Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 2, April 2024, pp. 519 -530

DOI: https://doi.org/10.52436/1.jutif.2024.5.2.1535
p-ISSN: 2723-3863
e-ISSN: 2723-3871

STREAM CIPHER ALGORITHM FOR ENCRYPTING TEXT USING LOGISTIC MAP, AUTO PARAMETERS LINEAR CONGRUENTIAL GENERATOR (APLCG), AND GRAY CODE

Adriana Fanggidae[*1], Yulianto Triwahyuadi Polly[2], Derwin Rony Sina[3], Kornelis Letelay[4], Yelly Yosiana Nabuasa[5], Meiton Boru[6], Juan Rizky Mannuel Ledoh[7]

[1,2,3,4,5,6,7]Department of Computer Science, Faculty of Science and Engineering, Universitas Nusa Cendana, Indonesia
Email: [1]adrianafanggidae@staf.undana.ac.id, [2]yuliantopolly@staf.undana.ac.id, [3]derwinsina@staf.undana.ac.id, [4]kornelis@staf.undana.ac.id, [5]yellynabuasa@staf.undana.ac.id, [6]meitonboru@staf.undana.ac.id, [7]juanledoh@staf.undana.ac.id

(Article received: November 18, 2023; Revision: December 17, 2023; published: April 15, 2024)

Abstract

One aspect frequently posing a challenge in cryptography pertains to the length of the secret key that users must remember. Achieving the requisite key length for cryptographic algorithms necessitates key padding. However, it is crucial to note that key padding is susceptible to predictable patterns. Both the Linear Congruential Generator (LCG) and gray code are algorithms employed to generate sequences of padded key bits. Regrettably, LCG requires the determination of two pre-defined parameters, whereas the Auto Parameters Linear Congruential Generator (APLCG) automatically establishes these parameters. These parameters play a pivotal role in generating unique sequences of random integers. To fortify key security, the generation of new keys is performed using a modified logistic map, an enhancement of the standard logistic map that exhibits random behavior consistently. Stream cipher, an encryption algorithm, necessitates a continuous key stream matching the bit or byte length of the message. We conducted experiments on stream cipher algorithms employing key streams generated from APLCG, gray code, and modified logistic map. Twenty text documents were utilized as test samples. The outcomes indicate that stream ciphers employing APLCG, gray code, and modified logistic map demonstrate high-security performance based on the statistical analysis conducted.

Keywords: *gray code, linear congruential generator, logistic map, statistical analysis, stream cipher, text data.*

ALGORITMA CIPHER ALIRAN UNTUK ENKRIPSI TEKS MENGGUNAKAN LOGISTIC MAP, AUTO PARAMETERS LINEAR CONGRUENTIAL GENERATOR (APLCG), DAN GRAY CODE

Abstrak

Salah satu aspek yang sering menjadi tantangan dalam kriptografi adalah panjang kunci rahasia yang harus diingat oleh pengguna. Untuk mencapai panjang kunci yang diperlukan oleh algoritma kriptografi, perlu dilakukan *padding* kunci. Namun, penting untuk diingat bahwa *padding* kunci rentan terhadap pola yang dapat dengan mudah diprediksi. *Linear Congruential Generator* (LCG) dan *gray code* merupakan algoritma yang dapat digunakan untuk melakukan pengacakan rangkaian bit kunci hasil *padding*. Sayangnya, LCG memerlukan dua parameter yang harus ditentukan terlebih dahulu, sementara *Auto Parameters Linear Congruential Generator* (APLCG) mampu secara otomatis menentukan parameter tersebut. Parameter ini memegang peran penting dalam menghasilkan rangkaian bilangan bulat acak yang unik. Untuk memperkuat keamanan kunci, dilakukan pembangkitan kunci baru menggunakan *logistic map* modifikasi, sebuah perbaikan dari *logistic map standar* yang menampilkan perilaku acak sepanjang waktu. *Stream cipher* adalah algoritma kriptografi yang membutuhkan aliran kunci sepanjang bit atau byte pesan. Kami melakukan eksperimen terhadap algoritma *stream cipher* dengan aliran kunci yang dihasilkan dari APLCG, *gray code*, dan *logistic map*. Dokumen uji yang digunakan sebanyak dua puluh dokumen teks. Hasilnya menunjukkan bahwa *stream cipher* dengan APLCG, *gray code*, dan *logistic map* modifikasi memberikan kinerja keamanan yang tinggi berdasarkan analisis statistik yang telah kami lakukan.

Kata kunci: *analisis statistik, data teks, gray code, linear congruential generator, logistic map, stream cipher.*

1. PENDAHULUAN

Stream cipher adalah algoritma kriptografi yang dikenal karena efisiensinya dalam melakukan enkripsi data secara *real-time*. Algoritma ini beroperasi dengan mengenkripsi pesan secara berurutan, bit per bit atau byte per byte, menggunakan bit atau byte dari aliran kunci yang dihasilkan oleh algoritma kunci. Penting untuk dicatat bahwa algoritma kunci harus menghasilkan aliran kunci yang memiliki sifat acak dan sulit diprediksi guna menjaga keamanan pesan yang dienkripsi [1].

Pada era ini, banyak individu memilih internet sebagai sarana untuk mengirim dan menyimpan data dalam bentuk teks. Fenomena ini terbukti dari pertumbuhan ekstrim dalam transfer data serta volume data itu sendiri [2]. Data yang bersifat privasi dan rahasia harus dijaga keamanannya, dan ini menjadi tantangan yang signifikan bagi penyedia sistem komunikasi [3]. Oleh karena itu, diperlukan kecepatan dalam proses transmisi informasi, sambil tetap menjaga tingkat keamanan untuk menghindari potensi risiko kebocoran data. Penggunaan teori *chaos* dalam bidang kriptografi tetap menarik karena sifatnya yang acak, sensitivitas terhadap kondisi awal dan parameter kontrol, serta beban komputasi yang rendah [4], [5]. Dalam konteks kriptografi, metode ini digunakan untuk menghasilkan kunci dengan urutan semu yang acak [6], [7].

Selain teori *chaos*, *Linear Congruential Generator* (LCG) dan *gray code* dapat diaplikasikan untuk menghasilkan kunci yang bersifat acak. LCG merupakan algoritma pembangkit bilangan acak yang sederhana dan efisien. Namun, dalam proses pembangkitan urutan bilangan acak yang bersifat unik, perlu dilakukan pemilihan parameter dengan sangat hati-hati. Dalam beberapa situasi, LCG dapat digunakan dalam konteks kriptografi, tetapi penting untuk menetapkan nilai parameter yang tepat agar siklus bilangan acak yang dihasilkan mencapai maksimal [8]. *Auto Parameters Linear Congruential Generator* (APLCG) dapat membantu dalam menentukan parameter yang optimal. Sementara itu, *gray code* adalah urutan sistematis dari bilangan biner di mana dua bilangan berturut-turut hanya berbeda satu bit. Dalam konteks kriptografi, *gray code* dapat digunakan sebagai salah satu langkah untuk meningkatkan keamanan data dengan mengurangi risiko terdeteksinya pola dalam data yang dapat dimanfaatkan oleh penyerang.

Logistic map standar termasuk dalam *discrete time chaotic systems*, dan menurut beberapa penelitian sebelumnya, peta ini memiliki kelemahan yaitu perilaku kekacauan yang terbatas pada wilayah tertentu [9]–[12]. Hal ini membuat *logistic map* standar menjadi rentan terhadap serangan. Dalam penelitian [9]–[12] dilakukan modifikasi terhadap *logistic map* standar untuk membuatnya memiliki perilaku kekacauan sepanjang waktu. Dalam *logistic map*, nilai parameter awal x_0 adalah kondisi awal dari sistem yang digunakan untuk menghasilkan deret nilai selanjutnya dalam iterasi *logistic map*. Dalam penelitian sebelumnya [11], [12], nilai x_0 ditentukan dari rangkaian bit kunci rahasia dengan panjang 128 bit atau 16 byte, yang tentunya menjadi sulit untuk diingat. Untuk mengatasi masalah panjang kunci yang tidak memadai, perlu dilakukan *padding* kunci. Namun, perlu diperhatikan bahwa teknik *padding* yang tidak tepat dapat menimbulkan celah keamanan. Dalam penelitian ini, kami mengusulkan penerapan algoritma APLCG dan *gray code* untuk mengacak kunci yang dihasilkan dari proses *padding*. Untuk meningkatkan tingkat keamanan, dilakukan pembangkitan kunci baru menggunakan *logistic map*. Dengan pendekatan ini, algoritma *stream cipher* memiliki pola kunci yang tidak dapat diprediksi, dan menjaga keamanan data yang optimal. Kontribusi utama dari penelitian ini mencakup:

- Algoritma baru APLCG menghasilkan deretan bilangan bulat acak yang unik.
- Aliran kunci dengan APLCG, *gray code*, dan *logistic map*.

