Optimasi Algoritma Blum Blum Shub sebagai Pembangkit Bilangan Acak dalam Algoritma Elgamal untuk Mengamankan File Image

Primanda Adyatma Hafiz - 13520022
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520022@std.stei.itb.ac.id

***Abstract*—Algoritma Blum Blum Shub (BBS) merupakan salah satu algoritma CSPRNG yang paling sederhana untuk digunakan dalam algoritma kriptografi. Akan tetapi, algoritma BBS memiliki kelemahan karena terlalu sederhana sehingga rentan terhadap penyerangan dengan *dictionary attack* dan memiliki *confussion dan diffusion* yang masih kurang baik. Oleh karena itu, dilakukan optimasi algoritma BBS sehingga pembangkitan bilangan mengambil parameter dari dua bilangan acak sebelumnya untuk meningkatkan *randomness*. Selain itu, untuk meningkatkan *confussion* dan *diffusion* dilakukan *swapping* parameter koefisien dan eksponen secara rutin. Hasil algoritma BBS digunakan pada algoritma Elgamal sebagai pembangkit bilangan acak untuk parameter pada algoritma Elgamal. Algoritma Elgamal digunakan dalam proses enkripsi dan dekripsi *file image* yang cenderung berukuran besar. Dalam hal ini, algoritma Elgamal sudah berjalan secara cukup efisien dan memiliki kompleksitas waktu linear**

Keywords—Blum Blum Shub, Elgamal, File Image Encryption

I. PENDAHULUAN

Saat ini banyak terjadi masalah keamanan informasi yang terkait dengan kurang amannya proses pengiriman dari sebuah data yang melibatkan pengirim dan penerima. Dalam hal tersebut, masalah keamanan yang terjadi yaitu terkait dengan mudahnya data dibaca atau dipecahkan oleh pihak ketiga. Oleh karena itu diperlukan algoritma kriptografi yang mumpuni untuk menjaga keabsahan, kerahasiaan, kredibilitas, integritas, dan autentikasi data. Adapun secara umum algoritma kriptografi dapat dibagi menjadi dua yaitu kriptografi klasik dan modern, akan tetapi karena algoritma kriptografi klasik mudah dipecahkan secara *bruteforce* dengan menggunakan *resource* komputasi yang ada saat ini maka kriptografi klasik sudah tidak banyak digunakan sehingga kriptografi lebih didominasi oleh algoritma kriptografi modern.

Dalam pembuatan algoritma kriptografi modern cukup banyak diperlukan pembangkit bilangan acak sebagai salah satu komponen untuk implementasi algoritma. Akan tetapi, secara teoritis tidak ada prosedur komputasi yang mampu menghasilkan deret bilangan acak yang benar-benar sempurna (*truly random*). Oleh karena itu, bilangan acak yang dihasilkan oleh algoritma-algoritma yang ada saat ini menggunakan prosedur komputasi *pseudo-random number generator* (PRNG). PRNG bersifat deterministik dan periodik, akan tetapi dengan membuat periode PRNG sebesar mungkin serta meningkatkan *confusion* dan *diffusion* dalam pembangkitan bilangan acak sehingga PRNG bisa memiliki performa yang dapat diandalkan.

Kemudian algoritma kriptografi sendiri juga dapat menimbulkan permasalahan bila data yang ingin diproses terlalu besar sehingga diperlukan algoritma yang efisien untuk pemrosesan data yang berukuran besar seperti halnya data file gambar.

Oleh karena itu, pada makalah ini akan dilakukan optimasi sebuah algoritma pembangkitan bilangan acak yaitu Blum Blum Shub (BBS) untuk meningkatkan *randomness* tetapi tetap menjaga performa dan efisiensi. Selanjutnya hasil pembangkitan bilangan acak dengan menggunakan algroitma BBS akan digunakan pada algoritma enkripsi Elgamal untuk proses enkripsi file gambar dengan implementasi algoritma Elgamal yang efisien.

