JITE, Vol. 1 (2) Januari (2018) p-ISSN: 2549-6247e-ISSN: 2549-6255

JOURNAL OF INFORMATICS AND TELECOMMUNICATION ENGINEERING

Available online http://ojs.uma.ac.id/index.php/jite

Perancangan Kunci Public RSA dan ElGamal pada Kriptografi untuk Kemananan Informasi

RSA and ElGamal Public Key Design on Cryptography for Information Security

Susilawati *

Universitas Medan Area

*E-mail Korespondensi: susi.shilawati@gmail.com

Abstrak

The use of very large primes in public key cryptography is necessary to avoid unauthorized decryption of messages. In its implementation, the generation of large prime numbers is constrained on the operation of the powers with large numbers. This paper discusses the technique of generating very large numbers using Linear Congruential Generator (LCG) algorithms and probabilistic prime number testing using the Miller-Rabin algorithm. The prime numbers generated are tested and analyzed on RSA and ElGamal public key cryptosystems for the purpose of information security.

Kata Kunci: Linear Congruential Generator (LCG), Miller-Rabin, RSA, ElGamal.

Abstract

The use of very large primes in public key cryptography is necessary to avoid unauthorized decryption of plaintext. In the implementation, the generation of large prime numbers is constrained on the operation of the powers with large numbers. This paper discusses the technique of generating very large numbers using Linear Congruential Generator (LCG) algorithms and probabilistic prime number testing using the Miller-Rabin algorithm. The prime numbers generated are tested and analyzed on the RSA and ElGamal public key cryptosystems for the purpose of information security.

Keywords: Linear Congruential Generator (LCG), Miller-Rabin, RSA, ElGamal

PENDAHULUAN

Kriptografi merupakan teknik menyandikan pesan yang memanfaatkan ilmu matematika diantaranya adalah bilangan prima dengan segala sifat bilangan bulatnya seperti aritmatika modulo, perpangkatan, faktor persekutuan terbesar dan faktor kuadrat.

Teori bilangan (*Number Theory*) merupakan teori mendasar dalam memahami kriptografi. khususnya pada sistem kriptografi dengan kunci asimetrik atau kriptografi kunci publik. Sebagian besar kriptografi kunci publik menggunakan bilangan prima sebagai parameternya. Bilangan prima yang disarankan adalah bilangan prima yang berukuran besar yakni yang terdiri dari lebih dari seratus angka.

Makalah ini membahas tentang pembangkitan bilangan acak menggunakan algoritma *Linear Congruential Generator* (LCG), pengujian keprimaan bilangan menggunakan algoritma Miller-Rabin dan implementasi kriptografinya menggunakan algoritma RSA dan ElGamal.

HASIl DAN PEMBAHASAN

Linear Congruential Generator

Bilangan acak (*random*) banyak digunakan dalam kriptografi kunci publik sebagai pembangkit parameter kunci, tujuannya agar bilangan tidak mudah diprediksi. Tidak ada prosedur komputasi yang benar-benar menghasilkan deret bilangan acak secara sempurna. Bilangan acak yang dihasilkan dengan rumus-rumus matematika adalah bilangan acak semu (*pseudo*), karena pembangkitan bilangannya dapat diulang kembali secara periodik. Pembangkitan seperti ini disebut *pseudo–random number generator* (*PRNG*).

Metode pembangkitan bilangan acak yang paling umum digunakan adalah *Linear Congruential Generator* (LCG). LCG merupakan pembangkit bilangan acak yang sederhana dan mudah untuk diimplementasikan. LCG didefinisikan dalam bentuk berikut :

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

dimana :
X_n = bilangan acak ke-n dari deretnya
X_{n-1} = bilangan acak sebelumnya
a = faktor pengali
b = *increment*
m = modulus
(*a, b,* dan *m* adalah konstan)

Persamaan (1) memiliki nilai awal X_0 sebagai kunci pembangkit (*seed*). 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* kelipatan 4 jika *m* juga kelipatan 4.
4. *m* > maks(a, b, X_0).
5. $a > 0, b > 0$.

