Available Online at www.ijcsmc.com/journal

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 11, Issue. 6, June 2022, pg.105 – 124

Detailed Analysis of Chaotic Logistic Map Model and Applications

Prof. Ziad Alqadi, Naseem Asad
Ismail Shayeb, Taha Othman
Albalqa Applied University
Jordan-Amman
DOI: https://doi.org/10.47760/ijcsmc.2022.v11i06.008

Abstract: Chaotic logistic map can be used to generate private key with different dimensions, this can be easily used to secure the process of data cryptography, required to encrypt-decrypt messages, images or audio files. Some of the encrypted-decrypted data (messages images) requires inter values keys, other like audio files require fraction values keys. In this research paper we will perform a detailed study of chaotic logistic map model to give some valuable recommendation of many aspects such as: the suitable range of the growth rate parameter, the suitable range of the initial generation, the sensitivity feature, the required time to generate private keys.
It will be show how to generate 1D and 2D private keys with integer values and with fractional values, these key will be generated taking in mined the process of optimizing the generation time and the selected length of key which can optimize the process of cryptography
Keywords: GRP, generation, population, chaotic key, private key, sensitivity.

Introduction

Chaotic logistic map model (CLMM) can be implemented using equation 1, this equation defines the rules (dynamics) of the chaotic system, where x represents the system population at any given time n, and r represents the growth rate, and here the population level (value) at any given time is a function of the growth rate parameter (GRP) value and the previous time step's population value [1-7].

$$x_{n+1} = r x_n (1 - x_n) \qquad (1)$$

The CLMM always starts with a population level of the starting value of x and it's set up to represent population as a ratio between 0 (extinction) and 1 (the maximum carrying capacity of our system). CLMM will be stable from run to another if we use the same parameters values, and it does not like the generated random values which change from run to run as shown in table 1.

Table 1: CLMM values vs random values

First run		Second run		Third run	
CLMM	Random	CLMM	Random	CLMM	Random
0.0386	0.0445	0.0386	0.3115	0.0386	0.5107
0.1448	0.2526	0.1448	0.9526	0.1448	0.2499
0.4829	0.1978	0.4829	0.1321	0.4829	0.0680
0.9739	0.5703	0.9739	0.9855	0.9739	0.1083
0.0993	0.4402	0.0993	0.2908	0.0993	0.0518
0.3488	0.8835	0.3488	0.9893	0.3488	0.7351
0.8859	0.6112	0.8859	0.8746	0.8859	0.3015
0.3943	0.9559	0.3943	0.7006	0.3943	0.0404
0.9314	0.0151	0.9314	0.8912	0.9314	0.4505
0.2492	0.6911	0.2492	0.1903	0.2492	0.9623
0.7296	0.1100	0.7296	0.0875	0.7296	0.8258
0.7694	0.2775	0.7694	0.9095	0.7694	0.3027
0.6920	0.9001	0.6920	0.1542	0.6920	0.4604
0.8313	0.3509	0.8313	0.1878	0.8313	0.4936
0.5471	0.8486	0.5471	0.0264	0.5471	0.8392

The previous property allows us to use the generated chaotic populations as a private key, the populations can be processed to get an integer values which can suit messages, text files, audio files and images encryption-decryption, as shown in table 2 [21-30].

Table 2: Converting CLMM values to PKs

CLMM1	PK1	CLMM2	PK2
0.0434	11	0.0040	1
0.1657	42	0.0158	4
0.5515	141	0.0622	16
0.9869	252	0.2328	59
0.0516	13	0.7125	182
0.1951	50	0.8173	208
0.6266	160	0.5959	152
0.9335	238	0.9608	245
0.2477	63	0.1501	38
0.7435	190	0.5091	130
0.7609	194	0.9972	254
0.7258	185	0.0113	3
0.7940	202	0.0445	11
0.6527	166	0.1695	43
0.9045	231	0.5616	143
0.3446	88	0.9823	250
0.9012	230	0.0692	18
0.3554	91	0.2570	66
0.9141	233	0.7620	194
0.3134	80	0.7237	185

PK= uint8(255*CLM values)

CLMM Behavior and Attractors

If tracing down the column under growth rate 1.5, we'll see the population level settles toward a final value of 0.333… after 20 generations. In the column for growth rate 2.0, you'll see an unchanging population level across each generation. This makes sense in the real world – if two parents produce two children, the overall population won't grow or shrink. So the growth rate of 2.0 represents the replacement rate.
Let's visualize table 3 and the chat representing this table (see figure 1).

Table 3: Various population rates values

r=0.5	r=1	r=1.5	r=2	r=2.5	r=3	r=3.5	r=3.99
0.0054	0.0109	0.0163	0.0218	0.0272	0.0326	0.0381	0.0434
0.0027	0.0108	0.0241	0.0426	0.0661	0.0947	0.1282	0.1657
0.0013	0.0106	0.0352	0.0815	0.1544	0.2572	0.3912	0.5515
0.0007	0.0105	0.0510	0.1497	0.3264	0.5732	0.8335	0.9869
0.0003	0.0104	0.0726	0.2546	0.5497	0.7339	0.4856	0.0516
0.0002	0.0103	0.1010	0.3796	0.6188	0.5858	0.8743	0.1951
0.0001	0.0102	0.1362	0.4710	0.5897	0.7279	0.3847	0.6266
0.0000	0.0101	0.1765	0.4983	0.6049	0.5942	0.8285	0.9335
0.0000	0.0100	0.2180	0.5000	0.5975	0.7234	0.4974	0.2477
0.0000	0.0099	0.2557	0.5000	0.6012	0.6003	0.8750	0.7435
0.0000	0.0098	0.2855	0.5000	0.5994	0.7198	0.3829	0.7609
0.0000	0.0097	0.3060	0.5000	0.6003	0.6050	0.8270	0.7258
0.0000	0.0096	0.3185	0.5000	0.5998	0.7169	0.5008	0.7940
0.0000	0.0095	0.3256	0.5000	0.6001	0.6089	0.8750	0.6527
0.0000	0.0094	0.3294	0.5000	0.6000	0.7144	0.3828	0.9045
0.0000	0.0093	0.3313	0.5000	0.6000	0.6120	0.8269	0.3446
0.0000	0.0093	0.3323	0.5000	0.6000	0.7123	0.5009	0.9012
0.0000	0.0092	0.3328	0.5000	0.6000	0.6147	0.8750	0.3554
0.0000	0.0091	0.3331	0.5000	0.6000	0.7105	0.3828	0.9141
0.0000	0.0090	0.3332	0.5000	0.6000	0.6171	0.8269	0.3134
0.0000	0.0089	0.3333	0.5000	0.6000	0.7089	0.5009	0.8586
0.0000	0.0088	0.3333	0.5000	0.6000	0.6191	0.8750	0.4844
0.0000	0.0088	0.3333	0.5000	0.6000	0.7075	0.3828	0.9965
0.0000	0.0087	0.3333	0.5000	0.6000	0.6209	0.8269	0.0138
0.0000	0.0086	0.3333	0.5000	0.6000	0.7062	0.5009	0.0544
0.0000	0.0085	0.3333	0.5000	0.6000	0.6225	0.8750	0.2052
0.0000	0.0085	0.3333	0.5000	0.6000	0.7050	0.3828	0.6508
0.0000	0.0084	0.3333	0.5000	0.6000	0.6239	0.8269	0.9067
0.0000	0.0083	0.3333	0.5000	0.6000	0.7039	0.5009	0.3374
0.0000	0.0083	0.3333	0.5000	0.6000	0.6253	0.8750	0.8920

