Rancangan Algoritma Simetris Kriptografi Null-BBS:

Algoritma Kriptografi yang Memanfaatkan Algoritma Steganografi Null Cipher dan Algoritma Random Blum Blum Shub (BBS)

Wilhelmus Andrian Tanujaya
Teknik Informatika / Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
tanujaya.wilhelmus@gmail.com

***Abstrak*—Null Cipher merupakan sebuah algoritma steganografi yang digunakan pada perang dunia II. Algoritma ini menyembunyikan pesan yang ingin dikirimkan pada sebuah pesan lain dengan memberikan informasi di indeks mana saja karakter pesan dapat ditemukan. Penulis menggunakan ide null cipher ini dan mengganti beberapa hal yang dilakukan pada null cipher. Medium penyimpanan pesan adalah berupa citra, dan pesan yang disisipkan adalah bit per bit. Ada pun indeks dari masing-masing bit akan diacak dengan algoritma Blum Blum Shub, dan hasil pengacakan dijadikan sebagai cipherteks.**

Keywords—Null-cipher; Kriptografi; Steganografi; Blum Blum Shub; BBS; Penyisipan pesan; citra

I. Pendahuluan

Steganografi merupakan sebuah teknik menyembunyikan pesan dengan suatu cara agar tidak menimbulkan kecurigaan tentang keberadaan pesan tersebut. Dalam sejarah, steganografi telah dilakukan dengan berbagai cara. Mulai dari menyimpan pesan pada kepala tahanan yang kemudian ditutupi rambut, hingga cara menyembunyikan pesan dalam sebuah kain yang ditelan oleh pembawa pesan dari Cina[1].

Salah satu cara menyembunyikan pesan yang terkenal pada masa perang dunia II adalah null cipher. Pada null cipher, pesan disembunyikan pada teks lain seperti pada artikel di surat kabar dengan cara pesan dipecah-pecah per karakter dan diselipkan pada indeks pertama, kedua, atau indeks lainnya per kata pada artikel. Penerima pesan dapat kemudian mengambil setiap karakter pada indeks yang telah disetujui dan keumdian digabungkan menjadi pesan yang ingin disampaikan[1].

Pada awalnya, Penulis memiliki ide untuk menggunakan kemampuan pemrosesan bit pada zaman sekarang untuk membuat sebuah null cipher modern. Penulis memilih media penyisipan yaitu media citra. Ide utamanya adalah menggunakan indeks pada pixel citra berwarna(*true color*), setiap bit pesan akan disisipkan ke LSB warna merah, hijau atau biru. Akan tetapi, cara ini masih memiliki kelemahan yaitu penyisipan dapat dideteksi dengan adanya teknik *fragile digital watermarking*.

Melihat fenomena ini, maka muncullah ide baru lainnya yaitu penyisipan yang ingin dilakukan oleh Penulis tidak akan mengubah pixel citra sama sekali. Penulis hanya akan menyimpan bit pesan pada LSB pixel dengan bit yang sama. Artinya, pengirim pesan dan penerima pesan memerlukan sebuah cara untuk mengetahui pada bit mana sajakah pesan tersebut disisipkan. Dari pemikiran ini lah Penulis ingin menggunakan sifat bilangan random semu, yaitu dengan *seed* yang sama, angka-angka random dapat dibangkitkan kembali dalam urutan yang sesuai. Dengan demikian, kedua belah pihak cukup menyepakati nilai *seed* terlebih dahulu[3].

Akan tetapi, mengingat algoritma ini akan menghasilkan sebuah kode koordinat pesan yang bersifat kompleks(dijelaskan lebih lanjut pada bab 3), dan membutuhkan pihak pengirim memberitahukan kode koordinat kepada penerima, Penulis merasa rancangan algoritma ini lebih tepat dikategorikan sebagai algoritma

kriptografi dibandingkan steganografi. Di dalam algoritma ini, gambar yang digunakan dapat disetujui kedua belah pihak terlebih dahulu, dan koordinat pesan dapat diibaratkan seperti cipherteks yang dihasilkan.

