Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15 yrs june 202 3 . Vol. 2 . No. 3; e-ISSN : 2808-4519

Implementation Of Affine Cipher Combination And Merkle Hellman On The Process Digital Image Security

Muhammad Fadillah Azmi [1*] , Achmad Fauzi [2] , Husnul Khair [3]

[1,2,3] Informatics Engineering, STMIK KAPUTAMA
Jl. Veterans No. 4A-9A, Binjai, North Sumatra, Indonesia
muhammadfadillahazmi5@gmail.com[1], fauzyrivai88@gmail.com[2], husnul.khair@gmail.com[3]

Abstract

The use of image media information has several weaknesses, one of which is the ease with which it can be manipulated by irresponsible people. Efforts made in increasing the security of this image is cryptography, namely the science and art of maintaining the security and confidentiality of images. Cryptography is used so that the confidentiality of the image can be maintained, so that it is not known by others. In general, there are 2 (two) techniques for carrying out cryptographic processes, namely encryption and decryption. In this study, the encryption process was first carried out using the Affine Cipher algorithm and then continued with the Merkle Hellman algorithm, while the decryption was carried out first by Merkle Hellman and then decrypted again with the Affine Cipher. The results of this study indicate that applying the Affine Cipher and Merkle Hellman algorithms can secure images. At the end of this system is an image that is in the form of blur so that it is not known and not understood by others. That way people who are not entitled cannot understand from the image.

Keywords: *cryptography, image, affine cipher algorithm, merkle hellman algorithm.*

1. Introduction

Computers are a technology that is often used to store important information, especially digital images. Digital images are often intercepted and manipulated by irresponsible people. Many new techniques seem to appear frequently so that the image can be misused by people who want to commit crimes with harmful purposes. Therefore, the security and confidentiality of digital images is very important. Using Affine Cipher and Merkle Hellman is one way to secure images. Both of these methods can guarantee security so that the image does not leak which can harm the owner. To encrypt the image by changing it to make the object opaque, or not too clear so that it is not known by many people using the Affine Cipher and Merkle Hellman methods. Meanwhile, what has been encrypted or kept secret will be decrypted back into a digital image that is like the original. Affine Cipher is a special method for encryption and decryption per character. Therefore, affine is classified as a symmetric key encryption method because this method has a key for image encryption and decryption. Merkle Hellman is a method that uses asymmetry, because it uses different key pairs for encryption and decryption of the image.

2. Research methodology

2.1. Cryptography

Cryptography comes from the Greek words crypto and graphia, crypto means secret and graphia means writing. According to terminology, cryptography is the art and science of ensuring the security of images sent from one place to another. In the development of cryptography it is also used to protect digital images and send digital signature messages on message authenticity using digital fingerprints or fingerprints [1] .

2.2 Classical Algorithms and Modern Algorithms

The classical algorithm is an algorithm that uses one key to protect data. This technique has been used for several centuries. This algorithm randomizes letters in bright words / plaintext. This algorithm only randomizes the letters AZ and is not recommended for storing important data because it can be completed in a short time [2] .

The modern algorithm is a refinement of the classic algorithm. This algorithm uses binary symbol processing, which consists of ASCII (American Standard Code For Information Interchange) codes, because it works on the basis of a digital computer. Therefore, basic knowledge of mathematics is required to master it. The level of difficulty of this algorithm makes it very difficult for cryptanalysts to crack ciphertexts without knowing the key [3] .

2.3 Affine Cipher Algorithm

Affine Cipher is an extension of Caesar Cipher which switches plaintext with a value and adds it with a shift [4] .
The encryption and decryption formulas for the Affine Cipher algorithm are as follows:

Affine Cipher Encryption Formula:
E(P)=(ax+b) mods m

Affine Cipher Decryption Formula:
D(x)=a(xb) mods m

Information :
E(p) = encrypted text of the letter p.
D(x) = light text of encoded letter y.
a and b = key parameters.
m = alphabet or ASCII size (usually 26 or 256 characters).

2.4 Merkle Hellman's Algorithm

Merkle Hellman is a cryptographic system that uses asymmetric key types. This Merkle Hellman system uses two different locks. One key for encryption and one key for decryption. The Merkle-Hellman algorithm system uses an asymmetric encryption key type. Therefore, the keys used to encrypt and decrypt messages are different. In this case, if Alice is to decrypt messages received from Bob, for example, Alice must first know her secret key to get Bob's public key. For example, Alice has the following increasing integer sequence w = 1 + 3 + 6 + 11 + 32 + 87 + 141 + 354 = 635. Since sum is 281, Alice chooses any prime q that is greater than 635, say 881, and any number r with a value between 1-w-1, namely 588 [5] .

So Alice's secret key is:
w = (1, 3, 6, 11, 32, 87, 141, 354)
q = 881
r = 588

Information :
w = super increasing number
r = private key
q = divisor number
To get the public key, the formula is used:

$$ti = r * w \bmod q$$

Information :
ti = public key
r = secret key
w = super increasing number
q = number of divisors and calculate like the way below:

Table 1: Process Search Key Public

r		W		q		t_i
588	*	1	mod	881	=	588
588	*	3	mod	881	=	2
588	*	5	mod	881	=	297

588	*	11	mod	881	=	301
588	*	32	mod	881	=	315
588	*	87	mod	881	=	58
588	*	141	mod	881	=	94
588	*	354	mod	881	=	236

From the table above, the public key results are obtained as follows:
ti = (588, 2, 297, 301, 315, 58, 94, 236)

For the encryption process, each character is converted into an ASCII number and then converted back into a binary number. For example, the character to be encrypted is the letter "S". The letter "S" is converted to ASCII (S=83) so that it can be decomposed into a binary number. The binary number for the letter "S" is 1010011 [5] .

After getting the binary number of the character in question, the encryption process is carried out with the formula:

$$\boldsymbol{ti} * \boldsymbol{x} = \sum \boldsymbol{y}$$

Information :
ti = public key
x = binary number of characters
y = multiplication result

Merkle Hellman's Decryption Formula:

z = y mod q

Information :
z = target sum / multiplication result
r = private key
= inverse of r
y = received ciphertext
q = divisor number

2.5 Digital Image

Image is a representation (image), likeness or imitation of an object. The image output of a data recording system can be optical in the form of photographs, analog in the form of video signals such as images on television, or digital in nature and stored on storage media. Images are grouped into two parts, namely still images (still images) and moving images (animations). A still image is a single image that is not moving. Conversely, moving images are a series of still images that are displayed one after another in such a way as to give the eye the impression of moving images. Each image in a series is called a frame [6]

3. Results and Discussion

3.1. Calculation of Affine Cipher Encryption

Figure 1: Pixel Images and Samples 8X8 Pixel Images

In this process aims to secure the image. The image size used is 1080 x 720 pixels. The process of securing this digital image will be sampled on an image with a size of 8 x 8 pixels, and the pixels will be converted to RGB values using Photoshop, which will start from the pixel point (0.0) to (7.7).
The digital image security that will be used in the affine cipher algorithm process is as follows. The following are the steps for Affine cipher plaintext encryption, namely:
Plaintext :