Misalnya nilai-nilai untuk :
$X_n \leftarrow a = 11, b = 17, m = 23$ dan $X_0 = 0$

Maka akan didapat rumusan sebagai berikut:

$$X_n = (11X_{n-1} + 17) \bmod 23$$

Kemudian dihitung nilai X_n seperti dapat dilihat pada tabel 1.

n	X_n
1	17
2	20
3	7
4	2
5	16
6	9
7	1
8	5

9	3
10	4
11	15
12	21
13	18
14	8
15	13
16	22
17	6
18	14
19	10
20	12
21	11
22	0
23	17
24	20

Tabel 1. Rangkaian bilangan acak n dan X_n untuk a = 11, b = 17, m = 23 dan X0 = 0

Dari tabel 1 tampak bahwa barisan bilangan berulang pada *n* = 23. Ini membuktikan bahwa bilangan acak semu memiliki suatu periode tertentu dan akan kembali ke keadaan awal setelah periode terlewati.

LCG memiliki keunggulan lebih cepat dan membutuhkan sedikit operasi bit dibandingkan dengan pembangkit bilangan semu lainnya. Kekurangan LCG terletak pada ketidakcocokannya bila digunakan untuk kriptografi karena bilangan acak yang dihasilkan dapat dengan mudah diprediksi urutan kemunculannya. Secara teoritis LCG mampu menghasilkan bilangan acak dengan baik, hal ini sangat sensitif terhadap pemilihan nilai-nilai *a*, *b* dan *m*. Pemilihan nilai-nilai yang buruk dapat mengarah pada implementasi LCG yang tidak efisien. Tabel 2 menyajikan nilai konstanta yang baik untuk LCG berdasarkan hasil analisis.

A	B	M
106	1283	6075
211	1663	7875
421	1663	7875
430	2351	11979
936	1399	6655
1366	1283	6075
171	11213	53125
859	2531	11979
419	6173	29282
967	3041	14406
141	28411	134456
625	6571	31104
1541	2957	14000
1741	2731	12960
1291	4621	21870
205	29573	139968
421	17117	81000
1255	6173	29282
281	284111	134456

Tabel 2. Konstanta baik untuk implementasi LCG

Miller-Rabin

Pada beberapa sistem kriptografi kunci publik dibutuhkan fungsi untuk memilih bilangan prima. Persoalannya adalah bagaimana algoritma menguji keprimaan suatu bilangan integer. Ada 2 algoritma pengujian bilangan prima yakni pengujian bersifat deterministik (pasti) dan pengujian bersifat probabilistik. Algoritma pengujian bersifat deterministik secara pasti menyatakan suatu bilangan integer termasuk bilangan prima atau bukan. Sedangkan algoritma pengujian bersifat probabilistik hanya memberikan status keprimaan suatu bilangan integer dengan nilai probabilitas. Algoritma pengujian yang banyak dipakai dalam sistem kriptografi adalah algoritma probabilistik Miller-Rabin. Algoritma ini memiliki 2 keunggulan yakni ringan dalam komputasi dan nilai probabilitas yang dihasilkan tinggi.

Algoritma Rabin-Miller untuk pengujian keprimaan

Input : (*n*,*a*) {*n* adalah bilangan integer positif dan *a* adalah basis}

Output : Status keprimaan *n* dengan probabilitas ¾

Temukan *m* dan *k* sehingga $n-1 = m \times 2^k$

$T = a^m \bmod n$

If $T = \pm 1$ then

return “Mungkin Bilangan Prima”
End if
For $i = 1 \rightarrow k - 1$ do
$T = T^2 \bmod n$
If $T = +1$ then
return “Bilangan Komposit”
End if
If $T = -1$ then
return“Mungkin Bilangan Prima”
End if
End for
return “Bilangan Komposit”

Perancangan Kunci Publik

Kriptografi kunci publik atau sering juga disebut sebagai kriptografi kunci asimetrik pertama kali diusulkan oleh Deffie dan Hellman pada tahun 1976, yang memungkinkan pengguna berkomunikasi secara aman tanpa perlu berbagi kunci rahasia. Dikatakan kriptografi kunci publik, karena kunci untuk enkripsi bersifat publik sehingga dapat diketahui oleh siapapun, sedangkan kunci untuk dekripsi bersifat rahasia karena hanya diketahui oleh penerima pesan.