II. Dasar Teori

A. Null Cipher

Null Cipher adalah sebuah algoritma penyembunyian pesan yang digunakan pada perang dunia II. Algoritma ini membagi pesan ke dalam karakter-karakter yang kemudian disisipkan ke dalam indeks pertama atau indeks lainnya dari sebuah artikel atau pesan lain yang tidak akan menimbulkan kecurigaan.

Cara membuat *null cipher* dapat dibagi ke dalam tiga langkah. Langkah pertama adalah memecah pesan ke dalam huruf-huruf. Misalkan pesan yang ingin dikirimkan adalah kata "code", maka pesan dipisah menjadi huruf c,o,d, dan e. Kemudian dipilih indeks yang diinginkan misalkan indeks pertama. Huruf c,o,d,e kamudian dirangkai menjadi satu atau lebih kalimat dengan huruf c,o,d,e sebagai huruf pertama dari tiap kata dalam kalimat sehingga pesan yang dikirimkan adalah "Can our daughter eat?". Pihak penerima pesan perlu mengetahui bahwa pesan yang ingin dikirimkan berada pada indeks pertama dari kalimat yang dikirimkan[1].

B. Citra Bitmap

Citra di dunia ada dua macam yaitu citra bitmap dan citra vector. Citra bitmap adalah citra yang terbentuk dari berbagai warna yang memenuhi masing-masing pixel(picture element) citra. Dikarenakan masing-masing pixel disimpan warnanya, maka semakin banyak pixel yang membentuk sebuah citra makin besar pula memori yang dibutuhkan untuk menyimpan citra.Ketika sebuah citra bitmap diperbesar hingga titik tertentu, maka kotak-kotak kecil berisi warna-warna dapat dilihat.

Citra bitmap sering dimodelkan sebagai sebuah matriks dengan masing-masing elemennya berupa warna. Karena masing-masing elemen dapat menyimpan bit dengan ukuran beragam, maka dengan membesarkan ukuran bit semakin banyak jenis informasi warna yang dapat disimpan sebuah citra[2].

Sebuah citra bitmap umunya memiliki ukuran pixel 1, 8, 24 dan 32 bit. Untuk warna monokrom, satu pixel cukup menyimpan 1 bit saja yang menyatakan warna hitam dan putih. 8 bit per pixel citra akan membuat citra memiliki palet warna *grayscale,* sedangkan 24 bit per pixel membuat citra memiliki warna *true-color.* Citra dengan kategori *deep color* memiliki ukuran 32 bit per pixel. 8 bit pertama pixel digunakan untuk menyimpan informasi warna merah, 8 bit berikutnya untuk warna hijau, kemudian biru dan sisanya untuk menyimpan nilai alpha atau nilai lainnya[4].

C. Least Significatnt Bit(LSB)

Citra bitmap merupakan citra yang terbentuk dari pixel-pixel yang disimpan dalam sebuah matriks. Seperti yang telah dijelaskan sebelumnya, setiap pixel umumnya menyimpan informasi warna yang disimpan dalam ukuran 1,8,24, atau 32 pixel.

Apabila informasi pada bit pixel ditampilkan dari kiri ke kanan, maka kira-kira tampilannya akan seperti ini "1100 0101"(ukuran 8 bit). Dari contoh ini, dapat dilihat bahwa ketika info dari bit pertama diubah sehingga menjadi "0100 0101", nilai dari informasi tersebut menjadi berubah drastis, dari 197 menjadi 69. Namun bila bit paling kanan yang diubah sehingga pesan menjadi "1100 0100", maka nilai hanya akan berubah dari 197 menjadi 196. Bit paling kanan ini lah yang disebut sebagai *least significant bit* pada sebuah citra, di mana ketika nilai bit ini diubah, pixel citra tidak berubah drastis[2].

D. Pembangkit Bilangan Acak Semu

Bilangan acak semu merupakan bilangan acak yang karena adanya keterbatasan pada rumus matematika, maka bilangan acak yang dihasilkan dapat dibangkitkan kembali dengan syarat memiliki bilangan pembangkit(*seed*) yang sama. Bilangan acak sendiri dikatakan bilangan acak karena pola himpunannya yang susah ditebak. Bilangan 1,2,4,8,12, dst bukan merupakan bilangan acak sebab anggota himpunan berikutnya akan mudah ditebak. Namun bilangan 3,11,277, 182,59, dst merupakan bilangan acak sebab polanya yang susah untuk ditebak.