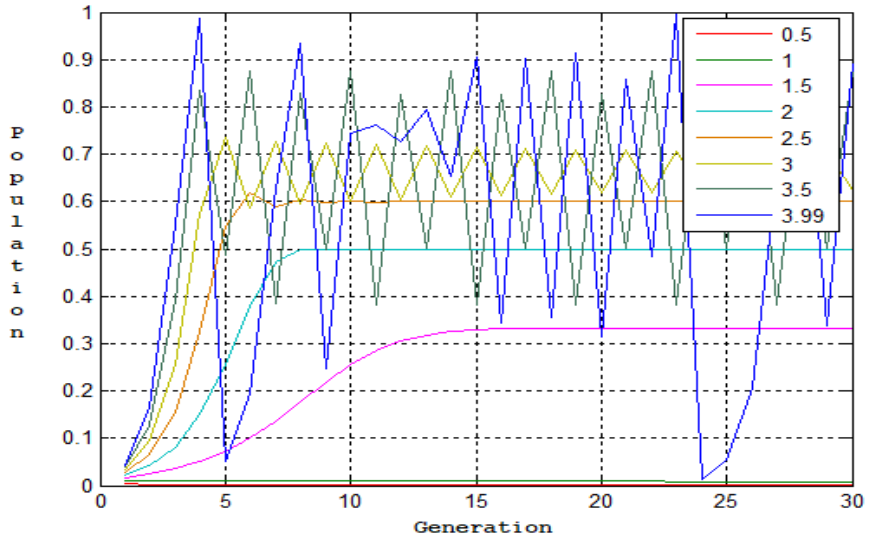

Figure 1: Population values for table 3

Here you can easily see how the population changes over time, given different growth rates. The blue line represents a growth rate of 0.5, and it quickly drops to zero. The population dies out. The cyan line represents a growth rate of 2.0 (remember, the replacement rate) and it stays steady at a population level of 0.5. The growth rates of 3.0 and 3.5 are more interesting. While the yellow line for 3.0 seems to be slowly converging toward a stable value, the gray line for 3.5 just seems to bounce around.

An *attractor* is the value, or set of values, that the system settles toward over time. When the growth rate parameter is set to 0.5, the system has a fixed-point attractor at population level 0 as depicted by the blue line. In other words, the population value is drawn toward 0 over time as the model iterates. When the growth rate parameter is set to 3.5, the system oscillates between four values, as depicted by the gray line. This attractor is called a limit cycle.

But when we adjust the growth rate parameter beyond 3.5, we see the onset of chaos. A chaotic system has a *strange attractor*, around which the system oscillates forever, never repeating itself or settling into a steady state of behavior. It never hits the same point twice and its structure has a fractal form, meaning the same patterns exist at every scale no matter how much you zoom into it.

CLMM Analysis

Below we will analyze several aspects of using CLMM to generate various chaotic keys needed in various methods of data cryptography:

1) Analysis of growth rate parameter (GRP) values

GRP has a double data type (represented by 8 bytes (64 bits)), this type provides a huge parameter space, and below we will analyze the effects of GRP values

First range: 0<r<=1

The following CLMM (written in matlab: see figure 2) was implemented using various values of r within this range and fixing x to the value 0.011


```
R=500;                  %number of generations
r1=0.75;                %GRP
x1=0.011;               %initial population value
for i=1:R
      x1=r1*x1*(1-x1);
      k12(1,i)=x1;       %CLM key

end
k2=uint8(255*k12);      %integer key
```

Figure 2: CLM operations

Table 4 shows the generated populations, while figure 3 shows the plots on the generated integer keys

Table 4: Population using the first rang of GRP values

r=0.1	r=0.5	r=0.75	r=1
0	1	2	2
0	1	2	2
0	0	1	1
0	0	1	1
0	0	1	1
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

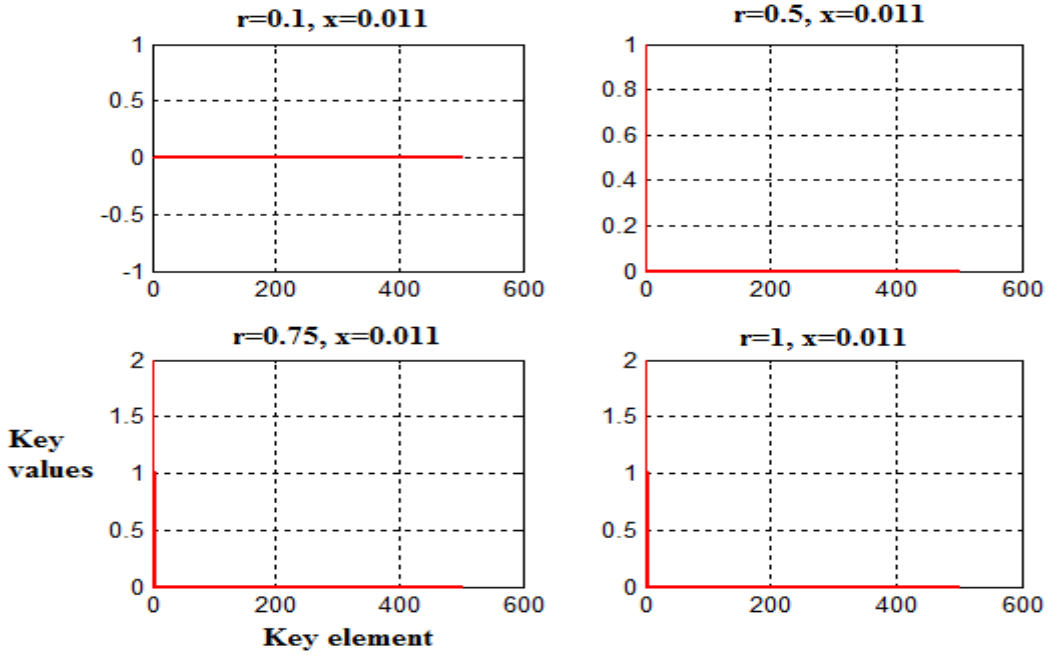

Figure 3: Integer key using the first range of GRP values

As we can see from table 4 and figure 3 the population values will remain zeros after applying a few generations, this case (range will not be recommended to be used to generate private keys).

Second range: 1<r<=2

The CLMM sequence of operations (see figure 2) was implemented using various values of r within second range and fixing x to the value 0.011, Table 5 shows the generated populations, while figure 4 shows the plots of the generated integer keys. Tracing down the column under growth rate 1.5, we'll see the population level settles toward a final value of 0.333… after 16 generations. In the column for growth rate 2.0, we'll see an unchanging population level across each generation. This range can be used to generate a few changeable generations, the limit range of changed generations depend on the r values, also the maximum reached value of an integer keys will range from 23 to 104 depending on the value of r (see figure 4), so the second range of r values cannot be recommended to be used to generate a key with any length.

Table 5: Population using the second rang of GRP values

r=1.1, x=0.011	r=1.5, x=0.011	r=1.75, x=0.011	r=2, x=0.011
0.0120	0.0163	0.0190	0.0218
0.0130	0.0241	0.0327	0.0426
0.0141	0.0352	0.0553	0.0815
0.0153	0.0510	0.0915	0.1497
0.0166	0.0726	0.1454	0.2546
0.0179	0.1010	0.2175	0.3796
0.0194	0.1362	0.2978	0.4710
0.0209	0.1765	0.3660	0.4983
0.0225	0.2180	0.4061	0.5000
0.0242	0.2557	0.4221	0.5000
0.0260	0.2855	0.4269	0.5000
0.0278	0.3060	0.4281	0.5000
0.0298	0.3185	0.4285	0.5000
0.0318	0.3256	0.4285	0.5000
0.0338	0.3294	0.4286	0.5000
0.0360	0.3313	0.4286	0.5000
0.0381	0.3323	0.4286	0.5000
0.0404	0.3328	0.4286	0.5000
0.0426	0.3331	0.4286	0.5000
0.0449	0.3332	0.4286	0.5000
0.0471	0.3333	0.4286	0.5000
0.0494	0.3333	0.4286	0.5000
0.0517	0.3333	0.4286	0.5000
0.0539	0.3333	0.4286	0.5000
0.0561	0.3333	0.4286	0.5000