Sistem kriptografi kunci-publik yang aman memiliki dua karakteristik sebagai berikut :

1. Komputasi untuk enkripsi dan dekripsi pesan mudah dilakukan
2. Secara komputasi hampir tidak mungkin menurunkan kunci privat, yang disimbolkan dengan *d* , bila diketahui kunci publik yang disimbolkan dengan *e*, pasangannya.

Dua permasalahan matematis yang sering dijadikan dasar perancangan sepasang kunci pada kriptografi kunci publik adalah pemfaktoran, tujuan dilakukannya pemfaktoran adalah untuk memperoleh kunci privat. Contoh algoritma kriptografi yang menggunakan prinsip ini adalah RSA. Kesulitan RSA terletak pada pemfaktorkan bilangan yang besar menjadi faktor-faktor prima. Selain pemfaktoran, permasalahan berikutnya adalah logaritma diskrit, hal ini ditemukan pada algoritma kriptografi ElGamal.

Kriptografi RSA

RSA (*Rivest-Shamir-Adelman*) adalah algoritma kunci publik yang paling umum digunakan. RSA dapat digunakan baik untuk enkripsi pesan maupun untuk tanda tangan digital, makalah ini membahas RSA untuk enkripsi pesan. RSA umumnya aman bila kunci yang digunakan panjang (512 bit tidak aman, 768 bit cukup aman, 1024 bit 2048 bit aman). Semua algoritma kunci publik membutuhkan kunci yang sangat panjang, semakin panjang bit kunci semakin tinggi pula tingkat keamanannya.

Keamanan RSA bergantung pada kesulitan memfaktorkan bilangan integer besar menjadi faktor-faktor primanya. Algoritma RSA memiliki besaran-besaran sebagai berikut :

1. $\Phi(n) = (p - 1)(q - 1)$ (rahasia)
2. *e* (kunci enkripsi) (tidak rahasia)
3. *d* (kunci dekripsi) (rahasia)
4. *m* (plainteks) (rahasia)
5. *c* (cipherteks) (tidak rahasia)

Pembangkit kunci RSA

1. Pilih 2 bilangan prima besar seperti *p*,*q* dimana *p* tidak sama dengan *q*.
2. Hitung $n = p \times q$
3. Hitung $\Phi(n) = (p - 1) * (q - 1)$
4. Pilih ***e*** yang relatif prima terhadap $\Phi(n)$

   Relatif prima terhadap $\Phi(n)$ artinya faktor pembagi terbesar keduanya adalah 1, secara matematis dinotasikan $gcd(e, \Phi(n)) = 1$.

5. Hitung ***d*** integer sehingga $e * d = 1$ mod n atau ($1 + m \cdot n$) / e.

Algortima Enkripsi dan Dekripsi RSA

Enkripsi :

1. Ambil kunci publik penerima pesan e dan modulus n
2. Nyatakan plainteks m menjadi blok-blok m_1, m_2, ..., sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0,n -1].
3. Setiap blok m_i dienkripsi menjadi blok Ci dengan rumus :

$$C_i = m_i^e \bmod n \qquad (2)$$

Dekripsi :

Setiap blok cipherteks C_i didekripsi kembali menjadi blok m_i dengan rumus :

$$M_i = C_i^d \bmod n \qquad (3)$$

Kriptografi El Gamal

Sistem kriptografi ElGamal ditemukan pada tahun 1984 oleh Taher ElGamal. Algoritma ini pada mulanya digunakan untuk *digital signature*, namun kemudian dimodifikasi sehingga juga bisa digunakan untuk enkripsi dan dekripsi. Keamanan algoritma ElGamal terletak pada sulitnya menghitung logaritma diskrit.