Pengukuran kinerja algoritma *stream cipher* yang diusulkan menggunakan analisis statistik, yaitu koefisien korelasi [13], Hamming *distance* [14], dan entropi [15], [16].

2. METODE PENELITIAN

2.1. *Logistic Map*

Logistic map standar adalah salah satu fungsi sederhana yang digunakan untuk memahami prinsip dasar teori *chaos*. Dalam *logistic map*, terlihat perilaku *chaos*, di mana sedikit perubahan pada kondisi awal populasi dapat menghasilkan perubahan drastis dalam sistem. Pada nilai-nilai parameter tertentu, *logistic map* menunjukkan perilaku acak, bahkan jika awalnya sistem berperilaku teratur. Persamaan *logistic map* standar ditunjukkan pada Persamaan 1, di mana x_0 adalah kondisi awal, parameter kontrol $r \in [0,4]$, dan $x \in [0,1]$.

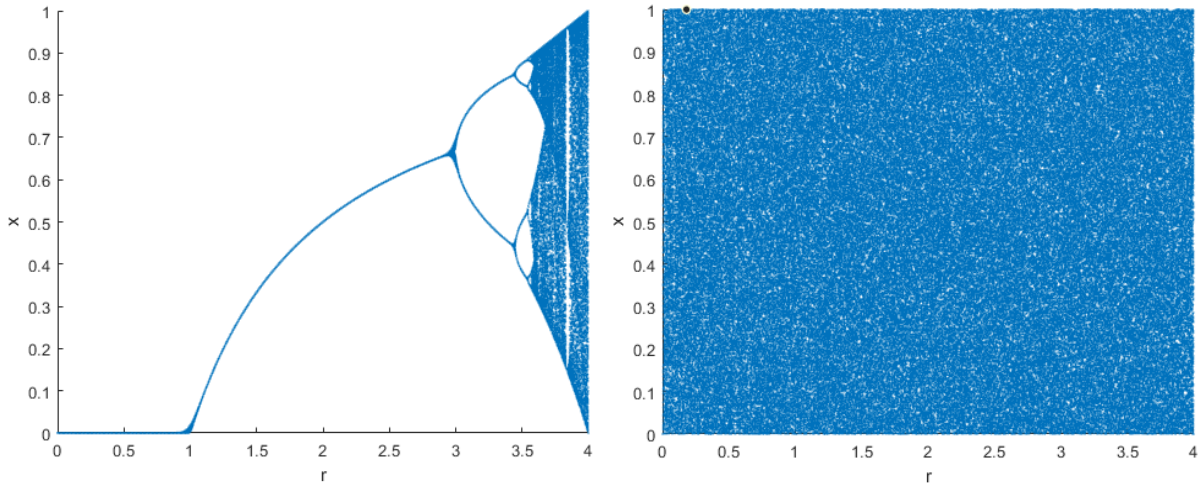
$$x_{n+1} = rx_n(1 - x_n) \quad (1)$$

Dalam perkembanganya, digunakan pendekatan lain untuk lebih menghasilkan *chaotic map* dari *logistic map*. Pada penelitian Alawida, Teh, Mehmood, Shoufan, Alshoura (2022) diusulkan *logistic map* dengan sensitivitas tinggi akibat penggunaan fungsi inversi perkalian [12], sebagaimana Persamaan 2.

$$x_{n+1} = \left(\frac{2^k}{2^{(rx_n(1-x_n))}}\right) \; mod \; 1 \quad (2)$$

Di mana k adalah parameter yang menyeimbangkan kompleksitas komputasi peta baru dengan sifat statistiknya, $k \in [5,15]$ dan direkomendasikan penggunaan $k = 10$.

Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 2, April 2024, pp. 519 -530
DOI: https://doi.org/10.52436/1.jutif.2024.5.2.1535
p-ISSN: 2723-3863
e-ISSN: 2723-3871



(a) (b)

Gambar 1. Diagram Bifurcation dengan Parameter $r \in [0,4]$, $x_0 = 0{,}5$: (a) *Logistic Map* Standar, dan (b) *Logistic Map* Modifikasi [12]

Perilaku kacau dari dua pendekatan *logistic map* dapat divisualisasikan melalui diagram bifurkasi, sebagaimana terlihat pada Gambar 1. Area yang diarsir sepanjang rentang parameter kontrol mengindikasikan tingkat kekacauan dari variabel sistem x yang dihasilkan, menjadikannya lebih sulit untuk diserang.

Logistic map menunjukkan perilaku *chaos* dalam rentang $3{,}67 < r < 4$ [11], [12]. Penelitian ini mengusulkan penggunaan nilai r yang bersifat dinamis dan dipengaruhi oleh nilai x yang dihasilkan, sebagaimana terlihat pada Persamaan 3. Hal ini bertujuan untuk membuat *logistic map* menjadi lebih sulit diprediksi.

$$r = r_{min} + x_{n+1}(r_{max} - r_{min}) \quad (3)$$

2.2. *Auto Parameters Linear Congruential Generator* (APLCG)

Linear Congruential Generator (LCG) diperkenalkan oleh Lehmar pada tahun 1951 sebagai pembangkit bilangan acak seperti pada Persamaan 4.

$$y_{n+1} = (ay_n + c)\ mod\ m \quad (4)$$

Parameter m menunjukkan rentang dari bilangan acak, sementara y_0 adalah nilai awal yang digunakan untuk menginisialisasi deretan bilangan acak. Dalam penelitian ini, penggunaan APLCG diusulkan untuk mengacak posisi bit atau byte dari kunci atau *plaintext* atau *ciphertext*, sehingga nilai m setara dengan panjang bit atau byte dari kunci atau *plaintext* atau *ciphertext* tersebut. Penentuan parameter a dan c untuk menghasilkan deretan bilangan bulat acak yang unik dilakukan berdasarkan nilai m menggunakan algoritma yang dijelaskan pada Gambar 2.

```
Fungsi DetermAC(m)
%v adalah semua bilangan prima yang lebih kecil dari (m-1)
%d adalah banyaknya v
%Menentukan nilai c
for i = 1 to d
  if (m mod v(d)) mod 2 <> 0
    c = v(d)
if c empty
  c = 1
%Menentukan nilai a
u = m
for i = 1 to d
  if u mod v(d) = 0
    a = v(d)
    u = round(u/a)
  end
if (u mod 4 == 0)
  a = (a*4)+1
  if a > m
    a = u + 1
else
  a = 1
end
```

Gambar 2. Algoritma untuk Menentukan Parameter a dan c pada APLCG

2.3. *Gray Code*

Gray code telah luas digunakan dalam berbagai bidang, seperti rekayasa, telekomunikasi, genetika, dan bahkan dalam teka-teki matematika [17]. Dalam *gray code*, dua nilai biner berurutan dibuat berbeda hanya dalam satu bit. Proses konversi satu bit ini dapat dilakukan dengan menggunakan Persamaan 5, dengan b adalah bit biner, l mewakili panjang bit biner, g adalah bit biner hasil *gray code*, dan $i = 1,2,\dots,l$.

$$g_i = \begin{cases} b_l \oplus b_i, & i = 1 \\ b_{i-1} \oplus b_i, & i > 1 \end{cases} \quad (5)$$

2.4. *Stream Cipher*

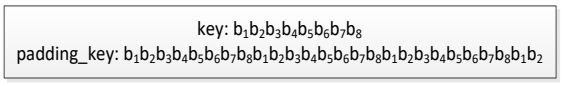
Stream cipher mengenkripsi *plaintext* dengan aliran kunci dalam bentuk bit yang dihasilkan dari

logistic map. *Plaintext* dibagi menjadi beberapa bagian dengan ukuran yang sama dengan kunci, lalu setiap bagian dienkripsi menggunakan aliran kunci yang dihasilkan melalui beberapa tahapan:

1. Karakter kunci rahasia diacak menggunakan APLCG berdasarkan panjang byte kunci dan nilai y_0 diperoleh dari Persamaan 6.

$$y_0 = (ac)^4 \quad (6)$$

2. Setiap byte kunci diubah menjadi representasi biner 8 bit. Tingkat kekacauan kunci ditetapkan $w = 26$ bit, nilai ini merupakan titik tengah dari format *floating-point* 52-bit [11]. Jika jumlah bit kunci belum mencapai kelipatan w, maka kunci perlu di-*padding*. Ilustrasinya dapat dilihat pada Gambar 3.