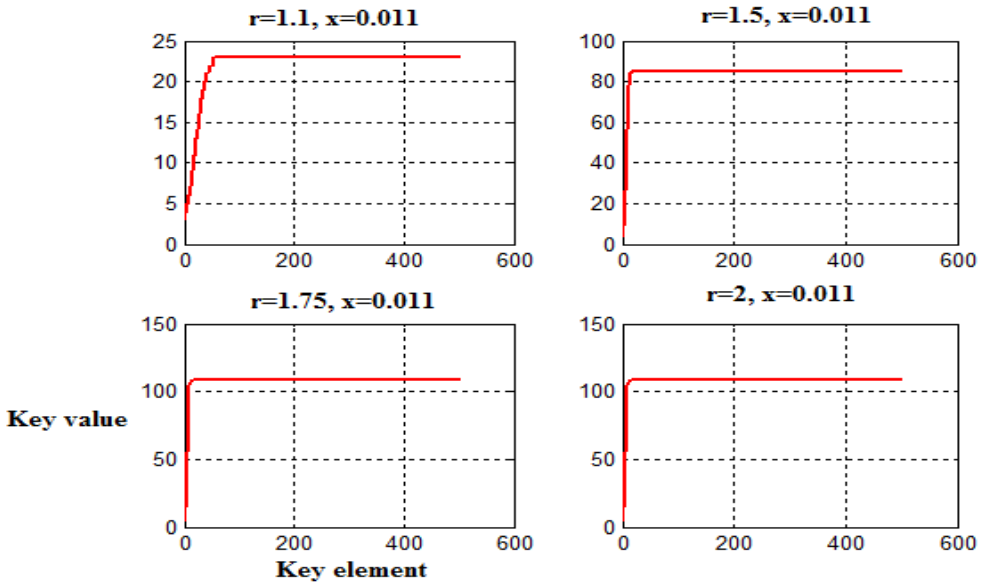

Figure 4: Integer key using the second range of GRP values

Third range: 2<r<=3

The CLMM sequence of operations (see figure 2) was implemented using various values of r within third range and fixing x to the value 0.011, Table 6 shows the generated populations, while figure 5 shows the plots of the generated integer keys. Tracing down the column under growth rate 2.1, we'll see the population level settles toward a final value of 0.5238 after 8 generations, the population level under column r equal 2.5 the populations settles toward a final value of 0.6000 after 14 generations, and so on. This range can be used to generate a few range of changeable generations, this range grow up when r increases, the limit range of changed generations depend on the r values, also the maximum reached value of an integer keys will range from 130 to 160 depending on the value of r (see figure 5), so the third range of r values can be recommended for short length key. And it is not cannot be recommended to be used to generate a key with big length.

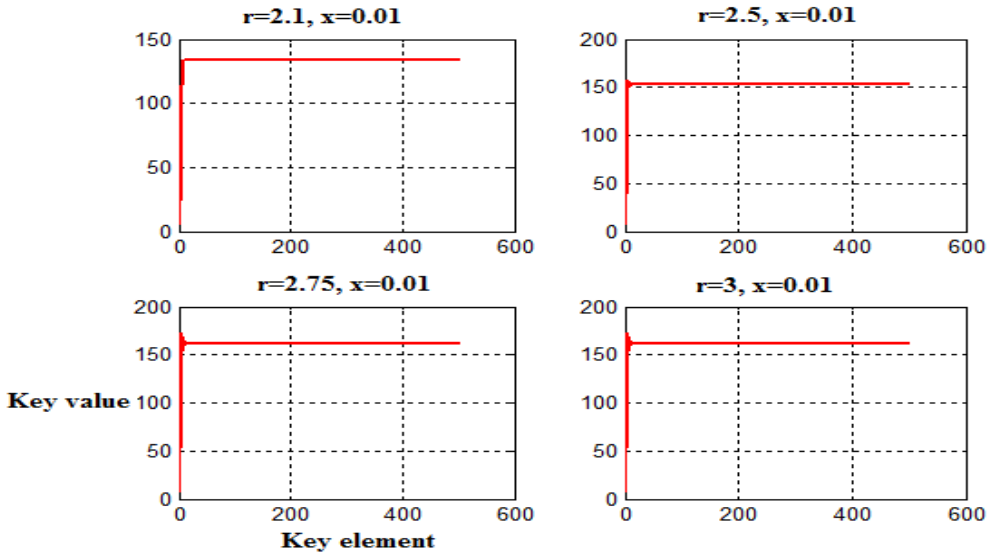

Figure 5: Integer key using the third range of GRP values

Table 6: Population using the third rang of GRP values

r=2.1,x=0.011	r=2.5,x=0.011	r=2.75,x=0.011	r=3,x=0.011
0.0228	0.0272	0.0299	0.0326
0.0469	0.0661	0.0798	0.0947
0.0938	0.1544	0.2020	0.2572
0.1786	0.3264	0.4432	0.5732
0.3080	0.5497	0.6786	0.7339
0.4476	0.6188	0.5997	0.5858
0.5192	0.5897	0.6601	0.7279
0.5242	0.6049	0.6170	0.5942
0.5238	0.5975	0.6499	0.7234
0.5238	0.6012	0.6257	0.6003
0.5238	0.5994	0.6440	0.7198
0.5238	0.6003	0.6305	0.6050
0.5238	0.5998	0.6407	0.7169
0.5238	0.6001	0.6331	0.6089
0.5238	0.6000	0.6388	0.7144
0.5238	0.6000	0.6345	0.6120
0.5238	0.6000	0.6377	0.7123
0.5238	0.6000	0.6353	0.6147
0.5238	0.6000	0.6371	0.7105
0.5238	0.6000	0.6358	0.6171
0.5238	0.6000	0.6368	0.7089
0.5238	0.6000	0.6360	0.6191
0.5238	0.6000	0.6366	0.7075
0.5238	0.6000	0.6362	0.6209
0.5238	0.6000	0.6365	0.7062

Forth range: 3<r<4

The CLMM sequence of operations (see figure 2) was implemented using various values of r within forth range and fixing x to the value 0.011, Table 7 shows the generated populations, while figure 6 shows the plots of the generated integer keys. Tracing down the columns under growth rates 3.1 to 3.99, we'll see the population level changeable. This range can be used to generate a range of changeable generations, this range can be used to generate a private key of any length, and it can be recommended to be use to generate a variable length private keys, the length of the key will equal the selected number of generations.

For r greater or equal 4, the generated populations will be unexpected and they will grow to infant, so these value cannot be recommended to be used to generate private keys, as shown in table 8.

