DAFTAR PUSTAKA

Arab, A., Rostami, M. J. dan Ghavami B. (2019). *An Image Encryption Method Based on Chaos System and AES Algorithm*. The Journal of Supercomputing, 75, 6663-6682.

Buchmann, J. A. (2001). *Introduction on to Cryptography*. Springer-Verlag, New York.

Cao, Y. (2013). *A New Hybrid Chaotic Map and Its Application on Image Encryption and Hiding*. Mathematical Problems in Engineering, 2013.

Faires, J. D., dan Burden, R. L. (2002). *Numerical Methods 3rd Edition*. Brooks Cole, Pacific Grove.

Gallian, J. A. (2021). *Contemporaray Abstract Algebra 10th Edition*. Taylor & Francis Group, Abingdon-on-Thames.

Glendinning, P. (1994). *Stability, Instability and Chaos: An Introduction to the Theory of Nonlinier Differential Equations*. Cambridge University Press, Cambridge.

Gonzalez, R. C., dan Woods, R. E. (2008). *Digital Image Processing 3rd Edition*. Pearson Education, New Jersey.

Hénon, M. (1976). *A Two-dimensional Mapping with a Strange Attractor*. Communications in Mathematical Physics, 50, 69-77.

Hussein, K. A., dan Mehdi, S. A., Hussein, S. A. (2019). *Image Encryption Based on Parallel Algorithm via Zigzag Manner with A New Chaotic System*. Journal of Southwest Jiaotong University, 54(4).

Ikeda, K. (1979). *Multiple-Valued Stationary State and Its Instability of the Transmitted Light by a Ring Cavity System*. Optics Communications, 30(2), 257-261.

Jacobson, N. (1985). *Basic Algebra I 2nd Edition*. W.H Freeman and Company, New York.

Jain, A. K. (1989). *Fundamentals of Digital Image Processing*. Prentice Hall, New Jersey.

Jung, K. H., dan Yoo, K. Y. (2009). *Data Hiding Method with Quality Control for Binary Images*. J. Software Engineering & Applications, 2, 20-24.

Kumar, T., dan Verma, K. (2010). *A Theory Based on Conversion of RGB image to Gray image*. International Journal of Computer Applications, 7(2), 7-10.

Liu, H., Kadir, A. dan Liu, J. (2019). *Color pathological image encryption algorithm using arithmetic over Galois field and coupled hyper chaotic system*. Optics and Lasers in Engineering, 122, 123-133.

Mardhiyah, A. dan Agus H. (2011). Metode Segmentasi Paru-Paru dan Jantung pada Citra X-Ray Thorax. Indonesian Journal of Elecronics and Instrumentation Systems, 1(2), 35-44.

May, R. M. (1974). *Biological Populations with Nonoverlapping Generations: Stable Points, Stable Cycles, and Chaos*. American Association for the Advancement of Science, 186(4164), 645-647.

Nagashima, H., dan Baba, Y. (1999). *Introduction to Chaos: Physics and Mathematics of Chaotic Phenomena*. IOP Publishing, Bristol.

Ruskey, F., Sawada, J., Cattell, K., Miers, C. R., dan Serra, M. (1970). *Generating Unlabeled Necklaces and Irreducible Polynomials over GF(2)*. Diakses dari https://www.researchgate.net/publication/228867414_Generating_Unlabeled_Necklaces_and_Irreducible_Polynomials_over_GF_2.

Vaidyanathan, S. (2017). *Analysis, synchronisation and circuit implementation of novel jerk chaotic system and its application for voice encryption*. Int. J. Modelling, Identification and Control, 28(2), 153-166.