II. LANDASAN TEORI

A. Kriptografi Modern

Kriptografi modern adalah era kriptografi setelah penemuan komputer digital. Perkembangan teknologi komputer digital membuat ilmu kriptografi berkembang dengan pesat. Algoritma kriptografi modern sendiri dapat beroperasi dalam mode bit atau byte. Walaupun begitu, algoritma kriptografi modern tetap menggunakan dua teknik dasar yang digunakan dalam kriptografi klasik yaitu teknik substitusi dan transposisi tetapi operasinya dibuat lebih kompleks.

Algoritma kriptografi modern dapat dibagi menjadi *symmetric cipher* dan *asymmetric cipher*. Pada *symmetric cipher* hanya digunakan satu buah kunci yang digunakan bersama oleh pengirim dan penerima, contohnya yaitu algoritma Diffie-Hellman. Sedangkan pada *asymmetric cipher* digunakan dua kunci, satu untuk pengirim dan satu untuk penerima, contohnya yaitu algoritma Elgamal.

B. Cryptographically Secure Pseudorandom Generator (CSPRNG)

Pembangkit bilangan acak yang cocok untuk kriptografi dinamakan CSPRNG. Adapun persyaratan dari CSPRNG yaitu:

1) Secara statistik lolos uji keacakan (*randomness test*)
2) Tahan terhadap serangan yang serius. Serangan ini bertujuan untuk memprediksi bilangan acak yang dihasilkan dari data bilangan-bilangan acak yang dihasilkan sebelumnya

Akan tetapi, meskipun memenuhi syarat tersebut CSPRNG tetap termasuk pembangkit bilangan acak yang tidak sempurna karena memiliki sifat deterministik dan periodik.

Adapun salah satu contoh dari algoritma CSPRNG yaitu algoritma Blum Blum Shub (BBS).

C. File Image

File image merupakan salah satu jenis format file yang bertujuan untuk mengirimkan *digital image*. Terdapat beberapa format dari *file image* antara lain yaitu JPG, JPEG, PNG, dan GIF. *File image* menyimpan digital image dalam bentuk 2D dengan komponen terkecilnya berupa pixel. *File image* tergolong ke dalam jenis file yang cenderung berdata besar, misalnya yaitu *file image* yang pixelnya memiliki lebar w dan panjang l berarti terdiri atas $w \times l$ pixel. Pada kasus umum, lebar dan panjang dari *file image* berada dalam orde ribuan sehingga diperlukan skema pemrosesan yang efisien untuk memproses data pixel dari *file image*.

D. Blum Blum Shub (BBS)

Blum Blum Shub (BBS) adalah salah satu algoritma CSPRNG yang paling sederhana tetapi tetap memiliki *randomness* yang baik. BBS dibuat pada tahun 1986 oleh Lenore Blum, Manuel Blum, dan Michael Shub dan berbasis *number theory*.

BBS memanfaatkan *number theory* berupa persamaan berikut:

$$y = x^2 \ mod \ n$$

Dari persamaan di atas cukup mudah untuk melakukan perhitungan y bila x diketahui karena hanya membutuhkan kompleksitas waktu $O(1)$. Tetapi untuk inversinya tergolong sulit, karena jika diketahui y maka x tidak dapat dihitung secara mudah terutama untuk n yang besar karena memerlukan kompleksitas waktu sekitar $O(\sqrt{n})$.

Oleh karena itu, dari persoalan di atas maka dirumuskan algoritma BBS sebagai berikut :

1) Pilih dua bilangan prima rahasia, p dan q yang masing-masing kongruen dengan 3 mod 4.
2) Kalikan keduanya menjadi $n = pq$. Bilangan n ini disebut dengan bilangan bilat Blum
3) Pilih bilangan bulat acak lain s yang digunakan sebagai $seed$ sedemikian sehingga
   a. $2 \leq s < n$
   b. s dan n relatif prima
4) Hitung $x_0 = s^2 \ mod \ n$
5) Barisan bilangan acak dihasilkan dengan melakukan iterasi beikut sepanjang yang diinginkan:
   a. Hitung $x_{i+1} = x_i^2 mod \ n$
   b. $z_i = LSB(x_i)$
6) Barisan bit acak yang dihasilkan berupa $z_1, z_2, \ldots, z_m$ dengan m merupakan panjang bit bilangan acak yang ingin dihasilkan

