entity that sends a message to another entity. The receiver is the entity that receives the message.
c. An eavesdropper is a person who tries to capture a message as it is transmitted.
d. Cryptoanalysis and Cryptology: Cryptoanalysis (cryptanalysis) is the science and art of breaking ciphertext into plaintext without knowing the aty used. The culprit is called a cryptanalyst. Cryptology (cryptology) is the study of cryptography and cryptanalysis.
e. Encryption and Decryption: The process of encoding plaintext into ciphertext is called encryption or enciphering. While the process of returning ciphertext to plaintext is called decryption or deciphering.
f. Ciphers and Atys: Cryptographic algorithms, also called ciphers, are rules for encryption and decryption, or mathematical functions used for encryption and decryption. Aty is a parameter used for enciphering and dechipering transformations. The aty is usually a string or series of numbers.

Blowfish is designed and is expected to have the following design criteria: 1. Fast, Blowfish encrypts data on a 32-bit microprocessor with a rate of 26 clock cycles per byte. 2. Compact, Blowfish can run on less than 5K memory. 3. Simple, Blowfish only uses simple operations, such as: addition, XOR, and table lookup on 32-bit operands. 4. Has varying levels of security, the length of the aty used by Blowfish can vary and can be up to a minimum of 32-bits, a maximum of 448-bits, Multiple 8-bits, default 128-bits (Maradona & Basorudin, 2017)

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 6, Number 1, January 2022
DOI : https://doi.org/10.33395/sinkron.v7i1.11286
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Blowfish is optimized for various applications where atys do not change frequently, such as in network communications or automatic file encryption. In its implementation in a 32-bit microprocessor computer with a large data cache (Pentium and Power PC) Blowfish proved to be much faster than DES. But Blowfish is not suitable for applications with frequent aty changes. Blowfish also cannot be used on smart card applications because it requires a large memory (Siregar & Ariwibowo, 2014)

Blowfish uses a large sub-aty that must be computed before data encryption and decryption. The Blowfish algorithm applies a Feistel network consisting of 16 rounds. The input is a 64-bit element, X for the encryption algorithm path with the Blowfish method is described as follows:

a. Initialization of the P-array is followed by four S-boxes with a string consisting of hexadecimal Pi.
b. P1 is XORed with the first 32-bit aty, P2 is XORed with the second 32-bit aty, the process is repeated until all P-arrays have been XORed.
c. The algorithm is then used to encrypt the empty string filled with sub-atys in steps 1 and 2.
d. P1 and P2 are replaced with stage 3 outputs.
e. Stage 3 output encryption with the Blowfish algorithm using a modified sub-aty.
f. The output of stage 5 is used to replace P3 and P4.
g. The process will be repeated until all P-arrays have been replaced, followed by all 4 S-boxes, with the output constantly changing. (Suhandinata et al., 2019)

Data encryption begins with a block of 64-bit plain text elements converted to 64-bit cipher text. First, the plain text segment is divided into 2 equal parts which form the basis of Blowfish. The next step is the implementation of the XOR operation which is performed between the first 32-bit block segment (L) and the first P-array. The 32-bit data obtained from the second step is transferred to the F function where it is converted into a 32-bit data segment block, which is then XORed with the second 32-bit block segment (R) of the 64-bit plain text. After the XOR process is complete, the L and R segments are used for the next iteration of the Blowfish algorithm. The process of decrypting data is the same as encryption, only P arrays are used in reverse order. The F function of Blowfish is probably the most complicated part of this algorithm because only this part maats use of S-boxes. The F function accepts 32-bit data and divides it into 4 8-bit parts. Each part is converted to 32-bit using the S-box associated with its part. Then the 32-bit data received is XORed or combined to produce the final 32-bit data for the Blowfish permutation (Suhandinata et al., 2019).

