Pengembangan Random Number Generator dengan Video dan Suara

Yohanes Andika / 13507067
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10
Bandung 40132, Indonesia
if17067@students.if.itb.ac.id

Pembimbing Tugas Akhir
Dr. Ir. Rinaldi Munir, M.T. / NIP 132084796
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10
Bandung 40132, Indonesia
rinaldi@informatika.org

Bilangan acak (random number) merupakan hal yang sangat penting dalam kriptografi. Kebutuhan akan bilangan acak yang kuat namun cepat dalam proses menghasilkannya, membuat kemunculan bermacam-macam algoritma pembangkitan bilangan acak. Pseudo-random number generator adalah salah satu sumber utama untuk menghasilkan bilangan acak. Untuk pseudo-random number generator, hasil keluaran deret bilangan acak akan sangat tergantung kepada seed awal. Oleh karena itu dibutuhkan suatu faktor luar yang selalu berubah-ubah untuk dijadikan seed, dan faktor luar tersebut dapat dimanfaatkan kembali dalam proses pembangkitan sebagai penambah tingkat keacakan deret bilangan acak yang dihasilkan.

Dalam makalah ini, akan digunakan sumber luar video dan suara sebagai seed dan sumber keacakan. Video dan suara akan diambil ketika perangkat lunak melakukan pembangkitan bilangan acak. Video dan suara digunakan sebagai sumber keacakan, karena diasumsikan bahwa file video dan suara pasti memiliki keacakan yang tidak dapat diprediksi, seperti terdapatnya noise pada hasil rekaman video dan rekaman suara. Untuk algoritma pembangkitan bilangan acak, digunakan algoritma modifikasi LSFR (Linear Shift Feedback Register).

Dalam makalah ini juga dibuat perangkat lunak sebagai implementasi algoritma. Perangkat lunak diimplementasikan dalam bahasa C#, menggunakan development tools Microsoft Visual Studio 2008 untuk sistem operasi Windows. Untuk perekaman video dan suara, pada implementasi ini bersifat machine specific, yakni hanya bekerja untuk perangkat tempat perangkat lunak ini dibuat, untuk dapat digunakan di perangkat lain, dibutuhkan penyesuaian dalam pengambilan video dan suara.

Dengan menggunakan tes uji statistik NIST, didapatkan bahwa algoritma pembangkitan bilangan acak telah mencukupi kriteria keacakan bilangan acak dengan melewati pengujian untuk berbagai kriteria keacakan.

Kata kunci—random number, random number generator, algoritma, seed video, seed sound

I. PENDAHULUAN

Bilangan acak (*random*) merupakan hal yang sangat penting dalam kriptografi. Tanpa bilangan *random*, sebagian besar algoritma kriptografi akan dengan mudah dipecahkan. Dengan menggunakan bilangan acak dalam suatu algoritma kriptografi, akan mempersulit penyerang untuk menebak kunci maupun hasil enkripsi dari suatu kriptografi.

Algoritma kriptografi dianggap paling kuat, yaitu *One time pad*, merupakan algoritma kriptografi yang sangat tergantung dengan bilangan acak. Kunci yang digunakan diharuskan merupakan bilangan acak murni dan sepanjang pesan yang dikirim juga. Hal ini akan membuat kriptanalisis tidak mungkin dilakukan. Akan tetapi pada praktiknya, *One time pad* tidak dapat diterapkan secara praktis karena tidak mungkin untuk membangkitkan bilangan acak secara murni.

Selain *One time pad*, banyak sekali algoritma lain yang menggunakan bilangan *random* sebagai unsur penting dalam algoritmanya. Dengan memasukkan bilangan *random*, dianggap dapat menghilangkan kemungkinan penyerang menebak hasil dengan mengetahui algoritmanya.

Telah banyak algoritma *generator* bilangan acak yang diusulkan dan digunakan hingga saat ini. Algoritma-algoritma tersebut menggunakan berbagai pendekatan berbeda untuk menghasilkan bilangan *random* seacak mungkin. Hingga saat ini kita tidak dapat membuat suatu *generator* bilangan acak yang dapat menghasilkan bilangan acak secara murni.

