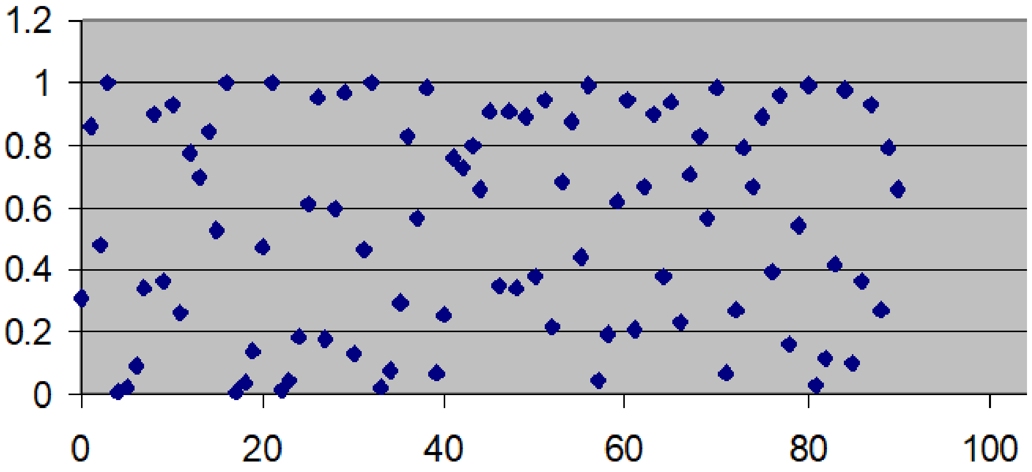


Pembangkit Bilangan Acak

Bahan Kuliah IF4020 Kriptografi

Program Studi Teknik Informatika

STEI-ITB

Bilangan Acak

- Bilangan acak: bilangan yang tidak dapat diprediksi nilainya

- Bilangan acak (*random*) sering digunakan di dalam kriptografi

- Contoh:

  - untuk pembangkitan parameter kunci pada algoritma kunci-publik,

  - pembangkitan *initialization vector* (*IV*) pada block-cipher

- Tidak ada prosedur komputasi yang menghasilkan deret bilangan acak yang benar-benar sempurna (*truly random*).

- Bilangan acak yang dihasilkan dengan prosedur komputasi adalah bilangan acak semu (*pseudo-random*), karena pembangkitan bilangannya dapat diulang kembali.

- Pembangkit deret bilangan acak semacam itu disebut *pseudo-random number generator* (*PRNG*).

- PRNG bersifat deterministik, artinya bilangan acak bisa ditentukan asalkan kunci (umpan) yang digunakan untuk pembangkitannya diketahui.

Linear Congruential Generator (LCG)

Pembangkit bilangan acak kongruen-lanjar adalah *PRNG* yang berbentuk:

$$X_n = (aX_{n-1} + b) \bmod m$$

X_{n} = bilangan acak ke-*n* dari deretnya

X_{n-1} = bilangan acak sebelumnya

a = faktor pengali

b = *increment*

m = modulus

Kunci pembangkit adalah X_0 yang disebut **umpan** (*seed*).

Contoh: $X_n = (7X_{n-1} + 11) \bmod 17$, dan $X_0 = 0$

n	X_n
0	0
1	11
2	3
3	15
4	14
5	7
6	9
7	6
8	2
9	8
10	16
11	4
12	5
13	12
14	10
15	13
16	*0*
17	*11*
18	*3*
19	*15*
20	*14*
21	*7*
22	*9*
23	*6*
24	*2*

- *LCG* mempunyai periode tidak lebih besar dari *m*, dan pada kebanyakan kasus periodenya kurang dari itu.

- *LCG* mempunyai periode penuh (m – 1) jika memenuhi syarat berikut:
  1. *b* relatif prima terhadap *m*.
  2. *a* – 1 dapat dibagi dengan semua faktor prima dari *m*
  3. *a* – 1 adalah kelipatan 4 jika *m* adalah kelipatan 4
  4. $m >$ maks(a, b, x_0)
  5. $a > 0$, $b > 0$

- Keunggulan *LCG* terletak pada kesederhanaannya dan komputasi yang cepat.

