PENGKODEAN TEKS MENGGUNAKAN MODIFIKASI ALGORITMA *ELECTRONIC CODE BOOK* DAN *MERKLE-HELLMAN KNAPSACK*

(*Text Encryption Using Modified Electronic Code Book and Merkle-Hellman Knapsack Algorithm*)

Innafajri Insyirah[1*)], Kiswara Agung Santoso[2)], Ahmad Kamsyakawuni[3)]

[1,2,3)]Jurusan Matematika, Fakultas MIPA, Universitas Jember
Jl. Kalimantan No. 37 Jember, 68121, Indonesia
innafajri@gmail.com[1)], kiswaras@gmail.com[2)], kamsyakawuni.fmipa@gmail.com[3)]
*)penulis korespondensi

Abstract. Data security needs to be maintained so that the messages sent to someone are not known by an unauthorized users. One of the data security techniques that can be applied is cryptography. Cryptography is a science and art to keep messages secure when messages are sent from one place to another. In this study, modification of Electronic Code Book and Merkle-Hellman Knapsack algorithms will be applied to secure messages. This research uses Matlab to create a program. The weaknesses of ECB and Merkle-Hellman Knapsack algorithms will be overcome by modifying key and XOR operations. The result of ciphertext is a random numbers which are difficult to understand by unauthorized users. In addition, the message pattern on the same plaintext no longer produces the same ciphertext. Ciphertext can be decrypted well even though the size is two times or longer than plaintext. This is proved by plaintext from the results of decryption is same as the original plaintext.

Keywords: Cryptography, Electronic Code Book, Merkle-Hellman Knapsack

1. Pendahuluan

Keamanan data perlu dijaga agar data tidak dapat dimengerti oleh orang lain kecuali pengirim dan penerima data [2]. Salah satu upaya pengamanan data yang dapat diterapkan adalah teknik kriptografi. Menurut Ariyus [1], kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat lain. Kriptografi berdasarkan kuncinya dibagi menjadi dua, yaitu algoritma simetri dan algoritma asimetri.

Salah satu algoritma dalam kriptografi kunci simetri adalah *Electronic Code Book* (ECB). Murdowo [6] meneliti algoritma ECB dengan memodifikasi jumlah karakter pada kunci. Modifikasi tersebut menghasilkan *ciphertext* yang lebih rahasia, namun blok *plaintext* dengan karakter yang sama menghasilkan *ciphertext* yang sama pula. Penelitian lain dilakukan oleh Widarma dkk [10] dengan menggabungkan *Vigenere Cipher* dan ECB untuk mengamankan data. Hasil dari penelitian tersebut dapat meningkatkan keamanan data karena kompleksitas dua algoritma lebih rumit daripada satu algoritma. Karo [4] menggabungkan *Affine Cipher* dan ECB untuk mengamankan pesan teks. Hasil *ciphertext* tidak dikonversi dalam bentuk heksadesimal, melainkan dikonversi sesuai kode ASCII

sehingga terdapat beberapa karakter yang tidak dapat ditampilkan oleh komputer.

Salah satu algoritma dalam kriptografi kunci asimetri adalah *Merkle-Hellman Knapsack*. Leman dan Rahman [5] mengaplikasikan *Merkle-Hellman Knapsack* dan menghasilkan *ciphertext* berupa angka. Namun, *plaintext* yang sama juga menghasilkan *ciphertext* yang sama. Penelitian lain dilakukan oleh Fadlan dan Hadriansa [3] dengan mengkombinasikan *Merkle-Hellman Knapsack* dan *Affine Cipher*. Implementasi kombinasi tersebut dapat digunakan untuk mengamankan sebuah data karena *ciphertext* hasil enkripsi dapat didekripsi menghasilkan *plaintext* semula. Sulaiman [9] melakukan penelitian menggunakan kombinasi XOR *Cipher* dan *Merkle-Hellman Knapsack*. Penelitian tersebut menghasilkan dua kali perlindungan sehingga kerahasiaan pesan digital lebih meningkat.