Gambar 3. *Padding* Kunci 1 Byte atau 8 Bit

3. Kunci hasil *padding* kemudian diacak menggunakan APLCG dan *gray code*.
4. Kondisi awal dari *logistic map* dihitung menggunakan Persamaan 7.

$$x_0 = \left(\sum_{i=1}^{q} x_{0i}\right) mod\ 1 \quad (7)$$

dimana, $x_{0i} = \frac{\text{decimal}(g)}{2^w}$, $q = \text{round}\left(\frac{\text{length}(padding\_key)}{w}\right)$

```
%Input plaintext, key
w = 26
p = length(plaintext)
DetermAC(p)
y0 = (a*c)^4
plain = LCG(plaintext,p,y0,a,c)
h = length(key)
DetermAC(h)
y0 = (a*c)^4
key =LCG(key,h,y0,a,c)
b = padding(key,h,w)
l = length(b)
e = 1
r = 3.999
k = 15
z = 15   %Putaran
q = round(l/w)
while e <= p
  if e = 1 then
    DetermAC(l)
    y0 = (a*c)^4
    b = LCG(b,l,y0,a,c)
    g = gray(b,l)
  else
    b = LCG(b,l,y0,a,c)
    g = gray(b,l)
  end
  for i =1 to q
    x0(i) = mod(bin2dec(g(((i-1)*w+1:i*w))/2^w),1)
  x(1) = mod(sum(x0),1)
  for f = 1 to h
    for i = 2 to z+1
      x(i) = (2^k / 2^(r*x(i-1)*(1-x(i-1)))  mod 1)   %Logistic map
    t(f) = floor(mod(x(i)*(2^w),256))
    if e <= p
      ciphertext(e) = bitxor(t(f),plaintext(e))
      key(e) = t(f)
    end
    r = 3.671+x(i)*(3.999-3.671)
    k = round(5+x(i)*(15-5))
    z = round(5+x(i)*(15-5))
    x(1) = x(i)
    e = e+1
  end
  y0 = floor(x(1)*(10^4))
  b = padding(key,h,w)
end
%Output ciphertext
```

(a)

```
%Input ciphertext, key
w = 26
p = length(ciphertext)
h = length(key)
DetermAC(h)
y0 = (a*c)^4
key =LCG(key,h,y0,a,c)
b = padding(key,h,w)
l = length(b)
e = 1
r = 3.999
k = 15
z = 15   %Putaran
q = round(l/w)
while e <= p
  if e = 1 then
    DetermAC(l)
    y0 = (a*c)^4
    b = LCG(b,l,y0,a,c)
    g = gray(b,l)
  else
    b = LCG(b,l,y0,a,c)
    g = gray(b,l)
  end
  for i =1 to q
    x0(i) = mod(bin2dec(g(((i-1)*w+1:i*w))/2^w),1)
  x(1) = mod(sum(x0),1)
  for f = 1 to h
    for i = 2 to z+1
      x(i) = (2^k / 2^(r*x(i-1)*(1-x(i-1)))  mod 1)   %Logistic map
    t(f) = floor(mod(x(i)*(2^w),256))
    if e <= p
      plain(e) = bitxor(t(f),ciphertext(e))
      key(e) = t(f)
    end
    r = 3.671+x(i)*(3.999-3.671)
    k = round(5+x(i)*(15-5))
    z = round(5+x(i)*(15-5))
    x(1) = x(i)
    e = e+1
  end
  y0 = floor(x(1)*(10^4))
  b = padding(key,h,w)
end
DetermAC(p);
y0 = (a*c)^4
plaintext = invxLCG(plain,p,y0,a,c)
%Output plaintext
```

(b)

Gambar 4. Algoritma *Stream Cipher*: (a) Enkripsi, (b) Dekripsi

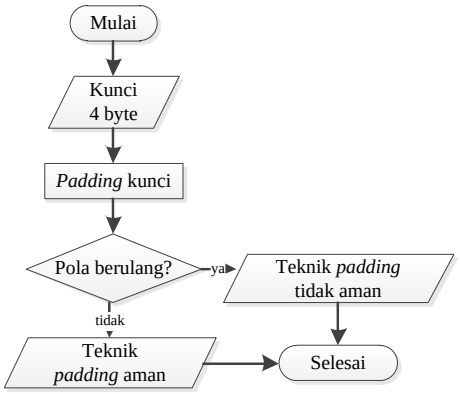
DOI: https://doi.org/10.52436/1.jutif.2024.5.2.1535
p-ISSN: 2723-3863
e-ISSN: 2723-3871

5. Bangkitkan kunci baru sebanyak length(key) menggunakan fungsi *logistic map*. Kunci baru pertama dihasilkan dari 15 putaran. Kunci berikutnya dihasilkan dari putaran [5,15] yang nilainya ditentukan dari nilai *chaos* x putaran terakhir. Prosedur untuk menentukan putaran, juga digunakan dalam menentukan nilai k.
6. Tahapan 5 diulang hingga panjang kunci baru sama dengan panjang *plaintext*, di mana nilai x_0 diperbarui berdasarkan nilai b yang diperoleh dari kunci baru sebelumnya.

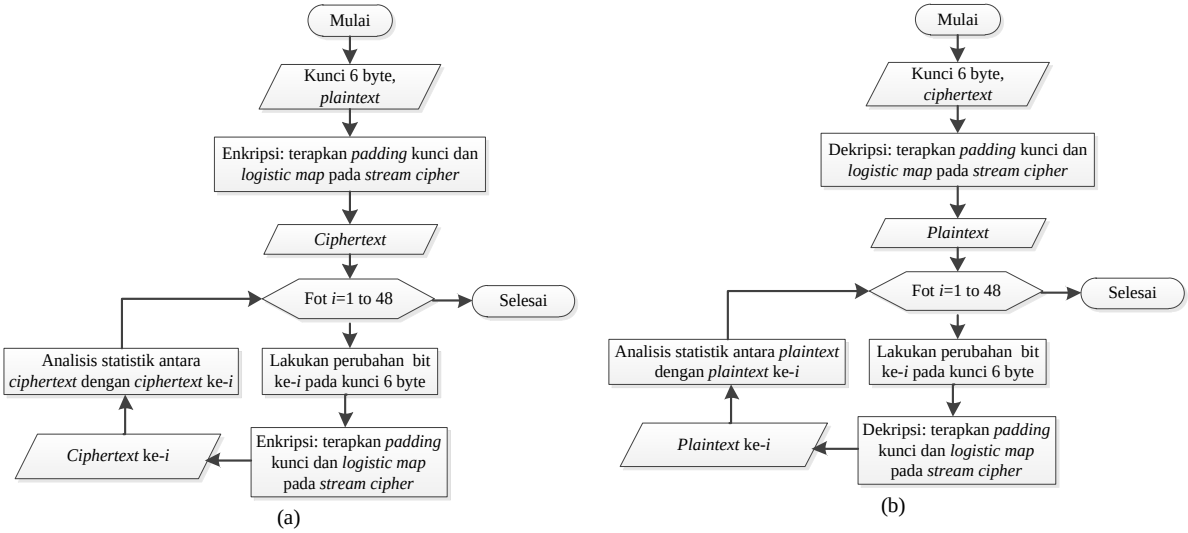
Untuk penjelasan yang lebih rinci, silahkan lihat Gambar 4.

2.5. Pengukuran kinerja algoritma *stream cipher*

Dalam mengevaluasi kinerja algoritma *stream cipher* yang diusulkan, dilakukan serangkaian tahapan pengujian, termasuk pengujian pola kunci sebagaimana ditunjukkan pada Gambar 5, dan pengujian sensitivitas kunci sebagaimana terlihat pada Gambar 6.



Gambar 5. Pengujian Pola Kunci



Gambar 6. Pengujian Sensitivitas Kunci pada: (a) *Ciphertext*, (b) *Plaintext*

Analisis statistik yang digunakan dalam Gambar 6 mencakup koefisien korelasi, Hamming *distance*, dan entropi. Dalam uji koefisien korelasi, diambil nilai absolut hasil koefisien korelasi, sehingga nilainya berada dalam rentang [0,1], dengan nilai terbaik adalah 0. Uji Hamming *distance* dan uji entropi memiliki nilai terbaik berturut-turut sebesar 100% dan 8.

3. HASIL DAN PEMBAHASAN

Sistem ini dibangun menggunakan bahasa pemrograman MATLAB, dengan spesifikasi perangkat keras yang mencakup Intel Core i7-4510U *configuration*, CPU @ 2.0GHz *with Turbo Boost up* to 3.1GHz, 8GB RAM, 64-bit *operating system*. Eksperimen dilakukan terhadap dua metode, yaitu *logistic map* standar dan *logistic map* modifikasi. Dokumen uji terdiri dari dua puluh dokumen teks yang memuat cerita-cerita dongeng Inggris, yang diperoleh dari [18]. Gambar 7 menampilkan judul-judul cerita dongeng Inggris. Baik *plaintext* maupun *ciphertext* disimpan dalam format unit8, yaitu tipe data yang merepresentasikan bilangan bulat tanpa tanda (non-negatif) dalam 8 bit.