E. Algortima Elgamal

Algoritma Elgamal dibuat oleh Taher Elgamal pada tahun 1985. Algrotima Elgamal memanfaatkan persoalan logaritma diskrit, misalnya yaitu pada persamaan berikut:

$$y = g^x mod \ p$$

Jika p adalah bilangan prima maka menghitung y bila diketahui g dan x akan mudah karena hanya membutuhkan kompleksitas waktu minimal sekitar $O(\log x)$ untuk proses perpangkatan g terhadap x. Akan tetapi, bila diketahui y dan g maka perhitungan x tidak akan mudah karena harus dilakukan *bruteforce* setiap kemungkinan yang ada sehingga membutuhkan kompleksitas waktu sekitar $O(p)$.

Berikut adalah properti dari algrotima Elgamal:

1) Bilangan prima, p (tidak rahasia)
2) Bilangan acak, g ($g < p$, g adalah akar primitif dari p) (tidak rahasia)
3) Bilangan acak x ($2 \leq x \leq p - 2$) (rahasia)
4) $y = g^x mod \ p$ (tidak rahasia)
5) m (plainteks) (rahasia)
6) a dan b (cipherteks) (tidak rahasia)

Berikut adalah prosedur pembangkitan kunci pada algoritma Elgamal:

1) Pilih sembarang bilangan prima p
2) Pilih dua buah bilangan acak, g dan x, dengan syarat g akar primitif dari p, dan $2 \leq x \leq p - 2$
3) Hitung $y = g^x mod \ p$
4) Hasil dari algoritma ini berupa:
   a. Kunci publik : (y, g, p)
   b. Kunci privat: (x, p)

Berikut ini adalah prosedur enkkripsi dari algoritma Elgamal:

1) Pilih bilangan acak k, dengan $1 \leq k < p$
2) Setiap blok dienkripsi dengan menggunakan rumus berikut:
   $$a = g^k mod \ p$$

$$b = y^k m \bmod p$$

3) Pasangan (a,b) adalah cipherteks dari blok pesan m

Berikut ini adalah prosedur dekripsi dari algoritma Elgamal:

1) Hitung plainteks dengan menggunakan mersamaan

$$m = \frac{b}{a^x} \bmod p = b \times (a^x)^{-1} \bmod p$$

III. PEMBAHASAN

A. Kelemahan Algoritma BBS

Secara umum, algoritma BBS akan melakukan pembangkitan bilangan acak secara sekuensial dengan formula:

$$x_{i+1} = x_i^2 \bmod n$$

Dari persamaan tersebut maka akan dihasilkan bilangan acak berupa LSB dari $(x_0)^2, (x_0)^4, (x_0)^8, \dots, (x_0)^{2^m}$. Oleh karena itu bila dilakukan observasi maka sebenarnya kita dapat memprediksi nilai bilangan acak yang akan dibangkitkan selanjutnya bila diketahui sebuah nilai x_i. Misalkan:

$$z_1 = (x_0)^2 \bmod n$$
$$z_2 = (x_0)^4 \bmod n$$
$$z_3 = (x_0)^8 \bmod n$$

dan seterusnya hingga

$$z_m = (x_0)^{2^m} \bmod n$$

Maka z_m dapat diprediksi bila diketahui z_{m-i} melalui persamaan:

$$z_m = (z_{m-i})^{2^i}$$

Oleh karena itu, hal ini dapat menimbulkan masalah keamanan bila penyerang melakukan pencocokan bit LSB yang dihasilkan oleh algoritma BBS dengan kemungkinan z_i nya dengan memanfaatkan *dictionary attack* karena begitu ditemukan suatu z_{m-i} yang sesuai dengan hasil bit LSB yang dikeluarkan maka keseluruhan output bilangan acak dapat diketahui dengan mudah meskipun penyerang tetap tidak dapat melacak nilai x_0 yang merupakan nilai rahasia.