Besaran-besaran yang digunakan di dalam algoritma ElGamal adalah :

1. Bilangan prima, p (tidak rahasia)
2. Bilangan acak, α $(\alpha < p)$ (tidak rahasia)
3. Bilangan acak, d $(d < p)$ (rahasia, kunci privat)
4. $\beta = \alpha^d \bmod p$ (tidak rahasia, kunci publik)
5. P (plainteks) (rahasia)
6. $c1$ dan $c2$ (cipherteks) (tidak rahasia)

Langkah-langkah yang dilakukan dalam pembangkit kunci adalah

1. Memilih sebuah bilangan prima p untuk membentuk grup perkalian (Z_p^*, x). Kemudian pilih akar primitif
2. Pilih akar primitif α pada (Z_p^*, x). α merupakan akar primitif pada (Z_p^*, x) bila order α dinotasikan $\alpha = p - 1$.
3. Pembangkit kunci memilih sebuah bilangan integer d yang memenuhi $1 < d < p - 2$ dan menghitung $\beta = \alpha^d \bmod p$
4. Kemudian menetapkan kunci publik $K_{publik} = (p, \alpha, \beta)$, dan kunci privat $K_{privat} = d$

Algoritma Pembangkitan Kunci ElGamal

1. Pilih bilangan prima p besar sebagai basis grup perkalian (Z_p^*, x)
2. Pilih α sebagai akar primitif pada grup (Z_p^*, x)
3. Pilih d yang memenuhi $1 \le d \le p - 2$
4. Hitung $\beta = \alpha^d \bmod p$
5. $K_{publik} = (p, \alpha, \beta)$, $K_{privat} = d$

Algoritma Ekripsi dan Dekripsi ElGamal

Enkripsi :

Input : $K_{publik} = (p, \alpha, \beta)$ $P \in Z_p^*$

Output : C_1, C_2 Z_n^*

$r \leftarrow Z_n^*$ { r dipilih acak }

$C_1 = \alpha^r \bmod p$

$C_2 = (P \times \beta^r) \bmod p$

Dekripsi :

Input : $K_{private} = d, C1, C2 \in Z_p^*$

Output : $P \in Z_n^*$

$P = [\, C_2 \times C_1^d)^{-1} \bmod p$

Untuk membangkitkan kunci publik pada kriptografi RSA digunakan bilangan prima sebagai parameternya. Bilangan prima yang disarankan adalah bilangan prima yang berukuran besar, terdiri lebih dari seratus angka. LCG secara teoritis mampu menghasilkan bilangan acak yang baik, akan tetapi hal ini sangat bergantung pada pemilihan nilai a, b dan m. Pemilihan nilai-nilai yang buruk dapat mengarah pada implementasi LCG yang tidak efisien. LCG mampu menghasilkan bilangan acak sesuai batasan nilai dan iterasi, keprimaannya dapat diuji dengan algoritma Miller-Rabin.

Langkah pertama yang dilakukan dalam pengujian ini adalah membangkitkan bilangan acak sebanyak n bilangan dengan menggunakan algoritma LCG sesuai dengan persamaan (1), disarankan untuk menggunakan nilai a, b dan m berdasarkan tabel 2.

Misalnya, nilai a, b dan m yang dipilih adalah :