BINNORIE	CAP O' RUSHES	CHILDE ROWLAND	HENNY-PENNY	HOW JACK WENT TO SEEK HIS FORTUNE
JACK AND THE BEANSTALK	JACK HANNAFORD	JACK THE GIANT-KILLER	MOUSE AND MOUSER	MR. VINEGAR
NIX NOUGHT NOTHING	TEENY-TINY	THE FISH AND THE RING	THE LAIDLY WORM OF SPINDLESTON HEUGH	THE OLD WOMAN AND HER PIG
THE ROSE-TREE	THE STORY OF THE THREE BEARS	THE STORY OF THE THREE LITTLE PIGS	THE THREE SILLIES	TOM TIT TOT

Gambar 7. Dongeng Inggris

Pengujian pola kunci bertujuan untuk mengevaluasi pola kunci yang dihasilkan baik dengan maupun tanpa pengacakan APLCG dan *gray code*. Pengujian ini dilakukan terhadap tiga kunci berukuran 4 byte atau 32 bit, dengan tahapan berdasarkan Gambar 5. Tabel 1 memperlihatkan bahwa pola berulang dimulai pada bit ke-33 untuk

kunci tanpa pengacakan APLCG dan *gray code*, sementara pada kunci dengan pengacakan APLCG dan *gray code*, tidak terdapat pola berulang.

Pengujian berikutnya bertujuan menganalisis kinerja dua fungsi *logistic map* yang dilakukan pada tiga kunci berukuran 6 byte atau 48 bit dengan $r = [0{,}001\ ,\ 3{,}999]$ dan $r = [3{,}671\ ,\ 3{,}999]$, serta menggunakan dua puluh dokumen teks sebagai *input*. Hasilnya menunjukkan bahwa *logistic map* modifikasi memiliki tingkat keamanan data yang lebih baik, meskipun ada peningkatan dalam waktu eksekusi akibat keterlibatan variabel k dalam fungsi tersebut, ditunjukkan pada Tabel 2 dan 3.

Tabel 1. Pola Kunci Hasil *Padding* Kunci

Kunci rahasia	***Bit key***	**Kunci *padding* tanpa APLCG dan *gray code***	**Kunci *padding* dengan APLCG dan *gray code***
1H9d	00110001010010000011100101100100	**0011000101001000001**11001011001 00**0011000101001000001**1	0000100011001111101100110000001 0101010111001101000000
vTuG	01110110010101000111010101000111	**0111011001010100011**10101010001 11**0111011001010100011**1	1001101011001111111111111110011 0011001100110010011001
1708	00110001001101110011000000111000	**0011000100110111001**10000001110 00**0011000100110111001**1	0110010101010101001100110110111 1111101110011001010110

Tabel 2. Hasil Pengujian Algoritma *Stream Cipher* dengan APLCG, *Gray Code*, dan Dua Fungsi *Logistic Map* dengan $r = [0{,}001\ ,\ 3{,}999]$

Pengujian	**Fungsi**	**Kunci rahasia**			**Mean**
		1H9d5X	**vTuGrb**	**170823**	
Koefisien korelasi	*Logistic map* standar	0,0096	0,0090	0,0121	**0,0102**
	Logistic map modifikasi	0,0153	0,0116	0,0121	0,0130
Hamming *distance* (%)	*Logistic map* standar	93,2887	99,5701	99,5888	97,4825
	Logistic map modifikasi	99,6172	99,6444	99,6120	**99,6245**
Entropi	*Logistic map* standar	4,4859	7,9479	7,9482	6,7940
	Logistic map modifikasi	7,9620	7,9616	7,9607	**7,9614**
Waktu (detik)	*Logistic map* standar	0,8312	0,8410	0,8436	**0,8386**
	Logistic map modifikasi	0,8561	0,8524	0,8528	0,8538

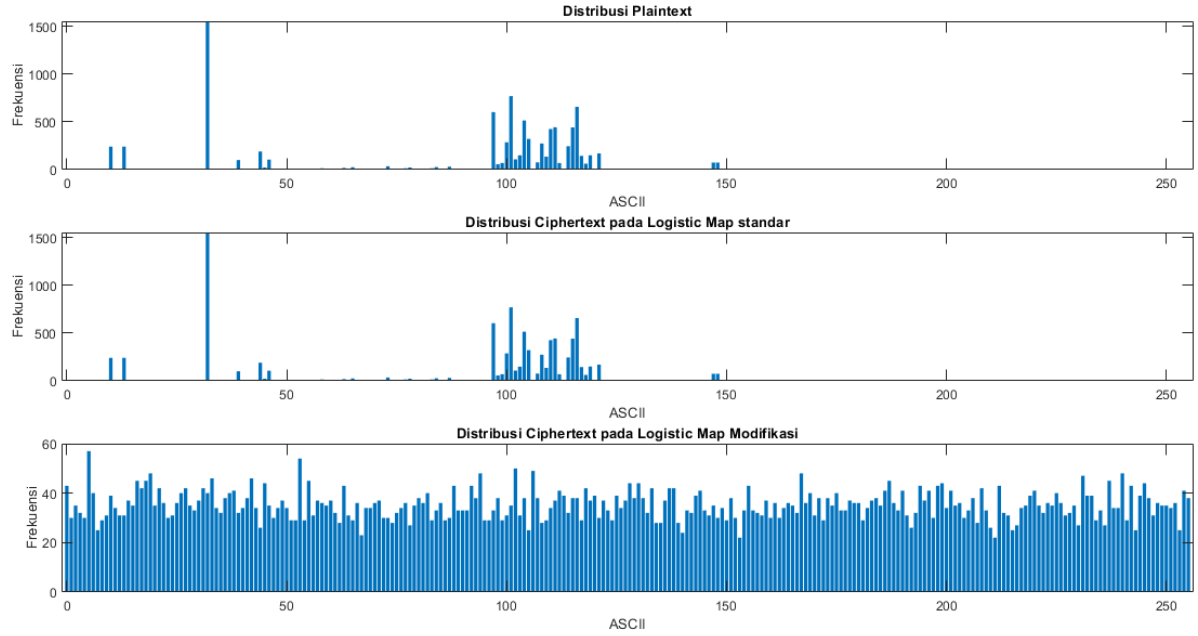
Tabel 3. Hasil Pengujian Algoritma *Stream Cipher* dengan APLCG, *Gray Code*, dan Dua Fungsi *Logistic Map* dengan $r = [3{,}671\ ,\ 3{,}999]$

Pengujian	**Fungsi**	**Kunci rahasia**			**Mean**
		1H9d5X	**vTuGrb**	**170823**	
Koefisien korelasi	*Logistic map* standar	0,0095	0,0120	0,0109	0,0108
	Logistic map modifikasi	0,0105	0,0088	0,0126	**0,0106**
Hamming *distance* (%)	*Logistic map* standar	99,6291	99,5780	99,6410	99,6160
	Logistic map modifikasi	99,6096	99,6599	99,6053	**99,6249**
Entropi	*Logistic map* standar	7,9610	7,9626	7,9610	7,9615
	Logistic map modifikasi	7,9615	7,9620	7,9618	**7,9618**
Waktu (detik)	*Logistic map* standar	0,8876	0,8499	0,8594	**0,8656**
	Logistic map modifikasi	0,8880	0,8668	0,8600	0,8716

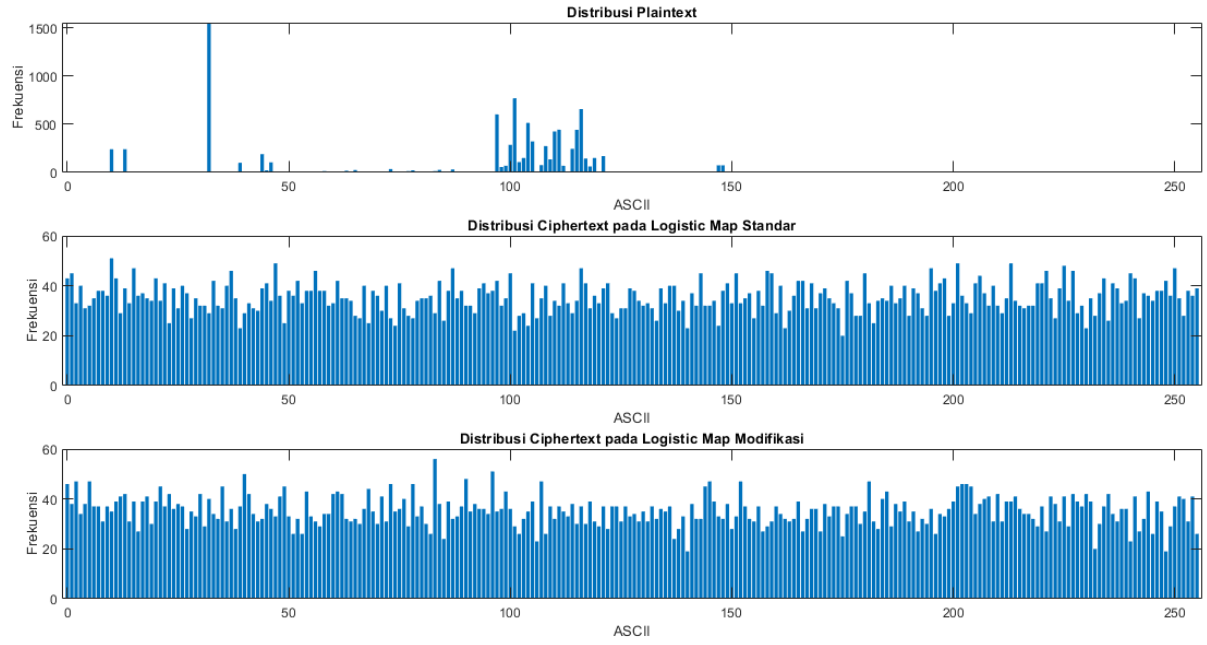
Pada Tabel 2, terlihat bahwa nilai entropi yang dihasilkan oleh *logistic map* standar lebih kecil dibandingkan dengan *logistic map* standar pada Tabel 3. Hal ini mengindikasikan bahwa *logistic map* standar memiliki rentang kekacauan yang terbatas. Sebaliknya, untuk *logistic map* modifikasi, nilai entropi mendekati delapan, menunjukkan distribusi karakter-karakter *ciphertext* yang lebih acak seperti pada Gambar 8. *Logistic map* modifikasi memiliki rentang kekacauan yang lebih luas. Selain itu, perilaku *chaos* fungsi *logistic map* sangat dipengaruhi oleh wilayah dari parameter kontrol r.