- Sayangnya, *LCG* tidak dapat digunakan untuk kriptografi karena bilangan acaknya dapat diprediksi urutan kemunculannya.

- Oleh karena itu *LCG* tidak aman digunakan untuk kriptografi.

- Namun demikian, *LCG* tetap berguna untuk aplikasi non-kriptografi seperti simulasi, sebab *LCG* mangkus dan memperlihatkan sifat statistik yang bagus dan sangat tepat untuk uji-uji empirik

Pembangkit Bilangan Acak yang Aman Secara Kriptografi

- Pembangkit bilangan acak yang cocok untuk kriptografi dinamakan *cryptographically secure pseudorandom generator* (*CSPRNG*).

- Persyaratan *CSPRNG* adalah:
  1. Secara statistik ia mempunyai sifat-sifat yang bagus (yaitu lolos uji keacakan statistik).
  2. Tahan terhadap serangan (*attack*) yang serius. Serangan ini bertujuan untuk memprediksi bilangan acak yang dihasilkan.

Blum Blum Shub (BBS)

- *CSPRNG* yang paling sederhana dan paling mangkus (secara kompleksitas teoritis).

- *BBS* dibuat pada tahun 1986 oleh Lenore Blum, Manuel Blum, dan Michael Shub.

- Berbasis teori bilangan (*number theory*)

Algoritma:

1. Pilih dua buah bilangan prima rahasia, *p* dan *q*, yang masing-masing kongruen dengan 3 (mod 4).
2. Kalikan keduanya menjadi $n = pq$. Bilangan *n* ini disebut **bilangan bulat Blum**
3. Pilih bilangan bulat acak lain, *s*, sebagai umpan sedemikian sehingga:
   (i) $2 \leq s < n$
   (ii) *s* dan *n* relatif prima

   kemudian hitung $x_0 = s^2 \bmod n$
4. Barisan bit acak dihasilkan dengan melakukan iterasi berikut sepanjang yang diinginkan:
   (i) Hitung $x_i = x_{i-1}^{\ 2} \bmod n$
   (ii) z_i = bit *LSB* (*Least Significant Bit*) dari x_i

   Barisan bit acak adalah $z_1, z_2, z_3, \ldots$

Contoh. Misalkan kita memilih p = 11 dan q = 23 sehingga $n = pq$ = 253.

Kita pilih s = 3 dan kita hitung $x_0 = 3^2$ mod 253 = 9.

Barisan bit acak kita hasilkan sebagai berikut:

$x_1 = x_0^2$ mod n = 9^2 mod 253 = 81 ➔ z_1 = 1 (karena 81 ganjil, bit *LSB*-nya pasti 1)

$x_2 = x_1^2$ mod n = 81^2 mod 253 = 236 ➔ z_2 = 0 (karena 236 genap, bit *LSB*-nya pasti 0)

$x_3 = x_2^2$ mod n = 236^2 mod 253 = 36 ➔ z_3 = 0

$x_4 = x_3^2$ mod n = 36^2 mod 253 = 31 ➔ z_4 = 1

$x_5 = x_4^2$ mod n = 31^2 mod 253 = 202 ➔ z_5 = 0

dst

Barisan bit acak yang dihasilkan 10010..

- Perhatikan rumus pembangkitan bilangan acak:

$$x_0 = s^2 \bmod n$$
$$x_i = x_{i-1}^2 \bmod n$$

- Keamanan BBS: sulit menghitung akar kuadratik sebuah bilangan dalam modulus *n* jika *n* adalah hasil kali bilangan prima *p* dan *q* dan faktorisasi *n* menjadi *p* dan *q* tidak diketahui.