a = 106 , b = 1283, m = 6075

$$X_n = (106.\, X_{n-1} + 1283) \bmod 6075;\ X_0 = 0$$

```
iter=1, lcg = 1283
iter=2, lcg = 3631
iter=3, lcg = 3444
iter=4, lcg = 1847
iter=5, lcg = 2665
iter=6, lcg = 4323
iter=7, lcg = 3896
iter=8, lcg = 1159
iter=9, lcg = 2637
iter=10, lcg = 1355
iter=11, lcg = 5188
iter=12, lcg = 4461
iter=13, lcg = 299
iter=14, lcg = 2602
iter=15, lcg = 3720
iter=16, lcg = 728
iter=17, lcg = 5551
iter=18, lcg = 414
iter=19, lcg = 2642
iter=20, lcg = 1885
iter=21, lcg = 618
iter=22, lcg = 6041
iter=23, lcg = 3754
iter=24, lcg = 4332
iter=25, lcg = 4850
iter=26, lcg = 5083
iter=27, lcg = 5481
iter=28, lcg = 5144
iter=29, lcg = 5872
iter=30, lcg = 4065
iter=31, lcg = 848
iter=32, lcg = 46
iter=33, lcg = 84
iter=34, lcg = 4112
iter=35, lcg = 5830
iter=36, lcg = 5688
iter=37, lcg = 2786
iter=38, lcg = 4999
iter=39, lcg = 2652
iter=40, lcg = 2945
iter=41, lcg = 3628
iter=42, lcg = 3126
iter=43, lcg = 4589
iter=44, lcg = 1717
iter=45, lcg = 1035
iter=46, lcg = 1643
iter=47, lcg = 5341
iter=48, lcg = 2454
iter=49, lcg = 182
iter=50, lcg = 2350
iter=51, lcg = 1308
iter=52, lcg = 206
iter=53, lcg = 4894
iter=54, lcg = 3672
iter=55, lcg = 1715
iter=56, lcg = 823
iter=57, lcg = 3471
iter=58, lcg = 4709
iter=59, lcg = 2287
iter=60, lcg = 705
iter=61, lcg = 3113
iter=62, lcg = 3211
iter=63, lcg = 1449
iter=64, lcg = 3002
iter=65, lcg = 3595
```

```
iter=66, lcg = 5703
iter=67, lcg = 4376
iter=68, lcg = 3439
iter=69, lcg = 1317
iter=70, lcg = 1160
iter=71, lcg = 2743
iter=72, lcg = 441
iter=73, lcg = 5504
iter=74, lcg = 1507
iter=75, lcg = 3075
iter=76, lcg = 5258
iter=77, lcg = 5806
iter=78, lcg = 3144
iter=79, lcg = 422
iter=80, lcg = 3490
iter=81, lcg = 648
iter=82, lcg = 3146
iter=83, lcg = 634
iter=84, lcg = 1662
iter=85, lcg = 1280
iter=86, lcg = 3313
iter=87, lcg = 111
iter=88, lcg = 899
iter=89, lcg = 5452
iter=90, lcg = 2070
iter=91, lcg = 2003
iter=92, lcg = 976
iter=93, lcg = 1464
iter=94, lcg = 4592
iter=95, lcg = 2035
iter=96, lcg = 4368
iter=97, lcg = 2591
iter=98, lcg = 2554
iter=99, lcg = 4707
iter=100, lcg = 2075
iter=101, lcg = 2533
iter=102, lcg = 2481
iter=103, lcg = 3044
iter=104, lcg = 1972
iter=105, lcg = 3765
iter=106, lcg = 5498
iter=107, lcg = 871
iter=108, lcg = 2484
iter=109, lcg = 3362
iter=110, lcg = 5305
iter=111, lcg = 4713
iter=112, lcg = 2711
iter=113, lcg = 3124
iter=114, lcg = 4377
iter=115, lcg = 3545
iter=116, lcg = 403
iter=117, lcg = 1476
iter=118, lcg = 5864
iter=119, lcg = 3217
iter=120, lcg = 2085
iter=121, lcg = 3593
iter=122, lcg = 5491
iter=123, lcg = 129
iter=124, lcg = 2807
iter=125, lcg = 1150
iter=126, lcg = 1683
iter=127, lcg = 3506
iter=128, lcg = 2344
iter=129, lcg = 672
iter=130, lcg = 5690
```

```
iter=131, lcg = 2998
iter=132, lcg = 3171
iter=133, lcg = 3284
iter=134, lcg = 3112
iter=135, lcg = 3105
iter=136, lcg = 2363
iter=137, lcg = 2686
iter=138, lcg = 474
iter=139, lcg = 2927
iter=140, lcg = 1720
iter=141, lcg = 1353
iter=142, lcg = 4976
iter=143, lcg = 214
iter=144, lcg = 5742
iter=145, lcg = 2435
iter=146, lcg = 4243
iter=147, lcg = 1491
iter=148, lcg = 1379
iter=149, lcg = 1657
iter=150, lcg = 750
iter=151, lcg = 1808