Untuk mendapatkan bilangan acak banyak ahli yang mengusulkan berbagai algoritma dari yang sederhana hingga yang paling rumit. Dari penggunaan variabel waktu sebagai faktor pengacak hingga penggunaan teori biologi cellular. Algoritma-algoritma ini masing – masing memiliki kelebihan maupun kekurangan.

Karena tidak ada algoritma bilangan acak yang benar-benar sempurna, hingga saat ini kita masih terus mengembangkan cara-cara untuk mendapatkan bilangan *random* yang sukar ditebak. Kita berupaya untuk mendapatkan *generator* bilangan yang menyerupai *generator* bilangan acak. *Generator* bilangan ini membangkitkan bilangan acak semu (*pseudo random number*) yang

memiliki tingkat kekuatan berbeda-beda menurut algoritma yang digunakan.

Pseudo-Random Number Generator menggunakan suatu state awal kemudian dengan suatu algoritma khusus akan menghasilkan bilangan acak semu. State awal yang digunakan diambil dari berbagai sumber yang dianggap cukup acak. Dengan demikian, *Pseudo Random Number Generator* ini akan menghasilkan suatu rangkaian bilangan yang menyerupai bilangan acak.

Pseudo Random Number Generator banyak sekali digunakan pada saat ini. Kemudahan implementasi dan kecepatannya merupakan faktor utama mengapa *Pseudo Random Number Generator* digunakan. Namun, sebagai algoritma *generator* bilangan acak, *Pseudo Random Number Generator* masih banyak memiliki kelemahan. Kelemahan utama adalah pada pemilihan state awal yang digunakan untuk proses *generate* bilangan acak.

II. TEORI YANG DIGUNAKAN

A. Random Number dan Random Number Generator

Definisi random number [MENEZES, 1996]:

1. Angka yang dipilih dari set angka tertentu dengan cara sedemikian sehingga setiap angka yang muncul memiliki probabilitas kemunculan yang sama
2. Urutan angka yang dinyatakan telah lulus tes statistik atau bebas dari kondisi dimana tidak dapat diduga atau ditebak kalkulasinya.

Random Number Generator adalah alat atau algoritma yang menghasilkan urutan angka yang secara statistik independen dan tidak dapat ditebak. Kelas *random number generator* :

a) *Deterministic* RNG

Dibuat menggunakan algoritma yang menghasilkan urutan bit berdasarkan nilai awal yang disebut *seed*.

b) *Non-deterministic* RNG

Menghasilkan keluaran yang tergantung kepada sumber luar yang tidak dipengaruhi oleh kontrol manusia.

Pseudo-Random Number Generator adalah *random number generator* yang menghasilkan nilai berdasarkan *seed* dan state saat ini. Dengan *seed* yang sama, sebuah PRNG akan menghasilkan nilai output yang sama. Kekuatan *pseudo-Random Number Generator* dalam menghasilkan deret bilangan acak sangat tergantung kepada kekuatan algoritma, serta keamanan *seed*.

Sebuah *random* bit *generator* membutuhkan sumber keacakan yang secara natural terjadi. Ada 2 macam cara memperoleh keacakan, yakni *hardware*-based dan *software*-based. *Hardware*-based adalah memperoleh keacakan dari fenomena fisik, diantaranya : waktu peluruhan radioaktif, suhu diode atau resistor, frekuensi osilator, suara dari mikrofon atau input video dari kamera. Sementara *software*-based adalah memperoleh nilai dari proses-proses *software* yang sedang berjalan, diantaranya : system clock, waktu keystroke, gerakan *mouse*, input user, nilai-nilai pada operating system seperti load dan network statistics.

B. Review beberapa *Random Number Generator*

a. *Random number generator oleh Lewis et al*

Generator ini akan menghasilkan urutan angka integer positif, IX, dengan rekursi :

$$IX(i + 1) = A + IX(i) \bmod P$$

dimana :

$P = 2^{31} - 1 = 2147483647$

$A = 7^5 = 16807$

Sehingga semua integer IX yang dihasilkan akan memenuhi $0 < IX < 2^{31} - 1$.

Generator ini dipanggil sebagai sebuah fungsi, FX = RAND (IX), yang pada saat penggunaan pertama kali nilai IX harus diinisialisasi.

Linus Schrage mengembangkan dan mengimplementasikan *generator* tersebut agar bersifat *machine independent*.