Video is a technology for capturing, recording, processing, storing, and reconstructing a sequence of images. Video was first developed for cathode ray tube television systems. Digital video is basically composed of a series of frames that are displayed at a certain rate (frames/second). If the frame rate is high enough, the human eye sees it as a continuous series. Each frame is a digital image / image. A digital image is represented by a matrix, each of which represents an intensity value. If I is a two-dimensional matrix, I(x,y) is the intensity value corresponding to the row x and column y positions in the matrix. The points at which the image is sampled are called picture elements, or pixels (Khairani & Nurwulan, 2018)

METHOD

Based on the research research framework using DCT and the blowfish algorithm in improving the security of digital video watermarking, in this case the author performs several stages as follows:

a. Initial research. At this stage, research materials are collected from various library sources, such as books, journals (both printed and online), proceedings, magazines, articles and other sources relevant to science.
b. Data collection. The data sources for this research are several digital video samples.
c. Data initialization. The collected data is identified and classified according to the group. It also determines the validity of the data and variables to be used.
d. Blowfish aty generation process. The blowfish aty is generated using a large up-aty. The aty must be pre-computed before computing the encryption and decryption of the data.
e. Data encryption process. The results of the blowfish aty generation then the data encryption process is carried out.
f. The process of merging the blowfish algorithm with DCT. The result of the encryption is insertion by taking the bit value through the DCT method.
g. Data decryption process. If the data that has been encrypted and combined with the DCT method is to be returned to its original state, then the data decryption process is carried out.
h. Conclusion. From several processes carried out, it can be concluded that the iteration process will continue to be carried out until the results obtained are maximal and in accordance with what the authors expect.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 6, Number 1, January 2022
DOI : https://doi.org/10.33395/sinkron.v7i1.11286
e-ISSN : 2541-2019
p-ISSN : 2541-044X

RESULT

In the test case on the watermarking system, data is needed that will be the source of reference and model making, while the procedure for data modeling can be explained from several steps as follows:

a. The input data is taatn from digital video sample data.
b. The data is the original data which is then continued with the data collection process. The data set process is in charge of changing the original data from the input into data to be initialized.
c. After being initialized data, aty generation will be carried out with the following steps:
   1) There is a permutation box (P-box) consisting of 18 32-bit sub-atys: P1, P2, P3,... P18. This P-box has been set from the start, the 4 initial P-boxes are as follows:P1 = 0x243f6a88
      P2 = 0x85a308d3
      P3 = 0x13198a2e
      P4 = 0x03707344
   2) XOR P1 with the initial 32 bits of the aty, Xor P2 with the next 32 bits of the aty, and continue until the entire length of the aty has been Xored (possibly up to P14, 14 x 32 = 448), maximum length)
   3) There are 64 bits with empty contents. The bits are entered into step
   4) Replace P1 and P2 with the output from step c.
   5) Re-encrypt the output of step c with step b, but this time with a different sub-aty.
   6) Replace P3 and P4 with the output of step e.
   7) Continue until all P-boxes are completely randomized.
d. After getting the results of all the required upa-atys, data encryption is carried out as follows:
   1) The input of this process is 64-bit data initialized by “x”.
   2) Divide x into 2 equal parts, xL (x left) is 32 bits long and xR (x right) is 32 bits long.
   3) Iterate over i=1 to i=16:xL = xL XOR P[i];
      xR = F(xL) XOR xR;
      Swap(xL, xR);
   4) The function of F is as follows: Divide xL into 4 8 bits a, b, c, and d.
      F(xL) = ((S[1,a] + S2[2,b] mod 232) XOR S[3,c] + S[4,d] mod 232.
   5) The final step is:
      Swap(xL, xR);
      xR = xR XOR P[17]; xL = xL XOR
      P[18];
e. The output generated from the encryption process is taatn the bit value to be inserted into the video through the Discrete Cosine Transform (DCT) method.

Before entering the program testing phase using the VB.NET application, manual calculations were carried out using the Discrete Cosine Transform (DCT) method and the blowfish algorithm cryptography in encrypting the aty for digital video watermarking. The plaintext of the calculation is UNHAR MEDAN with the aty used is UNHAR. After determining the plaintext and aty, the aty generator will be calculated using the blowfish algorithm. The calculations are as follows:

Plainteks : UNHAR MEDAN
Key : UNHAR
Key Generation:
P[0] XOR D[0] = 608135816 XOR 1431193665 = 1903239881
P[1] XOR D[1] = 2242054355 XOR 1381322312 = 3623241371
P[2] XOR D[2] = 320440878 XOR 1095914830 = 1380704096
P[3] XOR D[3] = 57701188 XOR 1212240469 = 1261510929
P[4] XOR D[4] = 2752067618 XOR 1313358162 = 3930159472
P[5] XOR D[5] = 698298832 XOR 1431193665 = 2094102929
P[6] XOR D[6] = 137296536 XOR 1381322312 = 1518056656
P[7] XOR D[7] = 3964562569 XOR 1095914830 = 2904308167
P[8] XOR D[8] = 1160258022 XOR 1212240469 = 225014707
P[9] XOR D[9] = 953160567 XOR 1313358162 = 1989693989
P[10] XOR D[10] = 3193202383 XOR 1431193665 = 3944361614
P[11] XOR D[11] = 887688300 XOR 1381322312 = 1723613732
P[12] XOR D[12] = 3232508343 XOR 1095914830 = 2180939001
P[13] XOR D[13] = 3380367581 XOR 1212240469 = 2168259208
P[14] XOR D[14] = 1065670069 XOR 1313358162 = 1909232871