Pembangkit deret bilangan acak disebut *pseudo-random number generator(PRNG).* Pembangkit bilangan acak yang aman untuk kriptografi disebut dengan *cryptographically secure pseudo-random number generator(CSPRNG).* Syarat agar sebuah PRNG dapat dikatakan aman ada 2 yaitu[3]:

1. Lolos uji statistik.
2. Tahan terhadap serangan yang serius(serangan untuk menebak angka random yang dihasilkan PRNG).

E. Algoritma Blum Blum Shub

Blum Blum Shub merupakan sebuah algoritma yang dikembangkan pada tahun 1986 oleh Lenore Blum, Manuel

Blum, dan Michael Shub. Algoritma ini merupakan algoritma paling sederhana dan mangkus secara teori, di mana algoritma ini digunakan dengan basis teori bilangan.

Berikut adalah langkah per langkah pembangkitan bilangan random dengan algoritma BBS[3].

1. Ambil dua buah bilangan prima rahasia, p dan q, yang masing-masing kongruen dengan 3 (mod 4).
2. p dan q kemudian dikalikan menjadi bilangan n. Bilangan n ini disebut bilangan bulat Blum.
3. Bilangan bulat acak lain diambil yaitu bilangan s. Bilangan s ini akan menjadi umpan sedemikian sehingga:
   (i) $2 <= s < n$
   (ii) s dan n relatif prima
4. Bit acak dihasilkan dengan melakukan iterasi berikut sepanjang yang diinginkan:
   (i) Hitung $xi = xi - 1^2 \bmod n$
   (ii) zi = bit LSB (Least Significant Bit) dari xi
   Barisan bit acak adalah z1, z2, z3, …

III. Rancangan Algoritma

Rancangan ini melibatkan dua pihak, sebut saja Alice dan Bob. Alice bertindak sebagai pengirim pesan dan Bob sebagai penerima pesan. Berikut adalah langkah-langkah yang dilakukan.

1. Alice dan Bob menyepakati nilai p dan q terlebih dahulu. Nilai p dan q yang dipilih kongruen dengan 3(mod 4). Misalkan Alice dan Bob menyetujui bilangan p=23 dan q=47.
2. Alice dan Bob kemudian menyepakati sebuah citra bitmap yang minimal berjenis *true-color(24 bits per pixel)*. Misalkan gambar yang dipilih berukuran 1024x1024 pixel.
3. Alice memilih sebuah angka seed, misalkan angka 640 yang juga diberitahukan ke Bob.
4. Pesan yang ingin dikirimkan Alice kemudian dipecah ke dalam ukuran bit.
5. Untuk bit pertama, Alice menggunakan *seed* 640 untuk membangkitkan bilangan acak dengan algoritma BBS. Hasilnya dimodulo dengan dimensi lebar gambar untuk menghasilkan koordinat x. Bilangan acak kedua kemudian dibangkitkan dengan *seed* baru dan dimodulo dengan dimensi tinggi gambar untuk menghasilkan koordinat y.
6. Kemudian pada pixel berkoordinat (x,y), dilihat apakah pixel merah memiliki LSB yang berbeda dengan pixel hijau atau biru. Apabila berbeda, maka pixel dapat digunakan, apabila tidak, perlu dibangkitkan nilai x dan y yang baru.
7. Bit pesan kemudian dicocokkan dengan LSB pixel merah. Apabila nilai keduanya sama, maka bit bernilai 1 akan disimpan sebagai pesan baru, kita sebut sebagai kode pesan. Apabila tidak sama, bit 0 akan disimpan sebagai cipherteks.
8. Demikian dilakukan per bit hingga semua pesan telah diproses dan menghasilkan sebuah cipherteks dengan panjang bit yang sama dengan plainteks.
9. Alice kemudian mengirimkan cipherteks ini ke Bob.

Untuk mendekripsi pesan, berikut adalah langkah-langkah yang dilakukan oleh Bob.