Oleh karena itu perlu dilakukan optimasi algoritma BBS untuk mengindari resiko adanya penyerangan dengan metode *dictionary attack* sehingga tanpa diketahui nilai x_0 penyerang hanya dapat melakukan *bruteforce* dengan kompleksitas waktu $O(\sqrt{n})$ untuk menebak nilai dari x_0 jika diketahui nilai output *list* LSB yang dihasilkan.

Selain itu, untuk meningkatkan keacakan dari hasil keluaran pada algoritma BBS perlu dilakukan beberapa pengubahan skema pembangkitan bilangan acak untuk meningkatkan *confussion* dan *diffusion* dari algoritma BBS sehingga walaupun *seed* yang digunakan hanya berbeda sedikit tetapi hasil keluaran bilangan acaknya bisa berbeda secara signifikan dengan hasil pembangkitan bilangan acak yang *seed* nya selisihnya berbeda tipis saja. Hal ini karena pada algoritma BBS, hasil LSB awal jika x_0 kecil dapat diprediksi yaitu akan menghasilkan bit 0 bila x_0 genap dan akan menghasilkan bit 1 bila x_0 ganjil.

B. Optimisasi Algoritma BBS

Bilangan acak ke-k pada algoritma BBS selain dapat dihitung dengan metode sekuensial yaitu dengan menghitung $z_1, z_2, z_3, \dots, z_{k-1}$ secara beurutan dapat pula dihitung secara langsung dengan menggunakan persamaan:

$$z_k(z_0) = (x_0)^{2^k \bmod \phi(n)} \bmod n$$

Dengan $\phi(n)$ menyatakan totient dari n. Dalam kasus ini karena $n = pq$ maka $\phi(n)$ dapat dihitung dengan mengunakan persamaan:

$$\phi(n) = lcm(p-1, q-1)$$

Oleh karena itu, dengan memanfaatkan persamaan berikut dapat digunakan algoritma sebagai berikut untuk meningkatkan kekuatan dari algoritma BBS. Berikut adalah usulan properti algoritma yang bertujuan untuk mengoptimasi algoritma BBS:

1) Pilih dua bilangan prima rahasia, p dan q yang masing-masing kongruen dengan 3 mod 4.
2) Kalikan keduanya menjadi $n = pq$.
3) Pilih dua bilangan acak lain yaitu s dan k sebagai *seed* sedemikian sehingga
   a. $2 \leq s < n$
   b. s relatif prima terhadap n
   c. $2 \leq k < n$
   d. k relatif prima terhadap n

Berikut adalah prosedur pembangkitan bilangan acak sebagai optimasi dari algoritma BBS:

1) Hitung nilai $\phi(n) = lcm(p-1, q-1)$
2) Gunakan persamaan berikut untuk membangkitkan bilangan acak

$$z_e(x) = (x)^{2^e \bmod \phi(n)} \bmod n$$

$$z_1 = z_k(s) = (s)^{2^k \bmod \phi(n)} \bmod n$$

3) Untuk pembangkitan bilangan acak selanjutnya gunakan $e = s$ dan ubah $x = z_1$
4) Lalu lakukan perhitungan z_2

$$z_2 = z_s(z_1) = (z_1)^{2^s \bmod \phi(n)} \bmod n$$

5) Untuk z_3 lakukan perhitungan dengan menggunakan $e = z_1$ dan $x = z_2$ sehingga akan dihitung nilai dari $z_{z_1}(z_2)$
6) Selanjutnya lakukan perhitungan hingga sebanyak m kali sehingga diperoleh $z_1, z_2, z_3, \dots, z_m$
7) Ambil LSB dari setiap z_i sehingga diperoleh bilangan acak yang dihasilkan berupa

$$LSB(z_1)LSB(z_2)LSB(z_3) \dots LSB(z_m)$$

Dengan menggunakan optimisasi algoritma BBS di atas maka potensi penyerangan dapat dikurangi. Misalnya:

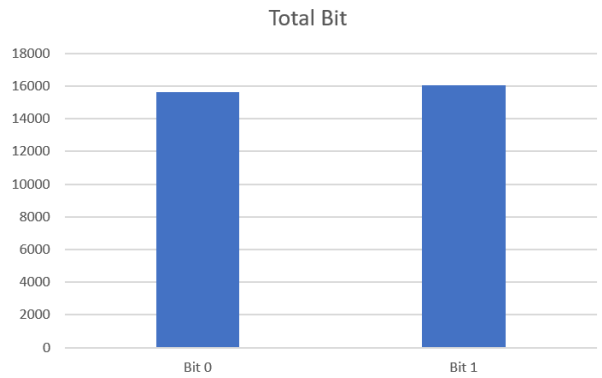
$$z_1 = z_k(s) = s^{2^k mod\ \phi(n)}\ mod\ n$$

$$z_2 = \left(s^{2^k mod\ \phi(n)}\ mod\ n\right)^{2^s mod\ \phi(n)}\ mod\ n$$

Dari contoh hasil dari z_1 dan z_2 di atas terlihat bahwa untuk perhitungan z_2 tetap diperlukan data yang terkait dengan nilai s karena z_2 merupakan z_1 yang dipangkatkan dengan $2^s\ mod\ \phi(n)$. Selain itu, untuk sembarang z_i untuk $i > 2$ hanya dapat diperoleh bila diketahui z_{i-1} dan z_{i-2} karena $z_i = z_{z_{i-2}}(z_{i-1})$ sehingga hal ini akan lebih menyulitkan penyerangan dengan *dictionary attack*.

Selain itu dengan menggunakan optimasi algoritma di atas maka *randomness* dari keluaran bilangan acak bisa ditingkatkan. Hal ini karena pada algoritma BBS sebelumnya hanya digunakan parameter s, p, q untuk inisialisasi. Sedangkan pada optimasinya digunakan parameter inisialisasi s, k, p, q sehingga nilainya dapat lebih variatif. Selain itu, karena dilakukan *swapping* variabel koefisien dan eksponen maka *confussion* dan *diffusion* dari algoritma bisa lebih baik dari metode sebelumnya yang hanya memanfaatkan perhitungan terurut secara sekuensial tanpa adanya penukaran variabel

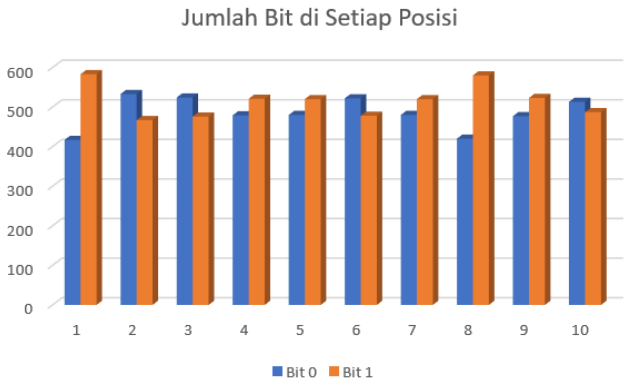
Berikut ini merupakan hasil uji *randomness* berdasarkan keluaran dari optimasi algoritma BBS yang diperoleh dari hasil pengujian 1000 kasus random untuk panjang keluaran sebesar 32 bit :



Gambar 1. Grafik Perbandingan Total Jumlah Bit
Diambil dari dokumentasi pribadi pada 20 Mei 2023

Dari hasil tersebut diperoleh bahwa jumlah bit 0 dan bit 1 seimbang sehingga hasil optimasi dari algoritma BBS tidak menghasilkan bit secara tidak merata.