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

P[15] XOR D[15] = 3041331479 XOR 1431193665 = 3758702934
P[16] XOR D[16] = 2450970073 XOR 1381322312 = 3225656209
P[17] XOR D[17] = 2306472731 XOR 1095914830 = 3358305877

BlockLeft XOR P[0] = 0 XOR 1903239881 = 1903239881
Encryption Round At - 1 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(0,1903239881) = 1550133945
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(1903239881,1550133945) = 3803788529
Encryption Round At - 3 :
BlockRight= FunctionF(BlockRight, BlockLeft) = FunctionF(1550133945,3803788529) = 179553141
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3803788529,179553141) = 410967972
Encryption Round At - 5 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(179553141,410967972) = 2238895262
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(410967972,2238895262) = 1279609848
Encryption Round At - 7 :
BlockRight= FunctionF(BlockRight, BlockLeft) = FunctionF(2238895262,1279609848) = 4230229400
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(1279609848,4230229400) = 2850051818
Encryption Round At - 9 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(4230229400,2850051818) = 3678391909
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2850051818,3678391909) = 568346407
Encryption Round At - 11 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(3678391909,568346407) = 2809326298
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(568346407,2809326298) = 445149594
Encryption Round At - 13 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(2809326298,445149594) = 1707252344
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(445149594,1707252344) = 362891151
Encryption Round At - 15 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(1707252344,362891151) = 2674638949
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(362891151,2674638949) = 3679413456
BlockRight XOR P[17] = 2674638949 XOR 3358305877 = 1463840304
Swap BlockRight with BlockLeft :
BlockLeft = 1463840304
BlockRight = 1463840304
P[0] = BlockLeft = 1463840304
P[1] = BlockRight = 3679413456

Key generation continues until the next key generation cycle:
BlockLeft XOR P[0] = 2192246169 XOR 1463840304 = 3588975529
Encryption Round At - 1 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(1878262694,3588975529) = 1925339637
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3588975529,1925339637) = 3703055392
Encryption Round At - 3 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(1925339637,3703055392) = 3336032817
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3703055392,3336032817) = 3630552731
Encryption Round At - 5 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(3336032817,3630552731) = 2211043282
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3630552731,2211043282) = 927807291
Encryption Round At - 7 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(2211043282,927807291) = 1558711166
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(927807291,1558711166) = 229678116
Encryption Round At - 9 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(1558711166,229678116) = 4014904364
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(229678116,4014904364) = 3701843627
Encryption Round At - 11 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(4014904364,3701843627) = 1558421055
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3701843627,1558421055) = 2075451769
Encryption Round At - 13 :
*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 6, Number 1, January 2022
DOI : https://doi.org/10.33395/sinkron.v7i1.11286
e-ISSN : 2541-2019
p-ISSN : 2541-044X

BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(1558421055,2075451769) = 1140663550
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2075451769,1140663550) = 1770324411
Encryption Round At - 15 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(1140663550,1770324411) = 652546610
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(1770324411,652546610) = 4285544314
BlockRight XOR P[17] = 652546610 XOR 2047768759 = 1558932101
Swap BlockRight with BlockLeft :
BlockLeft = 1558932101
BlockRight = 1558932101