iter=152, lcg = 4606
iter=153, lcg = 3519
iter=154, lcg = 3722
iter=155, lcg = 940
iter=156, lcg = 3723
iter=157, lcg = 1046
iter=158, lcg = 2809
iter=159, lcg = 1362
iter=160, lcg = 5930
iter=161, lcg = 4138
iter=162, lcg = 2511
iter=163, lcg = 149
iter=164, lcg = 4927
iter=165, lcg = 1095
iter=166, lcg = 1928
iter=167, lcg = 5176
iter=168, lcg = 3189
iter=169, lcg = 5192
iter=170, lcg = 4885
iter=171, lcg = 2718
iter=172, lcg = 3866
iter=173, lcg = 4054
iter=174, lcg = 5757
iter=175, lcg = 4025
iter=176, lcg = 2683
iter=177, lcg = 156
iter=178, lcg = 5669
iter=179, lcg = 772
iter=180, lcg = 4140
iter=181, lcg = 2723
iter=182, lcg = 4396
iter=183, lcg = 5559
iter=184, lcg = 1262
iter=185, lcg = 1405
iter=186, lcg = 4413
iter=187, lcg = 1286
iter=188, lcg = 3949
iter=189, lcg = 702
iter=190, lcg = 2795
iter=191, lcg = 5953
iter=192, lcg = 501
iter=193, lcg = 5789
iter=194, lcg = 1342
iter=195, lcg = 3810
```

```
iter=196, lcg = 4193
iter=197, lcg = 2266
iter=198, lcg = 4554
iter=199, lcg = 4082
iter=200, lcg = 2650
iter=201, lcg = 2733
iter=202, lcg = 5456
iter=203, lcg = 2494
iter=204, lcg = 4422
iter=205, lcg = 2240
iter=206, lcg = 1798
iter=207, lcg = 3546
iter=208, lcg = 509
iter=209, lcg = 562
iter=210, lcg = 105
iter=211, lcg = 263
iter=212, lcg = 4861
iter=213, lcg = 174
iter=214, lcg = 1502
iter=215, lcg = 2545
iter=216, lcg = 3753
iter=217, lcg = 4226
iter=218, lcg = 5764
iter=219, lcg = 4767
iter=220, lcg = 2360
iter=221, lcg = 2368
iter=222, lcg = 3216
iter=223, lcg = 1979
iter=224, lcg = 4507
iter=225, lcg = 5175
iter=226, lcg = 3083
iter=227, lcg = 31
iter=228, lcg = 4569
iter=229, lcg = 5672
iter=230, lcg = 1090
iter=231, lcg = 1398
iter=232, lcg = 3671
iter=233, lcg = 1609
iter=234, lcg = 1737
iter=235, lcg = 3155
iter=236, lcg = 1588
iter=237, lcg = 5586
iter=238, lcg = 4124
iter=239, lcg = 1027
iter=240, lcg = 795
iter=241, lcg = 503
iter=242, lcg = 6001
iter=243, lcg = 5589
iter=244, lcg = 4442
iter=245, lcg = 4360
iter=246, lcg = 1743
iter=247, lcg = 3791
iter=248, lcg = 2179
iter=249, lcg = 1407
iter=250, lcg = 4625
```

```
==DAFTAR BILANGAN PRIMA==
iterasi=1, lcg = 1283
iterasi=2, lcg = 3631
iterasi=4, lcg = 1847
iterasi=38, lcg = 4999
iterasi=56, lcg = 823
iterasi=59, lcg = 2287
iterasi=86, lcg = 3313
iterasi=91, lcg = 2003
iterasi=97, lcg = 2591
iterasi=112, lcg = 2711
iterasi=119, lcg = 3217
iterasi=121, lcg = 3593
iterasi=139, lcg = 2927
iterasi=146, lcg = 4243
iterasi=149, lcg = 1657
iterasi=163, lcg = 149
iterasi=176, lcg = 2683
iterasi=178, lcg = 5669
iterasi=191, lcg = 5953
iterasi=208, lcg = 509
iterasi=211, lcg = 263
iterasi=212, lcg = 4861
iterasi=223, lcg = 1979
iterasi=224, lcg = 4507
iterasi=226, lcg = 3083
iterasi=227, lcg = 31
iterasi=232, lcg = 3671
iterasi=233, lcg = 1609
iterasi=241, lcg = 503
iterasi=248, lcg = 2179
```

```
==URUTKAN BILANGAN PRIMA==
0 : 31
1 : 149
2 : 263
3 : 503
4 : 509
5 : 823
6 : 1283
7 : 1609
8 : 1657
9 : 1847
10 : 1979
11 : 2003
12 : 2179
13 : 2287
14 : 2591
15 : 2683
16 : 2711
17 : 2927
18 : 3083
19 : 3217
20 : 3313
21 : 3593
22 : 3631
23 : 3671
24 : 4243
25 : 4507
26 : 4861
27 : 4999
28 : 5669
29 : 5953
```