Berdasarkan penelitian-penelitian yang telah disebutkan, penulis akan menerapkan modifikasi algoritma *Electronic Code Book* dan *Merkle-Hellman Knapsack* untuk mengkodekan data teks. Penelitian ini dilakukan dengan membuat sebuah program menggunakan Matlab 2015. Algoritma ECB dan *Merkle-Hellman Knapsack* dipilih karena kedua algoritma tersebut sama-sama memanfaatkan nilai biner dalam proses penyandian pesan. Kelemahan algoritma ECB dan *Merkle-Hellman Knapsack* akan diatasi dengan cara memodifikasi kunci dan operasi XOR.

Kriptografi

Kriptografi berasal dari Bahasa Yunani, yaitu *crypto* (rahasia) dan *graphia* (tulisan). Kriptografi berdasarkan kuncinya dibagi menjadi dua, yaitu algoritma simetri dan algoritma asimetri. Algoritma simetri menggunakan kunci yang sama dalam proses enkripsi dan dekripsi, sedangkan algoritma asimetri menggunakan kunci yang berbeda.

Algoritma *Electronic Code Book*

Kriptografi dengan metode ECB adalah suatu teknik penyandian data yang menggunakan prinsip operasi logika XOR. Operasi XOR merupakan teknik penyandian dengan memanfaatkan nilai biner dari setiap karakter pesan dan kunci. Aturan operasi XOR dapat dilihat pada Tabel 1.

Tabel 1. Aturan operasi XOR

A	**B**	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Pada proses penyandian algoritma ECB, sebuah blok *plaintext* dipetakan secara individual menjadi blok *ciphertext* [7]. Contoh proses enkripsi dan dekripsi menggunakan algoritma ECB secara berturut-turut disajikan pada Tabel 2 dan Tabel 3.

Tabel 2. Proses enkripsi algoritma ECB

***Plaintext* (P)**	**Kunci (K)**	$P \oplus K$	**Pergeseran 1 bit ke kiri**
01001001 (I)	01111000 (x)	00110001	01100010 (b)
01001110 (N)	01111000 (x)	00110110	01101100 (l)
01001110 (N)	01111000 (x)	00110110	01101100 (l)
01000001 (A)	01111000 (x)	00111001	01110010 (r)

Tabel 3. Proses dekripsi algoritma ECB

Ciphertext	**Pergeseran 1 bit ke kanan (C)**	**Kunci (K)**	$C \oplus K$
01100010 (b)	00110001	01111000 (x)	01001001 (I)
01101100 (l)	00110110	01111000 (x)	01001110 (N)
01101100 (l)	00110110	01111000 (x)	01001110 (N)
01110010 (r)	00111001	01111000 (x)	01000001 (A)

Algoritma *Merkle-Hellman Knapsack*

Algoritma *Merkle-Hellman Knapsack* adalah algoritma yang menggeneralisasi enkripsi kunci asimetri yang didasarkan pada permasalahan *knapsack*. Ide algoritma ini adalah mengodekan pesan sebagai solusi untuk mengurutkan masalah *knapsack*. Mekanisme pengerjaan *Merkle-Hellman Knapsack* adalah dengan menentukan barisan *superincreasing* sebagai kunci rahasia. Barisan *superincreasing* dengan panjang k didefinisikan sebagai $S_j > \sum_{i=1}^{j-1} S_i$ dengan $2 \leq j \leq k$. Barisan *superincreasing* adalah barisan yang setiap elemennya lebih besar dari jumlah elemen-elemen sebelumnya [8].