Encryption :
Block Encryption:
85,78,72,65,82,32,77,69,
Break Block:
Block Left:
69,77,32,82,
Convert Block Left To Integer 32Bit: 1431193665
Block Right:
65,72,78,85,
Convert Block Right To Integer 32Bit: 1377848645
BlockLeft XOR P[0] = 1431193665 XOR 1463840304 = 34481777
Encryption Round At - 1 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(1377848645,34481777) = 874080579
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(34481777,874080579) = 2855446334
Encryption Round At - 3 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(874080579,2855446334) = 2259843998
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2855446334,2259843998) = 185647896
Encryption Round At - 5 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(2259843998,185647896) = 2286336335
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(185647896,2286336335) = 3970762800
Encryption Round At - 7 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(2286336335,3970762800) = 345472776
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3970762800,345472776) = 2788092794
Encryption Round At - 9 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(345472776,2788092794) = 1898591376
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2788092794,1898591376) = 2905875308
Encryption Round At - 11 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(1898591376,2905875308) = 3892648060
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2905875308,3892648060) = 1168791406
Encryption Round At - 13 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(3892648060,1168791406) = 631642274
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(1168791406,631642274) = 516824184
Encryption Round At - 15 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(631642274,516824184) = 3388049096
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(516824184,3388049096) = 1079505590
BlockRight XOR P[17] = 3388049096 XOR 2047768759 = 3019898495
Swap BlockRight dengan BlockLeft :
BlockLeft = 3019898495
BlockRight = 3019898495

Block Encryption:
68,65,78,0,0,0,0,0,
Break Block:
Block Left:
0,0,0,0,
Convert Block Left To Integer 32Bit: 1145130496
Block Right:
*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

0,78,65,68,
Convert Block Right To Integer 32Bit: 0
BlockLeft XOR P[0] = 1145130496 XOR 1463840304 = 318840880
Encryption Round At - 1 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(0,318840880) = 1637140925
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(318840880,1637140925) = 2383279111
Encryption Round At - 3 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(1637140925,2383279111) = 3639768287
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2383279111,3639768287) = 3268103476
Encryption Round At - 5 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(3639768287,3268103476) = 273580483
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3268103476,273580483) = 2795910494
Encryption Round At - 7 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(273580483,2795910494) = 88121977
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2795910494,88121977) = 448179905
Encryption Round At - 9 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(88121977,448179905) = 3902920877
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(448179905,3902920877) = 3919775186
Encryption Round At - 11 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(3902920877,3919775186) = 3818784025
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3919775186,3818784025) = 2956492819
Encryption Round At - 13 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(3818784025,2956492819) = 1441152094
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2956492819,1441152094) = 270130667
Encryption Round At - 15 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(1441152094,270130667) = 712325540
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(270130667,712325540) = 2122651022
BlockRight XOR P[17] = 712325540 XOR 2047768759 = 1350254867
Swap BlockRight dengan BlockLeft :
BlockLeft = 1350254867
BlockRight = 2122651022
Converting Block Left to a sequence of bits: 01010000 01111011 01000001 00010011
Converting Block Right to a sequence of bits: 01111110 10000101 00010101 10001110
The ciphertext results are : ³ÿþ• @Wò¶P{A_~…_Ž

Do it repeatedly until :
Block Description: 179,255,254,127,64,87,242,182,
Break Block:
Block Left:
182,242,87,64,
Convert Block Left To Integer 32Bit: 3019898495 Block Right:
127,254,255,179,
Convert Block Right To Integer 32Bit: 1079505590
BlockLeft XOR P[17] = 3019898495 XOR 2047768759 = 3388049096
Encryption Round At - 17 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(1079505590,3388049096) = 516824184
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3388049096,516824184) = 631642274
Encryption Round At - 15 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(516824184,631642274) = 1168791406
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(631642274,1168791406) = 3892648060
Encryption Round At - 13 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(1168791406,3892648060) = 2905875308
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3892648060,2905875308) = 1898591376
Encryption Round At - 11 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(2905875308,1898591376) = 2788092794
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(1898591376,2788092794) = 345472776
Encryption Round At - 9 :
*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(2788092794,345472776) = 3970762800
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(345472776,3970762800) = 2286336335
Encryption Round At - 7 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(3970762800,2286336335) = 185647896
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2286336335,185647896) = 2259843998
Encryption Round At - 5 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(185647896,2259843998) = 2855446334
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(2259843998,2855446334) = 874080579
Encryption Round At - 3 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(2855446334,874080579) = 34481777
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(874080579,34481777) = 1377848645
BlockRight XOR P[0] = 34481777 XOR 1463840304 = 1431193665
Swap BlockRight dengan BlockLeft :
BlockLeft = 1431193665
BlockRight = 1377848645