(x,y)	0	1	2	3	4	5	6	7
0	R : 112 G : 133 B : 28	R : 137 G : 158 B : 55	R : 147 G : 167 B : 70	R : 129 G : 148 B : 59	R : 76 G : 92 B : 17	R : 45 G : 59 B : 0	R : 52 G : 64 B : 16	R : 90 G : 101 B : 61
1	R : 150 G : 172 B : 64	R : 147 G : 163 B : 65	R : 166 G : 186 B : 89	R : 124 G : 142 B : 56	R : 62 G : 78 B : 5	R : 77 G : 90 B : 34	R : 87 G : 98 B : 55	R : 67 G : 77 B : 42
2	R : 171 G : 192 B : 87	R : 158 G : 179 B : 76	R : 173 G : 193 B : 98	R : 125 G : 143 B : 59	R : 63 G : 78 B : 0	R : 63 G : 76 B : 23	R : 84 G : 94 B : 57	R : 92 G : 101 B : 72
3	R : 183 G : 201 B : 101	R : 189 G : 206 B : 110	R : 183 G : 199 B : 110	R : 104 G : 118 B : 39	R : 117 G: 129 B : 67	R : 144 G: 153 B : 106	R : 90 G : 97 B : 64	R : 77 G : 82 B : 59
4	R : 191 G : 207 B : 118	R : 208 G : 224 B : 136	R : 166 G : 181 B : 100	R : 128 G : 141 B : 69	R : 178 G: 189 B : 129	R : 214 G: 223 B : 180	R : 209 G: 216 B : 185	R : 184 G : 189 B : 166
5	R : 201 G : 213 B : 137	R : 205 G : 217 B : 143	R : 210 G : 222 B : 150	R : 222 G : 233 B : 167	R : 198 G: 207 B : 152	R : 188 G: 195 B : 151	R : 222 G: 228 B : 194	R : 200 G : 204 B :177
6	R : 200 G : 210 B : 149	R : 145 G : 155 B : 92	R : 170 G : 180 B : 119	R : 127 G : 137 B : 77	R : 73 G : 82 B : 29	R : 125 G: 133 B : 86	R : 151 G: 157 B : 119	R : 154 G : 160 B : 126
7	R : 120 G : 129 B : 74	R : 102 G : 111 B : 56	R : 139 G : 148 B : 91	R : 160 G : 169 B : 112	R : 111 G: 120 B : 67	R : 97 G: 105 B : 58	R : 110 G: 117 B : 76	R : 129 G : 135 B : 97

Encrypted Affine Cipher with key :
a = 7 (since the value of a is relatively prime) and,
b = 10 (because the value of the shift (random value) starts from 1 to infinity).
m = 256 (because the alphabet used 256 ASCII tables).

Plaintext encryption is calculated by the formula:

E(P)=(ax+b) mod m

For this reason, the image encryption process with the affine cipher algorithm is as follows:

Pixels (0,0)
Red : 112
= 7 . 112 + 10 = 794 mod 256 = **26**
Green : 133
= 7 . 133 + 10 = 941 mod 256 =**173**
Blue : 28
= 7 . 28 + 10 = 206 mod 256 = **206**

Pixel (0,1)
Red : 137
= 7 . 137 + 10 = 969 mod 256 = **201**
Green : 158
= 7 . 158 + 10 = 1,116 mod 256 =**92**
Blue : 55
= 7 . 55 + 10 = 395 mod 256 = **139**

Pixel (0,2)
Red : 147
= 7 . 147 + 10 = 1,039 mod 256 = **15**
Green : 167

= 7 . 167 + 10 = 1,179 mod 256 = **155**
Blue : 70
= 7 . 70 + 10 = 500 mod 256 = **244**

Pixel (0,3)
Red : 129
= 7 . 129 + 10 = 913 mod 256 = **145**
Green : 148
= 7 . 148 + 10 = 1,046 mod 256 = **22**
Blue : 59
= 7 . 59 + 10 = 423 mod 256 = **167**

Pixel (0,4)
Red : 76
= 7 . 76 + 10 = 542 mod 256 = **30**
Green : 92
= 7 . 92 + 10 = 654 mod 256 = **142**
Blue : 70
= 7 . 17 + 10 = 129 mod 256 = **129**

Pixel (0,5)
Red : 45
= 7 . 45 + 10 = 325 mod 256 = **69**
Green : 59
= 7 . 59 + 10 = 423 mod 256 = **167**
Blue : 0
= 7 . 0 + 10 = 10 mod 256 = **10**
Pixel (0,6)
Red : 52
= 7 . 52 + 10 = 374 mod 256 = **118**
Green : 64
= 7 . 64 + 10 = 458 mod 256 = **202**
Blue : 16
= 7 . 16 + 10 = 122 mod 256 = **122**

Pixel (0,7)
Red : 90
= 7 . 90 + 10 = 640 mod 256 = **128**
Green : 101
= 7 . 101 + 10 = 717 mod 256 = **205**
Blue : 61
= 7 . 437 + 10 = 500 mod 256 = **181**
Then proceed to pixel (7,7) so that the results of the ciphertext in the Affine Cipher encryption process are known as follows:

(x,y)	0	1	2	3	4	5	6	7
0	**R : 26** **G : 173** **B : 206**	**R : 201** **G : 92** **B: 139**	**R : 15** **G : 155** **B : 244**	**R : 145** **G : 22** **B : 167**	**R : 30** **G: 142** **B : 129**	**R : 69** **G : 167** **B : 10**	**R : 118** **G: 202** **B : 122**	**R : 128** **G : 205** **B : 181**
1	R : 36 G : 190 B : 202	R : 15 G : 162 B : 209	R : 148 G : 32 B : 121	R : 110 G : 236 B : 146	R : 188 G : 44 B : 45	R : 37 G : 128 B : 148	R : 107 G: 184 B : 139	R : 223 G : 37 B : 48
2	R : 183 G : 74 B : 107	R : 92 G : 239 B : 30	R : 197 G : 81 B : 184	R : 117 G : 243 B : 167	R : 195 G : 44 B : 73	R : 195 G : 30 B : 171	R : 86 G: 156 B : 153	R : 142 G : 205 B : 2
3	R : 11 G : 137 B : 205	R : 53 G : 172 B : 12	R : 11 G : 123 B : 12	R : 226 G : 68 B : 27	R : 61 G: 145 B : 223	R : 250 G : 57 B : 240	R : 128 G: 177 B : 202	R : 37 G : 72 B : 167
4	R : 67 G : 179 B : 68	R : 186 G : 42 B : 194	R : 148 G : 253 B : 198	R : 138 G : 229 B : 237	R : 232 G : 53 B : 145	R : 228 G : 35 B : 246	R : 193 G: 242 B : 25	R : 18 G : 53 B : 148

5	R : 137 G : 221 B : 201	R : 165 G : 249 B : 243	R : 200 G : 28 B : 36	R : 28 G : 105 B : 155	R : 116 G: 179 B : 50	R : 46 G : 95 B : 43	R : 28 G : 70 B : 88	R : 130 G : 158 B :225
6	R :130 G : 200 B : 29	R : 1 G : 71 B: 142	R : 176 G : 246 B : 75	R : 131 G : 201 B : 37	R : 9 G : 72 B : 213	R : 117 G : 173 B :100	R : 43 G : 85 B : 75	R : 64 G : 106 B : 124
7	R : 82 G : 145 B : 16	R : 212 G : 19 B : 146	R : 215 G : 22 B: 135	R : 106 G : 169 B : 26	R : 19 G : 82 B : 223	R : 177 G : 233 B : 160	R : 12 G : 61 B : 30	R : 145 G : 187 B : 177

3.2. Calculation Enkripsi Merkle Hellman

Plaintext :