Algoritma *Merkle-Hellman Knapsack* terdiri atas tiga proses, yaitu proses pembangkitan kunci, proses enkripsi, dan proses dekripsi. Kunci publik dibangkitkan menggunakan Persamaan (1) berikut:

$$Kp_i = Kr_i \cdot n \ (mod\ m) \tag{1}$$

dengan,

Kp_i = kunci publik ke-i

Kr_i = kunci rahasia ke-i

Pada proses enkripsi, *ciphertext* diperoleh dengan cara mengalikan biner dengan kunci publik menggunakan Persamaan (2) berikut:

$$C_j = \sum_{i=1}^{n} (b_i \times Kp_i) \tag{2}$$

dengan,

C_j = *ciphertext* ke-j dengan j sebanyak ukuran *plaintext*

b_i = biner *plaintext* ke-i dengan $i = 1,2, \dots, n$

Kp_i = kunci publik ke-i dengan $i = 1,2, \dots, n$

Pada proses dekripsi, nilai *plaintext* dihitung menggunakan Persamaan (3) kemudian biner *plaintext* dicari dengan cara mengurangi nilai *plaintext* dengan elemen kunci rahasia yang besarnya mendekati nilai *plaintext* sampai hasil pengurangan menjadi 0. Nilai 1 diberikan pada indeks kunci rahasia apabila terpilih menjadi operasi pengurangan dan

nilai 0 diberikan jika tidak terpilih.

$$P_j = C_j \cdot n^{-1} \, (mod\ m) \tag{3}$$

dengan,

P_j = nilai *plaintext* ke-j dengan j sebanyak ukuran *ciphertext*

C_j = *ciphertext* ke-j dengan j sebanyak ukuran *ciphertext*

Contoh proses pembangkitan kunci, enkripsi, dan dekripsi menggunakan algoritma *Merkle-Hellman Knapsack* secara berturut-turut disajikan pada Tabel 4, Tabel 5, dan Tabel 6.

Tabel 4. Proses pembangkitan kunci algoritma *Merkle-Hellman Knapsack*

Kunci Rahasia	$Kr \times n \, (mod\ m)$	**Kunci Publik**
2	$2 \times 23 \, (mod\ 420)$	46
3	$3 \times 23 \, (mod\ 420)$	69
6	$6 \times 23 \, (mod\ 420)$	138
12	$12 \times 23 \, (mod\ 420)$	276
24	$24 \times 23 \, (mod\ 420)$	132
48	$48 \times 23 \, (mod\ 420)$	264
96	$96 \times 23 \, (mod\ 420)$	108
192	$192 \times 23 \, (mod\ 420)$	216

Tabel 5. Proses enkripsi algoritma *Merkle-Hellman Knapsack*

Plaintext	**Desimal**	**Biner**	**Kunci Publik**	***Ciphertext***
I	73	01001001	46, 69, 138, 276, 132, 264, 108, 216	417
N	78	01001110	46, 69, 138, 276, 132, 264, 108, 216	573
N	78	01001110	46, 69, 138, 276, 132, 264, 108, 216	573
A	65	01000001	46, 69, 138, 276, 132, 264, 108, 216	285

Tabel 6. Proses dekripsi algoritma *Merkle-Hellman Knapsack*

Ciphertext	**Nilai *Plaintext***	**Kunci Rahasia**	**Biner**	**Desimal**	***Plaintext***
417	219	2, 3, 6, 12, 24, 48, 96, 192	01001001	73	I
573	171	2, 3, 6, 12, 24, 48, 96, 192	01001110	78	N
573	171	2, 3, 6, 12, 24, 48, 96, 192	01001110	78	N
285	195	2, 3, 6, 12, 24, 48, 96, 192	01000001	65	A

2. Metodologi

Data yang digunakan pada penelitian ini adalah teks berupa karakter yang terdapat pada ASCII *printable characters*. Data tersebut akan dikonversi ke bentuk desimal dan bentuk biner sebelum dienkripsi dan setelah didekripsi. Data yang digunakan pada penelitian ini diantaranya *plaintext* = mAtLAb15, elemen pertama kunci rahasia = 2, dan $n = 23$.