Block Description:
80,123,65,19,126,133,21,142,
Break Block:
Block Left:
142,21,133,126,
Convert Block Left To Integer 32Bit: 1350254867
Block Right:
19,65,123,80,
Convert Block Right To Integer 32Bit: 2122651022
BlockLeft XOR P[17] = 1350254867 XOR 2047768759 = 712325540
Encryption Round At - 17 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(2122651022,712325540) = 270130667
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(712325540,270130667) = 1441152094
Encryption Round At - 15 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(270130667,1441152094) = 2956492819
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(1441152094,2956492819) = 3818784025
Encryption Round At - 13 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(2956492819,3818784025) = 3919775186
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3818784025,3919775186) = 3902920877
Encryption Round At - 11 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(3919775186,3902920877) = 448179905
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3902920877,448179905) = 88121977
Encryption Round At - 9 :
BlockRight = FunctionF(BlockRight, BlockLeft) = FunctionF(448179905,88121977) = 2795910494
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(88121977,2795910494) = 273580483
FunctionF(BlockLeft, BlockRight) = FunctionF(273580483,3268103476) = 3639768287
Encryption Round At - 5 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(3268103476,3639768287) = 2383279111
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(3639768287,2383279111) = 1637140925
Encryption Round At - 3 :
BlockRight =FunctionF(BlockRight, BlockLeft) = FunctionF(2383279111,1637140925) = 318840880
BlockLeft = FunctionF(BlockLeft, BlockRight) = FunctionF(1637140925,318840880) = 0
BlockRight XOR P[0] = 318840880 XOR 1463840304 = 1145130496
Swap BlockRight dengan BlockLeft :
BlockLeft = 1145130496
BlockRight = 0

Decryption result:
1431193665 = 01010101 01001110 01001000 01000001 = UNHA
1377848645 = 01010010 00100000 01001101 01000101 = R ME
1145130496 = 01000100 01000001 01001110 00000000 = DAN
*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 6, Number 1, January 2022
DOI : https://doi.org/10.33395/sinkron.v7i1.11286
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Final result decryption = UNHAR MEDAN

DISCUSSIONS

On the main page display for digital video watermarking using the Discrete Cosine Transform (DCT) method and blowfish algorithm cryptography, there is a watermarking menu where the menu has a function to enter the application. Atmudian there is an extraction menu that functions as a menu to extract digital videos that have been watermarked. On the encrypted watermarking page the user can choose which videos to watermark using the selected image. After the image is entered in the digital video as a watermark, the video is embedded. After the video watermarking process is complete, the encryption process is carried out using the blowfish algorithm with the key 12345. The encrypted watermarking can be seen in Figure 1

Figure 1. Encrypted Watermarking

On the watermark extraction page there is a way to describe a digital video that has been watermarked using the Discrete Cosine Transform (DCT) method and blowfish algorithm cryptography is a video that has been watermarked, then select the watermark detection command, it will be asked to enter a description key where the key is 12345. Atmudian will the output of the ctra that has been included in the digital video for water making. The display of watermark extraction can be seen in Figure 2.

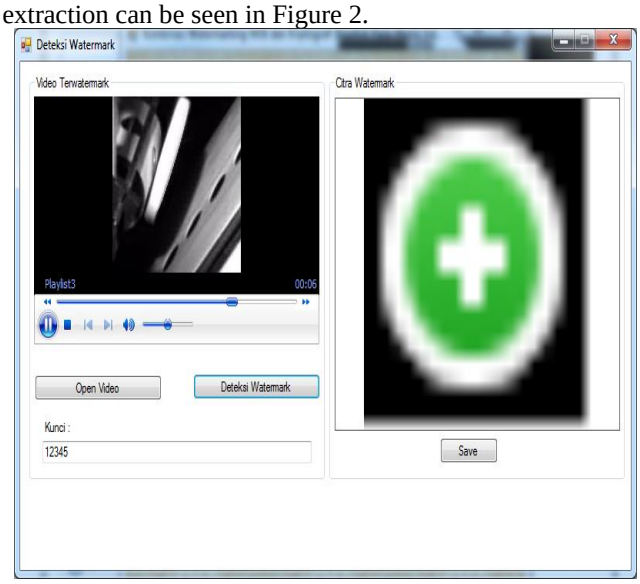

Figure 2. Extraction of Watermark

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

CONCLUSION

Based on the research that has been carried out with the title "Implementation of Discrete Cosine Transform (DCT) Watermarking Method With Blowfish Cryptography Algorithm in Digital Video Security" it can be concluded that: Blowfish cryptography method and Discrete Cosine Transform (DCT) watermarking digital video can be applied properly and in the process watermarking, digital image insertion will not damage the video to be watermarked, and is given a level of security with blowfish algorithm cryptographic encryption, so as to increase security in copyright protection.