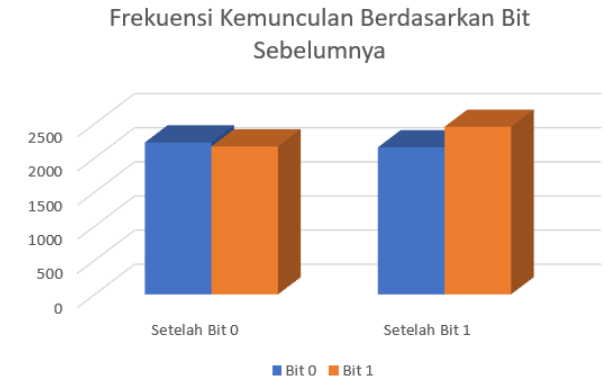
Berikut ini merupakan hasil uji *randomness* berdasarkan keluaran bit pada setiap posisinya untuk 1000 kasus uji dengan panjang keluaran sebesar 10 bit:



Gambar 2. Grafik Perbandingan Total Jumlah Bit di Setiap Posisi
Diambil dari dokumentasi pribadi pada 20 Mei 2023

Dari hasil tersebut diperoleh bahwa jumlah bit 0 dan bit 1 seimbang pada setiap posisinya sehingga hasil optimasi dari algoritma BBS tidak menghasilkan bit secara tidak merata relatif terhadap posisi.

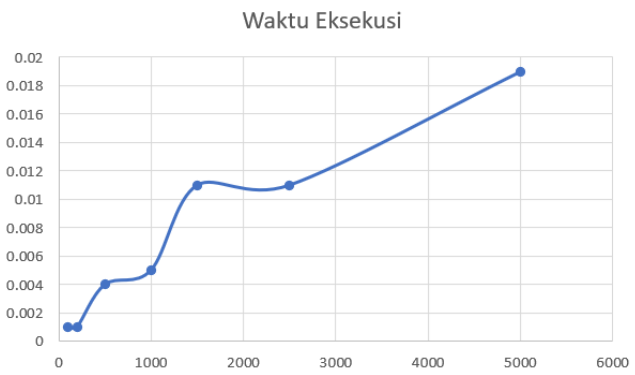
Berikut ini merupakan data hasil uji *randomness* berdasarkan kemungkinan suatu bit muncul setelah bit 0 atau bit 1 untuk 1000 kasus uji dengan panjang keluaran sebesar 10 bit:



Gambar 3. Grafik Perbandingan Total Jumlah Bit Berdasarkan Nilai Bit pada Posisi Sebelumnya
Diambil dari dokumentasi pribadi pada 20 Mei 2023

Dari hasil tersebut ditunjukkan bahwa tidak ada kecenderungan bahwa suatu bit akan lebih sering muncul setelah muncul sebuah bit X.

Selain itu, dari segi efisiensi optimasi algoritma BBS juga memerlukan waktu eksekusi yang singkat sehingga cocok jika ingin dilakukan pemrosesan yang cukup banyak frekuensinya, berikut adalah data waktu eksekusi relatif terhadap panjang bit yang ingin dihasilkan:



Gambar 4. Grafik Perbandingan Waktu Eksekusi terhadap Jumlah Bit yang ingin dihasilkan
Diambil dari dokumentasi pribadi pada 20 Mei 2023

C. *Enkripsi dan Dekripsi File Image dengan Algoritma Elgamal*

Dengan menggunakan algoritma Elgamal maka strategi yang dilakukan untuk enkripsi *file image* yaitu dengan memecah data *file image* ke satuan byte. Kemudian setiap *byte* akan dilakukan enkripsi satu per satu menghasilkan suatu cipherteks sehingga setiap plainteks akan berkorespondensi satu-satu dengan cipherteksnya. Berikut adalah langkah-langkah enkripsi dengan menggunakan algoritma Elgamal:

1) Menggunakan bilangan p yang memiliki ukuran 10-16 bit sehingga nantinya akan dihasilkan nilai a dan b yang berukuran 16 bit dari sebuah masukan
2) Untuk parameter k di-*generate* per pixel dengan menggunakan optimasi algoritma BBS dengan *seed* digunakan dari hasil *digest* fungsi SHA256 dengan menggunakan input *timestamp* saat ini
3) Dari proses enkripsi dihasil cipherteks berupa a dan b yang masing-masing berukuran 16 bit sehingga untuk setiap 8 bit plainteks akan dienkripsi ke 32 bit cipherteks
4) Byte hasil enkripsi akan ditulis terurut dimulai dari a kemudian b untuk setiap plainteks sehingga dihasilkan cipherteks berupa

$$a_1, b_1, a_2, b_2, \dots, a_m, b_m$$

Sedangkan untuk algoritma dekripsi yang dilakukan adalah sebagai berikut:

1) Mengambil data blok per 32 bit yang merepresentasikan cipherteks dari sebuah plainteks yang kemudian akan dipecah ke dalam variabel a dan b
2) Melakukan dekipsi cipherteks ke plainteks sehingga nantinya akan dihasilkan 8 bit plainteks dari 32 bit cipherteks
3) Melakukan penulisan data hasil dekripsi ke dalam *file image*

Beikut adalah hasil uji enkripsi sebuah *file image*:

1) Byte awal dari *file image*

[255, 216, 255, 224, 0, 16, 74, 70, 73, 70, 0, 1, 1, 1, 0, 240, 0, 240, 0, 0, 255, 237, 1, 24, 80, 104, 111, 116, 111, 115, 104, 111, 112, 32, 51, 46, 48, 0, 56, 66, 73, 77, 4, 4, 0, 0, 0, 0, 0, 252, 28, 2, 5, 0, 10, 49, 50, 55, 57, 56, 57, 57, 52, 56, 56, 28, 2, 40, 0, 18, 78, 111, 116, 32, 82, 101, 108, 101, 97, 115, 101, 100, 32, 40, 78, 82, 41, 32, 28, 2, 80, 0, 11, 77, 111, 108, 108, 121, 32, 65, ….]

2) Byte hasil enkripsi *file image*

[76, 9, 200, 0, 95, 100, 50, 24, 95, 100, 167, 67, 95, 100, 29, 115, 95, 100, 0, 0, 95, 100, 233, 53, 95, 100, 107, 105, 95, 100, 236, 123, 30, 60, 235, 114, 30, 60, 50, 110, 30, 60, 0, 0, 30, 60, 147, 1, 30, 60, 147, 1, 30, 60, 147, 1, 30, 60, 0, 0, 30, 60, 246, 121, 30, 60, 0, 0, 30, 60, 246, 121, 30, 60, 0, 0, 30, 60, 0, 0, 30, 60, 166, 17, 30, 60, 61, 117, 30, 60, 147, 1, 30, 60, 200, 37, 30, 60, 240, 125, ….]

3) Byte hasil dekripsi *file image*

[255, 216, 255, 224, 0, 16, 74, 70, 73, 70, 0, 1, 1, 1, 0, 240, 0, 240, 0, 0, 255, 237, 1, 24, 80, 104, 111, 116, 111, 115, 104, 111, 112, 32, 51, 46, 48, 0, 56, 66, 73, 77, 4, 4, 0, 0, 0, 0, 0, 252, 28, 2, 5, 0, 10, 49, 50, 55, 57, 56, 57, 57, 52, 56, 56, 28, 2, 40, 0, 18, 78, 111, 116, 32, 82, 101, 108, 101, 97, 115, 101, 100, 32, 40, 78, 82, 41, 32, 28, 2, 80, 0, 11, 77, 111, 108, 108, 121, 32, 65, ….]

Gambar 5. File Image Hasil Dekripsi yang Kembali ke File Semula
Diambil dari dokumentasi pribadi pada 20 Mei 2023

Dari kasus uji tersebut dapat disimpulkan bahwa pengamanan *file image* telah berjalan dengan baik karena *file* hasil dekripsi dapat kembali ke *file image* semula. Selain itu, terkait dengan performa proses enkripsi dan dekripsi membutuhkan kompleksitas waktu sekitar $O(nk)$ dengan $k \approx 300$ dan n menyatakan total pixel yang perlu diproses.