Table 7: Population using the forth rang of GRP values

r=3.1, x=0.011	r=3.5, x=0.011	r=3.75, x=0.011	r=3.99, x=0.011
0.0337	0.0381	0.0408	0.0434
0.1010	0.1282	0.1467	0.1657
0.2815	0.3912	0.4695	0.5515
0.6270	0.8335	0.9340	0.9869
0.7250	0.4856	0.2311	0.0516
0.6181	0.8743	0.6663	0.1951
0.7318	0.3847	0.8337	0.6266
0.6085	0.8285	0.5198	0.9335
0.7385	0.4974	0.9360	0.2477
0.5986	0.8750	0.2246	0.7435
0.7448	0.3829	0.6530	0.7609
0.5892	0.8270	0.8497	0.7258
0.7503	0.5008	0.4788	0.7940
0.5807	0.8750	0.9358	0.6527
0.7548	0.3828	0.2252	0.9045
0.5737	0.8269	0.6544	0.3446
0.7581	0.5009	0.8481	0.9012
0.5684	0.8750	0.4831	0.3554
0.7605	0.3828	0.9364	0.9141
0.5646	0.8269	0.2233	0.3134
0.7620	0.5009	0.6503	0.8586
0.5621	0.8750	0.8528	0.4844
0.7630	0.3828	0.4707	0.9965
0.5605	0.8269	0.9343	0.0138
0.7636	0.5009	0.2302	0.0544

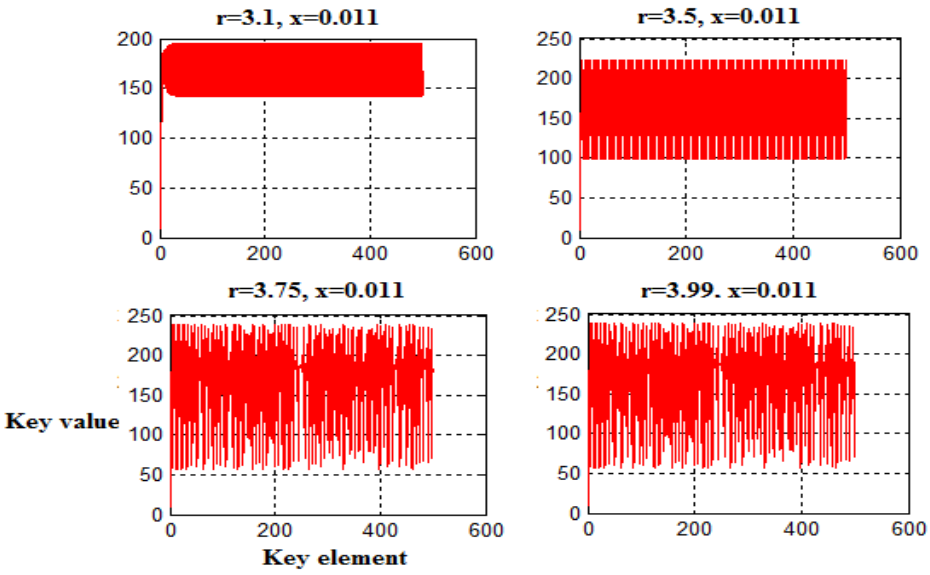

Figure 6: Integer key using the forth range of GRP values

Table 8: Populations for r>=4

1.0e+254 *			
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	-0.0000	0.0000
0.0000	-0.0000	-0.0000	0.0000
0.0000	-0.0000	-0.0000	-0.0000
0.0000	-0.0000	-0.0000	-0.0000
0.0000	-0.0000	-0.0000	-0.0000
0.0000	-0.0000	-0.0000	-0.0000
0.0000	-0.0000	-0.0000	-0.0000
0.0000	-0.0000	-0.0000	-0.0000
0.0000	-0.0000	-Inf	-0.0000
0.0000	-Inf	-Inf	-2.6632
0.0000	-Inf	-Inf	-Inf
0.0000	-Inf	-Inf	-Inf
0.0000	-Inf	-Inf	-Inf
0.0000	-Inf	-Inf	-Inf
0.0000	-Inf	-Inf	-Inf
0.0000	-Inf	-Inf	-Inf
0.0000	-Inf	-Inf	-Inf
0.0000	-Inf	-Inf	-Inf
0.0000	-Inf	-Inf	-Inf

r>=4

2) Analysis of initial population

The values of the initial generation x must be within the range 0<x<1 in order to get acceptable values to for a private key, table 9, and 10 this fact:

Table 9: Values of x are out of the required range

1.0e+291 *			
0	-0.0000	-0.0000	-0.0000
0	-0.0000	-0.0000	-0.0000
0	-0.0000	-0.0000	-0.0000
0	-0.0000	-0.0000	-0.0000
0	-0.0000	-0.0000	-0.0000
0	-0.0000	-0.0000	-0.0000
0	-0.0000	-0.0000	-0.0000
0	-0.0000	-0.0000	-9.1056
0	-Inf	-Inf	-Inf
0	-Inf	-Inf	-Inf
0	-Inf	-Inf	-Inf
0	-Inf	-Inf	-Inf
0	-Inf	-Inf	-Inf
0	-Inf	-Inf	-Inf
0	-Inf	-Inf	-Inf
x=1	x=2	x=3	x=4

r=3.99

1.0e+291 *			
-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-0.0000	-0.0000
-0.0000	-0.0000	-9.1056	-Inf
-Inf	-Inf	-Inf	-Inf
-Inf	-Inf	-Inf	-Inf
-Inf	-Inf	-Inf	-Inf
-Inf	-Inf	-Inf	-Inf
-Inf	-Inf	-Inf	-Inf
-Inf	-Inf	-Inf	-Inf
-Inf	-Inf	-Inf	-Inf
x=-1	x=-2	x=-3	x=-4

r=3.99

Table 10: The required range of x(0<x<1)

x=-0.01	x=-0.02	x=-0.022	x=-0.009	
0.0395	0.0782	0.0858	0.0356	
0.1514	0.2876	0.3131	0.1369	
0.5126	0.8175	0.8582	0.4716	
0.9969	0.5952	0.4856	0.9943	
0.0125	0.9614	0.9967	0.0227	
0.0491	0.1482	0.0132	0.0886	
0.1862	0.5036	0.0520	0.3221	
0.6046	0.9974	0.1968	0.8713	r=3.99
0.9538	0.0102	0.6306	0.4475	
0.1758	0.0401	0.9295	0.9865	
0.5781	0.1536	0.2616	0.0532	
0.9732	0.5188	0.7708	0.2010	
0.1042	0.9961	0.7049	0.6407	
0.3723	0.0156	0.8299	0.9185	
0.9324	0.0611	0.5632	0.2986	
0.2514	0.2290	0.9816	0.8356	
0.7510	0.7044	0.0722	0.5480	
0.7462	0.8307	0.2674	0.9883	
0.7557	0.5611	0.7816	0.0461	
0.7366	0.9826	0.6810	0.1756	

3) Sensitivity analysis

CLMM is characterized by it's sensitive dependence on initial population(x) and GRP (r). It doesn't have a basin of attraction that collects nearby points over time into a fixed-point or limit cycle attractor. Rather, with a strange attractor, close points *diverge* over time.

This makes real-world modeling and prediction difficult, because you must measure the parameters and system state with infinite precision. Otherwise, tiny errors in measurement or rounding are compounded over time until the system is thrown drastically off.

Based on the selected CLMM parameters (r and x) we can generate a required number of populations depending on the selected number of generation, the generated values are very sensitive to any minor changes in r and/or x, changing one these parameters or changing both parameters will produce new values, these is shown in tables 11, 12 1and 13.

Table 11: Changing x values will reproduce new population values

		r=3.99	
x=0.01	x=0.02	x=0.022	x=0.009
0.0395	0.0782	0.0858	0.0356
0.1514	0.2876	0.3131	0.1369
0.5126	0.8175	0.8582	0.4716
0.9969	0.5952	0.4856	0.9943
0.0125	0.9614	0.9967	0.0227
0.0491	0.1482	0.0132	0.0886
0.1862	0.5036	0.0520	0.3221
0.6046	0.9974	0.1968	0.8713
0.9538	0.0102	0.6306	0.4475
0.1758	0.0401	0.9295	0.9865
0.5781	0.1536	0.2616	0.0532
0.9732	0.5188	0.7708	0.2010
0.1042	0.9961	0.7049	0.6407
0.3723	0.0156	0.8299	0.9185
0.9324	0.0611	0.5632	0.2986