Pixel (0,0) : R : **26: 0001 1010** G : **173:1010 1101** B: **206: 1100 1110** **Pixel (0,4) :** R : **30 : 0001 1110** G:**142 : 1000 1110** B:**129 : 1000 0001**	**Pixel (0,1) :** R: **201: 1100 1001** G : **92 : 0101 1100** B: **139: 1000 1011** **Pixel (0,5) :** R : **69 : 0100 0101** G:**167 : 1010 0111** B : **10 : 0000 1010**	**Pixel (0,2) :** R : **15 : 0000 1111** G: **155: 1001 1011** B : **244 :1111 0100** **Pixel (0,6) :** R: **118 : 0111 0110** G: **202: 1100 1010** B :**122 : 0111 1010**	**Pixel (0,3) :** R: **145:1001 0001** G: **22 : 0001 0110** B: **167:1010 0111** **Pixel (0,7) :** R:**128: 1000 0000** G:**205: 1100 1101** B: **181:1011 0101**
Pixel (1,0) : R : 36 : 0010 0100 G :190: 1011 1110 B :202: 1100 1010 **Pixel (1,4) :** R: 188: 1011 1100 G : 44: 0010 1100 B : 45: 1100 1010	**Pixel (1,1) :** R : 15 : 0000 1111 G: 162: 1010 0010 B: 209: 1101 0001 **Pixel (1,5) :** R : 37: 0010 0101 G: 128: 1000 0000 B: 148: 1111 1000	**Pixel (1,2) :** R : 148: 1001 0100 G : 32: 0010 0000 B : 121: 0111 1001 **Pixel (1,6) :** R : 107: 0110 1011 G: 184: 1011 1000 B : 139: 1000 1011	**Pixel (1,3) :** R:110: 0110 1110 G:236: 1110 1100 B:146: 1001 0010 **Pixel (1,7) :** R:223: 1101 1111 G : 37: 0010 0101 B : 48: 0011 0000
Pixel (2,0) : R: 183: 1011 0111 G : 74: 0100 1010 B: 107: 0110 1011 **Pixel (2,4) :** R: 195: 1100 0011 G : 44: 0010 1100 B : 73: 0100 1001	**Pixel (2,1) :** R : 92: 0101 1100 G: 239: 1110 1111 B : 30: 0001 1110 **Pixel (2,5) :** R: 195: 1100 0011 G : 30: 0001 1110 B: 171: 1010 1011	**Pixel (2,2) :** R : 197: 1100 0101 G : 81: 0101 0001 B : 184: 1011 1000 **Pixel (2,6) :** R : 86: 0101 0110 G: 156: 1001 1100 B : 153: 1001 1001	**Pixel (2,3) :** R:117: 0111 0101 G:243: 1111 0011 B:167: 1010 0111 **Pixel (2,7) :** R:142: 1000 1110 G:205: 1100 1101 B : 2: 0000 0010
Pixel (3,0) : R : 11: 0000 1011 G: 137: 1000 1001 B: 205: 1100 1101 **Pixel (3,4) :** R : 61: 0011 1101 G: 145: 1001 0001 B: 223: 1101 1111	**Pixel (3,1) :** R : 53: 0011 0101 G: 172: 1010 1100 B : 12: 0000 1100 **Pixel (3,5) :** R: 250: 1111 1010 G : 57: 0011 1001 B: 240: 1111 0000	**Pixel (3,2) :** R : 11: 0000 1011 G: 123: 0111 1011 B : 12: 0010 1100 **Pixel (3,6) :** R : 128: 1000 0000 G: 177: 1011 0001 B : 202: 1100 1010	**Pixel (3,3) :** R:226: 1110 0010 G : 68: 0100 0100 B : 27: 0001 1011 **Pixel (3,7) :** R : 37: 0010 0101 G : 72: 0100 1000 B: 167:1010 0111
Pixel (4,0) : R : 67: 0100 0011 G: 179: 1011 0011 B : 68: 0100 0100 **Pixel (4,4) :** R : 232:1110 1000 G : 53: 0011 0101 B : 145:1001 0001	**Pixel (4,1) :** R: 186: 1011 1010 G : 42: 0010 1010 B: 194: 1100 0010 **Pixel (4,5) :** R: 228: 1110 0100 G : 35: 0010 0011 B: 246: 1111 0110	**Pixel (4,2) :** R : 148: 1001 0100 G: 253: 1111 1101 B : 198: 1100 0110 **Pixel (4,6) :** R : 193: 1100 0001 G: 242: 1111 0010 B : 25: 0001 1001	**Pixel (4,3) :** R:138: 1000 1010 G:229: 1110 0101 B:237: 1110 1101 **Pixel (4,7) :** R : 18: 0001 0010 G : 53: 0011 0101 B:148: 1001 0100

Pixel (5,0) :	**Pixel (5,1) :**	**Pixel (5,2) :**	**Pixel (5,3) :**
R: 137: 1000 1001	R: 165: 1010 0101	R : 200: 1100 1000	R : 28: 0001 1100
G: 221: 1101 1101	G: 249: 1111 1001	G : 28: 0001 1100	G:105: 0110 1001
B: 201: 1100 1001	B: 243: 1111 0011	B : 36: 0010 0100	B:155: 1001 1011
Pixel (5,4) :	**Pixel (5,5) :**	**Pixel (5,6) :**	**Pixel (5,7) :**
R: 116: 0111 0100	R : 46: 0010 1110	R : 28: 0001 1100	R:130: 1000 0010
G: 179: 1011 0011	G : 95: 0101 1111	G : 70: 0100 0110	G:158: 1001 1110
B : 50: 0011 0010	B : 43: 0010 1011	B : 88: 0101 1000	B:225: 1110 0001
Pixel (6,0) :	**Pixel (6,1) :**	**Pixel (6,2) :**	**Pixel (6,3) :**
R :130: 1000 0010	R : 1: 0000 0001	R : 176: 1011 0000	R:131: 1000 0011
G: 200: 1100 1000	G : 71: 0100 0111	G: 246: 1111 0110	G:201: 1100 1001
B : 29: 0001 1101	B: 142: 1000 1110	B : 75: 0100 1011	B : 37: 0010 0101
Pixel (6,4) :	**Pixel (6,5) :**	**Pixel (6,6) :**	**Pixel (6,7) :**
R : 9: 0000 1001	R: 117: 0111 0101	R : 43: 0010 1011	R : 64: 0100 0000
G : 72: 0100 1000	G: 173: 1010 1101	G : 85: 0101 0101	G:106: 0110 1010
B: 213: 1101 0101	B: 100: 0110 0100	B : 75: 0100 1011	B:124: 0111 1100
Pixel (7,0) :	**Pixel (7,1) :**	**Pixel (7,2) :**	**Pixel (7,3) :**
R : 82: 0101 0010	R: 212: 1101 0100	R : 215: 1101 0111	R:106: 0110 1010
G: 145: 1001 0001	G : 19: 0001 0011	G : 22: 0001 0110	G:169: 1010 1001
B : 16: 0001 0000	B: 146: 1001 0010	B : 135: 1000 0111	B : 26: 0001 1010
Pixel (7,4) :	**Pixel 7,5) :**	**Pixel (7,6) :**	**Pixel (7,7) :**
R : 19: 0001 0011	R: 177: 1011 0001	R : 12: 0000 1100	R:145: 1001 0001
G : 82: 0101 0010	G: 233: 1110 1001	G : 61: 0011 1101	G:187: 1011 1011
B: 223: 1101 1111	B: 160: 1010 0000	B : 30: 0001 1110	B:177: 1011 0001

Determining Superincreasing:

w = 1 + 3 + 6 + 11 + 32 + 87 + 141 + 354 = 635
q = 881, r = 31
The formula for obtaining the public key results is:
t = w * r mod q
w1 = 1 * 31 mod 881 = 31
w2 = 3 * 31 mod 881 = 93
w3 = 6 * 31 mod 881 = 186
w4 = 11 * 31 mod 881 = 341
w5 = 32 * 31 mod 881 = 111
w6 = 87 * 31 mod 881 = 54
w7 = 141 * 31 mod 881 = 847
w8 = 354 * 31 mod 881 = 402

so that the public key for the Hellman encryption is obtained as follows:

{31, 93, 186, 341, 111, 54, 847, 402}

Next, each block is multiplied by each p element, so as to get the ciphertext as follows:

Formula : **x * ti =** $\sum \boldsymbol{y}$

Pixels (0,0) :
Red = (0*31) + (0*93) + (0*186) + (1*341) + (1*111) + (0*54) + (1*847) + (0*402) = 341 + 111 +847 = **1.299**
Green = (1*31) + (0*93) + (1*186) + (0*341) + (1*111) + (1*54) + (0*847) + (1*402) = 31 + 186 +111 + 54 + 402 = **784**
Blue = (1*31) + (1*93) + (0*186) + (0*341) + (1*111) + (1*54) + (1*847) + (0*402) = 31 + 93 + 111 + 54 + 847 = **1.136**

Pixels (0,1) :
Red = (1*31) + (1*93) + (0*186) + (0*341) + (1*111) + (0*54) + (0*847) + (1*402) = 31 + 93 + 111 + 402 = **637**
Green = (0*31) + (1*93) + (0*186) + (1*341) + (1*111) + (1*54) + (0*847) + (0*402) = 93 + 341 +111 + 54 = **599**
Blue = (1*31) + (0*93) + (0*186) + (0*341) + (1*111) + (0*54) + (1*847) + (1*402) = 31 + 111 +847 + 402= **1,391**

Pixel (0,2) :
Red = (0*31) + (0*93) + (0*186) + (0*341) + (1*111) + (1*54) + (1*847) + (1*402) = 111 + 54 + 847 + 402 = **1,414**
Green = (1*31) + (0*93) + (0*186) + (1*341) + (1*111) + (0*54) + (1*847) + (1*402) = 31 + 341 +111 + 847 + 402 = **1, 732**
Blue = (1*31) + (1*93) + (1*186) + (1*341) + (0*111) + (1*54) + (0*847) + (0*402) = 31 + 93 + 186 + 341 + 54 = **705**

Pixel (0,3) :

Red = (1*31) + (0*93) + (0*186) + (1*341) + (0*111) + (0*54) + (0*847) + (1*402) = 31 + 341 + 402 = **774**
Green = (0*31) + (0*93) + (0*186) + (1*341) + (0*111) + (1*54) + (1*847) + (0*402) = 341 + 54 +847 = **1, 242**
Blue = (1*31) + (0*93) + (1*186) + (0*341) + (0*111) + (1*54) + (1*847) + (1*402) = 31 + 186 + 54 + 847 + 402 = **1, 520**

Pixel (0,4) :
Red = (0*31) + (0*93) + (0*186) + (1*341) + (1*111) + (1*54) + (1*847) + (0*402) = 341 + 111 + 54 + 847 = **1, 353**
Green = (1*31) + (0*93) + (0*186) + (0*341) + (1*111) + (1*54) + (1*847) + (0*402) = 31 + 111 +54 + 847 = **1, 043**
Blue = (1*31) + (0*93) + (0*186) + (0*341) + (0*111) + (0*54) + (0*847) + (1*402) = 31 + 402 = **433**

Pixel (0,5) :
Red = (0*31) + (1*93) + (0*186) + (0*341) + (0*111) + (1*54) + (0*847) + (1*402) = 93 + 54 + 402 = **549**
Green = (1*31) + (0*93) + (1*186) + (0*341) + (0*111) + (1*54) + (1*847) + (1*402) = 31 + 186 +54 + 847 + 402 = **1, 520**
Blue = (0*31) + (0*93) + (0*186) + (0*341) + (1*111) + (0*54) + (1*847) + (0*402) = 111 + 847 = **958**

Pixel (0,6) :
Red = (0*31) + (1*93) + (1*186) + (1*341) + (0*111) + (1*54) + (1*847) + (0*402) = 93 + 186 + 341 + 54 + 847 = **1, 521**
Green = (1*31) + (1*93) + (0*186) + (0*341) + (1*111) + (0*54) + (1*847) + (0*402) = 31 + 93 +111 + 847 = **1, 082**
Blue = (0*31) + (1*93) + (1*186) + (1*341) + (1*111) + (0*54) + (1*847) + (0*402) = 93 + 186 +341 + 111 + 847 = **1, 578**

Pixel (0,7) :
Red = (1*31) + (0*93) + (0*186) + (0*341) + (0*111) + (0*54) + (0*847) + (0*402) = 31 = **31**
Green = (1*31) + (1*93) + (0*186) + (0*341) + (1*111) + (1*54) + (0*847) + (1*402) = 31 + 93 +111 + 54 + 402 = **691**
Blue = (1*31) + (0*93) + (1*186) + (1*341) + (0*111) + (1*54) + (0*847) + (1*402) = 31 + 186 +341 + 54 + 402 = **1.014**

Then proceed to pixel (7,7) so that the results of the ciphertext in the Merkle Hellman encryption process are known as follows:

Pixels (0,0) : R : **1, 299** G : **784** B : **1, 136**	**Pixel (0,1) :** R : **637** G : **599** B : **1, 391**	**Pixel (0,2) :** R : **1, 414** G : **1, 732** B : **705**	**Pixel (0,3) :** R : **774** G : **1, 242** B : **1, 520**
Pixel (0,4) : R : **1, 353** G : **1, 043** B : **433**	**Pixel (0,5) :** R : **549** G : **1, 520** B : **958**	**Pixel (0,6) :** R : **1, 521** G : **1, 082** B : **1, 578**	**Pixel (0,7) :** R : **31** G : **691** B : **1, 014**
Pixel (1,0) : R : 240 G : 1, 570 B : 1, 082	**Pixel (1,1) :** R : 1, 414 G : 1, 064 B : 867	**Pixel (1,2) :** R : 426 G : 186 B : 1, 133	**Pixel (1,3) :** R : 1, 291 G : 475 B : 1, 219
Pixel (1,4) : R : 723 G : 351 B : 753	**Pixel (1,5) :** R : 642 G : 31 B : 762	**Pixel (1,6) :** R : 1, 639 G : 669 B : 1, 391	**Pixel (1,7) :** R : 1, 879 G : 642 B : 527
Pixel (2,0) : R : 1, 861 G : 1,051 B : 1, 639	**Pixel (2,1) :** R : 599 G : 1, 724 B : 1, 353	**Pixel (2,2) :** R : 580 G : 836 B : 669	**Pixel (2,3) :** R : 1, 076 G : 1, 900 B : 1, 520
Pixel (2,4) : R : 1, 373 G : 351 B : 606	**Pixel (2,5) :** R : 1, 373 G : 1, 353 B : 1, 577	**Pixel (2,6) :** R : 1, 335 G : 537 B : 885	**Pixel (2,7) :** R : 1, 043 G : 691 B : 847
Pixel (3,0) : R : 1, 360 G : 544 B : 691	**Pixel (3,1) :** R : 983 G : 382 B : 165	**Pixel (3,2) :** R : 1, 360 G : 1, 980 B : 165	**Pixel (3,3) :** R : 1, 157 G : 147 B : 1, 701
Pixel (3,4) : R : 1, 094 G : 774 B : 1, 879	**Pixel (3,5) :** R : 1, 609 G : 1, 040 B : 651	**Pixel (3,6) :** R : 31 G : 960 B : 1, 082	**Pixel (3,7) :** R : 642 G : 204 B : 1,520