Langkah-langkah pengodean teks menggunakan modifikasi algoritma *Electronic Code Book* dan *Merkle-Hellman Knapsack* terdiri atas proses pembangkitan kunci, proses enkripsi, dan proses dekripsi yang dijabarkan sebagai berikut:

1. Proses Pembangkitan Kunci
   a) Tentukan elemen pertama kunci rahasia (Kr_1). Elemen kedua sampai kedelapan kunci rahasia dihitung dari hasil penjumlahan elemen-elemen sebelumnya ditambah satu.
   b) Tentukan bilangan prima n. Setelah itu, akan ditentukan bilangan m dengan syarat m lebih besar dari jumlah semua elemen pada kunci rahasia serta m dan n relatif prima.
   c) Bangkitkan kunci publik menggunakan Persamaan (1).
   d) Modifikasikan kunci XOR dengan cara membangkitkannya dari kunci publik dengan Persamaan (4) kemudian konversi bentuk desimal kunci XOR ke bentuk karakter sehingga didapatkan kunci XOR berupa teks.
   $$Kx_i = Kp_i \ (mod\ 95) + 32 \quad (4)$$
   dengan,
   $Kx_i =$ nilai desimal kunci XOR ke-i
   $Kp_i =$ kunci publik ke-i
2. Proses Enkripsi
   a) Konversi *plaintext*, kunci XOR, dan bilangan n ke bentuk desimal dan biner sesuai ASCII. Bagi bilangan biner tersebut menjadi blok-blok dengan 8 digit di setiap blok.
   b) Enkripsi *plaintext* menggunakan modifikasi algoritma ECB. Lakukan operasi XOR pada *plaintext* dan kunci XOR sesuai bloknya dan geser hasil biner pada tiap blok satu bit ke kiri kemudian modifikasikan hasilnya dengan cara meng-XOR-kan lagi dengan kunci n sehingga diperoleh *ciphertext* ECB.
   c) Enkripsi *ciphertext* hasil ECB menggunakan algoritma *Merkle-Hellman Knapsack* dengan kunci publik. Tentukan *ciphertext* dengan Persamaan (2) sehingga diperoleh *ciphertext* berbentuk angka.
3. Proses Dekripsi
   a) Dekripsi *plaintext* menggunakan algoritma *Merkle-Hellman Knapsack*. Tentukan nilai invers dari n modulo m kemudian hitung nilai *plaintext* menggunakan Persamaan (3). Tentukan bentuk biner dengan cara mengurangi nilai *plaintext* dengan elemen kunci rahasia yang besarnya mendekati nilai *plaintext*. Proses tersebut berlanjut sampai hasil pengurangan menjadi 0. Berikan nilai 1 pada indeks kunci rahasia apabila terpilih menjadi operasi pengurangan dan berikan nilai 0 jika tidak terpilih.
   b) Dekripsi *plaintext* hasil *Merkle-Hellman Knapsack* menggunakan algoritma ECB. Bagi bilangan biner yang sudah diperoleh menjadi blok-blok bit dengan 8 digit di setiap blok kemudian XOR-kan dengan biner n dan geser hasilnya satu bit ke kanan pada tiap blok biner. Lakukan operasi XOR antara biner hasil pergeseran dan kunci XOR untuk mendapatkan biner *plaintext*.
   c) Konversi biner yang diperoleh ke bentuk desimal dan karakter sesuai ASCII sehingga diperoleh *plaintext* semula.

3. Hasil dan Pembahasan

3.1 Perhitungan Manual

a. Proses Pembangkitan Kunci

1) Tentukan elemen pertama kunci rahasia (Kr_1). Elemen kedua sampai kedelapan kunci rahasia dihitung dari hasil penjumlahan elemen-elemen sebelumnya ditambah satu. Penambahan nilai satu tersebut disesuaikan dengan proses pengodean pada program untuk memudahkan pembuatan program menggunakan Matlab. Pada perhitungan manual ini digunakan $Kr_1 = 2$ sehingga diperoleh kunci rahasia yaitu 2, 3, 6, 12, 24, 48, 96, 192.
2) Tentukan bilangan prima n. Setelah itu, tentukan bilangan m dengan syarat m lebih besar dari jumlah semua elemen pada kunci rahasia serta m dan n relatif prima. Pada perhitungan manual ini digunakan $n = 23$ dan $m = 384$.
3) Bangkitkan kunci publik dengan Persamaan (1). Hasil perhitungan kunci publik dapat dilihat pada Tabel 7.