Table 12: Changing r values will reproduce new population values

	x=0.009		
r=3.99	r=3.98	r=3.97	r=3.96
0.0356	0.0355	0.0354	0.0353
0.1369	0.1363	0.1356	0.1349
0.4716	0.4684	0.4653	0.4622
0.9943	0.9910	0.9877	0.9843
0.0227	0.0354	0.0481	0.0610
0.0886	0.1358	0.1819	0.2269
0.3221	0.4670	0.5908	0.6947
0.8713	0.9907	0.9598	0.8399
0.4475	0.0368	0.1533	0.5324
0.9865	0.1411	0.5152	0.9858
0.0532	0.4824	0.9916	0.0553
0.2010	0.9938	0.0331	0.2068
0.6407	0.0247	0.1271	0.6495
0.9185	0.0957	0.4406	0.9015
0.2986	0.3445	0.9785	0.3516

Table 13: Changing r and x values will reproduce new population values

r=3.99;x=0.009	r=3.98;x=0.011	r=3.97;x=0.012	r=3.96;x=0.013
0.0356	0.0433	0.0471	0.0508
0.1369	0.1649	0.1781	0.1910
0.4716	0.5480	0.5810	0.6119
0.9943	0.9858	0.9664	0.9404
0.0227	0.0556	0.1288	0.2218
0.0886	0.2089	0.4455	0.6835
0.3221	0.6578	0.9807	0.8567
0.8713	0.8959	0.0751	0.4862
0.4475	0.3710	0.2756	0.9892
0.9865	0.9288	0.7927	0.0421
0.0532	0.2631	0.6525	0.1598
0.2010	0.7717	0.9002	0.5317
0.6407	0.7012	0.3566	0.9860
0.9185	0.8339	0.9108	0.0546
0.2986	0.5512	0.3224	0.2043

The sensitivity is very important when using the populations as a private key in data cryptography method, any minor changes in r and/or x in the decryption phase will cause generating different key, the decrypted data will be corrupted [31-47] and will not match the original data, the changes in CLMM parameters will be considered as a hacking attempt by producing damaged decrypted data.

4) Processing time analysis

CLMM populations required processing time to be generated, this time will grow up when increasing the number of generations, table 14 shows the required time to generate populations with various lengths (generations) (the selected ranges are: $3.5<r<4$ and $0<x<1$)

Table 14: Key generation time

Population (key) length (values)	Generation time (seconds)	Population (key) length (values)	Generation time (seconds)
20	0.0011	3000	0.0055
50	0.0015	4000	0.0079
100	0.0016	5000	0.0113
200	0.0018	7500	0.0241
300	0.0019	10000	0.0375
400	0.0020	20000	0.1362
500	0.0022	50000	2.1908
1000	0.0025	100000	10.3759
2000	0.0040	1000000	1739.4

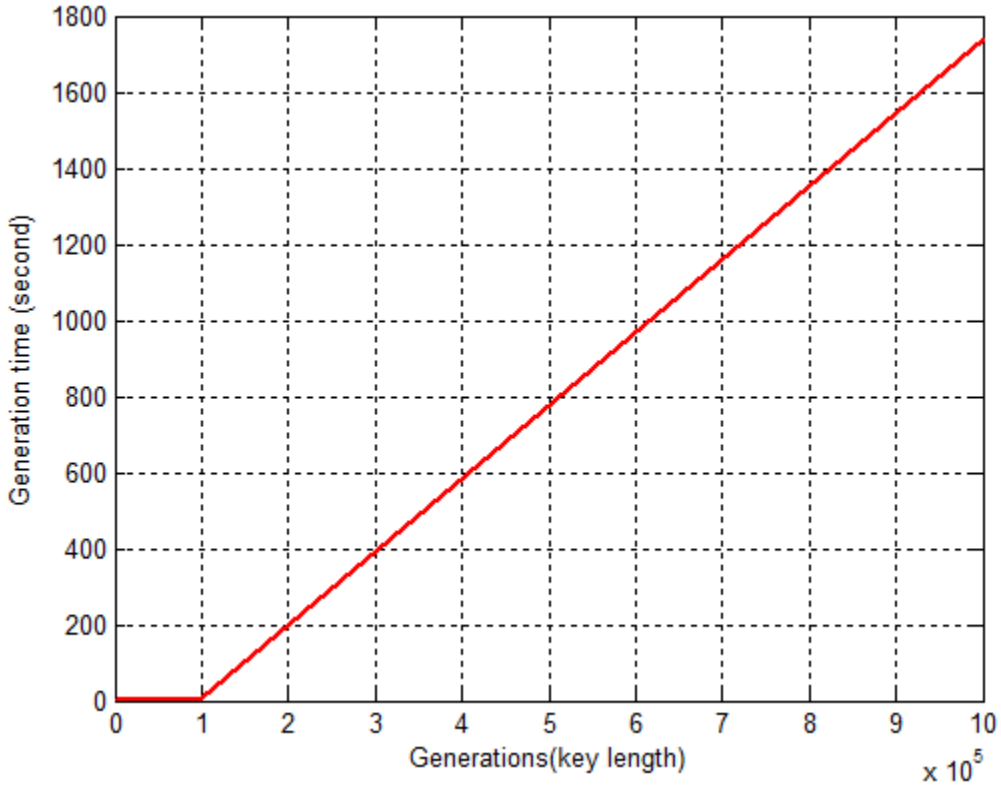

Figure 7: Generation time vs key length

From table 14 we can see that the required time to generate a long key will rapidly grow up (see figure 7), so we can recommend here generating a key with a short length, this key can be easily resized to match the size of the data to be encrypted-decrypted.

CLMM Applications

CLMM can be easily used to generate private keys, required to apply data encryption-decryption [9], [10]. The generated private keys may be one or 2 dimensional depending on the data (to be encrypted-decrypted) dimensions, figure 8 shows a sequence of operations required to use CLMM to generate a 2D private key[16-20].


```
R=10;C=8;    %Diminsions
r1=3.99;x1=0.02;
for i=1:R
   for j=1:C
      x1=r1*x1*(1-x1);
      k12(i,j)=x1;
   end
   x1=x1+0.001;
end
k2=uint8(255*k12);
k12
k2
```

Figure 8: CLMM operation to generate 2D PK

Table 15, shows the generated key, while figure 9 shows the plot of the first 4 columns of this key.

Table 15: Generated 2D key

0.0782	0.2876	0.8175	0.5952	0.9614	0.1482	0.5036	0.9974
0.0062	0.0245	0.0954	0.3444	0.9010	0.3561	0.9148	0.3109
0.8563	0.4910	0.9972	0.0112	0.0443	0.1688	0.5599	0.9832
0.0621	0.2323	0.7115	0.8190	0.5915	0.9641	0.1380	0.4747
0.9951	0.0193	0.0756	0.2788	0.8022	0.6330	0.9269	0.2703
0.7888	0.6648	0.8891	0.3934	0.9521	0.1818	0.5936	0.9625
0.1401	0.4808	0.9960	0.0158	0.0620	0.2319	0.7107	0.8203
0.5856	0.9683	0.1225	0.4289	0.9773	0.0884	0.3215	0.8703
0.4474	0.9865	0.0533	0.2012	0.6413	0.9179	0.3008	0.8392
0.5357	0.9924	0.0300	0.1161	0.4096	0.9649	0.1351	0.4664