- jika n dapat difaktorkan menjadi p dan q, maka bilangan acak dapat dibangkitkan langsung dengan persamaan:

$$x_i = x_0^{2^i \bmod ((p-1)(q-1))}$$

- Bilangan acak tidak harus 1 bit *LSB* tetapi bisa juga *j* buah bit (*j* adalah bilangan bulat positif yang tidak melebihi $\log_2(\log_2 n)$).

- Perhatikan contoh berikut:

Contoh . Misalkan p = 11351 dan q = 11987 sehingga $n = pq$ = 136064437.
Kita pilih s = 80331757 dan j = 4
(j tidak melebihi $\log_2(\log_2 136064437) = 4.75594$).

Hitung: $x_0 = 80331757^2 \bmod 136064437 = 1312737111$.
Barisan bit acak yang dihasilkan adalah sebagai berikut:

$x_1 = x_0^2 \bmod n = 131273718^2 \bmod 136064437 = 47497112$

$z_1 = 47497112 \equiv 8 \pmod{2^4} = 1000_{\text{basis 2}}$ (4 bit *LSB* dari 47497112)

$x_2 = x_1^2 \bmod n = 47497112^2 \bmod 136064437 = 69993144$

$z_2 = 69993144 \equiv 8 \pmod{2^4} = 1000_{\text{basis 2}}$ (4 bit *LSB* dari 69993144)

…

$x_3 = x_2^2 \bmod n = 69993144^2 \bmod 136064437 = 13810821$

$z_3 = 13810821 \equiv 5 \pmod{2^4} = 0101_{\text{basis 2}}$ (4 bit *LSB* dari 13810821)

…

Barisan blok bit acak yang dihasilkan: 1000 1000 0101 …

- Keamanan *BBS* terletak pada sulitnya memfaktorkan *n*. Nilai *n* tidak perlu rahasia dan dapat diumumkan kepada publik.

- *BBS* tidak dapat diprediksi dari arah kiri (*unpredictable to the left*) dan tidak dapat diprediksi dari arah kanan (*unpredictable to the kanan*),

- artinya jika diberikan barisan bit yang dihasilkan oleh *BBS*, kriptanalis tidak dapat memprediksi barisan bit sebelumnya dan barisan bit sesudahnya

CSPRNG Berbasis *RSA*

Algoritma:

1. Pilih dua buah bilangan prima rahasia, p dan q, dan bilangan bulat e yang relatif prima dengan $(p - 1)(q - 1)$
2. Kalikan keduanya menjadi $n = pq$
3. Pilih bilangan bulat acak lain, s, sebagai x_0 yang dalam hal ini $2 \leq s \leq n$
4. Barisan bit acak dihasilkan dengan melakukan iterasi berikut sepanjang yang diinginkan:

   (i) Hitung $x_i = x_{i-1}^{\,e} \bmod n$ dengan $x_0 = s$.

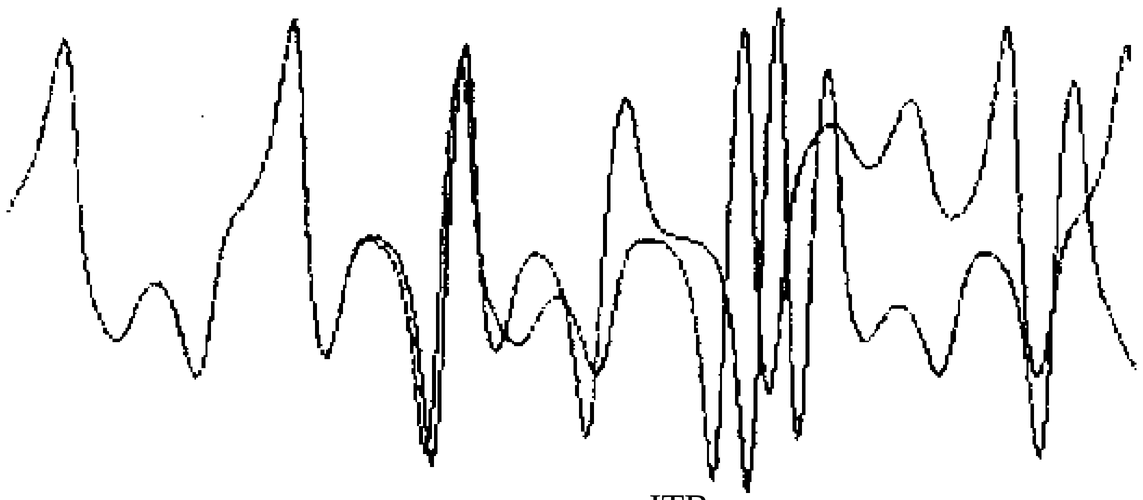
   (ii) z_i = bit *LSB* (*Least Significant Bit*) dari x_i