Pixel (4,0) :	**Pixel (4,1) :**	**Pixel (4,2) :**	**Pixel (4,3) :**
R : 1, 342	R : 1, 516	R : 426	R : 989
G : 1, 807	G : 1, 144	G : 1, 218	G : 766
B : 147	B : 971	B : 1, 025	B : 877
Pixel (4,4) :	**Pixel (4,5) :**	**Pixel (4,6) :**	**Pixel (4,7) :**
R : 421	R : 364	R : 526	R : 1, 188
G : 983	G : 1, 435	G : 1, 498	G : 983
B : 774	B : 1,552	B : 854	B : 426
Pixel (5,0) :	**Pixel (5,1) :**	**Pixel (5,2) :**	**Pixel (5,3) :**
R : 544	R : 673	R : 235	R : 506
G : 1, 032	G : 1, 164	G : 506	G : 792
B : 637	B : 1, 900	B : 240	B : 1, 732
Pixel (5,4) :	**Pixel (5,5) :**	**Pixel (5,6) :**	**Pixel (5,7) :**
R : 674	R : 1, 198	R : 506	R : 878
G : 1, 807	G : 1, 848	G : 994	G : 1, 384
B : 1, 374	B : 1, 546	B : 545	B : 712
Pixel (6,0) :	**Pixel (6,1) :**	**Pixel (6,2) :**	**Pixel (6,3) :**
R : 878	R : 402	R : 558	R : 1, 280
G : 235	G : 1, 396	G : 1, 552	G : 637
B : 908	B : 1, 043	B : 1, 453	B : 642
Pixel (6,4) :	**Pixel (6,5) :**	**Pixel (6,6) :**	**Pixel (6,7) :**
R : 513	R : 1,076	R : 1, 546	R : 93
G : 204	G : 784	G : 890	G : 1, 237
B : 921	B : 333	B : 1, 453	B : 785
Pixel (7,0) :	**Pixel (7,1) :**	**Pixels (7,2) :**	**Pixels (7,3) :**
R : 1, 281	R : 519	R : 1, 768	R : 877
G : 774	G : 1, 590	G : 1, 242	G : 730
B : 341	B : 1, 219	B : 1, 334	B :1, 299
Pixel (7,4) :	**Pixel 7,5) :**	**Pixels (7,6) :**	**Pixels (7,7) :**
R : 1, 590	R : 960	R : 165	R : 774
G : 1, 281	G : 823	G : 1, 094	G : 1.918
B : 1, 879	B : 217	B : 1, 353	B : 960

3.3. Calculation of Merkle Hellman's Decryption

For the process of decrypting the Merkle Hellman Algorithm, plaintext is used as follows:

Pixels (0,0) :	**Pixel (0,1) :**	**Pixel (0,2) :**	**Pixel (0,3) :**
R : **1, 299**	R : **637**	R : **1, 414**	R : **774**
G : **784**	G : **599**	G : **1, 732**	G : **1, 242**
B : **1, 136**	B : **1, 391**	B : **705**	B : **1, 520**
Pixel (0,4) :	**Pixel (0,5) :**	**Pixel (0,6) :**	**Pixel (0,7) :**
R : **1, 353**	R : **549**	R : **1, 521**	R : **31**
G : **1, 043**	G : **1, 520**	G : **1, 082**	G : **691**
B : **433**	B : **958**	B : **1, 578**	B : **1, 014**
Pixel (1,0) :	**Pixel (1,1) :**	**Pixel (1,2) :**	**Pixel (1,3) :**
R : 240	R : 1, 414	R : 426	R : 1, 291
G : 1, 570	G : 1, 064	G : 186	G : 475
B : 1, 082	B : 867	B : 1, 133	B : 1, 219
Pixel (1,4) :	**Pixel (1,5) :**	**Pixel (1,6) :**	**Pixel (1,7) :**
R : 723	R : 642	R : 1, 639	R : 1, 879
G : 351	G : 31	G : 669	G : 642
B : 753	B : 762	B : 1, 391	B : 527

Pixel (2,0) :	**Pixel (2,1) :**	**Pixel (2,2) :**	**Pixel (2,3) :**
R : 1, 861	R : 599	R : 580	R : 1, 076
G : 1,051	G : 1, 724	G : 836	G : 1, 900
B : 1, 639	B : 1, 353	B : 669	B : 1, 520
Pixel (2,4) :	**Pixel (2,5) :**	**Pixel (2,6) :**	**Pixel (2,7) :**
R : 1, 373	R : 1, 373	R : 1, 335	R : 1, 043
G : 351	G : 1, 353	G : 537	G : 691
B : 606	B : 1, 577	B : 885	B : 847
Pixel (3,0) :	**Pixel (3,1) :**	**Pixel (3,2) :**	**Pixel (3,3) :**
R : 1, 360	R : 983	R : 1, 360	R : 1, 157
G : 544	G : 382	G : 1, 980	G : 147
B : 691	B : 165	B : 165	B : 1, 701
Pixel (3,4) :	**Pixel (3,5) :**	**Pixel (3,6) :**	**Pixel (3,7) :**
R : 1, 094	R : 1, 609	R : 31	R : 642
G : 774	G : 1, 040	G : 960	G : 204
B : 1, 879	B : 651	B : 1, 082	B : 1,520
Pixel (4,0) :	**Pixel (4,1) :**	**Pixel (4,2) :**	**Pixel (4,3) :**
R : 1, 342	R : 1, 516	R : 426	R : 989
G : 1, 807	G : 1, 144	G : 1, 218	G : 766
B : 147	B : 971	B : 1, 025	B : 877
Pixel (4,4) :	**Pixel (4,5) :**	**Pixel (4,6) :**	**Pixel (4,7) :**
R : 421	R : 364	R : 526	R : 1, 188
G : 983	G : 1, 435	G : 1, 498	G : 983
B : 774	B : 1,552	B : 854	B : 426
Pixel (5,0) :	**Pixel (5,1) :**	**Pixel (5,2) :**	**Pixel (5,3) :**
R : 544	R : 673	R : 235	R : 506
G : 1, 032	G : 1, 164	G : 506	G : 792
B : 637	B : 1, 900	B : 240	B : 1, 732
Pixel (5,4) :	**Pixel (5,5) :**	**Pixel (5,6) :**	**Pixel (5,7) :**
R : 674	R : 1, 198	R : 506	R : 878
G : 1, 807	G : 1, 848	G : 994	G : 1, 384
B : 1, 374	B : 1, 546	B : 545	B : 712
Pixel (6,0) :	**Pixel (6,1) :**	**Pixel (6,2) :**	**Pixel (6,3) :**
R : 878	R : 402	R : 558	R : 1, 280
G : 235	G : 1, 396	G : 1, 552	G : 637
B : 908	B : 1, 043	B : 1, 453	B : 642
Pixel (6,4) :	**Pixel (6,5) :**	**Pixel (6,6) :**	**Pixel (6,7) :**
R : 513	R : 1,076	R : 1, 546	R : 93
G : 204	G : 784	G : 890	G : 1, 237
B : 921	B : 333	B : 1, 453	B : 785
Pixel (7,0) :	**Pixel (7,1) :**	**Pixels (7,2) :**	**Pixels (7,3) :**
R : 1, 281	R : 519	R : 1, 768	R : 877
G : 774	G : 1, 590	G : 1, 242	G : 730
B : 341	B : 1, 219	B : 1, 334	B :1, 299
Pixel (7,4) :	**Pixels 7.5) :**	**Pixels (7,6) :**	**Pixels (7,7) :**
R : 1, 590	R : 960	R : 165	R : 774
G : 1, 281	G : 823	G : 1, 094	G : 1.918
B : 1, 879	B : 217	B : 1, 353	B : 960

Then, the first decryption step will be carried out on the Merkle Hellman algorithm with the formula n. $n^{-1} \equiv 1 \pmod{m}$ then:

n^{-1} can be obtained from: $n^{-1} = (1 + 881\ k) / 31$, try k = 0,1,2,....., an integer is obtained, namely $n^{-1} = 540$. Next, decrypt the ciphertext:

Formula : **z * y mod q**

Pixels (0,0)
Red : 1, 299*540 mod 881 = 184 = 141 + 32 + 11 obtained: **0001 1010**
Green : 784*540 mod 881 = 480 = 354 + 87 + 32 + 6 + 1 obtained: **1010 1101**
Blue : 1,136*540 mod 881 = 264 = 141 + 87 + 32 + 3 + 1 gain: **1100 1110**

Pixels (0.1)
Red : 637*540 mod 881 = 390 = 354 + 32 + 3 + 1 obtained: **1100 1001**
Green : 599*540 mod 881 = 133 = 87 + 32 + 11 + 3 obtained: **0101 1100**
Blue : 1,391*540 mod 881 = 528 = 354 + 141 + 32 + 1 obtained: **1000 1011**

pixels (0,2)
Red : 1,414*540 mod 881 = 614 = 354 + 141 + 87 + 32 obtained: **0000 1111**
Green : 1,732*540 mod 881 = 539 = 354 + 141 + 32 + 11 + 1 obtained: **1001 1011**
Blue :705*540 mod 881 = 108 = 87 + 11 + 6 + 3 + 1 obtained: **1111 0100**

pixels (0,3)
Red : 774*540 mod 881 = 366 = 354 + 11 + 1 obtained: **1001 0001**
Green : 1, 242*540 mod 881 = 239 = 141 + 87 + 11 obtained: **0001 0110**
Blue : 1,520*540 mod 881 = 589 = 354 + 141 + 87 + 6 + 1 obtained: **1010 0111**

pixels (0,4)
Red : 1, 353*540 mod 881 = 271 = 141 + 87 + 32 + 11 obtained: **0001 1110**
Green : 1, 043*540 mod 881 = 261 = 141 + 87 + 32 + 1 obtained: **1000 1110**
Blue : 433*540 mod 881 = 355 = 354 + 1 obtained: **1000 0001**

pixels (0.5)
Red : 549*540 mod 881 = 444 = 354 + 87 + 3 obtained: **0100 0101**
Green : 1, 520*540 mod 881 = 589 = 354 + 141 + 87 + 6 + 1 obtained: **1010 0111**
Blue : 958*540 mod 881 = 173 = 141 + 32 obtained: **0000 1010**

pixels (0.6)
Red : 1, 521*540 mod 881 = 248 = 141 + 87 + 11 + 6 + 3 obtained: **0111 0110**
Green : 1, 082*540 mod 881 = 177 = 141 + 32 + 3 + 1 obtained: **1100 1010**
Blue : 1, 578*540 mod 881 = 193 = 141 + 32 + 11 + 6 + 3 obtained: **0111 1010**

Pixels (0.7)
Red : 31*540 mod 881 = 1 = 1 obtained: **1000 0000**
Green : 691*540 mod 881 = 477 = 354 + 87 + 32 + 3 + 1 obtained: **1100 1101**
Blue : 1,014*540 mod 881 = 459 = 354 + 87 + 11 + 6 + 1 obtained: **1011 0101**

Then proceed to pixel (7,7) so that the results of the ciphertext in the Merkle Hellman Decryption process are known as follows:

Pixels (0,0) :	**Pixel (0,1) :**	**Pixel (0,2) :**	**Pixel (0,3) :**
R: **0001 1010: 26**	R:**1100 1001: 201**	R:**0000 1111: 15**	R:**1001 0001: 145**
G:**1010 1101: 173**	G : **0101 1100: 92**	G:**1001 1011: 155**	G : **0001 0110: 22**
B : **1100 1110: 206**	B:**1000 1011: 139**	B:**1111 0100: 244**	B:**1010 0111: 167**
Pixel (0,4) :	**Pixel (0,5) :**	**Pixel (0,6) :**	**Pixel (0,7) :**
R : **0001 1110: 30**	R : **0100 0101: 69**	R:**0111 0110: 118**	R:**1000 0000: 128**
G : **1000 1110: 142**	G:**1010 0111: 167**	G:**1100 1010: 202**	G:**1100 1101: 205**
B : **1000 0001: 129**	B : **0000 1010: 10**	B:**0111 1010: 122**	B:**1011 0101: 181**
Pixel (1,0) :	**Pixel (1,1) :**	**Pixel (1,2) :**	**Pixel (1,3) :**
R : 0010 0100: 36	R : 0000 1111: 15	R:1001 0100: 148	R:0110 1110: 110
G : 1011 1110: 190	G:1010 0010: 162	G : 0010 0000: 32	G:1110 1100: 236
B : 1100 1010: 202	B:1101 0001: 209	B:0111 1001: 121	B:1001 0010: 146
Pixel (1,4) :	**Pixel (1,5) :**	**Pixel (1,6) :**	**Pixel (1,7) :**
R : 1011 1100: 188	R : 0010 0101: 37	R:0110 1011: 107	R:1101 1111: 223
G : 0010 1100: 44	G:1000 0000: 128	G:1011 1000: 184	G : 0010 0101: 37
B : 1100 1010: 45	B:1111 1000: 148	B:1000 1011: 139	B : 0011 0000: 48
Pixel (2,0) :	**Pixel (2,1) :**	**Pixel (2,2) :**	**Pixel (2,3) :**
R : 1011 0111: 183	R : 0101 1100: 92	R:1100 0101: 197	R:0111 0101: 117
G : 0100 1010: 74	G:1110 1111: 239	G : 0101 0001: 81	G:1111 0011: 243
B : 0110 1011: 107	B : 0001 1110: 30	B:1011 1000: 184	B:1010 0111: 167