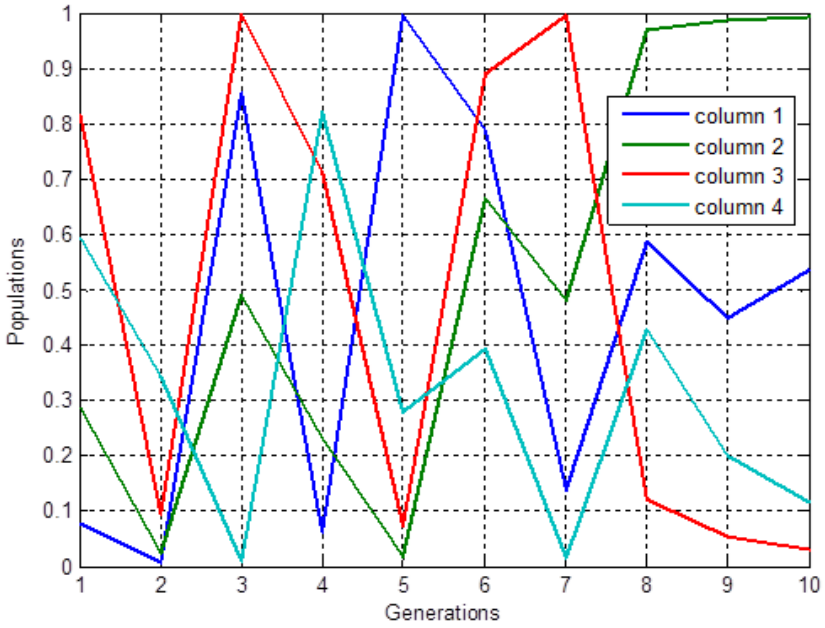

Figure 9: 2D private key plot

The previous key can be easily changed to a key with inter values, the integer key can be used to apply message [8], [13-15] or digital image cryptography [9-12], table 16 shows the generated key with integer values, while figure 9 shows the plot of this key

Table 16: Generated key with integer values

20	73	208	152	245	38	128	254
2	6	24	88	230	91	233	79
218	125	254	3	11	43	143	251
16	59	181	209	151	246	35	121
254	5	19	71	205	161	236	69
201	170	227	100	243	46	151	245
36	123	254	4	16	59	181	209
149	247	31	109	249	23	82	222
114	252	14	51	164	234	77	214
137	253	8	30	104	246	34	119

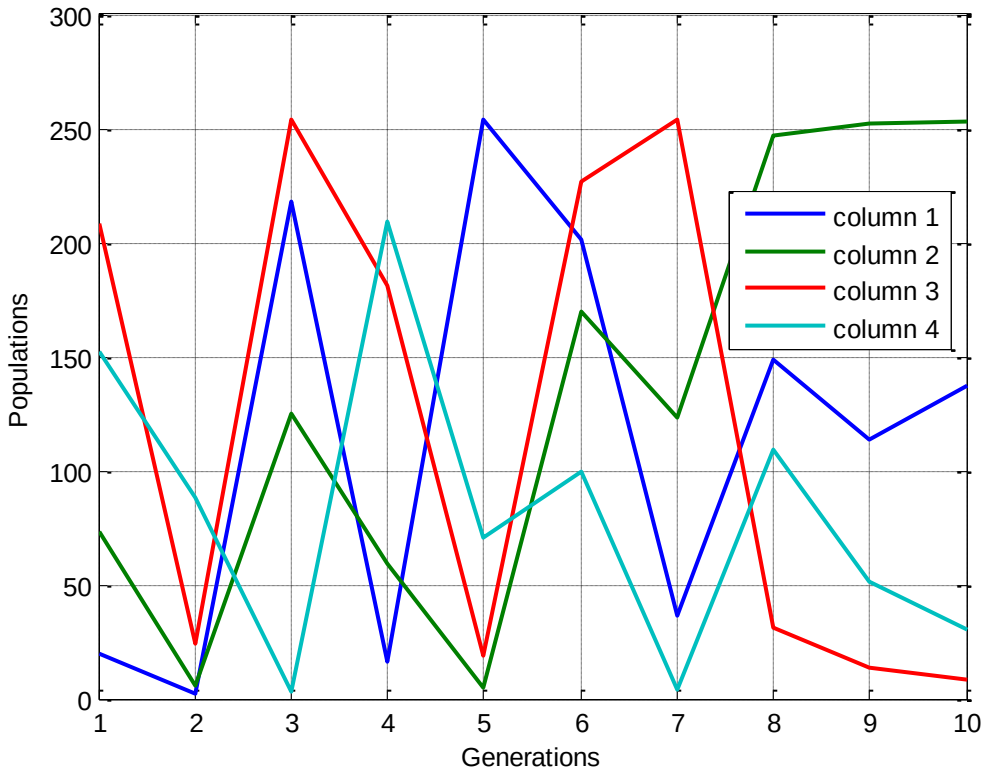

Figure 10: Plot of integer values key

Two or more CLMM can be used to generate different 2d keys; the sequence of operations shown in figure 11 can uses two CLMM to generate two different 2D keys, tables 17 and 18 show the generated keys:

Table 17: Generated first key

10	39	131	254	3	13	47	154
243	45	147	248	27	95	238	64
191	190	193	188	197	178	215	136
253	7	26	93	236	70	203	165
232	83	223	112	251	16	61	185
202	167	230	89	231	86	227	100
242	48	156	241	52	164	233	79
217	129	254	3	10	39	133	254
5	18	66	195	184	205	160	237
65	193	186	200	172	224	110	249

Table 17: Generated second key

244	41	138	253	10	37	125	254
3	12	45	147	248	27	95	238
64	191	192	190	193	187	200	173
223	113	251	15	57	177	216	132
254	4	15	55	172	223	111	250
19	70	202	166	231	88	229	92
234	76	213	139	252	11	41	138
253	9	34	119	253	7	27	97
240	58	178	214	137	253	8	30
105	246	34	117	253	9	34	117

Below is an example of using CLMM generation to encrypt-decrypt the gray image 'cameraman.tif'

1) Read the image
2) Get the image size
3) Use the CLMM (see figure 8) to generate a private key
4) Resize the key to the image size.
5) Apply XORing the image with the resized key to get the encrypted image

The following matlab code was executed:

```
clear all,close all,clc
a=imread('cameraman.tif');
[n1 n2]=size(a);
tic
R=120;C=100;
r1=3.99;x1=0.02;
for i=1:R
   for j=1:C
      x1=r1*x1*(1-x1);
      k12(i,j)=x1;
   end
   x1=x1+0.001;
end
k2=uint8(255*k12);
t2=toc;
k12;
k2
key=imresize(k2,[n1 n2]);
imhist(key)
en=bitxor(a,key);
figure,
subplot(2,2,1),imshow(a)
subplot(2,2,2),imhist(en)
subplot(2,2,3),imshow(en)
subplot(2,2,4),imhist(en)
[m1 m2]=PSNR_RGB(a,en)
```

The following are the outputs of the program running (figures 11 and 12):