Pengujian sensitivitas kunci dilakukan sesuai dengan tahapan pada Gambar 6. Kami melakukan perbandingan yang adil dengan menggunakan *logistic map* standar pada nilai $r = [3{,}671\ ,\ 3{,}999]$. Dalam pengujian sensitivitas kunci pada *ciphertext*, kami mengamati hasil enkripsi dengan mengubah satu bit pada kunci rahasia, dimulai dari bit ke-1 hingga bit ke-48. *Ciphertext* hasil dari perubahan satu bit kunci dibandingkan dengan *ciphertext* hasil kunci rahasia. Hasilnya menunjukkan pola yang sangat berbeda, tercermin dalam nilai koefisien korelasi mendekati nol, *Hamming distance* mendekati seratus persen, dan nilai entropi mendekati delapan. Sementara itu, pengujian sensitivitas kunci pada *plaintext* dilakukan dengan melakukan perubahan satu bit pada kunci rahasia selama dekripsi, yang juga dimulai dari bit ke-1 hingga bit ke-48. Pengamatan ini membandingkan *plaintext* hasil dekripsi dengan *plaintext* asli, dan hasilnya menunjukkan bahwa keduanya memiliki pola yang sangat berbeda. Hal ini juga didukung oleh nilai koefisien korelasi mendekati nol, *Hamming distance* mendekati seratus persen, dan nilai entropi mendekati delapan. Semua ini terdokumentasi dalam Tabel 4-5.

Jurnal Teknik Informatika (JUTIF)
Vol. 5, No. 2, April 2024, pp. 519 -530
DOI: https://doi.org/10.52436/1.jutif.2024.5.2.1535
p-ISSN: 2723-3863
e-ISSN: 2723-3871



(a)



(b)

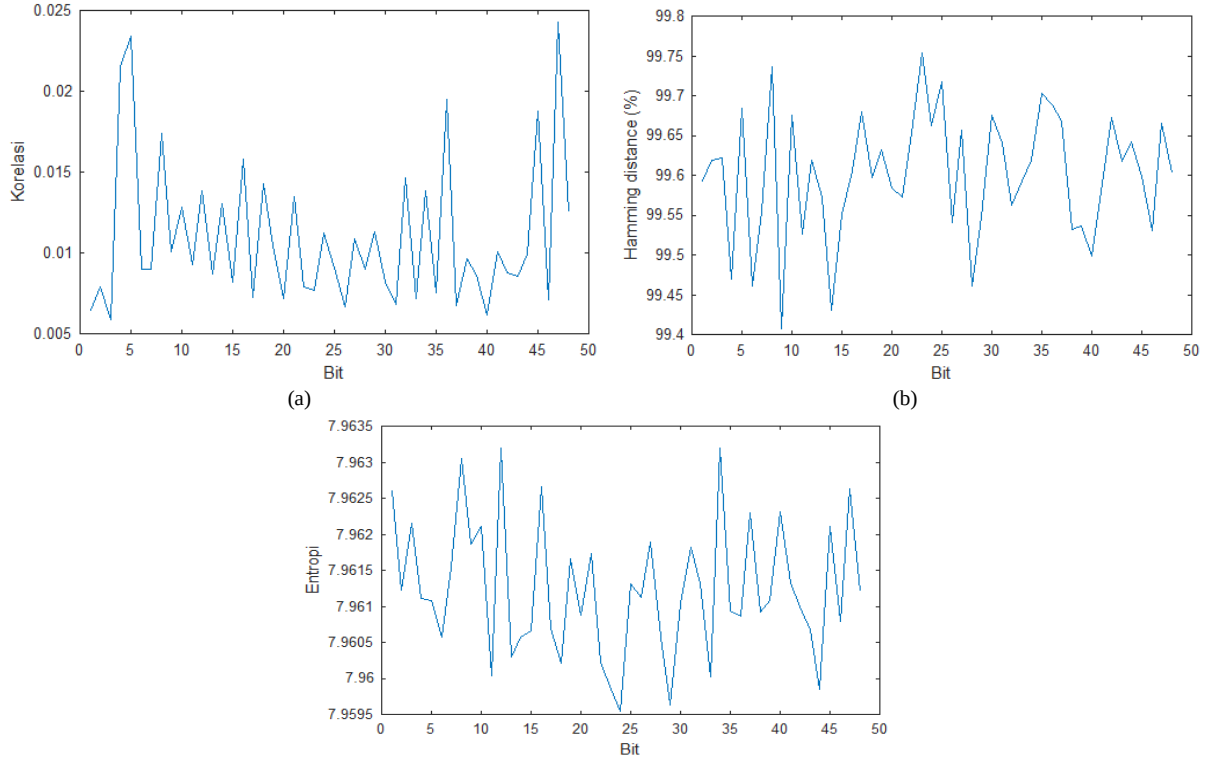
Gambar 8. Distribusi Frekuensi Karakter *Ciphertext* pada Dokumen "TOM TIT TOT.txt" dengan Kunci Rahasia 1H9d5X: (a) $r = [0{,}001\ ,\ 3{,}999]$; (b) $r = [3{,}671\ ,\ 3{,}999]$

Tabel 4. Hasil Pengujian Sensitivitas Kunci pada Algoritma *Stream Cipher* dengan APLCG, *Gray Code*, dan *Logistic Map* Standar

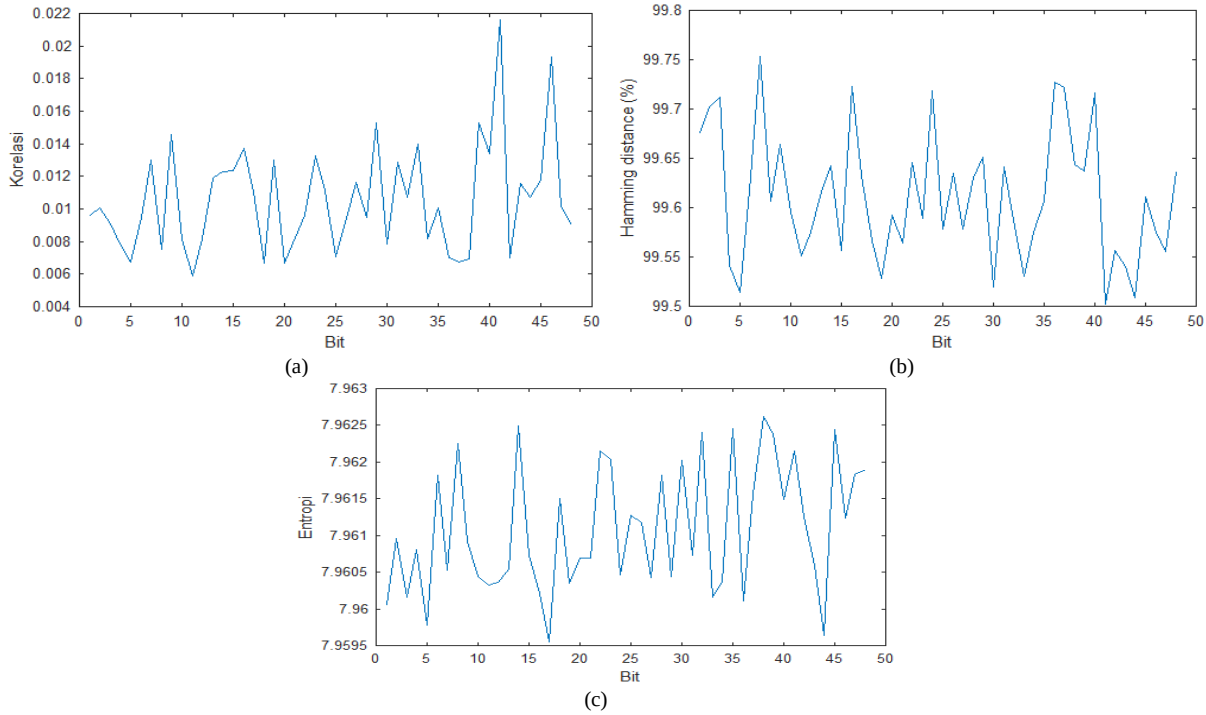
Kunci rahasia	Enkripsi			Dekripsi		
	Koefisien korelasi	Hamming *distance* (%)	Entropi	Koefisien korelasi	*Hamming distance* (%)	Entropi
1H9d5X	0,0110	99,6005	7,9612	0,0119	99,6370	7,2692
vTuGrb	0,0099	99,6017	7,9610	0,0117	99,6381	7,2693
170823	0,0103	99,6000	7,9612	0,0108	99,6364	7,2692
Mean	**0,0104**	**99,6007**	**7,9611**	**0,0115**	**99,6372**	**7,2692**