Tabel 7. Hasil pembangkitan kunci publik

Kunci Rahasia	$Kr \times n \ (mod\ m)$	**Kunci Publik**
2	46 (mod 384)	46
3	69 (mod 384)	69
6	138 (mod 384)	138
12	276 (mod 384)	276
24	552 (mod 384)	168
48	1104 (mod 384)	336
96	2208 (mod 384)	288
192	4416 (mod 384)	192

4) Bangkitkan kunci XOR dengan Persamaan (4). Kemudian konversi bentuk desimal kunci XOR ke bentuk karakter sesuai ASCII sehingga didapatkan kunci XOR berupa teks. Hasil perhitungan dapat dilihat pada Tabel 8.

Tabel 8. Hasil pembangkitan kunci XOR

Kunci Publik	**Desimal Kunci XOR**	**Kunci XOR**
46	78	N
69	101	e
138	75	K
276	118	v
168	105	i
336	83	S
288	35	#
192	34	”

b. Proses Enkripsi

1) Konversi *plaintext*, kunci XOR, dan bilangan n ke bentuk desimal dan biner sesuai ASCII. Bagi bilangan biner tersebut menjadi blok-blok dengan 8 digit di setiap blok seperti pada Tabel 9.

Tabel 9. Hasil konversi *plaintext* pada proses enkripsi

Karakter	Desimal	Biner
m	109	01101101
A	65	01000001
t	116	01110100
L	76	01001100
A	65	01000001
b	98	01100010
1	49	00110001
5	53	00110101
N	78	01001110
e	101	01100101
K	75	01001011
v	118	01110110
i	105	01101001
S	83	01010011
#	35	00100011
”	34	00100010
2	50	00110010
3	51	00110011

2) Lakukan operasi XOR pada *plaintext* dan kunci XOR sesuai bloknya dan geser hasil biner pada tiap blok satu bit ke kiri kemudian XOR-kan lagi dengan kunci n sehingga diperoleh *ciphertext* ECB seperti pada Tabel 10.

Tabel 10. Hasil enkripsi pertama

No	***Plaintext* (P)**	**Kunci XOR (Kx)**	$P \oplus Kx$	**Pergeseran 1 bit ke kiri (B)**	**Kunci n (n)**	***Ciphertext*** $(B \oplus n)$
1	01101101 (m)	01001110 (N)	00100011	01000110	00110010 (2)	01110100
2	01000001 (A)	01100101 (e)	00100100	01001000	00110011 (3)	01111011
3	01110100 (t)	01001011 (K)	00111111	01111110	00110010 (2)	01001100
4	01001100 (L)	01110110 (v)	00111010	01110100	00110011 (3)	01000111
5	01000001 (A)	01101001 (i)	00101000	01010000	00110010 (2)	01100010
6	01100010 (b)	01010011 (S)	00110001	01100010	00110011 (3)	01010001
7	00110001 (1)	00100011 (#)	00010010	00100100	00110010 (2)	00010110
8	00110101 (5)	00100010 (“)	00010111	00101110	00110011 (3)	00011101

3) Enkripsi *ciphertext* hasil ECB menggunakan algoritma *Merkle-Hellman Knapsack* dengan kunci publik. Tentukan *ciphertext* dengan Persamaan (2) sehingga diperoleh *ciphertext* berbentuk angka seperti pada Tabel 11.