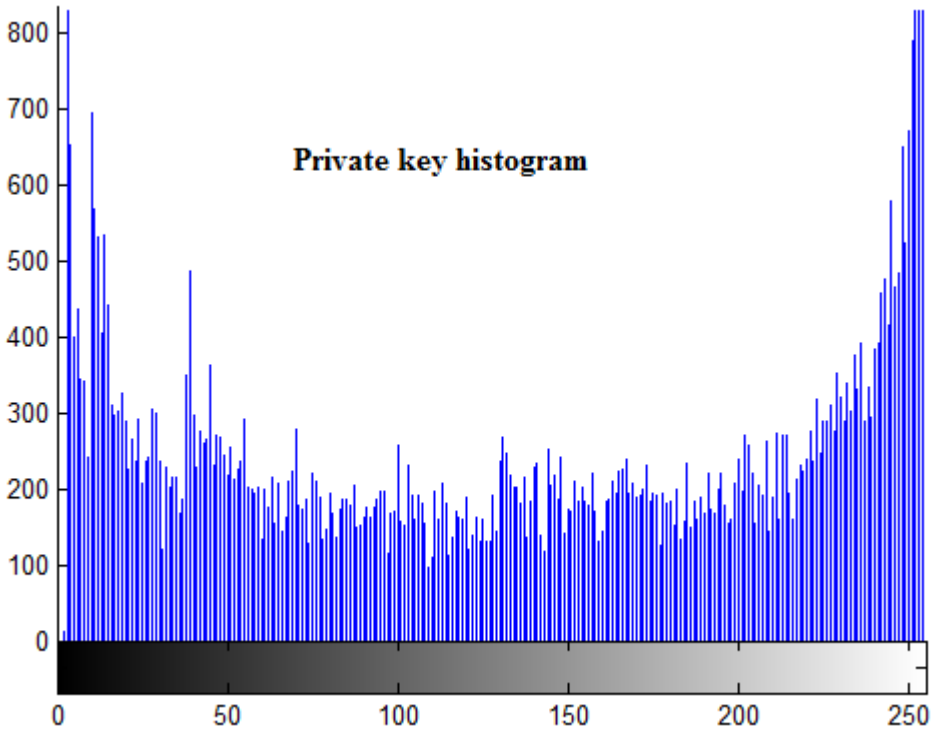

Figure 11: Private key histogram

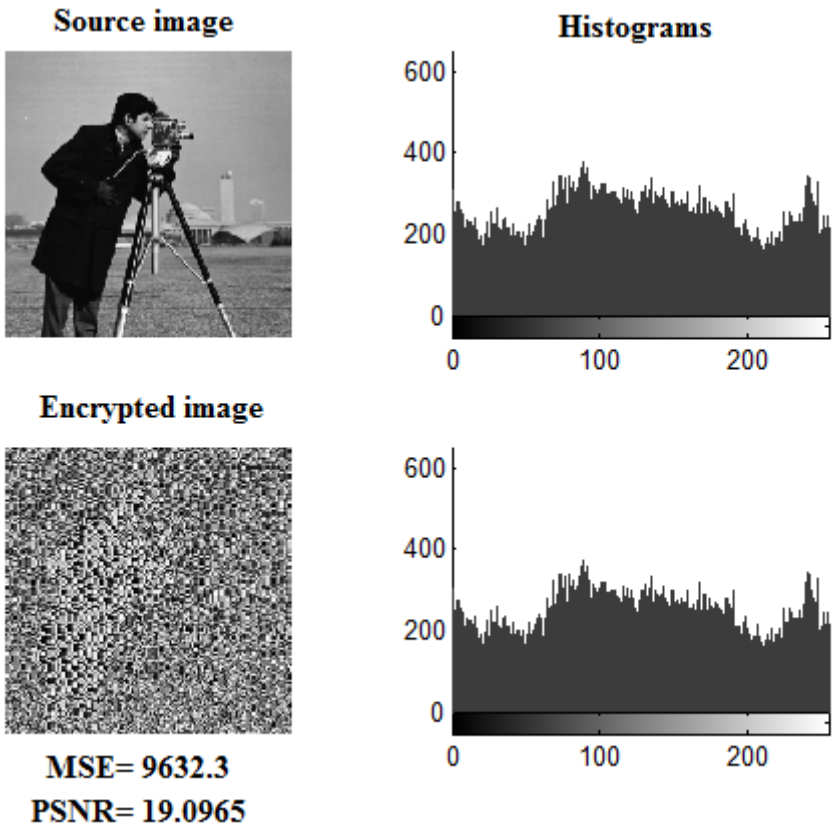

Figure 12: Images outputs

Conclusion

A detailed analysis of chaotic logistic map model was performed, the CLMM parameters were studied. It was shown that to produce a changeable generation it was recommended to use one rang for the growth rate parameter r and a range for the initial generation x, the first range was 305<r<4, the second range was 0<x<1, these ranges will guarantee generating a suitable for data cryptography keys. It was shown that the generated keys are very sensitive to any minor changes in CLMM parameters, changing ant parameter values will lead to generate a new key.

The key generation time was examined and it was recommended to use small generation number to create a key, this key can be easily converted to integers and can easily resized to match the data to be encrypted size, using this key will optimize the encryption-decryption times. Some data requires 2D keys, it was shown how to use two CLMM to generate theses keys.

References

[1]. Aamer Nadeem, Dr M. Younus Javed, A Performance Comparison of Data Encryption Algorithms, Conference Paper, September 2005 DOI: 10.1109/ICICT.2005.1598556 · Source: IEEE Xplore.

[2]. M. Bala Kumara, P. Karthikkab , N. Dhivyac , T. Gopalakrishnan, A Performance Comparison of Encryption Algorithms for Digital Images, International Journal of Engineering Research & Technology (IJERT), Vol. 3 Issue 2, February – 2014.

[3]. Lee Mariel Heucheun Yepdia, Alain Tiedeu, and Guillaume Kom, A Robust and Fast Image Encryption Scheme Based on a Mixing Technique, Security and Communication Networks, Volume 2021 |Article ID 6615708 | https://doi.org/10.1155/2021/6615708

[4]. Z. Hua, Y. Zhou, and H. Huang, "Cosine-transform-based chaotic system for image encryption," *Information Sciences*, vol. 480, pp. 403–419, 2019.