1. Dengan menggunakan *seed* yang sama dengan *seed* Alice, Bob menghitung masing-masing koordinat x dan y dengan cara yang sama dengan Alice.
2. Bob mencocokkan bit pada cipherteks dengan bit pada pixel. Apabila bit cipherteks bernilai 1, maka LSB pada pixel merah akan diambil sebagai nilai bit plainteks. Apabila bernilai 0, maka nilai LSB dari pixel biru atau hijau yang memiliki nilai tak sama dengan LSB pixel merah lah yang akan dijadikan bit plainteks.
3. Bob menggabungkan semua bit sehingga terbentuklah pesan yang ingin dikirimkan Alice.

IV. Eksperimen dan Pembahasan Hasil

Penulis menggunakan gambar Lenna berukuran 1024x1024 dan juga sebuah kalimat yang dibuat oleh Penulis. Berikut adalah state awal percobaan.

Citra bitmap yang digunakan:

Gambar 1. Lenna
(Sumber: https://paulschlessinger.files.wordpress.com/2014/06/image2.jpg)

Pesan yang dikirimkan:
Tidak ada yang instan dalam hidup. Segala sesuatu yang ingin kita dapat harus kita perjuangkan terlebih dahulu.

Dengan pesan di atas, Penulis menyisipkan pesan dan mendapatkan cipherteks berikut.

Nilai p : 23

Nilai q : 47

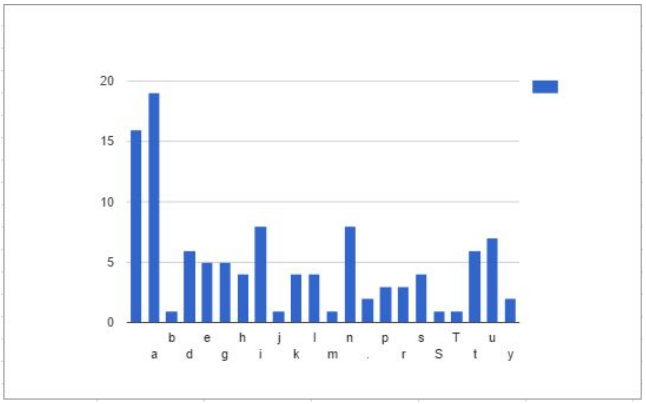
Nilai s : 640

Cipherteks yang dihasilkan:
jÃs7,L{´+Òèñ¯ë¦K"¥g£þÌÅË¿þÒØ×ÙÛ;+RÀtVÍ~zWÄp?,
Fv¡jçæî¥½^1¾4Õ©ëÀÅÓº±6ÖÑGÚ2=ZÜ\0NÄu/

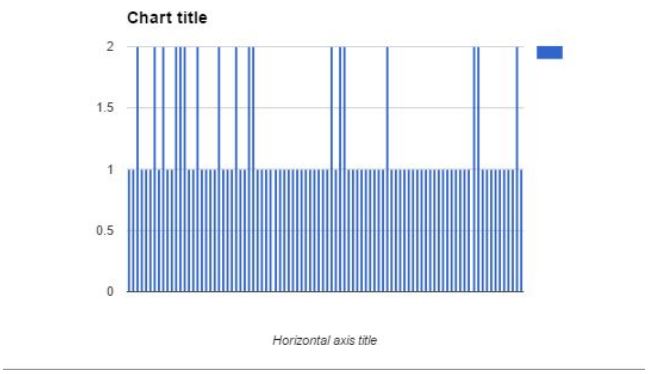
Dalam hexadecimal:

6AC3837337C2942C4C7BC2B42BC3921EC3A8C3B1C2AFC283C3ABC2A64B22C2A567C291C2A3C286C3BEC38CC385C38B1305C280C2BFC3BEC39202C398C397C29B0B09C399C39B3B2BC28852C380097456C38DC2907E7A57C384703FC2912C4676C2A16AC28B1BC3A7C3A6C3AEC29EC2A5C2BD5E31C2BE34C395C2A9C283C3ABC380C385C3931F13C29FC2BAC2B1C29C36C396C391C294471CC29CC39A323DC29F5AC39C5C304EC384C28B752F10

Distribusi karakter pada plainteks:



Distribusi karakter pada cipherteks:



Pada hasil di atas dapat diamati bahwa panjang plainteks adalah sama dengan panjang cipherteks. Namun distribusi karakter pada cipherteks menjadi lebih merata. Ini disebabkan panjang karakter yang masih sebatas 111 karakter saja dan kebetulan warna pada citra yang digunakan cukup bervariasi. Apabila karakter plainteks melebihi jumlah karakter ASCII yang ada dan citra memiliki variasi warna yang minim, Penulis tidak dapat menjamin distribusi karakter cipherteks yang dihasilkan akan tetap seragam.