5. Barisan bit acak adalah $z_1, z_2, z_3, \ldots$

Teori *Chaos*

- Teori *chaos* menggambarkan perilaku sistem dinamis nirlanjar yang menunjukkan fenomena *chaos*.

- Salah satu karakteristik sistem *chaos*: **peka pada nilai awal** (*sensitive dependence on initial condition*).



Logistic Map

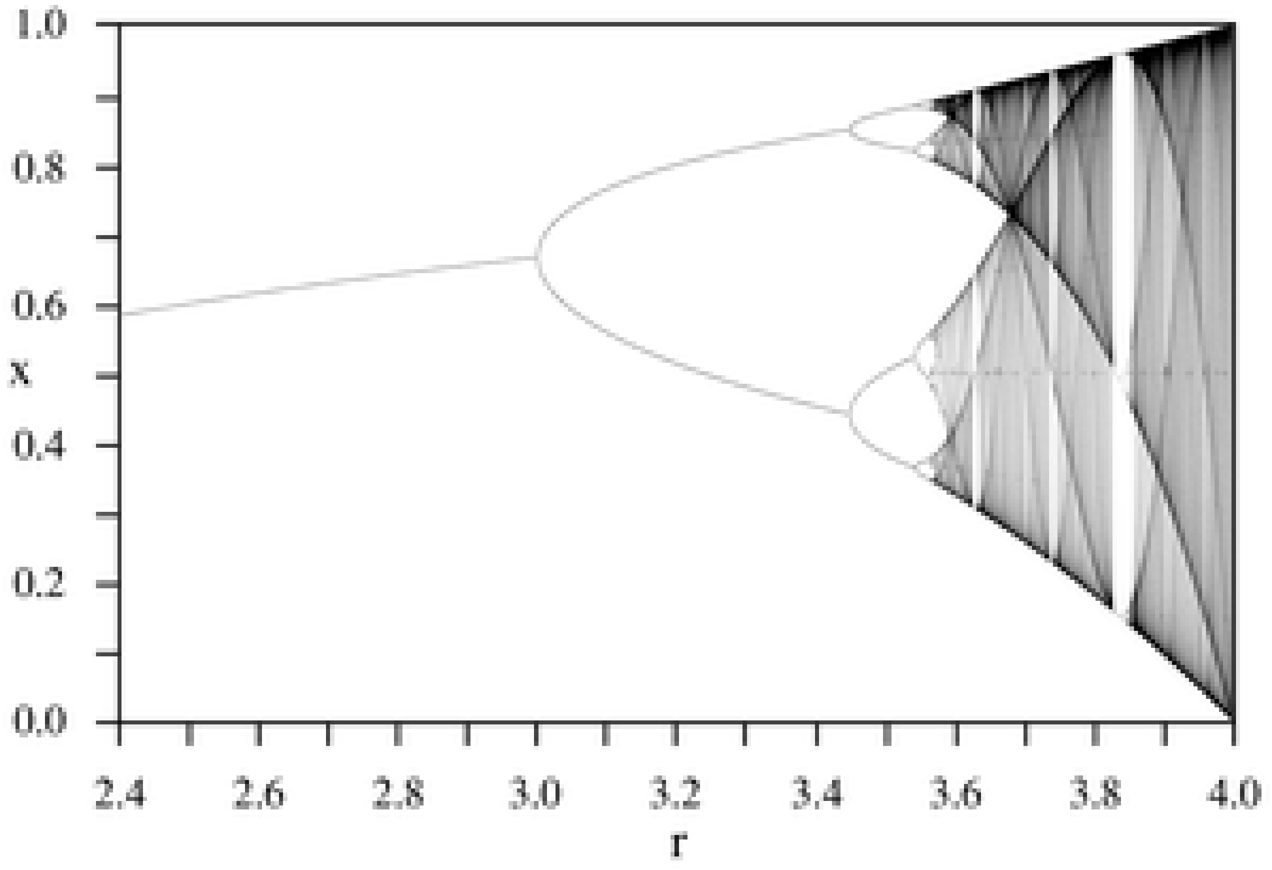
- Contoh fungsi *chaos*: persamaan logistik (*logistic map*)