Tabel 5. Hasil Pengujian Sensitivitas Kunci pada Algoritma *Stream Cipher* dengan APLCG, *Gray Code*, dan *Logistic Map* Modifikasi

Kunci rahasia	Enkripsi			Dekripsi		
	Koefisien korelasi	Hamming *distance* (%)	Entropi	Koefisien korelasi	*Hamming distance* (%)	Entropi
1H9d5X	0,0106	99,6118	7,9611	0,0119	99,6477	7,2695
vTuGrb	0,0108	99,6202	7,9613	0,0116	99,6549	7,2690
170823	0,0131	99,5862	7,9613	0,0117	99,6231	7,2693
Mean	**0,0115**	**99,6061**	**7,9612**	**0,0117**	**99,6419**	**7,2693**

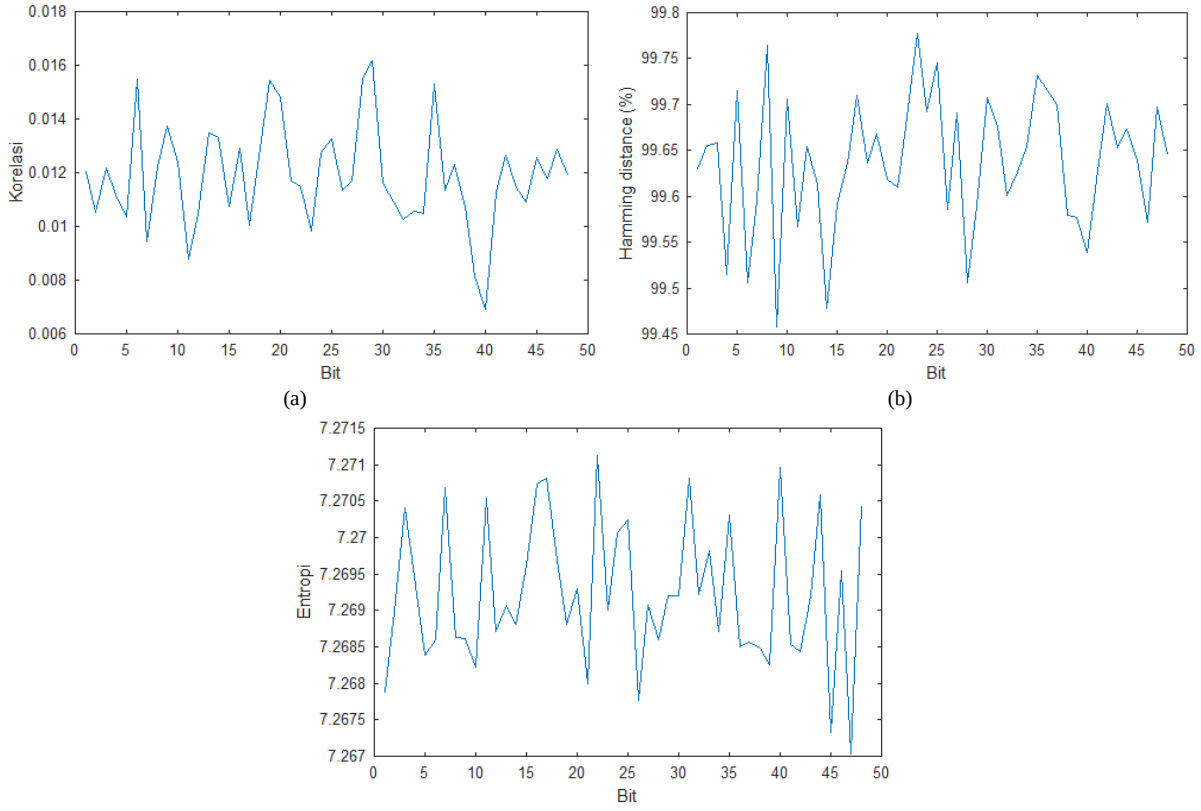


Gambar 9. Sensitivitas Kunci terhadap Perubahan Satu Bit Kunci Rahasia 1H9d5X pada Enkripsi dengan *Logistic Map* Standar: (a) Koefisien Korelasi; (b) Hamming *Distance*; (c) Entropi

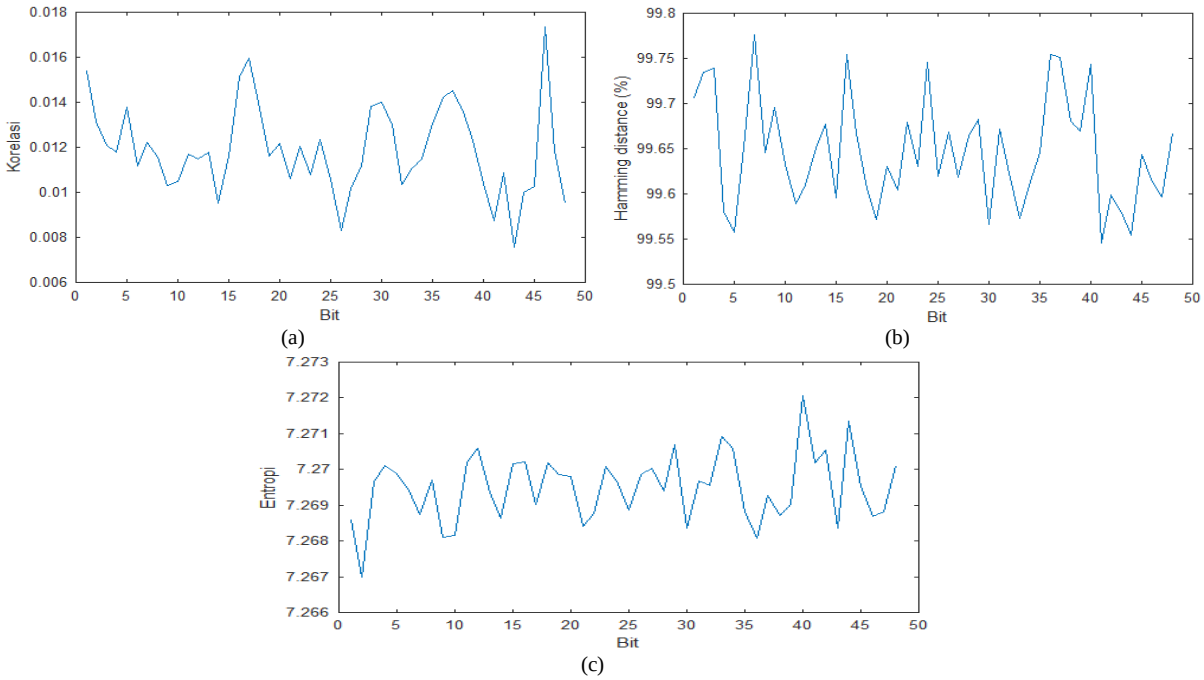


Gambar 10. Sensitivitas Kunci terhadap Perubahan Satu Bit Kunci Rahasia 1H9d5X pada Enkripsi dengan *Logistic Map* modifikasi: (a) Koefisien Korelasi; (b) Hamming *Distance*; (c) Entropi

Gambar 9 dan 10 menampilkan hasil analisis statistik untuk pengujian sensitivitas kunci pada *ciphertext* di salah satu dokumen uji dengan salah satu kunci. Tahapan yang digunakan merujuk pada Gambar 6(a). Hasil analisis statistik untuk pengujian sensitivitas kunci pada *plaintext* dalam salah satu dokumen uji dengan penggunaan kunci tertentu tergambar pada Gambar 11 dan 12. Langkah-langkah dalam tahap ini mengacu pada Gambar 6(b).