Pixel (2,4) :	**Pixel (2,5) :**	**Pixel (2,6) :**	**Pixel (2,7) :**
R : 1100 0011: 195	R:1100 0011: 195	R : 0101 0110: 86	R:1000 1110: 142
G : 0010 1100: 44	G : 0001 1110: 30	G:1001 1100: 156	G:1100 1101: 205
B : 0100 1001: 73	B:1010 1011: 171	B:1001 1001: 153	B : 0000 0010: 2
Pixel (3,0) :	**Pixel (3,1) :**	**Pixel (3,2) :**	**Pixel (3,3) :**
R : 0000 1011: 11	R : 0011 0101: 53	R : 0000 1011: 11	R:1110 0010: 226
G : 1000 1001: 137	G:1010 1100: 172	G:0111 1011: 123	G : 0100 0100: 68
B : 1100 1101: 205	B : 0000 1100: 12	B : 0010 1100: 12	B : 0001 1011: 27
Pixel (3,4) :	**Pixel (3,5) :**	**Pixel (3,6) :**	**Pixel (3,7) :**
R : 0011 1101: 61	R:1111 1010: 250	R:1000 0000: 128	R : 0010 0101: 37
G : 1001 0001: 145	G : 0011 1001: 57	G:1011 0001: 177	G : 0100 1000: 72
B : 1101 1111: 223	B:1111 0000: 240	B:1100 1010: 202	B:1010 0111: 167
Pixel (4,0) :	**Pixel (4,1) :**	**Pixel (4,2) :**	**Pixel (4,3) :**
R : 0100 0011: 67	R:1011 1010: 186	R:1001 0100: 148	R:1000 1010: 138
G : 1011 0011: 179	G : 0010 1010: 42	G:1111 1101: 253	G:1110 0101: 229
B : 0100 0100; 68	B:1100 0010: 194	B:1100 0110: 198	B:1110 1101: 237
Pixel (4,4) :	**Pixel (4,5) :**	**Pixel (4,6) :**	**Pixel (4,7) :**
R : 1110 1000: 232	R:1110 0100: 228	R:1100 0001: 193	R : 0001 0010:18
G : 0011 0101: 53	G : 0010 0011: 35	G:1111 0010: 242	G : 0011 0101: 53
B : 1001 0001: 145	B:1111 0110: 246	B : 0001 1001: 25	B:1001 0100: 148
Pixel (5,0) :	**Pixel (5,1) :**	**Pixel (5,2) :**	**Pixel (5,3) :**
R : 1000 1001: 137	R:1010 0101: 165	R:1100 1000: 200	R : 0001 1100: 28
G : 1101 1101: 221	G:1111 1001: 249	G : 0001 1100: 28	G:0110 1001: 105
B : 1100 1001: 201	B:1111 0011: 243	B : 0010 0100: 36	B:1001 1011: 155
Pixel (5,4) :	**Pixel (5,5) :**	**Pixel (5,6) :**	**Pixel (5,7) :**
R : 0111 0100: 116	R : 0010 1110: 46	R : 0001 1100: 28	R:1000 0010: 130
G : 1011 0011: 179	G : 0101 1111: 95	G : 0100 0110: 70	G:1001 1110: 158
B : 0011 0010: 50	B : 0010 1011: 43	B : 0101 1000: 88	B:1110 0001: 225
Pixel (6,0) :	**Pixel (6,1) :**	**Pixel (6,2) :**	**Pixel (6,3) :**
R : 1000 0010: 130	R : 0000 0001: 1	R:1011 0000: 176	R:1000 0011: 131
G : 1100 1000: 200	G : 0100 0111: 71	G:1111 0110: 246	G:1100 1001: 201
B : 0001 1101: 29	B:1000 1110: 142	B : 0100 1011: 75	B : 0010 0101: 37
Pixel (6,4) :	**Pixel (6,5) :**	**Pixel (6,6) :**	**Pixel (6,7) :**
R : 0000 1001: 9	R:0111 0101: 117	R : 0010 1011: 43	R : 0100 0000: 64
G : 0100 1000: 72	G:1010 1101: 173	G : 0101 0101: 85	G:0110 1010: 106
B : 1101 0101: 213	B:0110 0100: 100	B : 0100 1011: 75	B:0111 1100: 124
Pixel (7,0) :	**Pixel (7,1) :**	**Pixels (7,2) :**	**Pixels (7,3) :**
R : 0101 0010: 82	R:1101 0100: 212	R:1101 0111:215	R:0110 1010:106
G : 1001 0001: 145	G : 0001 0011: 19	G : 0001 0110: 22	G:1010 1001: 169
B : 0001 0000: 16	B:1001 0010: 146	B:1000 0111:135	B : 0001 1010: 26
Pixel (7,4) :	**Pixel 7,5) :**	**Pixels (7,6) :**	**Pixels (7,7) :**
R : 0001 0011: 19	R:1011 0001:177	R : 0000 1100: 12	R:1001 0001:145
G : 0101 0010: 82	G:1110 1001:233	G : 0011 1101: 61	G:1011 1011: 187
B : 1101 1111: 223	B:1010 0000:160	B : 0001 1110: 30	B:1011 0001:177

3.4. Calculation of Affine Cipher Decryption

For the process of decrypting this affine cipher algorithm, plaintext is used as follows:

(x,y)	0	1	2	3	4	5	6	7
0	**R : 26** **G : 173** **B : 206**	**R : 201** **G : 92** **B: 139**	**R : 15** **G : 155** **B : 244**	**R : 145** **G : 22** **B : 167**	**R : 30** **G: 142** **B : 129**	**R : 69** **G : 167** **B : 10**	**R : 118** **G: 202** **B : 122**	**R : 128** **G : 205** **B : 181**

1	R : 36 G : 190 B : 202	R : 15 G : 162 B : 209	R : 148 G : 32 B : 121	R : 110 G : 236 B : 146	R : 188 G : 44 B : 45	R : 37 G : 128 B : 148	R : 107 G: 184 B : 139	R : 223 G : 37 B : 48
2	R : 183 G : 74 B : 107	R : 92 G : 239 B : 30	R : 197 G : 81 B : 184	R : 117 G : 243 B : 167	R : 195 G : 44 B : 73	R : 195 G : 30 B : 171	R : 86 G: 156 B : 153	R : 142 G : 205 B : 2
3	R : 11 G : 137 B : 205	R : 53 G : 172 B : 12	R : 11 G : 123 B : 12	R : 226 G : 68 B : 27	R : 61 G: 145 B : 223	R : 250 G : 57 B : 240	R : 128 G: 177 B : 202	R : 37 G : 72 B : 167
4	R : 67 G : 179 B : 68	R : 186 G : 42 B : 194	R : 148 G : 253 B : 198	R : 138 G : 229 B : 237	R : 232 G : 53 B : 145	R : 228 G : 35 B : 246	R : 193 G: 242 B : 25	R : 18 G : 53 B : 148
5	R : 137 G : 221 B : 201	R : 165 G : 249 B : 243	R : 200 G : 28 B : 36	R : 28 G : 105 B : 155	R : 116 G: 179 B : 50	R : 46 G : 95 B : 43	R : 28 G : 70 B : 88	R : 130 G : 158 B :225
6	R :130 G : 200 B : 29	R : 1 G : 71 B: 142	R : 176 G : 246 B : 75	R : 131 G : 201 B : 37	R : 9 G : 72 B : 213	R : 117 G : 173 B :100	R : 43 G : 85 B : 75	R : 64 G : 106 B : 124
7	R : 82 G : 145 B : 16	R : 212 G : 19 B : 146	R : 215 G : 22 B: 135	R : 106 G : 169 B : 26	R : 19 G : 82 B : 223	R : 177 G : 233 B : 160	R : 12 G : 61 B : 30	R : 145 G : 187 B : 177

The plaintext decryption is calculated by the formula:

$$D(x) = a(x - b) \bmod m$$

For this reason, the image decryption process with the affine cipher algorithm is as follows:

Pixels (0,0)
Red : 183 . (26 - 10) = 2.928 mod 256 = **112**
Green : 183 . (173 - 10) = 29, 829 mod 256 = **133**
Blue : 183 . (206 - 10) = 35.868 mod 256 = **28**

Pixels (0.1)
Red : 183 . (201 - 10) = 34.953 mod 256 = **137**
Green : 183 . (92 - 10) = 15,006 mod 256 = **158**
Blue : 183 . (139 - 10) = 23,607 mod 256 = **55**

Pixel (0,2)
Red : 183 . (15 - 10) = 915 mod 256 = **147**
Green : 183 . (155 - 10) = 26, 535 mod 256 = **167**
Blue : 183 . (244 - 10) = 42,822 mod 256 = **70**

Pixel (0,3)
Red : 183 . (145 - 10) = 24,705 mod 256 = **129**
Green : 183 . (22 - 10) = 2,196 mod 256 = **148**
Blue : 183 . (167 - 10) = 28,731 mod 256 = **59**
Pixel (0,4)
Red : 183 . (30 - 10) = 3,660 mod 256 = **76**
Green : 183 . (142 - 10) = 24,156 mod 256 = **92**
Blue : 183 . (129 - 10) = 21, 777 mod 256 = **17**

Pixel (0,5)
Red : 183 . (69 - 10) = 10,797 mod 256 = **45**
Green : 183 . (167 - 10) = 28, 731 mod 256 = **59**
Blue : 183 . (10 - 10) = 0 mod 256 = **0**

Pixel (0,6)
Red : 183 . (118 - 10) = 19,764 mod 256 = **52**
Green : 183 . (202 - 10) = 35,136 mod 256 = **64**
Blue : 183 . (122 - 10) = 20, 496 mod 256 = **16**

Pixel (0,7)

Red : 183 . (128 - 10) = 21, 594 mod 256 = **90**
Green : 183 . (205 - 10) = 35.685 mod 256 = **101**
Blue : 183 . (181 - 10) = 31.293 mod 256 = **61**