$$x_{i+1} = r\, x_i\, (1 - x_i)$$

r : laju pertumbuhan ($0 \leq r \leq 4$)

x : nilai-nilai *chaos* ($0 \leq x \leq 1$)

Logistic Map



$$x_{i+1} = r\, x_i\,(1 - x_i)$$

Gambar 1 Diagram *bifurcation* untuk persamaan logistik $x_{i+1} = r\, x_i\,(1 - x_i)$

Logistic Map

- Sistem *chaos* berguna untuk pembangkitan bilangan acak

$$x_{i+1} = r\, x_i\, (1 - x_i)$$

- Misal r = 4.0 dan nilai awal x_0 = 0.456

$x_1 = 4.0x_0(1 - x_0) = 0.992256$

$x_2 = 4.0x_1(1 - x_1) = 0.030736$

…

$x_{99} = 4.0x_{98}(1 - x_{98}) = 0.914379$

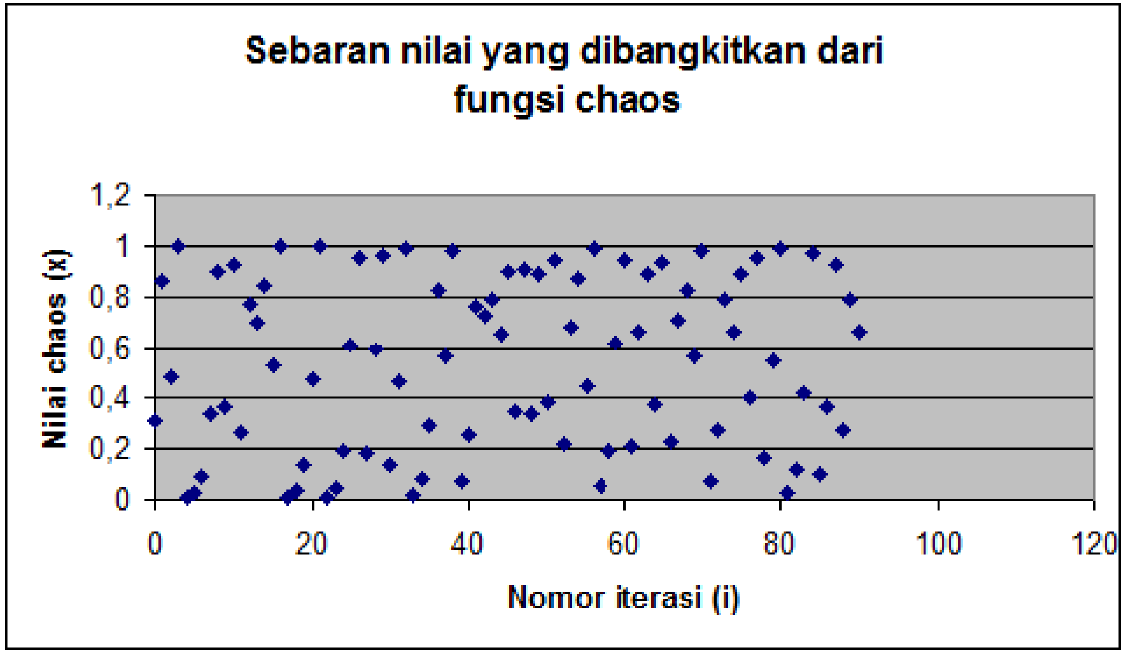
$x_{100} = 4.0x_{99}(1 - x_{99}) = 0.313162$