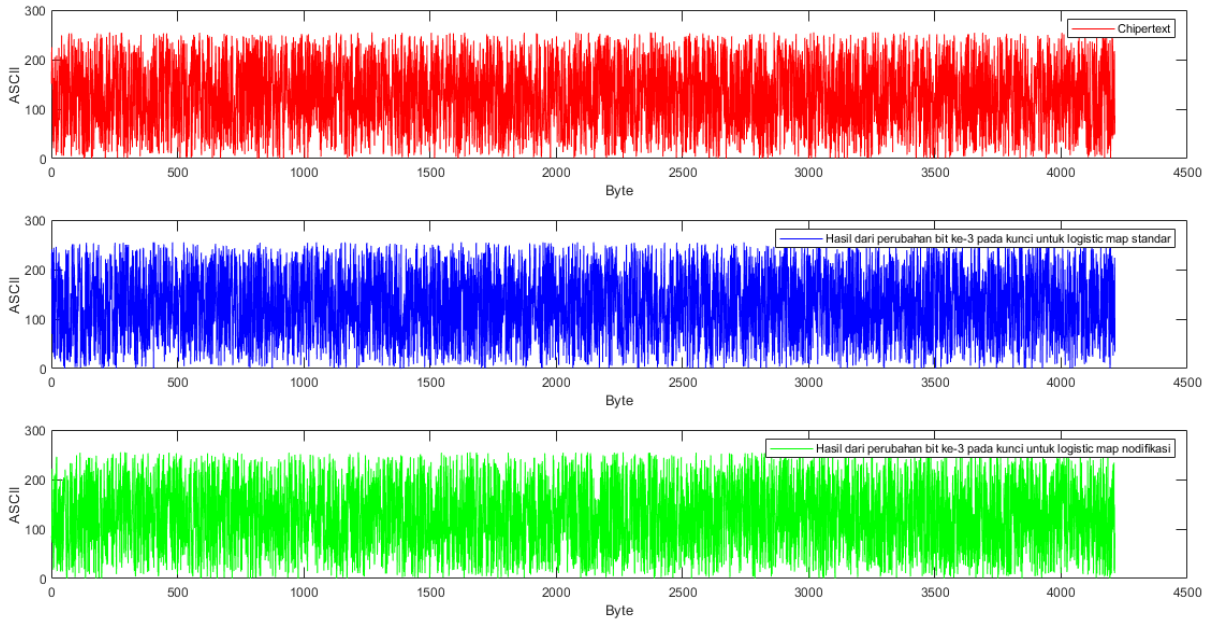
Gambar 11. Sensitivitas Kunci terhadap Perubahan Satu Bit Kunci Rahasia 1H9d5X pada Dekripsi dengan *Logistic Map* Standar: (a) Koefisien Korelasi; (b) Hamming *Distance*; (c) Entropi



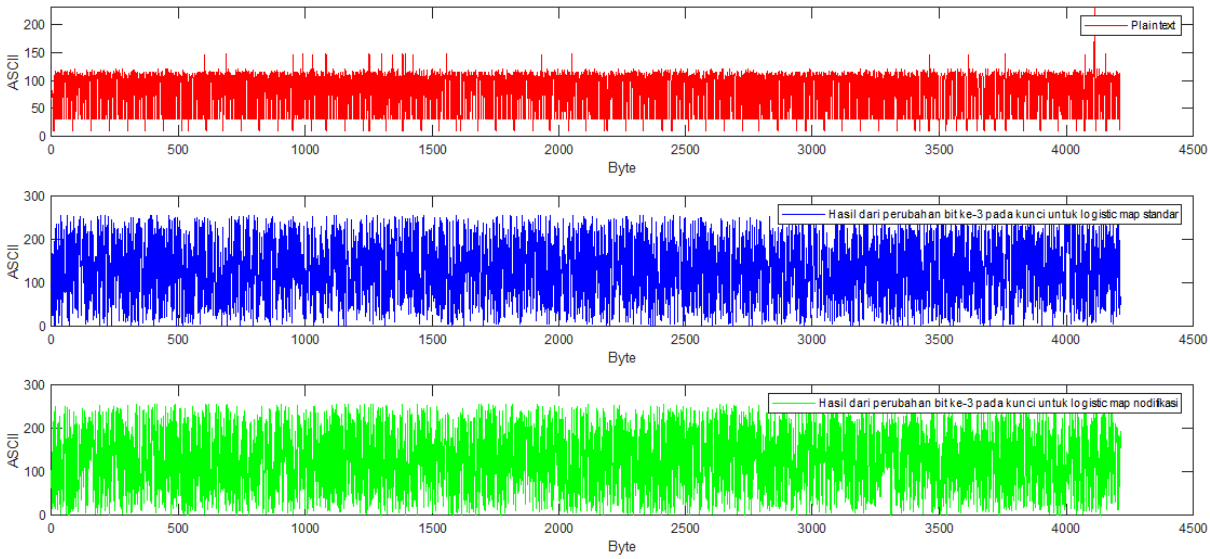
Gambar 12. Sensitivitas Kunci terhadap Perubahan Satu Bit Kunci Rahasia 1H9d5X pada Dekripsi dengan *Logistic Map* modifikasi: (a) Koefisien Korelasi; (b) Hamming *Distance*; (c) Entropi

Hasil Tabel 4-5 dapat diperkuat dengan memvisualisasikan hubungan nilai ASCII dari setiap byte *ciphertext* atau *plaintext*. Tampak pola yang berbeda baik dari kunci sebelum maupun setelah perubahan satu bit, sebagaimana tergambarkan pada Gambar 13. Serangkaian pengujian menunjukkan sensitivitas yang tinggi dari algoritma *stream cipher* kami terhadap perubahan satu bit kunci rahasia, dengan nilai koefisien korelasi, Hamming *distance*, dan entropi mendekati nilai terbaiknya.

DOI: https://doi.org/10.52436/1.jutif.2024.5.2.1535
p-ISSN: 2723-3863
e-ISSN: 2723-3871



(a)



(b)

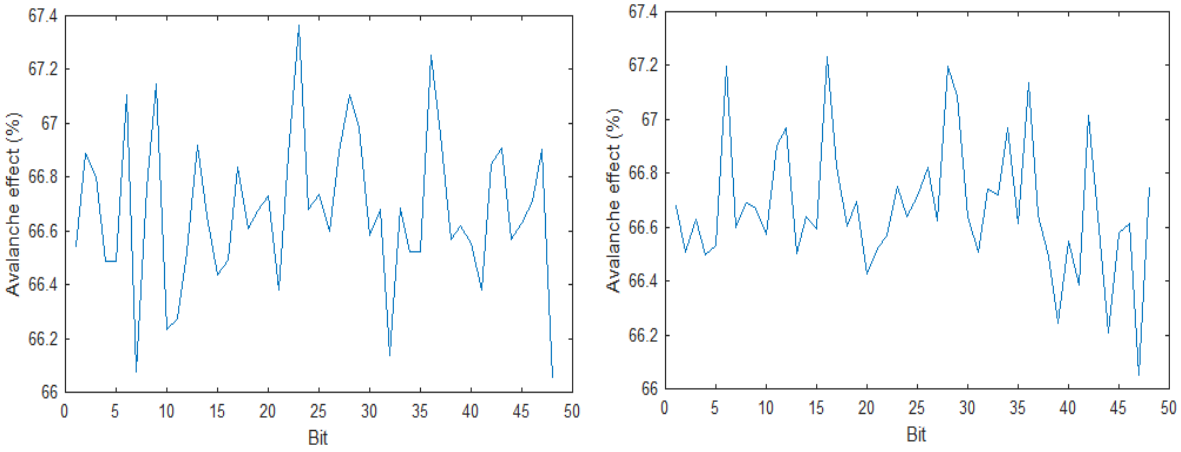
Gambar 13. Pola Hubungan Byte dan ASCII Akibat Perubahan bit ke-3 pada Kunci Rahasia 1H9d5X untuk Dokumen 'BINNORIE.txt': (a) Sensitivitas Kunci pada *Ciphertext*; (b) Sensitivitas Kunci pada *Plaintext*

4. DISKUSI

Pada penelitian [6], digunakan *tent map* untuk *Pseudo Random Number Generator* (PRNG), dimana dua parameter yang memiliki presisi 10^{-16} harus ditentukan sebelumnya. Hasil pengujian terhadap *plaintext* dengan ukuran 2000 karakter menghasilkan nilai entropi sebesar 7,2991. Pada penelitian kami, pengujian dilakukan terhadap 20 dokumen teks dengan rata-rata 6669,4 karakter pada setiap dokumen. Hasilnya menunjukkan bahwa nilai entropi rata-rata adalah 7,9615 untuk *logistic map* standar dan 7,9618 untuk *logistic map* modifikasi.

Suatu algoritma kriptografi dapat dianggap aman apabila menunjukkan *avalanche effect* yang kuat, yang tercermin dalam nilai *avalanche effect* yang tinggi [19]. Pengujian *avalanche effect* dilakukan dengan mengukur perubahan nilai bit pada *ciphertext* akibat adanya perubahan pada kunci. Dalam penelitian sebelumnya [20], algoritma Diffie-Hellman digunakan untuk mengubah kunci asli menjadi kunci baru. Dalam proses penciptaan kunci baru, terdapat empat parameter yang bersifat publik dan dua parameter yang bersifat privat. Keempat parameter publik ini harus ditentukan terlebih dahulu oleh kedua pihak yang berkomunikasi. Pada penelitian tersebut, satu byte atau karakter pada kunci rahasia diubah, dan rata-rata *avalanche effect* untuk 5 dokumen teks sebesar 45,5120%. Dalam penelitian kami, kami melakukan perubahan satu bit pada kunci rahasia, dimulai dari bit ke-1 hingga bit ke-48, dengan menggunakan 20 dokumen teks. Hasilnya adalah

rata-rata *avalanche effect* sebesar 66,6748% untuk *logistic map* standar dan 66,6740% untuk *logistic map* modifikasi. Informasi selengkapnya dapat dilihat pada Gambar 14.