Then proceed to pixel (7,7) so that the results of the ciphertext return to the Affine Cipher Decryption process are known as follows:

(x,y)	0	1	2	3	4	5	6	7
0	R : **112** G : **133** B : **28**	R : **137** G : **158** B : **55**	R : **147** G : **167** B : **70**	R : **129** G : **148** B : **59**	R : **76** G : **92** B : **17**	R : **45** G : **59** B : **0**	R : **52** G : **64** B : **16**	R : **90** G : **101** B : **61**
1	R : 150 G : 172 B : 64	R : 147 G : 163 B : 65	R : 166 G : 186 B : 89	R : 124 G : 142 B : 56	R : 62 G : 78 B : 5	R : 77 G : 90 B : 34	R : 87 G : 98 B : 55	R : 67 G : 77 B : 42
2	R : 171 G : 192 B : 87	R : 158 G : 179 B : 76	R : 173 G : 193 B : 98	R : 125 G : 143 B : 59	R : 63 G : 78 B : 0	R : 63 G : 76 B : 23	R : 84 G : 94 B : 57	R : 92 G : 101 B : 72
3	R : 183 G : 201 B : 101	R : 189 G : 206 B : 110	R : 183 G : 199 B : 110	R : 104 G : 118 B : 39	R : 117 G: 129 B : 67	R : 144 G: 153 B : 106	R : 90 G : 97 B : 64	R : 77 G : 82 B : 59
4	R : 191 G : 207 B : 118	R : 208 G : 224 B : 136	R : 166 G : 181 B : 100	R : 128 G : 141 B : 69	R : 178 G: 189 B : 129	R : 214 G: 223 B : 180	R : 209 G: 216 B : 185	R : 184 G : 189 B : 166
5	R : 201 G : 213 B : 137	R : 205 G : 217 B : 143	R : 210 G : 222 B : 150	R : 222 G : 233 B : 167	R : 198 G: 207 B : 152	R : 188 G: 195 B : 151	R : 222 G: 228 B : 194	R : 200 G : 204 B :177
6	R : 200 G : 210 B : 149	R : 145 G : 155 B : 92	R : 170 G : 180 B : 119	R : 127 G : 137 B : 77	R : 73 G : 82 B : 29	R : 125 G: 133 B : 86	R : 151 G: 157 B : 119	R : 154 G : 160 B : 126
7	R : 120 G : 129 B : 74	R : 102 G : 111 B : 56	R : 139 G : 148 B : 91	R : 160 G : 169 B : 112	R : 111 G: 120 B : 67	R : 97 G: 105 B : 58	R : 110 G: 117 B : 76	R : 129 G : 135 B : 97

4. Discussion

The stages that the authors do is the design of the software interface. So in this discussion, we will discuss some of the views of the existing menus using the affine cipher and Merkle Hellman algorithms, as follows:

4.1. Main Form

The main form is the display form which acts as an opening in the application process.

Figure 2: Main Form

4.2. Encryption Form

The form of this encryption form can be seen to retrieve the image by clicking the Browse button (search) and the image and file name will appear, then enter the key for affine cipher encryption, then click the process button on the affine cipher, then the image will move the process to Merkle Hellman and will enter the key for the Merkle Hellman encryption and the encryption results are successful, then click save.

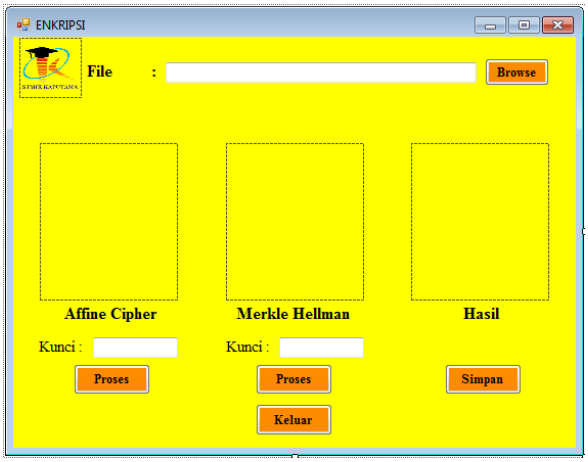

Figure 3: Encryption Form

4.3. Decryption Form

In this decryption form it can be seen that to retrieve the image by clicking the Browse button (search) and an image will appear along with the name of the file that was encrypted earlier, then enter the key for the Merkle Hellman decryption, then click the process button on Merkle Hellman, then the image moves to the process affine cipher and the key will be entered for the affine cipher decryption and the decryption results return to the original image, then click save.

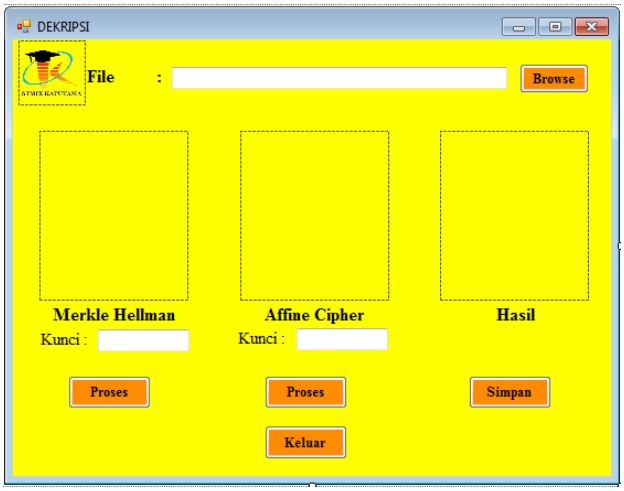

Figure 1 Decryption Form

5. Conclusion

With this digital image encryption and decryption application system, the authors can draw the following conclusions:

1. Affine Cipher is a classic encryption method that uses a linear affinity function to convert each character in an image into another character. This method provides a higher level of security than simple replacement methods such as the Caesar Cipher. By applying Affine Cipher to digital images, information in images can be scrambled and difficult for unauthorized parties to understand.
2. Merkle-Hellman is a public key cryptographic algorithm that is used to secure the key exchange process between sender and receiver. By applying Merkle-Hellman to the digital image security process, transmission and storage of encryption keys can be carried out safely, thereby minimizing the risk of information leakage.
3. By combining Affine Cipher and Merkle-Hellman, digital image security can be significantly improved. Affine Cipher provides a strong encryption method to protect information in images, while Merkle-Hellman provides secure key exchange. This combination ensures that the digital image remains protected from unauthorized access and disclosure of unwanted information.

References

[1] H. Mukhtar, *Cryptography for Data Security* . Yogyakarta, 2018.
[2] MS Soeb Arifin, "Modification Analysis of Classical Cryptographic Algorithms Using the Blum-Micali Generator Algorithm," vol. 6, pp. 136–147, 2022.
[3] Yusfrizal, "DESIGN OF CRYPTOGRAPHIC APPLICATIONS ON TEXT USING THE REVERSE CHIPER METHOD," vol. 3, no. 2, pp. 29–37, 2019.
[4] A. Rambe, "Modification of Affine Ciphers Methods in Classical Cryptography," *Ready Star* , no. m, pp. 256–261, 2019, [Online]. Available: https://ptki.ac.id/jurnal/index.php/readystar/article/view/64
[5] M. Simanjuntak, T. Pasaribu, and S. Rahmadilla, "Implementation of the Merkle Hellman Algorithm for Database Security," *MEANS (Media Inf. Anal. and Sist.* , vol. 4, no. 1, pp. 46–50 , 2019, doi: 10.54367/means.v4i1.327.
[6] Y. Antika, "Implementation of the Affine Cipher and Tripple Des Algorithms in Securing Image Files," *Maj. CORE SCIENCES* , vol. 14, no. 2, pp. 200–206, 2019.