Keunggulan *generator* ini antara lain bersifat *machine independent* serta g*enerator* bekerja secara satu siklus penuh, yaitu dari 1 hingga $2^{31} - 1$ dihasilkan tepat sekali selama satu siklus. Siklus ini empat kali lebih besar daripada *random number* Fortran lainnya.

b. *Intel RNG*

Intel RNG menggunakan *noise source* sebagai dasar keacakan dari *random number* yang dihasilkan. Osilator kemudian mencatat *noise* yang sudah diamplifikasi. Terdapat dua osilator yang berbeda frekuensinya, yang satu cepat, yang satu lambat yang digunakan untuk memodulasikan frekuensi osilator yang lambat.

Hasil osilator kemudian akan digabungkan (lambat dan cepat) menjadi output *random number* yang dilanjutkan ke pembuatan *random number* secara *software*. Dengan menggunakan algoritma SHA-1, dibuat sehingga tidak mungkin mendapatkan *seed* dari *random number* yang ada.

c. *Fortuna : Cryptographically Secure Pseudo-Random Number Generation In Software And Hardware*

Dalam membangkitkan bilangan acak, fortuna memiliki tiga bagian utama, yaitu :

i. Entropy Accumulator

Akumulator akan bekerja dengan cara mengumpulkan data acak dari berbagai sumber entropi eksternal, dan menggunakannya untuk memberi *seed* dan re*seed* pada *generator*. Algoritma ini memungkinkan penggunaan hingga 256 sumber entropi.

ii. *Generator*

Menerima *seed random* dari entropy accumulator, kemudian menghasilkan sekuens panjang dari *pseudo-random* data. Dalam implementasinya, AES-256 digunakan sebagai block cipher.

iii. *Seed* File Manager

Pada saat Fortuna dijalankan pertama kali, sebuah '*seed* file' memberikan *seed* kepada *generator*. *Seed* awal ini memungkinkan *generator* menghasilkan data *random*, sebelum cukup entropi berhasil dikumpulkan oleh akumulator. *Seed* kemudian akan diganti oleh data yang terkumpul oleh akumulator.

Beberapa sumber entropi utama untuk akumulator adalah pergerakan *mouse, t*iming penekanan tombol *keyboard* dan *noise* pada *soundcard.*

Keunggulan Fortuna antara lain : mudah digunakan oleh pengguna PC pada umumnya, karena sumber entropi yang digunakan dapat diakses oleh kebanyakan pengguna PC, serta penggunaan sumber entropi yang cukup banyak membuat algoritma ini cukup kuat terhadap serangan.

d. Linear Congruential *Generator*

Diusulkan oleh Lehmer pada tahun 1951. Menggunakan fungsi linear pada modulus aritmatika. *Generator* ini paling banyak digunakan, umumnya di aplikasi simulasi.

Diproduksi secara iteratif dengan persamaan berikut :

$X_i = (BX_{i-1} + c) \bmod m, i \geq 1$

dimana X_i, B, c, dan m adalah integer positif dan disebut parameter LCG (X_i adalah semua integer antara 0 sampai m-1). Kualitas *generator* sangat tergantung kepada pemilihan nilai kenaikan c, faktor pengali B, *seed* awal X_0, dan modulus m.

e. Linear Feedback Shift Register (LSFR) *Generator*

LSFR digunakan untuk menghasilkan deret biner bit. Menggunakan *seed* pendek untuk menghasilkan deret bit yang memiliki cycle yang sangat panjang.

Bentuk umum produksi LSFR :

$X_i = (\square_1 X_i + ... + \square_k X_{i-k}) \bmod 2$

f. Blum-Micali (BM) *Generator*

Berdasarkan tingkat kesulitan untuk menghitung logaritma diskrit, yang berdasarkan kepercayaan bahwa modular eksponensiasi modulo adalah prima dan fungsi satu arah.

Bentuk umum BM :

$X_{i+1} = aX_i \bmod m, i \geq 0$

Output dari *generator* adalah 1 jika $X_i < m/2$, selain itu menghasilkan 0.

g. Blum Blum Shub (BBS) *Generator*

Blum, Blum dan Shub menciptakan BBS pada tahun 1986. *Generator* yang bersifat *secure cryptographic*. Berdasarkan kekuatan residu kuadratik dan faktorisasi antar field modulus.