Setelah dilakukan pengujian sebanyak 250 iterasi, tidak ditemukan adanya periode perulangan LCG, seperti pada gambar 1. Langkah berikutnya adalah menguji keprimaan bilangan-bilangan acak yang dihasilkan LCG dengan menggunakan algoritma Miller-Rabin. Setelah dilakukan pengujian diperoleh bilangan-bilangan prima seperti terlihat pada gambar 1.

Gambar 1. Bilangan hasil algoritma LCG dan bilangan prima hasil algoritma Miller-Rabin

Kemudian diseleksi dua bilangan prima yang bernilai besar untuk dijadikan nilai p dan q pada kriptografi RSA, dan nilai p pada kriptografi ElGamal.

```
==AMBIL 2 BILANGAN PRIMA TERBESAR==
prima1 = 5953
prima2 = 5669
```

Selanjutnya kedua bilangan prima tersebut diimplementasi untuk merancang pembangkitan kunci publik pada algoritma RSA dan algoritma ElGamal. Implementasi pembangkitan kunci publik RSA sebagai berikut :

1. p = 5953
   q = 5669

nilai p dan q dipilih dari bilangan prima yang dihasilkan pembangkitan bilangan acak di atas.

Berdasarkan algoritma pembangkitan pasangan kunci RSA di atas, diperoleh hasil sebagai berikut :

2. n = 5953 x 5669
   n = 33747557
3. $\Phi(n)$ = (p-1) x (q-1)
   $\Phi(n)$ = (5953-1) x (5669-1)
   $\Phi(n)$ = 33735936
4. Menghitung e bilangan relatif prima terhadap n.
   e = 5, maka gcd(5, 33735936) = 1
5. Menghitung d dimana, $e.d$ mod $\Phi(n)$ = 1
   d = 26988749, karena
   5 x 26988749 mod 33735936 = 1

Dari perhitungan tersebut diperoleh kunci publik (5, 33747557) dan kunci privat (26988749, 33747557). Kunci publik dan kunci privat ini akan digunakan untuk proses enkripsi dan dekripsi pada RSA sebagai berikut :

Misal pesan adalah 'HUJAN'

Pesan	H	U	J	A	N
ASCII	72	85	74	65	78

Plainteks : 7285746578

Kemudian dipecah menjadi 4 blok

Plainteks :

Pesan	728	574	657	008

$C1 = 728^5$ mod 33747557 = 27155095

$C2 = 574^5$ mod 33747557 = 4861433

$C3 = 657^5$ mod 33747557 = 21008501

$C4 = 008^5$ mod 33747557 = 32768

Hasil enkripsi menjadi :

Ciphertek s	2715509 5	486143 3	2100850 1	3276 8

Selanjutnya cipherteks tersebut didekripsi agar kembali ke pesan aslinya dengan menggunaan kunci privat sebagai berikut :

$P1 = 27155095^{26988749}$ mod 33747557 = 728

$P2 = 4861433^{26988749}$ mod 33747557 = 574

$P3 = 21008501^{26988749}$ mod 33747557 = 657

$P4 = 32768^{26988749}$ mod 33747557 = 8

Hasil dekripsi menjadi :

Plainteks	728	574	657	8

Kemudian pesan tersebut disatukan kembali menjadi :