Tabel 11 Hasil enkripsi kedua

Biner	Kunci Publik	*Ciphertext*
01110100	46, 69, 138, 276, 168, 336, 288, 192	819
01111011	46, 69, 138, 276, 168, 336, 288, 192	1131
01001100	46, 69, 138, 276, 168, 336, 288, 192	573
01000111	46, 69, 138, 276, 168, 336, 288, 192	885
01100010	46, 69, 138, 276, 168, 336, 288, 192	495
01010001	46, 69, 138, 276, 168, 336, 288, 192	537
00010110	46, 69, 138, 276, 168, 336, 288, 192	900
00011101	46, 69, 138, 276, 168, 336, 288, 192	972

Berikut merupakan cara menghitung biner 01110100 sehingga diperoleh *ciphertext* 819:

$$C_1 = (0 \times 46) + (1 \times 69) + (1 \times 138) + (1 \times 276) + (0 \times 168) + (1 \times 336) + (0 \times 288) + (0 \times 192)$$
$$= 69 + 138 + 276 + 336 = 819$$

Dengan cara yang sama diperoleh C_2 hingga C_8. Dari Tabel 11 diperoleh hasil *ciphertext* menggunakan modifikasi algoritma *Electronic Code Book* dan *Merkle-Hellman Knapsack* yaitu 819 1131 573 885 495 537 900 972.

c. Proses Dekripsi

1) Dekripsi *plaintext* menggunakan algoritma *Merkle-Hellman Knapsack*. Tentukan nilai invers dari n modulo m kemudian hitung nilai *plaintext* dengan Persamaan (3). Tentukan bentuk biner dengan cara mengurangi nilai *plaintext* dengan elemen kunci rahasia yang besarnya mendekati nilai *plaintext*. Proses tersebut berlanjut sampai hasil pengurangan menjadi 0. Berikan nilai 1 pada indeks kunci rahasia apabila terpilih menjadi operasi pengurangan dan berikan nilai 0 jika tidak terpilih. Perhitungan mencari nilai invers modulo sebagai berikut:

$$\begin{aligned} 384 &= 23 \times 16 + 16 \rightarrow 23 = 16 \times 1 + 7 \rightarrow 16 = \\ 7 &\times 2 + 2 \rightarrow \quad 16 = 384 - 23 \times 16\ 7 = 23 - \\ 16 &\times 1\ 2 = 16 - 7 \times 2\ 7 \quad = 2 \times 3 + 1 \quad \rightarrow 2 \quad = \\ & 1 \times 2 + 0 \quad 1 = 7 - 2 \times 3 \qquad \} \end{aligned} \tag{5}$$

Substitusi Persamaan (5)

$$\begin{aligned} 1 &= 7 - 2 \times 3 \\ &= 7 - (16 - 7 \times 2) \times 3 \\ &= 7 \times 7 - 16 \times 3 \\ &= (23 - 16 \times 1) \times 7 - 16 \times 3 \\ &= 23 \times 7 - 16 \times 10 \\ &= 23 \times 7 - (384 - 23 \times 16) \times 10 \\ &= 23 \times 167 - 384 \times 10 \\ &= 23 \times 167 + 384 \times (-10) \end{aligned}$$

Dari hasil substitusi Persamaan (5) diketahui bahwa 167 merupakan invers dari 23 (mod 384). Selanjutnya, nilai *plaintext* dihitung berdasarkan rumus pada Persamaan (3) dan bentuk biner diperoleh dari hasil pengurangan nilai *plaintext* dan kunci

rahasia. Hasil perhitungan disajikan pada Tabel 12.

Tabel 12 Hasil dekripsi pertama

Ciphertext	**Nilai *Plaintext***	**Kunci Rahasia**	**Biner**
819	69	2, 3, 6, 12, 24, 48, 96, 192	01110100
1131	333	2, 3, 6, 12, 24, 48, 96, 192	01111011
573	75	2, 3, 6, 12, 24, 48, 96, 192	01001100
885	339	2, 3, 6, 12, 24, 48, 96, 192	01000111
495	105	2, 3, 6, 12, 24, 48, 96, 192	01100010
537	207	2, 3, 6, 12, 24, 48, 96, 192	01010001
900	156	2, 3, 6, 12, 24, 48, 96, 192	00010110
972	276	2, 3, 6, 12, 24, 48, 96, 192	00011101

Berikut merupakan perhitungan untuk memperoleh nilai *plaintext* dan biner dari *ciphertext* "819":

$C_1 = 819$

$P_1 = 819 \times 167 \ (mod\ 384) = 69 \ (mod\ 384) = 69$

$0 = 69 - (0 \times 2) - (1 \times 3) - (1 \times 6) - (1 \times 12) - (0 \times 24) - (1 \times 48) - (0 \times 96) - (0 \times 192)$

Sehingga diperoleh biner *plaintext*-1 = 01110100. Cara yang sama dilakukan untuk memperoleh biner *plaintext*-2 sampai biner *plaintext*-8.