REFERENCES

Abdullah, D., & Saputro, D. N. (2016). Implementasi Algoritma Blowfish Dan Metode Least Significant Bit Insertion Pada Video Mp4. *Jurnal Pseudocode*, *3*(2), 137–145. https://doi.org/10.33369/pseudocode.3.2.137-145

Agustina, R., & Asmara, R. A. (2015). Penyisipan Watermark Menggunakan Metode Discrete Cosine Transform Pada Citra Digital. *Jurnal Informatika Polinema*, *2*(1), 29–34.

Himawan, Usup, Irfansyah, P., & Lukman. (2012). Penggunaan Teknik Watermarking Menggunakan Metode Discrete Cosine Transform (DCT) dalam Perlindungan Hak Cipta Dokumen Citra Digital. *Konferensi Nasional ICT-M Politeknik Telkom (KNIP)*, 281–287.

Irfan, & AZ, N. (2013). Prototipe Teknik Penyisipan Dokumen Citra Digital Menggunakan Watermarking dengan Metode DCT (Discrete Cosine Transform). *Jurnal TICOM*, *2*(1), 21–27.

Khairani, M., & Nurwulan, N. (2018). Algoritma Blowfish Pada Watermarking Video Digital. *JURIKOM (Jurnal …*, *5*(4), 357–361. http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom/article/view/842

Maradona, H., & Basorudin. (2017). Analisis Algoritma Blowfish Pada Proses Enkripsi dan Deskripsi File. *Riau Journal Of Computer Science*, *3*(2), 156–166.

Moonlight, L. S., Irfansyah, A., & Widyarini, R. (2021). Digital Image Watermarking Pada Citra Medis Menggunakan Discrete Cosine Transform (Dct), Dan Metode Spread Spectrum. *Scan*, *16*(1), 27–33.

Putrana, C. B., Budiman, G., Nyoman, I., & Ramatryana, A. (2016). Merancang Citra Watermark Menggunakan DCT dan Terenkripsi – Kompresi Teks Menggunakan RSA-HUFFMAN Code. *Seminar Nasional Inovasi Dan Aplikasi Teknologi Di Industri (SENIATI)*, 128–133.

Reva, E. Y., Susilo, B., & Purwandari, E. P. (2016). Aplikasi Watermark Pada Citra Digital Menggunakan Kombinasi Metode Discrete Cosine Transform , Discrete Wavelet Transform Dan Singular Value Decomposition. *Jurnal Rekursif*, *4*(2), 152–160. ejournal.unib.ac.id

Sari, C. A., Sukamto, T. S., & Rachmawanto, E. H. (2018). Analisa Robustness Citra Digital Pada Watermarking DCT-DWT. *Prosiding SNST*, *1*(1), 19–22.

Siregar, K. P., & Ariwibowo, E. (2014). Implementasi Keamanan Data Menggunakan Algoritma Blowfish dan LSB Pada Citra. *Jurnal Sarjana Teknik Informatika*, *2*(3), 140–150.

Solikhin, M. (2017). WATERMARKING MENGGUNAKAN METODE DISCRETE COSINE TRANSFORM. *Jurnal Riset Dan Aplikasi Matematika*, *1*(1), 27–39.

Suhandinata, S., Rizal, R. A., Wijaya, D. O., Warren, P., & Srinjiwi. (2019). Analisis Performa Kriptografi Hybrid Algoritma Blowfish dan Algoritma RSA. *Jurteksi*, *VI*(1), 1–10.

Sutardi. (2015). Implementasi Algoritma Blowfish Untuk Keamanan Data Suara. *Jurnal Ilmiah Teknik Mesin*, *6*(2), 51–58.

Wulanningrum, R., & Lelitavistara, N. V. W. (2015). Discrete Cosine Transform Untuk Identifikasi Citra Hylocereus Costaricensis. *Simetris : Jurnal Teknik Mesin, Elektro Dan Ilmu Komputer*, *6*(2), 353–360. https://doi.org/10.24176/simet.v6i2.472

*name of corresponding author