Bentuk umum BBS :

$X_{i+1} = (X_i)2 \bmod m, i \geq 1$

$Z_i = X_i \bmod 2$

Z_i adalah output dari *generator*.

Modulus m pada BBS adalah blum interger. Hubungan antara modulus m dan dua integer prima p dan q adalah sebagai berikut :

$m = p \times q$ dan $p \equiv q \equiv 3 \bmod m$

h. RSA *Generator*

Diusulkan oleh Rivest, Shamir dan Adleman pada tahun 1978. Berdasarkan kekuatan faktorisasi integer dari angka modulus. RSA menghasilkan deret angka dimana setiap elemen pada deret adalah enkripsi RSA dari elemen sebelumnya.

Bentuk umum RSA :

$X_{i+1} = X_i e \bmod m, i \geq 1$

$Z_i = X_i \bmod 2$

Z_i adalah output dari *generator*.

Modulus m adalah produk dari 2 bilangan prima besar p dan q (prima 512-bit), e dipilih sehingga :

$\gcd(e, \varphi(m)) = 1$

dimana m dan e adalah kunci publik sementara p dan q adalah kunci rahasia. Keamanan RSA berdasarkan pada penemuan faktor m, jika m cukup besar, akan sangat sulit untuk memfaktorkannya.

C. **Format Audio Video**

a) Format Audio WAV

Secara garis besar, *format* WAV terdiri dari 3 bagian, "*RIFF" chunk descriptor, "fmt" sub-chunk* dan *"data" sub-chunk*. *"RIFF" chunk descriptor* akan menunjukkan jenis *file*, yakni disini *WAVE*, yang mengharuskan adanya 2 *sub-chunk*, yaitu "*fmt*" dan "*data*". "*fmt*" berisikan informasi format suara, dan "*data*" berisikan ukuran dari suara itu sendiri serta *data* suara *raw*.

Isi file WAV terdapat pada byte 44 dan seterusnya. Untuk mendapatkan keacakan yang selalu terjadi, hanya akan diambil byte 44 dan seterusnya. Umumnya *header* akan berisi byte yang sama karena direkam dengan device yang sama.

b) Format Video AVI

AVI adalah format turunan dari RIFF, yang mana format ini membagi data file menjadi blok-blok, atau biasa disebut dengan *chunk*. Setiap *chunk* diidentifikasi oleh *tag* FourCC. Biasanya terbagi menjadi 2 *chunk* utama dan sebuah *chunk* opsional.

Sub-chunk yang pertama adalah hdrl, yang berisikan metadata mengenai video, seperti ukuran, frame rate. Sub-chunk yang kedua adalah "movi", yang berisikan data aktual dari file.

D. Pendekatan Evaluasi

NIST Statistical Test Suite adalah tes statistik yang dikembangkan dari kolaborasi antara Computer Security Division dan Statistical Engineering Division di NIST. Framework NIST berdasarkan pada testing hipotesis. Dalam kasus ini, tes ini melibatkan penentuan apakah deret spesifik dari nol dan satu adalah random

[SOTO,2008]. Tabel dibawah ini menjelaskan proses masing-masing untuk melakukan evaluasi :

Langkah – langkah prosedur evaluasi :

1. *State your null hypothesis*.	Asumsikan sekuens biner adalah *random*.
2. *Compute a sequence test statistic*.	Pengetesan dilakukan pada level bit.
3. *Compute the P-value*.	P-value ϵ[0,1]
4. *Compare the P-value to α*.	Perbaikiα, dimana α (0.001, 0.01].Sukses jika P-value ≥ α; selain itu gagal melewati tes.

Tabel Tes Statistik NIST

III. ANALISIS

A. RNG yang Sudah Ada dan Permasalahannya

Beberapa RNG yang sudah biasa digunakan, menggunakan perkalian faktor primer untuk kekuatan keamannnya (RSA dan BBS). Faktor primer cukup sulit untuk ditebak jika nilai perkaliannya sangat besar. Namun apabila angka tersebut bocor, akan dapat dihasilkan deret *random* yang sama.