2) Bagi bilangan biner yang sudah diperoleh menjadi blok-blok dengan 8 digit di setiap blok dan XOR-kan dengan biner n kemudian geser hasilnya satu bit ke kanan pada tiap blok. Lakukan operasi XOR antara biner hasil pergeseran dan kunci XOR seperti pada Tabel 13.

Tabel 13. Hasil dekripsi kedua

No	***Ciphertext* (C)**	**Kunci n (n)**	$C \oplus n$	**Pergeseran 1 bit ke kanan (B)**	**Kunci XOR (Kx)**	***Plaintext*** $(B \oplus Kx)$
1	01110100	00110010 (2)	01000110	00100011	01001110 (N)	01101101
2	01111011	00110011 (3)	01001000	00100100	01100101 (e)	01000001
3	01001100	00110010 (2)	01111110	00111111	01001011 (K)	01110100
4	01000111	00110011 (3)	01110100	00111010	01110110 (v)	01001100
5	01100010	00110010 (2)	01010000	00101000	01101001 (i)	01000001
6	01010001	00110011 (3)	01100010	00110001	01010011 (S)	01100010
7	00010110	00110010 (2)	00100100	00010010	00100011 (#)	00110001
8	00011101	00110011 (3)	00101110	00010111	00100010 (")	00110101

3) Konversi biner yang diperoleh ke bentuk desimal dan karakter sesuai ASCII sehingga diperoleh *plaintext* semula. Dari Tabel 14 diperoleh *plaintext* semula yaitu mAtLAb15.

Tabel 14. Hasil konversi *plaintext* pada proses dekripsi

Biner	Desimal	Karakter
01101101	109	m
01000001	65	A
01110100	116	t
01001100	76	L
01000001	65	A
01100010	98	b
00110001	49	1
00110101	53	5

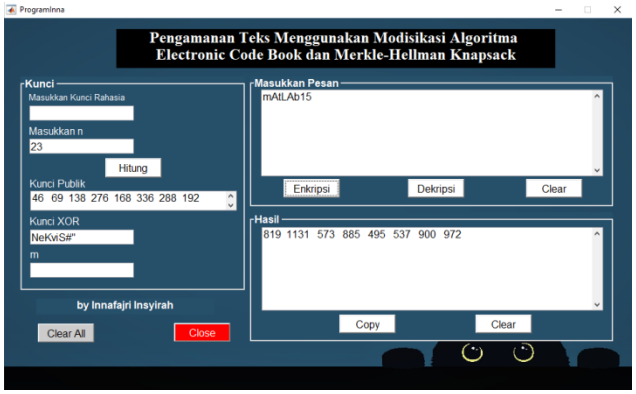
3.2 Simulasi Program

Program pada penelitian ini dibuat menggunakan *software* Matlab 2015. Proses pembangkitan kunci pada program dilakukan dengan cara memasukkan nilai elemen pertama kunci rahasia dan bilangan n. Setelah itu, tombol hitung ditekan untuk menampilkan kunci publik, kunci XOR, dan bilangan m. Gambar 1 menampilkan hasil pembangkitan kunci dengan nilai elemen pertama kunci rahasia = 2 dan $n = 23$.

Gambar 1. Hasil program pembangkitan kunci

Proses enkripsi pada program dijalankan dengan cara memasukkan nilai dari bilangan n, kunci publik, kunci XOR, dan *plaintext*. Setelah itu, tombol enkripsi ditekan untuk menampilkan hasil enkripsi. Gambar 2 menampilkan hasil program enkripsi dengan *plaintext* = mAtLAb15, $n = 23$, kunci XOR = NeKviS#", dan kunci publik = 46, 69, 138, 276, 168, 336, 388, 192 diperoleh *ciphertext* = 819 1131 573 885 495 537 900 972.