…

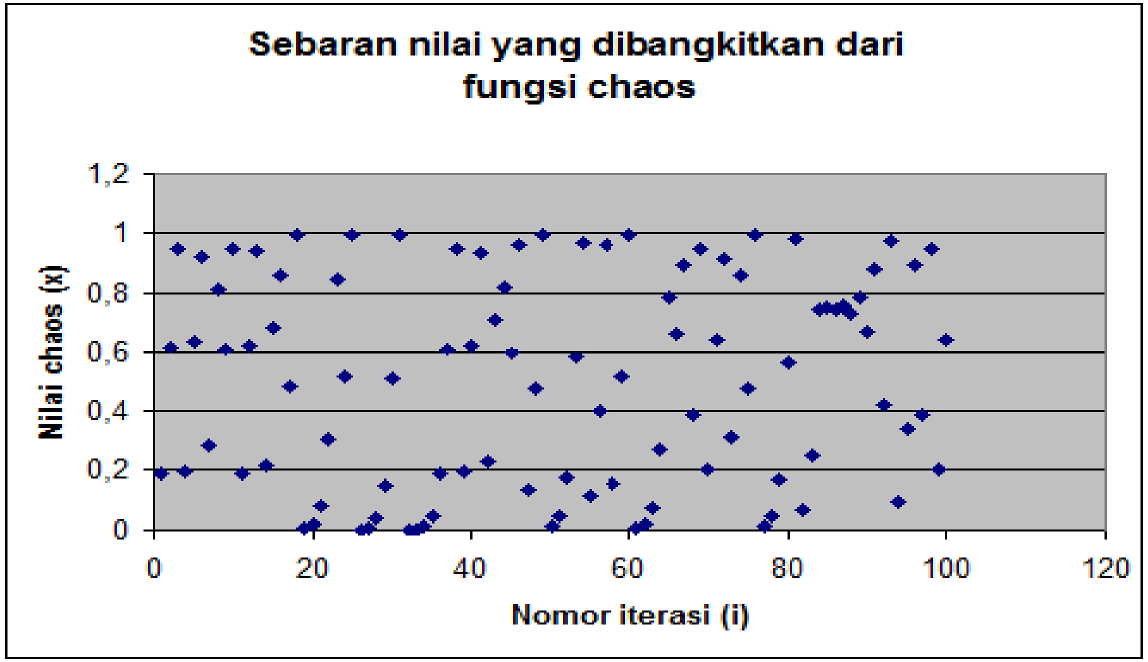
- Bilangan acak dengan *chaos* tidak punya periode

Logistic Map

```
main()
{
 int i, n
 double r, x;
 printf("Masukkan nilai awal (0 s/d 1) : "); scanf("%lf", &x);
 printf("Masukkan jumlah iterasi : "); scanf("%d", &n);
 r = 4.0;
 for (i = 1; i <= n; i++) {
    x = r * x * (1 - x);
    printf("x = %lf ‟, x);
 }
}
```



Gambar 1. Nilai-nilai *chaos* yang dibangkitkan dari fungsi *chaos* $x_{n+1} = 4.0\, x_n\, (1 - x_n)$ dengan $x_0 = 0.456$



Gambar 1. Nilai-nilai *chaos* yang dibangkitkan dari fungsi *chaos* $x_{n+1} = 4.0\, x_n\, (1 - x_n)$ dengan $x_0 = 0.456001$

Teori *Chaos*

Contoh *chaos map* lainnya:

1. *Henon map*

$$x_n = 1 + b(x_{n-2} - x_{n-3}) + cx^2_{n-2}$$

2. *Tent Map*

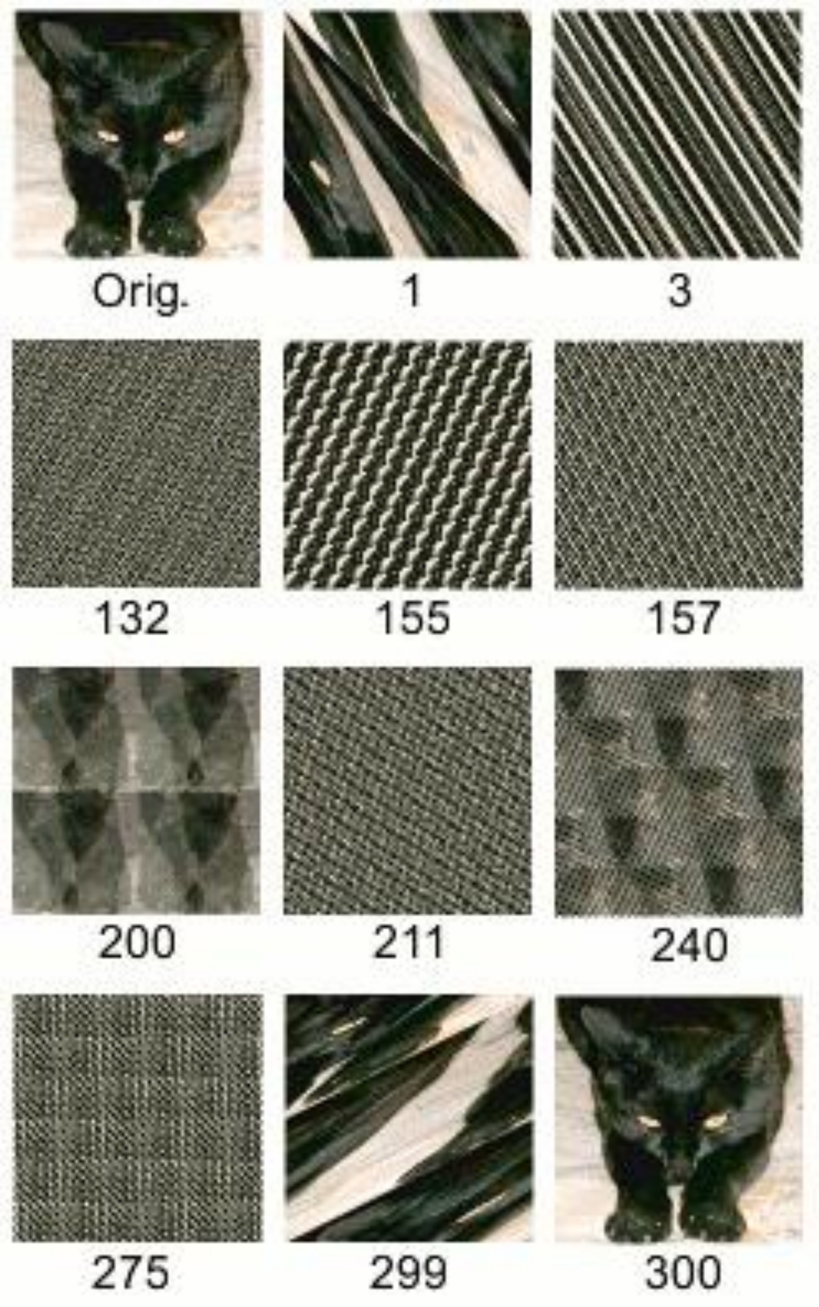
$$x_{i+1} = f_\mu(x_i) = \begin{cases} \mu x_i & , x_i < \frac{1}{2} \\ \mu(1 - x_i) & , x_i \geq \frac{1}{2} \end{cases}$$

3. *Arnold's cat map*:

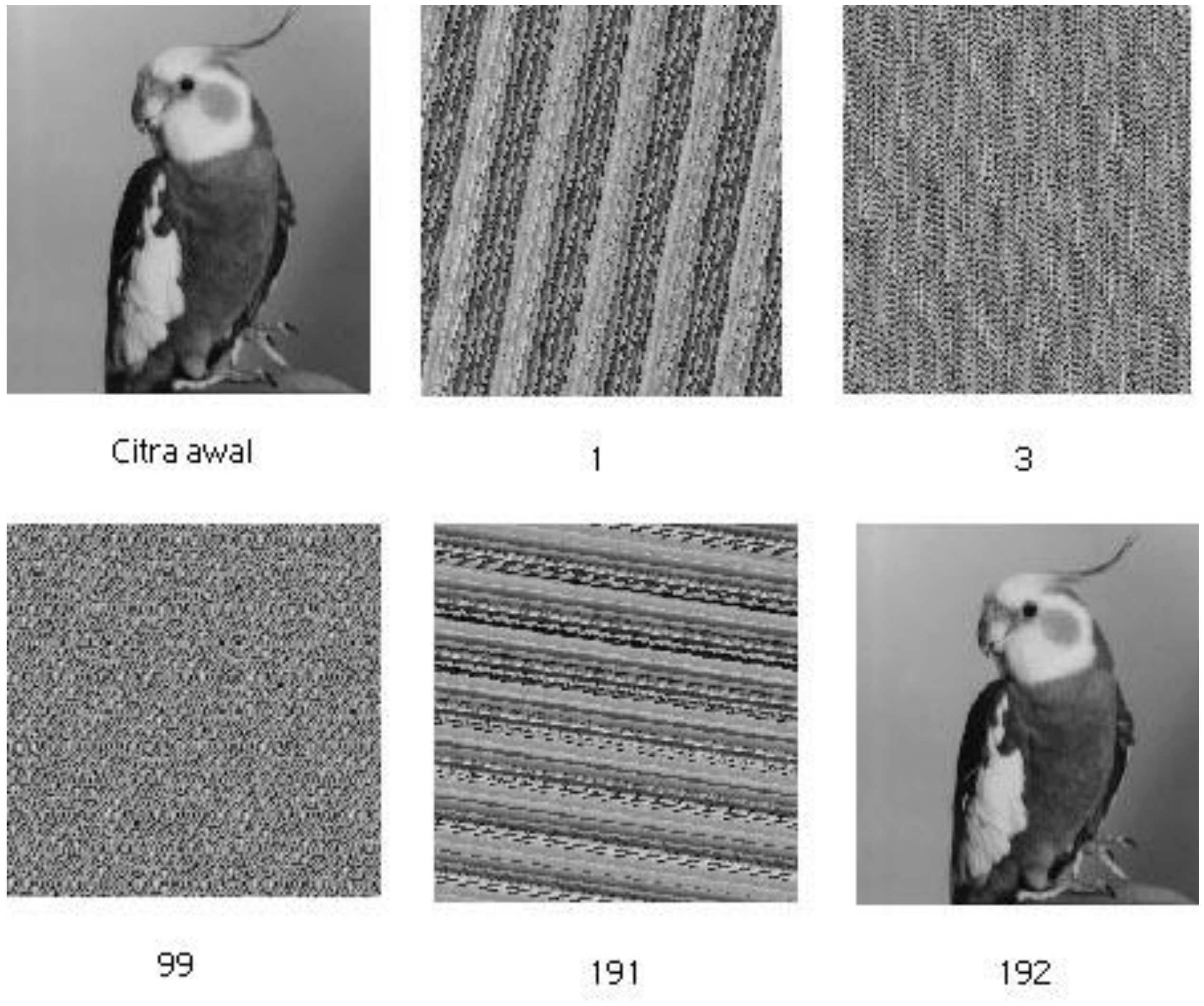
$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & b \\ c & bc+1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \operatorname{mod}(N)$$

b dan *c* adalah *integer* positif sembarang. Determinan matriks $\begin{bmatrix} 1 & b \\ c & bc+1 \end{bmatrix}$ harus sama dengan 1

Hasil lelaran dengan
Arnold Cat Map



$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = \begin{bmatrix} 1 & b \\ c & bc+1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \mathrm{mod}(N)$$


Citra awal
1
3
99
191
192