Akan tetapi, salah satu kelemahan dari proses pengamanan *file image* yang telah dilakukan yakni *file image* hasil enkripsi memiliki ukuran yang besarnya 4 kali *file image* semula sehingga masih bisa dioptimasi dengan memanfaatkan nilai k

yang sama untuk setiap pixel sehingga ukurannya hanya akan menjadi 2 kali file semula.

IV. Kesimpulan dan Saran

Optimasi dari algoritma BBS dapat dilakukan dengan tujuan untuk meningkatkan keamanan algoritma dari *dictionary attack* serta meningkatkan *randomness* serta *confussion* dan *diffusion* dari algoritma. Dalam hal ini, optimasi dilakukan dengan membuat algoritma BBS tidak melakukan pembangkitan bilangan acak dengan mengambil kuadrat dari bilangan sebelumnya saja kemudian diambil nilai LSB nya. Optimasi yang dilakukan yaitu dengan melakukan penambahan parameter pada algoritma dan melakukan *swapping* variabel secara rutin sehingga untuk setiap bit $LSB(z_i)$ dapat dibangkitkan dengan persamaan:

$$z_i = z_{i-1}(z_{i-2})$$

Dari hasil optimasi diperoleh bahwa *randomness* dari keluaran yang dihasilkan sudah cukup baik selain itu juga tidak ada kecenderungan bahwa suatu bit akan muncul setelah suatu bit X muncul sebelumnya.

Dengan menggunakan optimasi algoritma BBS tersebut dilakukan enkripsi *file image* dengan menggunakan algoritma Elgamal. Algoritma BBS sendiri digunakan untuk menghasilkan parameter k pada algoritma Elgamal.

Dari hasil uji enkripsi dan dekripsi pada algoritma Elgamal diperoleh bahwa proses enkripsi dan dekripsi sudah dapat berjalan dengan baik serta menghasilkan keluaran yang sesuai. Selain itu, implementasi yang dilakukan juga sudah dapat menghasilkan efisiensi yang baik yaitu dengan kompleksitas waktu sekitar $O(nk)$ dengan n menyatakan jumlah pixel dan $k \approx 300$.

Akan tetapi, implementasi dari algoritma Elgamal masih dapat dilakukan efisiensi karena untuk implementasi yang dibuat oleh penulis cipherteks berukuran 4 kali ukuran plainteks sehingga bila operasi dilakukan dalam mode bit maka ukuran yang dihasilkan bisa lebih kecil. Selain itu, untuk mengurangi ukuran dari cipherteks bisa juga dilakukan dengan menyetarakan parameter k untuk proses enkripsi di setiap pixelnya.

V. Tautan Kode Pada Github

Tautan kode dari implementasi program dapat diakses pada link berikut ini : primahafiz/elgamal-bbs (github.com)

VI. Ucapan Terima Kasih

Puji syukur kehadirat Tuhan yang Maha Esa atas segala rahmat, karunia, serta taufik dan hidayah-Nya sehingga penulis dapat menyelesaikan makalah yang berjudul "Optimasi Algoritma Blum Blum Shub sebagai Pembangkit Bilangan Acak dalam Algoritma Elgamal untuk Mengamankan File Image" sebagai pemenuhan tugas akhir semester II pada mata kuliah IF4020 Kriptografi.

Penulis juga ingin berterimakasih kepada Bapak Dr. Ir. Rinaldi, M.T. selaku dosen kelas K1 yang telah membagikan ilmunya kepada kami, para mahasiswa, selama satu semester ini. Penulis juga beterimakasih kepada seluruh pihak yang telah berkontribusi baik secara langsung maupun tidak langsung terhadap kelancaran penulisan makalah ini yang namanya tidak bisa saya sebutkan satu persatu. Terakhir, penulis ingin mengucapan permohonan maaf apabila terdapat kesalahan dalam penulisan makalah ini.

VII. Referensi

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/37-Pembangkit-bilangan-acak-2023.pdf Diakses pada 20 Mei 2023

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/20-Algoritma-Elgamal-2023.pdf Diakses pada 20 Mei 2023

VIII. Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Depok, 22 Mei 2023

Primanda Adyatma Hafiz (13520022)