Penulis kemudian melakukan beberapa eksperimen untuk menguji kekuatan algoritma ini. Eksperimen pertama adalah dengan melakukan penebakan nilai s. Pada eksperimen ini, Penulis menghitung total kemiripan plainteks dan hasil dekripsi dengan menghitung nilai mean squared error dari tiap karakter sebagai nilai pembanding. Berikut adalah hasilnya:

Nomor	**Nilai s**	**error**
001	**639**	7935.324
Hasil dekripsi: XÆ$		
002	**641**	6937.018
Hasil dekripsi: GÜ¦<?SÊí;ÁWË×²dÄd¡½2Ùç 9B¸0Eºûpà ö:m§o$¬ÙSJüÚÆvzÛ¥4:SÀà.ÎØ¥%ÙP´®ÑvÓ â!N®/@ÖôÏ~Ì°/(¦f3¤ÅäÓÝ#=		
003	**642**	7083.522
Hasil dekripsi: ÜV· H-¦t Õªæÿ"ÈO´HY\Ï7Xéìp·ÀBuö¸F£å¾xàXTé¦e&-Î^ú#÷ÒÊe»T é¡>_ÙÁ@aTÚêÕJü*9mÿ1)ÏËì|Ýgpb		

Dari hasil di atas, dapat dilihat bahwa perbedaan satu angka pada s saja sudah dapat menyebabkan seluruh plainteks gagal ditebak. Penulis kemudian melakukan eksperimen lain, kali ini dengan mengganti nilai p dan q. Berikut adalah hasilnya.

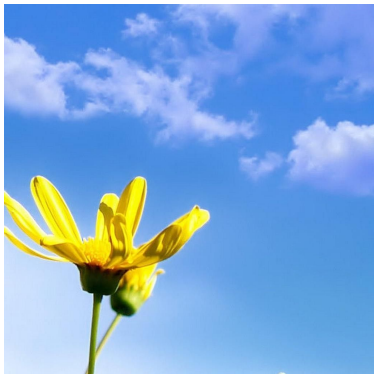
Nomor	**Nilai p**	**Nilai q**	**error**
004	**23**	**89**	8268.6036
Hasil dekripsi: N×|Å Si{Æ±Iê®F>¾<¢äAÁ^!ª{úëB¡Ýh§³¹ðç			
005	**17**	**47**	7130.468
Hasil dekripsi: ¼{µ:]ÁèÈÈÉP7-+ ~F}F@Ëb´FxYc¥)jÄíyêb8!ü0õ§Û¦Ì`|¶			

?]ËýÌ_ Û0Ô05m].EÞn´^tO| f$ðãåÊwÀ9(ê'éòAÃ¯×5Ç

Dengan nilai p dan q yang berbeda, tentu hasil dekripsi juga akan salah. Dapat dilihat pada kasus di atas, error yang dihasilkan tidak berbeda jauh dengan error yang dihasilkan ketika nilai *seed* diganti.

Penulis kemudian mencoba mengganti citra yang digunakan dengan citra lain. Berikut adalah hasilnya.

Citra yang digunakan:



Gambar 2. Flower (Sumber: http://www.hot-wallpaper.com/free_download/1024_1024_wallpapers/09201520/B/B_1024-x-102_IKrsL7adK6.jpg)

Hasil Dekripsi:
ÉøHÐvÀó1Å/L$yÈ4B}VD¿Nbþ³rEòj«/
øÐ¹ðÓáå#¬Ös£ÊøBÅ7äÓ,4Y?*>GhZ\³X}ûü<q
Nilai error:
8539.405

Citra yang digunakan:

Gambar 3. Tree (Sumber: http://www.wallpaper-mobile.com/free_download/1024_10

24_wallpapers/11201322/B/B_artistic-t_wwZsN3hp.jpg)

Hasil Dekripsi:
íDº|èxÂ¤ãk!E³À,ïTú^
Nilai error:
7498.738

Hasil dari penggantian gambar juga memiliki karakter yang sama dengan pergantian nilai p,q maupun nilai s, di mana nilai MSE berada di sekitar 7000 hingga 8000. Selain itu kata yang dienkripsi juga gagal untuk didekripsi kembali.