LAMPIRAN

Lampiran 1.1 Kode Enkripsi

```
%%Input%%------------------------------------------

%Input Citra (plaintext)
namaFile = 'plain.png';
P = imread(namaFile);
%imshow(P);


%Input Parameter Sistem
h = 0.001;

d = 0.4; %d elemen [0.1,0,4]
u = 0.9; %u elemen [0.8, 0.85)U(0.85,0.9]
beta = 4; alpha = 1.4; %beta > alpha
miu = 4; %miu > 1
a = 3; b = 0.01; c = 3.8; p = 0.1; q = 0.1;
B = 10^20;


%Input Kunci Luar (Nilai Awal Sistem)
w0 = 0.1;
s0 = 0.1;
x0 = 0.1;
y0 = 0.1;
t0 = d - 6/(1+x0^2+y0^2);
z0 = 0.3; z10 = 0.3; z20 = 0.3;

%Input Kunci Polinomial Primitif Tak Tereduksi di Z/2Z[X]
berderajat 8
GalPol = 357;
%Membuat Tabel Perkalian
ElemenGal = gf(0:255,8,GalPol);
TabelKali = transpose(ElemenGal)*ElemenGal;


%Input Kunci Nilai Awal CM(0)
CMR0 = 244;
CMG0 = 239;
CMB0 = 47;


%%-------------------------------------------------
%%Pembangkitan Kunci%%------------------------------------------

%Menghitung ukuran citra
[H,W,Col] = size(P);
L = W*H;

%Membangkitkan kunci internal dengan peta chaos
%Menyelesaikan sistem chaos dengan Runge Kuttas
```

```
[Zz,w,s,x,y,t] =
cubjerk(L,z0,z10,z20,a,b,c,p,q,h,B,w0,s0,x0,y0,t0,d,u,beta,alpha,m
iu);
z = Zz(1,:); z1 = Zz(2,:); z2 = Zz(3,:);
%z adalah x_1 pada sistem cubic jerk, akan dihitung dengan Runge
Kutta,
%pengganti z2
%z1 adalah x_2 pada sistem cubic jerk, akan dihitung dengan Runge
Kutta
%z2 adalah x_3 pada sistem cubic jerk, akan dihitung dengan Runge
Kutta


%Men-generate key-stream
xR = mod(fix((w+s+x-t-z)*(10^14)),256);
yG = mod(fix((w+s+x-y-z)*(10^14)),256);
zB = mod(fix((w+s+x-y-t)*(10^14)),256);

%%--------------------------------------------------


%%--------------------------------------------------
%%Merentangkan Citra Menjadi Plaintext Vektor Baris%%-------------
-----------------------------
PP = zeros(1, L, 3);
for i = 1:3
PP(:,:,i) = reshape(transpose(P(:,:,i)), 1, []);
end
%%--------------------------------------------------


%%--------------------------------------------------
%%%Enkripsi Lapis 1: Melakukan operasi xor plaintext dengan kunci
menghasilkan protochipertext CX %%--------------------------------
----------
CR = bitxor(PP(:,:,1), xR);
CG = bitxor(PP(:,:,2), yG);
CB = bitxor(PP(:,:,3), zB);

%%--------------------------------------------------

%%--------------------------------------------------
%%Mengembalikan dimensi protociphertext (CX) %%-------------------
-----------------------
CX = zeros(H, W, 3);
CX(:,:,1) = transpose(reshape(CR, W, H));
CX(:,:,2) = transpose(reshape(CG, W, H));
CX(:,:,3) = transpose(reshape(CB, W, H));

%%--------------------------------------------------

%%--------------------------------------------------
%%%Enkripsi Lapis 2: Permutasi baris dan kolom setiap komponen
warna berdasarkan kunci %%----------------------------------------
--
```

```
%Permutasi Pertama (Mengacak Baris dan Kolom Seragam)
z1Cut = z1(1:H); %Menggunakan kunci turunan pertama cubjerk
[Z1, rowInd] = sort(z1Cut); %Membuat permutasi dengan mengurutkan
elemen di z1

z2Cut = z2(1:W); %Menggunakan kunci turunan kedua cubjerk
[Z2, colmInd] = sort(z2Cut); %Membuat permutasi dengan mengurutkan
elemen di z2