Pseudo-random number generator menggunakan algoritma untuk menghasilkan deret *random* (LCG dan LSFR). Deret tersebut akan sangat tergantung kepada *seed*nya. Jika *seed* didapatkan, maka dapat didapatkan deret *random* yang bersesuaian. *Pseudo-random number* juga memiliki periode. Periode tersebut menyebabkan tingkat keacakan *random number* menjadi berkurang. Oleh karena itu dibutuhkan tambahan keacakan lain untuk menjadikan deret *random* memiliki periode lebih besar, atau bahkan mungkin tidak memiliki periode (deret yang dihasilkan awalnya memiliki periode, namun karena proses tambahan menjadi seperti tidak memiliki periode).

Untuk menambah kekuatan dari *Pseudo-Random number generator*, dibutuhkan faktor-faktor luar yang bersifat selalu berubah-ubah setiap waktu, namun harus dapat diobservasi. Intel RNG menggunakan *source* berupa suhu resistor, yang mana agak sulit untuk diobservasi secara langsung oleh manusia.

Fortuna menggunakan berbagai *source* luar untuk sumber keacakannya. Yang utama antara lain penekanan *keyboard*, penekanan *mouse*, serta *noise* pada *soundcard*. *Source-source* tersebut mudah diobservasi, namun dapat dengan mudah dimanipulasi oleh manusia atau oleh mesin.

Source luar berupa suara dan video memiliki tingkat keacakan yang cukup tinggi. Sebuah rekaman suara pada waktu yang sama, namun menggunakan *device* yang berbeda pasti memiliki perbedaan. Perbedaan tersebut disebabkan antara lain karena perbedaan kemampuan *device* untuk merekam, serta adanya *noise* yang terekam. Hal ini menyebabkan tingkat keacakan rekaman suara adalah tinggi, yang mana dapat digunakan sebagai sumber keacakan pada pembangkit bilangan acak.

Sumber lain yakni video, kurang lebih memiliki kriteria yang sama dengan rekaman suara. Hampir tidak mungkin terdapat rekaman video yang memiliki nilai (dalam hal ini byte) yang sama persis antara video satu dengan lainnya. Rekaman video pasti memiliki *noise* yang membuat suatu video memiliki perbedaan dengan video lainnya. Hal ini pula yang mendasari bahwa video adalah *source* keacakan yang dapat dimanfaatkan pada pembangkitan bilangan acak.

B. Usulan Solusi

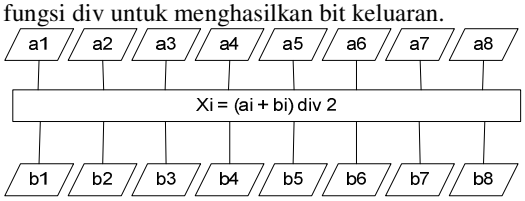
Berdasarkan analisis masalah diatas dapat disimpulkan bahwa dibutuhkan aplikasi dan algoritma yang bersifat : *secure cryptographic,* menghasilkan deret bit dengan cepat dan mudah dalam mengambil *seed.*

LSFR memberikan bit *stream* yang cukup kuat, namun dalam menghasilkan bit memiliki periode, yang mengakibatkan ada perulangan. Oleh karena itu dibutuhkan algoritma yang mampu menghasilkan bit *stream* tanpa menghasilkan pengulangan.

Random number generator yang menggunakan input lain sebagai *seed* terkadang sulit untuk diakses, seperti temperatur. Maka diharapkan dengan menggunakan mic dan kamera, dengan mudah dapat didapatkan *seed.*

Disini video dan suara yang ditangkap tidak hanya berfungsi sebagai *seed* belaka. Nilai bit yang didapatkan dari sound dan video tersebut digunakan kembali untuk menambah keacakan dari bit *stream* yang dihasilkan.

Program menerima input video dan gambar sebagai *seed* untuk pertama kali melakukan pembuatan deret *random* dengan caramengambil 8 bit awal dari file video dan 8 bit awal dari file suara, kemudian menjumlahkan nilai bit untuk masing-masing bit kemudian diproses dengan fungsi div untuk menghasilkan bit keluaran.