(a) (b)

Gambar 14. *Avalanche Effect* dari Sensitivitas Kunci terhadap Perubahan Satu Bit Kunci Rahasia 1H9d5X pada Enkripsi: (a) *Logistic Map* Standar (b) *Logistic Map* Modifikasi

Perbandingan dengan dua penelitian lain menunjukkan bahwa algoritma kami memiliki kinerja lebih baik, dengan nilai entropi dan *avalanche effect* yang lebih besar. Dalam penelitian kami, parameter-parameter yang diperlukan, baik dalam *logistic map* maupun LCG, dihasilkan secara otomatis oleh sistem untuk mempermudah pengguna.

5. KESIMPULAN

Dalam penelitian ini, diusulkan enkripsi data teks menggunakan algoritma *stream cipher* dengan memanfaatkan fungsi *logistic map*. Seringkali pengguna dihadapkan pada keharusan menggunakan kunci dengan panjang tertentu dalam kriptografi, yang dapat menyulitkan dalam hal mengingatnya. Teknik *padding* dapat digunakan untuk memenuhi persyaratan ini. Keacakan dari kunci hasil *padding* dapat dicapai dengan mengacak posisi bit kunci menggunakan *Auto Parameters Linear Congruential Generator* (APLCG) dan *gray code*. APLCG memiliki kapabilitas untuk menghasilkan deretan bilangan bulat acak yang unik, di mana dua parameter sebagai penentunya dihitung secara otomatis. Pengujian yang melibatkan dua puluh dokumen teks menunjukkan bahwa perilaku *chaos* fungsi *logistic map* sangat dipengaruhi oleh parameter kontrol, terdapat tingkat kekacauan yang sangat sensitif terhadap perubahan kunci rahasia, baik pada enkripsi maupun dekripsi. Hal ini menandakan bahwa algoritma *stream cipher* kami sangat aman. Penelitian selanjutnya diarahkan pada penerapan algoritma *stream cipher* yang kami kembangkan untuk data citra digital.

UCAPAN TERIMA KASIH

Ucapan terima kasih kami tujukan kepada Program Studi Ilmu Komputer FST UNDANA selaku penyandang dana untuk kegiatan penelitian ini.

DAFTAR PUSTAKA

[1] B. F. Vajargah and R. Asghari, "Cryptographic Secure Pseudo-random Generation : The Chaotic Linear Congruential Generator (CLCG)," *Sci. Int.*, vol. 27, no. 3, pp. 1797–1801, 2015, [Online]. Available: http://www.sci-int.com/pdf/659911501 a 1797-1801 Meisam Asghari--MATH--IRAN--COMPOSED.pdf

[2] L. Diedrich, L. Murati, and A. Wiesmaier, "Stream ciphers in the IoT: Grain and Trivium," pp. 1–19, 2015, [Online]. Available: http://download.hrz.tu-darmstadt.de/media/FB20/Dekanat/Publikationen/CDC/GrainAndTrivium.pdf

[3] M. Yusuf, A. Arizal, and I. R. Hikmah, "Implementation Cryptography and Access Control on IoT-Based Warehouse Inventory Management System," *Matrik J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 22, no. 1, pp. 37–50, 2022, doi: 10.30812/matrik.v22i1.1849.

[4] N. Balaska, Z. Ahmida, A. Belmeguenai, and S. Boumerdassi, "Image encryption using a combination of Grain-128a algorithm and Zaslavsky chaotic map," *IET Image Process.*, vol. 14, no. 6, pp. 1120–1131, May 2020, doi: 10.1049/iet-ipr.2019.0671.

[5] M. Alawida, A. Samsudin, N. Alajarmeh, J. Sen Teh, M. Ahmad, and W. H. Alshoura, "A Novel Hash Function Based on a Chaotic Sponge and DNA Sequence," *IEEE Access*, vol. 9, pp. 17882–17897, 2021, doi: 10.1109/ACCESS.2021.3049881.

[6] E. Abass Albhrany, L. Fayeq Jalil, and H. Hadi Saleh, "New Text Encryption Algorithm Based on Block Cipher and

Chaotic Maps," vol. 2, no. 2, pp. 67–73, 2016.

[7] J. Machicao, M. Alves, M. S. Baptista, and O. M. Bruno, "Exploiting ergodicity of the logistic map using deep-zoom to improve security of chaos-based cryptosystems," *Int. J. Mod. Phys. C*, vol. 30, no. 05, p. 1950033, May 2019, doi: 10.1142/S0129183119500335.

[8] M. Elveny, R. Syah, I. Jaya, and I. Affandi, "Implementation of Linear Congruential Generator (LCG) Algorithm, Most Significant Bit (MSB) and Fibonacci Code in Compression and Security Messages Using Images," *J. Phys. Conf. Ser.*, vol. 1566, no. 1, p. 012015, Jun. 2020, doi: 10.1088/1742-6596/1566/1/012015.

[9] C. Han, "An image encryption algorithm based on modified logistic chaotic map," *Optik (Stuttg).*, vol. 181, pp. 779–785, Mar. 2019, doi: 10.1016/j.ijleo.2018.12.178.

[10] H. Ogras and M. Turk, "An efficient method to improve the Logistic map: Design and implementation," in *2016 Third International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA)*, Beirut, Lebanon: IEEE, Apr. 2016, pp. 24–28. doi: 10.1109/EECEA.2016.7470760.

[11] M. Alawida, "A novel chaos-based permutation for image encryption," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 35, no. 6, p. 101595, Jun. 2023, doi: 10.1016/j.jksuci.2023.101595.

[12] M. Alawida, J. Sen Teh, A. Mehmood, A. Shoufan, and W. H. Alshoura, "A chaos-based block cipher based on an enhanced logistic map and simultaneous confusion-diffusion operations," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 10, pp. 8136–8151, Nov. 2022, doi: 10.1016/j.jksuci.2022.07.025.

[13] R. S. Bhogal, B. Li, A. Gale, and Y. Chen, "Medical Image Encryption using Chaotic Map Improved Advanced Encryption Standard," *Int. J. Inf. Technol. Comput. Sci.*, vol. 10, no. 8, pp. 1–10, Aug. 2018, doi: 10.5815/ijitcs.2018.08.01.

[14] B. Krishnaveni and S. Sridhar, "Role of Distance Measures in Approximate String Matching Algorithms for Face Recognition System," in *IFIP Advances in Information and Communication Technology*, vol. 578, 2020, pp. 157–169. doi: 10.1007/978-3-030-63467-4_12.

[15] X. Zhang, L. Wang, G. Cui, and Y. Niu, "Entropy-Based Block Scrambling Image Encryption Using DES Structure and Chaotic Systems," *Int. J. Opt.*, vol. 2019, pp. 1–13, Aug. 2019, doi: 10.1155/2019/3594534.

[16] H. Wen *et al.*, "Design and Embedded Implementation of Secure Image Encryption Scheme Using DWT and 2D-LASM," *Entropy*, vol. 24, no. 10, p. 1332, Sep. 2022, doi: 10.3390/e24101332.

[17] G. Alvarez, D. Arroyo, and J. Nunez, "Application of gray code to the cryptanalysis of chaotic cryptosystems," *3rd Int. IEEE Sci. ...*, 2007, [Online]. Available: http://www.researchgate.net/publication/215732242_Application_of_Gray_code_to_the_cryptanalysis_of_chaotic_cryptosystems/file/79e415057608d30b9d.pdf

[18] Anonymous, "English Fairy Tales." Accessed: Oct. 05, 2023. [Online]. Available: https://www.gutenberg.org/files/7439/7439-0.txt

[19] A. F. Shimal, B. H. Helal, and A. T. Hashim, "Extended of TEA: A 256 bits block cipher algorithm for image encryption," *Int. J. Electr. Comput. Eng.*, vol. 11, no. 5, p. 3996, Oct. 2021, doi: 10.11591/ijece.v11i5.pp3996-4007.

[20] Permana langgeng wicaksono ellwid putra, C. A. Sari, and F. O. Isinkaye, "Secure Text Encryption for IoT Communication Using Affine Cipher and Diffie-Hellman Key Distribution on Arduino ATMEGA2560 IoT Devices," *J. Tek. Inform.*, vol. 4, no. 4, pp. 849–855, Aug. 2023, doi: 10.52436/1.jutif.2023.4.4.1129.