%Permutasi Kedua (Mengacak Baris dan Kolom Tiap Komponen Warna)
%Menggunakan kunci untuk permutasi baris
tperm = t(1:H); %Permutasi baris komp merah
[permRowR, rowIndR] = sort(tperm);
rowIndR = rowInd(rowIndR);

wperm = w(1:H); %Permutasi beris komp hijau
[permRowG, rowIndG] = sort(wperm);
rowIndG = rowInd(rowIndG);

sperm = s(1:H); %Permutasi baris komp biru
[permRowB, rowIndB] = sort(sperm);
rowIndB = rowInd(rowIndB);

%Menggunakan kunci untuk permutasi kolom
xperm = x(1:W); %Permutasi baris komp merah
[permColmR, colmIndR] = sort(xperm);
colmIndR = colmInd(colmIndR);

yperm = y(1:W); %Permutasi beris komp hijau
[permColmG, colmIndG] = sort(yperm);
colmIndG = colmInd(colmIndG);

zperm = z(1:W); %Permutasi baris komp biru
[permColmB, colmIndB] = sort(zperm);
colmIndB = colmInd(colmIndB);

%Pengekseksuian permutasi terhadap CX menjadi C1
C1 = zeros(H, W, 3);
C1(:,:,1) = CX(rowIndR,colmIndR,1);
C1(:,:,2) = CX(rowIndG,colmIndG,2);
C1(:,:,3) = CX(rowIndB,colmIndB,3);

%%-------------------------------------------------

%%-------------------------------------------------
%%Merentangkan Citra C1 Menjadi Vektor Baris%%---------------------
----------------------
C1Vect = zeros(1, L, 3);
for i = 1:3
C1Vect(:,:,i) = reshape(transpose(C1(:,:,i)), 1, []);
end
%%-------------------------------------------------

%%------------------------------------------------
%%Konversi Elemen di C1 Menjadi Polinomial di GF(2^8) %%----------
--------------------------------
C1Pol = gf(C1Vect,8,GalPol);
```

```
%%-------------------------------------------------

%%-----------------------------------------------
%%Men-generate key-stream untuk perkalian polinomial di GF %%-----
-------------------------------------
xRM = mod(fix((y+t+z-s-x)*(10^14)),256);
xRM = max(xRM, 1);
yGM = mod(fix((y+t+z-w-x)*(10^14)),256);
yGM = max(yGM, 1);
zBM = mod(fix((y+t+z-w-s)*(10^14)),256);
zBM = max(zBM, 1);

%%-------------------------------------------------



%%-----------------------------------------------
%%%Enkripsi Lapis 3: Enkripsi menggunakan penjumlahan dan
perkalian di GF(2^8) menghasilkan chipertext CM%%-----------------
-------------------------
CM = gf(zeros(1, L, 3),8,GalPol);

CM(1,1,1) = (C1Pol(1,1,1) + CMR0)*xRM(1);
CM(1,1,2) = (C1Pol(1,1,2) + CMG0)*yGM(1);
CM(1,1,3) = (C1Pol(1,1,3) + CMB0)*zBM(1);

for i = 2:L
    alpha = (C1Pol(1,i,1) + CM(1,i-1,1));
    CM(1,i,1) = TabelKali(alpha.x + 1, xRM(i) + 1);

    alpha = (C1Pol(1,i,2) + CM(1,i-1,2));
    CM(1,i,2) = TabelKali(alpha.x + 1, yGM(i) +1);

    alpha = (C1Pol(1,i,3) + CM(1,i-1,3));
    CM(1,i,3) = TabelKali(alpha.x + 1, zBM(i) + 1);

end
%%-------------------------------------------------



%%-------------------------------------------------
%%Mengembalikan dimensi ciphertext (CM) %%------------------------
------------------
CMTrueDim = gf(zeros(H, W, 3),8,GalPol);
CMTrueDim(:,:,1) = transpose(reshape(CM(:,:,1), W, H));
CMTrueDim(:,:,2) = transpose(reshape(CM(:,:,2), W, H));
CMTrueDim(:,:,3) = transpose(reshape(CM(:,:,3), W, H));
imshow(uint8(CMTrueDim.x));

imwrite(uint8(CMTrueDim.x),'cipher.png');
%%-------------------------------------------------
```