V. Analisis Keamanan

Algoritma ini bergantung pada kekuatan algoritma acak BBS. Selama BBS masih belum dapat diprediksi, maka algoritma ini cukup aman untuk digunakan. Sudah dibuktikan dengan perubahan pada bilangan s, p, q maupun perubahan pada citra, hasil dekripsi pesan sebagian besar akan berubah sehingga pesan susah untuk dikenali.

Panjang dari cipherteks yang dihasilkan sama panjang dengan plainteks yang dimasukkan. Namun, tidak mungkin untuk melakukan penyerangan dengan menghitung frekuensi, sebab frekuensi kemunculan karakter akan sangat bergantung pada nilai acak yang dihasilkan oleh BBS. Dengan serangan known plainteks pun, kriptanalis memerlukan cara untuk mengetahui nilai p,q dan *seed* untuk memprediksi pixel yang digunakan untuk mengenkripsi setiap bit.

Kelemahan algoritma Null-BBS ini adalah penggunaan LSB sebagai indeks yang digunakan untuk mengacak plainteks. Misalkan citra yang digunakan memiliki *watermark* yang disimpan pada LSB citra. Dengan asumsi *watermark* tersebut tidak hanya digunakan pada citra tersebut, melainkan pada citra lain juga, maka kerahasiaan citra menjadi irelevan dibandingkan dengan citra dengan *watermark* yang sama.

Algoritma Null-BBS ini juga membutuhkan kedua belah pihak memiliki nilai p,q,*seed* dan citra yang sama terlebih dahulu. Artinya algoritma ini bersifat simetris sehingga dibutuhkan sebuah cara yang *reliable* untuk dapat bertukar kunci terlebih dahulu.

VI. Kesimpulan dan Saran

Algoritma ini masih membutuhkan percobaan yang lebih. Misalkan pada perhitungan frekuensi karakter yang dilakukan, distribusi karakter yang muncul masih memiliki frekuensi yang tersebar merata sebab plainteks hanya berukuran 111 karakter saja. Sebaiknya algoritma ini dicoba juga dengan plainteks berukuran jauh lebih besar agar lebih terlihat celah keamanannya.

Penggunaan LSB pada penentuan bit cipherteks juga hanya membuat bit lain pada pixel menjadi tidak berguna dan juga seperti yang telah dijelaskan pada bab sebelumnya, citra dengan LSB yang sama (karena memiliki *watermark* pada bit LSB) akan menjadi kelemahan algoritma ini. Sebaiknya digunakan sebuah cara untuk memanfaatkan bit selain bit LSB, misalkan dengan menggunakan algoritma random lagi untuk mengacak bit yang akan digunakan pada pixel dengan koordinat(x,y).

Nilai p dan q sebaiknya dibuat besar untuk menyulitkan penebakan nilai *seed*, sebab nilai *seed* akan bergantung pada hasil perkalian nilai p dan q. Apabila pxq besar, maka rentang kemungkinan nilai *seed* yang diambil pun semakin besar.

Daftar Referensi

[1] Munir, Rinaldi. Slide Kuliah: Steganografi. http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/kriptografi.htm. Diakses pada tanggal 17 Desember 2016.

[2] Munir, Rinaldi. Slide Kuliah: Digital Watermarking. http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/kriptografi.htm. Diakses pada tanggal 17 Desember 2016.

[3] Munir, Rinaldi. Slide Kuliah: Pembangkit Bilangan Acak Semu. http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/kriptografi.htm. Diakses pada tanggal 17 Desember 2016.

[4] Bourke, Paul. A Beginners Guide to Bitmaps. http://paulbourke.net/dataformats/bitmaps/. Diakses pada tanggal 18 Desember 2016.