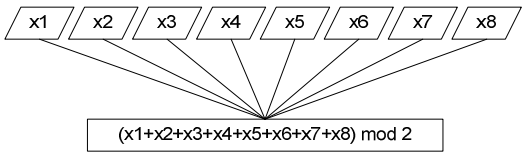


Skema diatas menggambarkan bagaimana seed dibuat pertama kali, dimana ai adalah bit ke-i dari file video, bi adalah bit ke-i dari file suara, dan xi adalah bit ke-i output (yakni seed). Alasan penggunaan cara tersebut untuk mendapatkan seed adalah dengan cara tersebut akan didapatkan seed yang unik untuk setiap file sound dan video. Akan didapatkan hasil yang sama apabila ternyata nilai delapan bit pertama adalah sama, oleh karena itu,

header dari file akan diabaikan (diasumsikan header dari file selalu sama) dan diambil langsung dari data yang bukan header.

Seed hasil keluaran tersebut akan diproses dengan menggunakan algoritma :

$X_i = (X1 + X2 + X3 + X4 + X5 + X6 + X7 + X8) \bmod 2$



Proses pembangkitan diatas seperti proses pembangkitan pada LSFR Dalam proses ini, digunakan semua bit, agar keacakan bit tergantung kepada state keseluruhan bit sebelumnya. Kemudian dengan proses berikutnya, deret bit tersebut akan tidak memiliki siklus.

Hasil keluaran algoritma tersebut kemudian diproses lagi dengan algoritma sebagai berikut :

$$Z = (X_i + m) \bmod 2$$

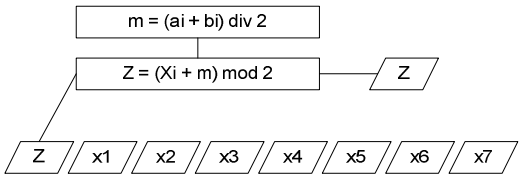
nilai m = 1, jika a dan b adalah sama (sama-sama 0 atau sama-sama 1), m = 0, jika nilai a dan b tidak sama

a, bit yang didapat dari *stream* bit suara

b, bit yang didapat dari *stream* bit gambar

Z, bit keluaran algoritma

Z kemudian akan masuk kembali ke feedback shift menggantikan X1, dan sisa bit akan bergeser ke kanan (ke arah X8).



Proses kedua, yakni proses dengan *stream* bit suara dan gambar adalah proses tambahan untuk menambahkan tingkat keacakan dari *stream* bit. Dengan asumsi bahwa *stream* bit gambar dan suara adalah sesuatu yang memiliki tingkat keacakan tinggi, maka *stream* bit yang dihasilkan akan memiliki tingkat keacakan yang lebih tinggi. Proses ini diharapkan menjadi sumber keacakan utama, yang bersumber pada bit – bit yang ada sebelumnya serta pengaruh bit luar, yakni bit dari video dan suara.

IV. PERANCANGAN

Perangkat lunak yang akan dibangun merupakan perangkat lunak berbasis *desktop*. Perangkat lunak akan secara otomatis bekerja untuk menghasilkan deret bilangan acak setelah diberikan perintah untuk membangkitkan bilangan. Perangkat lunak menerima masukan yang disamarkan dari user (perangkat lunak ketika dijalankan akan mengambil gambar dan suara melalui kamera dan *microphone* tanpa menunjukkan kepada pengguna bahwa perangkat lunak sedang merekam).

Program akan diimplementasikan dalam bahasa c#. Program yang dibuat disini membutuhkan implementasi *library* luar untuk melakukan pengolahan video. *Library* yang digunakan adalah DirectXCapture. *Library* ini memberikan fungsi-fungsi penting untuk program ini, yakni untuk melakukan deteksi *device* yang dapat digunakan untuk menangkap gambar dan suara.

Untuk pengolahan sound, c# sudah menyediakan *library* dari dalam yang mampu untuk menangkap suara melalui *microphone*.

Program akan dibuat dalam satu modul saja, yakni modul utama, yang terdiri dari berbagai fungsi antara lain : menangkap video, menangkap suara, konversi video dan suara menjadi *seed* dan *source* keacakan, proses utama yang menghasilkan deret bit *random*.

Program kemudian akan melakukan konversi nilai byte-byte data pada file suara dan video menjadi bit agar lebih mudah dalam proses menghasilkan deret bit *random*. Bit-bit tersebut akan dituliskan ke file eksternal untuk kemudahan pembacaan dan pengecekan. File eksternal tersebut akan dibaca kembali untuk menjadi *seed* dan kemudian menjadi penambah nilai keacakan deret bit yang dihasilkan.