Lampiran 1.2 Kode Fungsi Sistem *Chaos Hybrid* Baru

```
function [Zz,w,s,x,y,t] =
cubjerk(L,z0,z10,z20,a,b,c,p,q,h,B,w0,s0,x0,y0,t0,d,u,beta,alpha,m
iu)

N = L;

%Membangkitkan kunci internal, definisikan variabel
t = zeros(1, L); t(1) = t0; %t,w,s,x,y adalah variable pada sistem
hybrid
w = zeros(1, L); w(1) = w0;
s = zeros(1, L); s(1) = s0;
x = zeros(1, L); x(1) = x0;
y = zeros(1, L); y(1) = y0;

Zz = zeros(3,N);

Zz(1,1) = z0;
Zz(2,1) = z10;
Zz(3,1) = z20;

k = zeros(4,3);

for j = 1:N-1
  w(j+1) = 1 + u*(w(j)*cos(t(j))-s(j)*sin(t(j))); %pengganti x1
  s(j+1) =  u*(w(j)*sin(t(j))-s(j)*cos(t(j))); %pengganti x2
  x(j+1) = 1 - alpha*s(j+1)^2 + beta*w(j+1); %pengganti y1
  y(j+1) = miu*w(j+1)*(1-s(j+1)); %pengganti y2
  t(j+1) = d - 6/(1+x(j+1)^2+y(j+1)^2); %pengganti z1


k(1,1) = h*Zz(2,j);
k(1,2) = h*Zz(3,j);
k(1,3) = h*f3(a,b,c,p,q,Zz(1,j),Zz(2,j),Zz(3,j));

k(2,1) = h*(Zz(2,j) + k(1,2)/2);
k(2,2) = h*(Zz(3,j) + k(1,3)/2);
k(2,3) =
h*f3(a,b,c,p,q,Zz(1,j)+k(1,1)/2,Zz(2,j)+k(1,2)/2,Zz(3,j)+k(1,3)/2)
;

k(3,1) = h*(Zz(2,j) + k(2,2)/2);
k(3,2) = h*(Zz(3,j) + k(2,3)/2);
k(3,3) =
h*f3(a,b,c,p,q,Zz(1,j)+k(2,1)/2,Zz(2,j)+k(2,2)/2,Zz(3,j)+k(2,3)/2)
;

k(4,1) = h*(Zz(2,j) + k(3,2));
k(4,2) = h*(Zz(3,j) + k(3,3));
k(4,3) =
h*f3(a,b,c,p,q,Zz(1,j)+k(3,1),Zz(2,j)+k(3,2),Zz(3,j)+k(3,3));

for i = 1:3
    Zz(i,j+1) = Zz(i,j) + (k(1,i)+2*k(2,i)+2*k(3,i)+k(4,i))/6;
    if or(or(isnan(Zz(i,j+1)),Zz(i,j+1)==Inf),abs(Zz(i,j+1))>B)
```

```
        an = (t(j)+w(j)+x(j)+y(j)+s(j))/5;
        Zz(i,j+1) = Zz(i,1)+an*(10^-14);
    end
end

end

end
```