Proses dekripsi program dijalankan dengan memasukkan elemen pertama kunci rahasia, bilangan n, kunci XOR, bilangan m, dan *ciphertext*. Setelah itu, tombol dekripsi ditekan untuk menampilkan hasilnya. Gambar 3 menampilkan hasil program dekripsi dengan elemen pertama kunci rahasia = 2, $n = 23$, kunci XOR = NeKviS#", $m = 384$, dan *ciphertext* = 819 1131 573 885 495 537 900 972 menghasilkan *plaintext* = mAtLAb15.



Gambar 2. Hasil program enkripsi

Gambar 3. Hasil program dekripsi

4. Kesimpulan

Modifikasi algoritma *Electronic Code Book* dan *Merkle-Hellman Knapsack* dapat digunakan untuk mengodekan teks. Pesan teks yang dienkripsi menghasilkan *ciphertext* berupa angka acak yang sulit dipahami maknanya oleh orang yang tidak berkepentingan. Pola pesan pada *plaintext* yang sama sudah tidak menghasilkan *ciphertext* yang sama. *Ciphertext* yang dihasilkan dapat didekripsi dengan baik meskipun memiliki panjang dua kali atau lebih dari jumlah karakter *plaintext*. Hal ini dibuktikan dengan *plaintext* yang diperoleh dari hasil dekripsi tidak berubah atau sama seperti *plaintext* semula.

Daftar Pustaka

[1] Ariyus, D., (2008), *Pengantar Ilmu Kriptografi*, CV Andi Offset, Yogyakarta.

[2] Asti, M., Kamsyakawuni, A., Santoso, K.A., (2018), Pengamanan *Image* Dengan Modifikasi Algoritma *Electronic Code Book* (ECB), *Majalah Ilmiah Matematika dan Statistika* **18(2)**, 91-104.

[3] Fadlan, M. dan Hadriansa, (2017), Rekayasa Aplikasi Kriptografi Dengan Penerapan Kombinasi Algoritma *Knapsack Merkle-Hellman* dan *Affine Cipher*.

Jurnal Teknologi Informasi dan Ilmu Komputer **4(4)**, 268-274.

[4] Karo, E. S. B., (2020), Penerapan Algoritma *Affine Cipher* dan Algoritma *Electronic Code Book* (ECB) dalam Pengamanan Pesan Teks, *Jurnal Teknik Informatika Unika St. Thomas* **5(1)**, 28-32.

[5] Leman, D. dan Rahman, M., (2020), Metode *Merkle-Hellman* untuk Enkripsi dan Dekripsi Pesan *Whatapp*, *Riau Journal of Computer Science* **6(1)**, 45-49.

[6] Murdowo, S., (2019), Mengenal Kriptografi Modern Sederhana Menggunakan Electronic Code Book (ECB), *Jurnal INFOKAM* **15(1)**, 29-37.

[7] Setyaningsih, E., (2015), *Kriptografi dan Implementasinya Menggunakan Matlab*, CV Andi Offset, Yogyakarta.

[8] Simarmata, J., Sriadhi, dan Rahim, R., (2019), *Kriptografi Teknik Keamanan Data dan Informasi*, CV Andi Offset, Yogyakarta.

[9] Sulaiman, O. K., (2019), *Hybrid Cryptosystem* Menggunakan *XOR Cipher* dan *Merkle-Hellman Knapsack* untuk Menjaga Kerahasiaan Pesan Digital, *Jurnal Teknologi Informasi* **3(2)**, 169-173.

[10] Widarma, A., Siregar, H. F., dan Irawan, M. D., (2019), Teknik Keamanan Data Menggunakan *Vigenere Cipher* dan *Electronic Code Book* (ECB), *Jurnal Sains Komputer & Informatika* **3(2)**, 393-400.