[5]. M. Asgari-chenaghlu, M.-A. Balafar, and M.-R. Feizi-Derakhshi, "A novel image encryption algorithm based on polynomial combination of chaotic maps and dynamic function generation," *Signal Processing*, vol. 157, p. 1, 2019.
[6]. X. Zhang and X. Wang, *Multiple-image Encryption Algorithm Based on DNA Encoding and Chaotic System*, Springer, New York, NY, USA, 2019.
[7]. J. S. Zhenjun and R. Sun, "Multiple-image encryption with bit-plane decomposition and chaotic maps," *Optics and Lasers in Engineering*, vol. 80, pp. 1–11, 2016.
[8]. Ziad A AlQadi Amjad Y Hindi, O Dwairi Majed, PROCEDURES FOR SPEECH RECOGNITION USING LPC AND ANN, International Journal of Engineering Technology Research & Management, vol. 4, issue 2, pp. 48-55, 2020.
[9]. AlQaisi Aws, Tarawneh Mokhled, A Alqadi Ziad, A Sharadqah Ahmad, Analysis of Color Image Features Extraction using Texture Methods, TELKOMNIKA, vol. 17, issue 3, 2018.
[10]. Mua'ad M.Abu-Faraj, Ziad A. Alqadi, Rounds Reduction and Blocks Controlling to Enhance the Performance of Standard Method of Data Cryptography, IJCSNS International Journal of Computer Science and Network Security, VOL.21 No.12, December 2021.
[11]. M. Abu-Faraj, and Z. Alqadi, "Image Encryption using Variable Length Blocks and Variable Length Private Key," International Journal of Computer Science and Mobile Computing (IJCSMC), vol. 11, Iss. 3, pp. 138-151, 2022.
[12]. M. Abu-Faraj, A. Al-Hyari, and Z. Alqadi, "A Dual Approach for Audio Cryptography," Journal of Southwest Jiaotong University, vol. 57, no. 1, pp. 24-33, 2022.
[13]. M. Abu-Faraj, A. Al-Hyari, and Z. Alqadi, "Complex Matrix Private Key to Enhance the Security Level of Image Cryptography," Symmetry, vol. 14, Iss. 4, pp. 664-678, 2022.
[14]. M. Abu-Faraj, K. Aldebei, and Z. Alqadi, "Simple, Efficient, Highly Secure, and Multiple Pur- posed Method on Data Cryptography," Traitement du Signal, vol. 39, no. 1, pp. 173-178, 2022.
[15]. M. Abu-Faraj, Khaled Aldebe, and Z. Alqadi, "Deep Machine Learning to Enhance ANN Performance: Fingerprint Classifier Case Study," Journal of Southwest Jiaotong University, vol. 56, no. 6 , pp. 685-694, 2021.
[16]. M. Abu-Faraj, Z. Alqadi, and K. Aldebei, "Comparative Analysis of Fingerprint Features Ex- traction Methods," Journal of Hunan University Natural Sciences, vol. 48, iss. 12, pp. 177-182, 2021.
[17]. M. Abu-Faraj, and Z. Alqadi, "Improving the Efficiency and Scalability of Standard Meth- ods for Data Cryptography," International Journal of Computer Science and Network Security (IJCSNS), vol. 21, no.12 , pp. 451-458, 2021.
[18]. Rashad J. Rasras, Mohammed Abuzalata; Ziad Alqadi; Jamil Al-Azzeh; Qazem Jaber, Comparative Analysis of Color Image Encryption-Decryption Methods Based on Matrix Manipulation International Journal of Computer Science and Mobile Computing, Vol.8 Issue.3, March- 2019, pg. 14-26.
[19]. Jamil Al-Azzeh, Bilal Zahran, Ziad Alqadi, Belal Ayyoub, Muhammed Mesleh: A Novel Based On Image Blocking Method to Encrypt-Decrypt Color JOIV: International Journal on Informatics Visualization, 2019.
[20]. Mua'ad M. Abu-Faraj Prof. Ziad Alqadi, Using Highly Secure Data Encryption Method for Text File Cryptography, International Journal of Computer Science and Network Security, vol. 20, issue 11, pp. 53-60, 2021.
[21]. Ziad A. Alqadi Mua'ad M. Abu-Faraj, Improving the Efficiency and Scalability of Standard Methods for Data Cryptography, International Journal of Computer Science and Network Security, vol. 21, issue 12, pp. 451-458, 2021.
[22]. Dr. Mohamad Barakat Prof. Ziad Alqadi, Highly Secure Method for Secret Data Transmission, International Journal of Scientific Engineering and Science, vol. 6, issue 1, pp. 49-55, 2022.
[23]. Saleh Khawatreh, Belal Ayyoub, Ashraf Abu-Ein, Ziad Alqadi, A Novel Methodology to Extract Voice Signal Features, International Journal of Computer Applications, vol. 179, issue 9, pp. 40-43, 2018.
[24]. Hamdan, M., Subaih, B., Alqadi, Z.: Extracting Isolated Words from an Image of Text. International Journal of Computer Science & Mobile Computing 5(11), 29-36 (2016)
[25]. Hindi, A., Dr. Dwairi, M., Alqadi, Z.: Analysis of Procedures used to Build an Optimal Fingerprint Recognition System. International Journal of Computer Science and Mobile Computing 9(2), 21-37 (2020) https://doi.org/10.47760/ijcsmc.2020.v09i11.002
[26]. Rushdi Abu Zneit, R., Al-Azzeh, J. Alqadi, Z., Ayyoub, B., Sharadqh, A.: Using Color Image as a Stego-Media to Hide Short Secret Messages. International Journal of Computer Science and Mobile Computing, vol. 8, issue 6, pp. 106-123, 2019.
[27]. Dr. Mohamad Barakat Prof. Ziad Alqadi, Highly Secure Method for Secret Data Transmission, International Journal of Scientific Engineering and Science, vol. 6, issue 1, pp. 49-55,2022.
[28]. Naseem Asad, Ismail Shayeb, Qazem Jaber, Belal Ayyoub, Ziad Alqadi, Ahmad Sharadqh, creating a Stable and Fixed Features Array for Digital Color Image, IJCSMC, Vol. 8, Issue. 8, August 2019, pg.50 – 62.
[29]. Majed O. Al-Dwairi, Amjad Y. Hendi, Mohamed S. Soliman, Ziad A.A. Alqadi, A new method for voice signal features creation, International Journal of Electrical and Computer Engineering (IJECE), vol. 9, issue 5, pp. 4092-4098, 2018.
[30]. Akram A. Moustafa and Ziad A. Alqadi, A Practical Approach of Selecting the Edge Detector Parameters to Achieve a Good Edge Map of the Gray Image, Journal of Computer Science 5 (5): 355-362, 2009.
[31]. ZA Alqadi, Musbah Aqel, Ibrahiem MM El Emary, Performance analysis and evaluation of parallel matrix multiplication algorithms, World Applied Sciences Journal, vol. 5, issue 2, pp. 211-214, 2008.
[32]. Ayman Al-Rawashdeh, Ziad Al-Qadi, using wave equation to extract digital signal features, Engineering, Technology & Applied Science Research, vol. 8, issue 4, pp. 1356-1359, 2018.

[33]. Ziad Alqadi, Bilal Zahran, Qazem Jaber, Belal Ayyoub, Jamil Al-Azzeh, Enhancing the Capacity of LSB Method by Introducing LSB2Z Method, International Journal of Computer Science and Mobile Computing, vol. 8, issue 3, pp. 76-90, 2019.

[34]. Ziad A. Alqadi, Majed O. Al-Dwairi, Amjad A. Abu Jazar and Rushdi Abu Zneit, Optimized True-RGB color Image Processing, World Applied Sciences Journal 8 (10): 1175-1182, ISSN 1818-4952, 2010.

[35]. Waheeb, A. and Ziad AlQadi, Gray image reconstruction. Eur. J. Sci. Res., 27: 167-173, 2009.

[36]. A. A. Moustafa, Z. A. Alqadi, "Color Image Reconstruction Using a New R'G'I Model", Journal of Computer Science, Vol.5, No. 4, pp. 250-254, 2009.

[37]. K Matrouk, A Al-Hasanat, H Alasha'ary, Z. Al-Qadi Al-Shalabi, "Speech fingerprint to identify isolated word person", World Applied Sciences Journal, Vol. 31, No. 10, pp. 1767-1771, 2014.

[38]. J. Al-Azzeh, B. Zahran, Z. Alqadi, B. Ayyoub, M. Abu-Zaher, "A Novel zero-error method to create a secret tag for an image", Journal of Theoretical and Applied Information Technology, Vol. 96. No. 13, pp. 4081-4091, 2018.

[39]. Prof. Ziad A.A. Alqadi, Prof. Mohammed K. Abu Zalata, Ghazi M. Qaryouti, Comparative Analysis of Color Image Steganography, JCSMC, Vol.5, Issue. 11, November 2016, pg.37–43.

[40]. M. Jose, "Hiding Image in Image Using LSB Insertion Method with Improved Security and Quality", International Journal of Science and Research, Vol. 3, No. 9, pp. 2281-2284, 2014.

[41]. M. Juneja, P. S. Sandhu, An improved LSB based Steganography with enhanced Security and Embedding/Extraction, 3rd International Conference on Intelligent Computational Systems, Hong Kong China, January 26-27, 2013.

[42]. H. Alasha'ary, K. Matrouk, A. Al-Hasanat, Z. A alqadi, H. Al-Shalabi (2013), Improving Matrix Multiplication Using Parallel Computing, International Journal on Information Technology (I.RE.I.T.) Vol. 1, N. 6 ISSN 2281-2911.

[43]. Bilal Zahran, Ziad Alqadi, Jihad Nader, Ashraf Abu Ein A COMPARISON BETWEEN PARALLEL AND SEGMENTATION METHODS USED FOR IMAGE ENCRYPTION-DECRYPTION, International Journal of Computer Science & Information Technology (IJCSIT) Vol 8, No 5, October 2016.

[44]. Z.A. Alqadi, A. Abu-Jazar (2005), Analysis of Program Methods Used for Optimizing Matrix Multiplication, Journal of Engineering, vol. 15 n. 1, pp. 73-78.

[45]. Jamil Al-Azzeh, Bilal Zahran, Ziad Alqadi, Belal Ayyoub, Muhammed Mesleh: A Novel Based On Image Blocking Method to Encrypt-Decrypt Color JOIV: International Journal on Informatics Visualization, 2019.

[46]. Jamil Al-Azzeh, Bilal Zahran, Ziad Alqadi, Belal Ayyoub and Mazen Abu-Zaher: A Novel Zero-Error Method to Create a Secret Tag for an Image; Journal of Theoretical and Applied Information Technology 15th July 2018.

[47]. Jamil Al Azzeh, Ziad Alqadi Qazem, M. Jabber: Statistical Analysis of Methods Used to Enhanced Color Image Histogram; XX International Scientific and Technical Conference; Russia May 24-26, 2017.