Lampiran 1.3 Fungsi f_3

```
function hasil = f3(a,b,c,p,q,z1,z2,z3)

hasil = -a*z1 + b*z1*z2 - c*z3 + p*z1*z2*z2 - q*z1^3;

end
```

Lampiran 1.4 Kode Dekripsi

```
%%Input%%------------------------------------------

%Input Citra (ciphertext)
namaFile = 'cipher.png';
CMTrueDim = imread(namaFile);
%imshow(P);

%Input Parameter Sistem
h = 0.001;

d = 0.4; %d elemen [0.1,0,4]
u = 0.9; %u elemen [0.8, 0.85)U(0.85,0.9]
beta = 4; alpha = 1.4; %beta > alpha
miu = 4; %miu > 1
a = 3; b = 0.01; c = 3.8; p = 0.1; q = 0.1;
B = 10^20;


%Input Kunci Luar (Nilai Awal Sistem)
w0 = 0.1;
s0 = 0.1;
x0 = 0.1;
y0 = 0.1;
t0 = d - 6/(1+x0^2+y0^2);
z0 = 0.3; z10 = 0.3; z20 = 0.3;

%Input Kunci Polinomial Primitif Tak Tereduksi di Z/2Z[X]
berderajat 8
GalPol = 357;
%Membuat Tabel Perkalian
ElemenGal = gf(0:255,8,GalPol);
TabelKali = transpose(ElemenGal)*ElemenGal;


%Input Kunci Nilai Awal CM(0)
CMR0 = 244;
CMG0 = 239;
CMB0 = 47;


%%------------------------------------------------
%%Pembangkitan Kunci%%------------------------------------------

%Menghitung ukuran citra
[H,W,Col] = size(CMTrueDim);
L = W*H;

%Membangkitkan kunci internal dengan peta chaos
%Menyelesaikan sistem chaos dengan Runge Kuttas
[Zz,w,s,x,y,t] =
cubjerk(L,z0,z10,z20,a,b,c,p,q,h,B,w0,s0,x0,y0,t0,d,u,beta,alpha,m
iu);
z = Zz(1,:); z1 = Zz(2,:); z2 = Zz(3,:);
%z adalah x_1 pada sistem cubic jerk, akan dihitung dengan Runge
```

```
Kutta,
%pengganti z2
%z1 adalah x_2 pada sistem cubic jerk, akan dihitung dengan Runge
Kutta
%z2 adalah x_3 pada sistem cubic jerk, akan dihitung dengan Runge
Kutta


%Men-generate key-stream
xR = mod(fix((w+s+x-t-z)*(10^14)),256);
yG = mod(fix((w+s+x-y-z)*(10^14)),256);
zB = mod(fix((w+s+x-y-t)*(10^14)),256);

%%---------------------------------------------------

%%--------------------------------------------------
%%Men-generate key-stream untuk perkalian polinomial di GF %%-----
-------------------------------------
xRM = mod(fix((y+t+z-s-x)*(10^14)),256);
xRM = max(xRM, 1);
yGM = mod(fix((y+t+z-w-x)*(10^14)),256);
yGM = max(yGM, 1);
zBM = mod(fix((y+t+z-w-s)*(10^14)),256);
zBM = max(zBM, 1);

%%---------------------------------------------------



%%---------------------------------------------------
%%Merentangkan Citra Menjadi Ciphertext Vektor Baris%%------------
------------------------------
CM = zeros(1, L, 3);
for i = 1:3
CM(:,:,i) = reshape(transpose(CMTrueDim(:,:,i)), 1, []);
end
%%---------------------------------------------------

%%--------------------------------------------------
%%%Dekripsi Lapis 3: Enkripsi menggunakan penjumlahan dan
perkalian di GF(2^8) menghasilkan chipertext C1%%-----------------
-------------------------
CMPol = gf(CM,8,GalPol);
CMPolShift = gf(zeros(1, L, 3),8,GalPol);
CMPolShift(1,1,1) = CMR0; CMPolShift(1,2:L,1) = CMPol(1,1:L-1,1);
CMPolShift(1,1,2) = CMG0; CMPolShift(1,2:L,2) = CMPol(1,1:L-1,2);
CMPolShift(1,1,3) = CMB0; CMPolShift(1,2:L,3) = CMPol(1,1:L-1,3);
C1Pol = gf(zeros(1, L, 3),8,GalPol);

C1Pol(:,:,1) = (CMPol(:,:,1)./xRM) - CMPolShift(:,:,1);
C1Pol(:,:,2) = (CMPol(:,:,2)./yGM) - CMPolShift(:,:,2);
C1Pol(:,:,3) = (CMPol(:,:,3)./zBM) - CMPolShift(:,:,3);

C1 = C1Pol.x;

%%---------------------------------------------------
```

```
%%-----------------------------------------------------
%%Mengembalikan dimensi C1 %%-----------------------------------------
-----
C1TrueDim = zeros(H, W, 3);
C1TrueDim(:,:,1) = transpose(reshape(C1(:,:,1), W, H));