Pesan : 7285746578

Sehingga dihasilkan :

ASCII	72	85	74	65	78
Pesan :	H	U	J	A	N

Selanjutnya implementasi prima tersebut untuk pembangkitan kunci ElGamal sebagai berikut :

Prima yang dipilih adalah prima besar yang dihasilkan dari pembangkitan bilangan acak sebelumnya, dimana

1. p = 5953
2. Pilih α sebagai akar primitif pada (Z^*_{p}, x)

α = 4365

3. Pilih sebuah bilangan integer acak d yang memenuhi $1 \leq d \leq p - 2$. Maka nilai d = 4680
4. $\beta = 4365^{4680} \bmod 5953$, β = 1316
5. Maka diperoleh kunci publik = (5953, 4365, 1316) dan kunci privat = (4680)

Kunci publik dan kunci privat ini akan digunakan untuk proses enkripsi pada ElGamal. Berdasarkan algoritma enkripsi dan dekripsi ElGamal diperoleh hasil sebagai berikut :

1. $Kunci_{publik}$ = (5953, 4365, 1316)
2. r dipilih secara acak, maka nilai r = 1597
3. $C_1 = 4365^{1597} \bmod 5953$
   C_1 = 3778
4. Menghitung nilai C_2 sebagai berikut :

Teks	ASCII	C_2
H	72	$(72 * 1316^{1597}) \bmod 5953 = 262$
U	85	$(85 * 1316^{1597}) \bmod 5953 = 2459$
J	74	$(74 * 1316^{1597}) \bmod 5953 = 600$
A	65	$(65 * 1316^{1597}) \bmod 5953 = 5032$
N	78	$(78 * 1316^{1597}) \bmod 5953 = 1276$

Hasil enkripsi menjadi :

Cipherteks	262	2459	600	5032	1278

Selanjutnya cipherteks tersebut didekripsi agar kembali ke pesan aslinya dengan menggunaan kunci privat sebagai berikut :

1. $Kunci_{private}$ = (4680)
2. Menghitung nilai P sebagai berikut :

C_2	P
262	$262 \times 3778^{5953-1-4680} \bmod 5953 = 72$
2459	$2459 \times 3778^{5953-1-4680} \bmod 5953 = 85$
600	$600 \times 3778^{5953-1-4680} \bmod 5953 = 74$
5032	$5032 \times 3778^{5953-1-4680} \bmod 5953 = 65$
1276	$1278 \times 3778^{5953-1-4680} \bmod 5953 = 78$

Hasil dekripsi menjadi :

Plainteks	72	85	74	65	78

Sehingga dihasilkan :

ASCII	72	85	74	65	78
Pesan :	H	U	J	A	N

SIMPULAN

Dari hasil pengujian dan analisis sebelumnya, dapat disimpulkan bahwa, LCG mampu menghasilkan bilangan acak yang efisien dengan memperhatikan penggunaan nilai untuk parameter a, b dan m. Algoritma Miller-Rabin memiliki komputasi yang ringan dan memberikan nilai probabilitas yang tinggi untuk menguji keprimaan bilangan acak yang dihasilkan dari LCG. Algoritma RSA dan ElGamal dapat mengembalikan pesan yang telah dienkripsi dari hasil pemfaktoran bilangan prima yang sangat besar yang dihasilkan oleh LCG dan telah diuji menggunakan algoritma Miller-Rabin .

DAFTAR PUSTAKA

Munir, Rinaldi. 2006. *Kriptografi*. Informatika. Bandung.

Sadikin, Rifki. 2012. *Kriptografi Untuk Keamanan Jaringan*. Andi Offset. Yogyakarta.

Schneier, Bruce. 1996. *Applied Cryptography Second Edition*. John Wiley & Sons.

Menezes, Alfred J., Paul C. van Oorschot & Scott A. Vanstone. 2001. *Hand Book of Applied Cripthography*. CRC Press.

Mollin, Richard A. 2000. *Fundamental Number Theory with Application*. CRC Press.

Croft, Anthony., Davison, Robert. 2006. *Foundation Maths*. Ashford Colour Press.