C1TrueDim(:,:,2) = transpose(reshape(C1(:,:,2), W, H));
C1TrueDim(:,:,3) = transpose(reshape(C1(:,:,3), W, H));

%%-----------------------------------------------------



%%-----------------------------------------------------
%%%Dekripsi Lapis 2: Permutasi balikan baris dan kolom setiap
komponen warna berdasarkan kunci %%-------------------------------
-----------

%Permutasi dibuat terlebih dahulu kemudian seperti pada enkripsi
kemudian dicari permutasi inversnya

%Permutasi Pertama (Mengacak Baris dan Kolom Seragam)
z1Cut = z1(1:H); %Menggunakan kunci turunan pertama cubjerk
[Z1, rowInd] = sort(z1Cut); %Membuat permutasi dengan mengurutkan
elemen di z1


z2Cut = z2(1:W); %Menggunakan kunci turunan kedua cubjerk
[Z2, colmInd] = sort(z2Cut); %Membuat permutasi dengan mengurutkan
elemen di z1

%Permutasi Kedua (Mengacak Baris dan Kolom Tiap Komponen Warna)
%Menggunakan kunci untuk permutasi baris
tperm = t(1:H); %Permutasi baris komp merah
[permRowR, rowIndR] = sort(tperm);
rowIndR = rowInd(rowIndR);
[permRowRInv, rowIndRInv] = sort(rowIndR); %Membuat permutasi
invers dari rowIndR

wperm = w(1:H); %Permutasi beris komp hijau
[permRowG, rowIndG] = sort(wperm);
rowIndG = rowInd(rowIndG);
[permRowGInv, rowIndGInv] = sort(rowIndG); %Membuat permutasi
invers dari rowIndG

sperm = s(1:H); %Permutasi baris komp biru
[permRowB, rowIndB] = sort(sperm);
rowIndB = rowInd(rowIndB);
[permRowBInv, rowIndBInv] = sort(rowIndB); %Membuat permutasi
invers dari rowIndB

%Menggunakan kunci untuk permutasi kolom
xperm = x(1:W); %Permutasi baris komp merah
[permColmR, colmIndR] = sort(xperm);
colmIndR = colmInd(colmIndR);
[permColmRInv, colmIndRInv] = sort(colmIndR); %Membuat permutasi
invers dari colmIndR
```

```
yperm = y(1:W); %Permutasi beris komp hijau
[permColmG, colmIndG] = sort(yperm);
colmIndG = colmInd(colmIndG);
[permColmGInv, colmIndGInv] = sort(colmIndG); %Membuat permutasi
invers dari colmIndG

zperm = z(1:W); %Permutasi baris komp biru
[permColmB, colmIndB] = sort(zperm);
colmIndB = colmInd(colmIndB);
[permColmBInv, colmIndBInv] = sort(colmIndB); %Membuat permutasi
invers dari colmIndB

%Pengekseksuian permutasi invers terhadap C1 menjadi CX
CX = zeros(H, W, 3);
CX(:,:,1) = C1TrueDim(rowIndRInv,colmIndRInv,1);
CX(:,:,2) = C1TrueDim(rowIndGInv,colmIndGInv,2);
CX(:,:,3) = C1TrueDim(rowIndBInv,colmIndBInv,3);
%%-----------------------------------------------

%%-----------------------------------------------
%%Merentangkan CX Menjadi Vektor Baris%%--------------------------
----------------
CXVect = zeros(1, L, 3);
for i = 1:3
CXVect(:,:,i) = reshape(transpose(CX(:,:,i)), 1, []);
end
%%-----------------------------------------------



%%-----------------------------------------------
%%%Dekripsi Lapis 1: Melakukan operasi xor CX dengan kunci
menghasilkan plaintext P %%-------------------------------------
---
P = zeros(1,L,3);
P(:,:,1) = bitxor(CXVect(:,:,1), xR);
P(:,:,2) = bitxor(CXVect(:,:,2), yG);
P(:,:,3) = bitxor(CXVect(:,:,3), zB);

%%-----------------------------------------------

%%-----------------------------------------------
%%Mengembalikan dimensi P %%-------------------------------------
----
PTrueDim = zeros(H, W, 3);
PTrueDim(:,:,1) = transpose(reshape(P(:,:,1), W, H));
PTrueDim(:,:,2) = transpose(reshape(P(:,:,2), W, H));
PTrueDim(:,:,3) = transpose(reshape(P(:,:,3), W, H));


imshow(uint8(PTrueDim));

imwrite(uint8(PTrueDim),'decrypted.png');
%%-----------------------------------------------
```