Didalam implementasi fungsi utama, akan terjadi proses algoritma seperti yang sudah dituliskan pada rancangan solusi di analisis. Kemudian deret bit *random* akan dituliskan kembali di file eksternal agar dapat dibaca kembali.

Kebutuhan fungsional perangkat lunak antara lain :

i. Perangkat lunak mampu menangkap gambar dan suara menggunakan kamera dan *microphone*.

ii. Perangkat lunak mampu mampu menghasilkan deret bit bilangan acak dan menuliskannya ke file eksternal.

iii. Perangkat lunak mampu menerima masukan dari user untuk menspesifikasikan letak file output akan disimpan.

iv. Perangkat lunak mampu menerima masukan dari user berupa panjang output yang akan dihasilkan

Kebutuhan non-fungsional perangkat lunak adalah perangkat lunak memiliki antarmuka yang mudah dimengerti oleh pengguna, sehingga pengguna dapat dengan mudah menggunakan perangkat lunak tersebut.

V. IMPLEMENTASI

A. Lingkungan Implementasi

Implementasi perangkat lunak menggunakan perangkat keras berupa komputer dengan spesifikasi :

Processor Intel Core 2 Duo 2.1 GHz

RAM 2 GB
Hard Disk 320 GB
Perangkat masukan *keyboard* dan *mouse*
Perangkat keluaran monitor

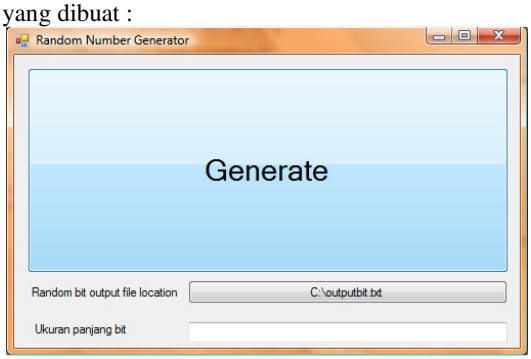
Perangkat lunak yang digunakan :
Sistem operasi Windows Vista Service Pack 2
Microsoft Visual Studio 2008
.Net Framework 3.5

B. Batasan Implementasi

Batasan-batasan yang ada dalam implementasi ini adalah program untuk saat ini spesifik menangani laptop tempat program ini dibuat. Untuk diimplementasikan di komputer atau laptop lain, dibutuhkan penyesuaian dalam pemilihan input device suara dan video.

C. Implementasi Antarmuka

Dibawah ini merupakan tampilan antarmuka yang dibuat :



Tombol Generate berfungsi untuk melalukan seluruh proses pembangkitan bilangan acak. Dibawahnya terdapat tombol untuk mengganti lokasi penyimpanan keluaran generator. Secara default sudah memiliki lokasi penyimpanan, untuk memudahkan user. Tulisan tombol untuk mengganti lokasi penyimpanan keluaran generator adalah lokasi tempat output akan dituliskan.

D. Pengujian Perangkat Lunak

Tujuan dari pengujian perangkat lunak ini adalah untuk memenuhi kebutuhan fungsional perangkat lunak seperti yang telah disebutkan pada bagian-bagian sebelumnya. Tujuan tersebut antara lain :

i. Menguji kemampuan perangkat lunak untuk merekam sound dan video dan menyimpannya sementara di hard disk.

ii. Menguji kemampuan perangkat lunak untuk menghasilkan deret bilangan acak dan menuliskannya ke file eksternal.

iii. Menguji kemampuan perangkat lunak untuk menghasilkan deret bilangan acak dengan panjang berbeda sesuai dengan masukan dari pengguna.

E. Hasil dan Analisis Hasil Uji

i. Kasus uji perangkat lunak untuk melakukan perekaman sound dan video.

Untuk kasus uji ini, perangkat lunak dijalankan seperti biasanya. Kemudian fungsi generate dipanggil / dieksekusi. Pada direktori C:\, akan ditemukan file record.avi dan temp.wav yang sebelumnya tidak ada. File tersebut ketika dibuka berisikan rekaman 5 detik video dan suara. Jika file tersebut sudah ada sebelumnya, file tersebut akan digantikan dengan rekaman terbaru file video dan suara.

ii. Kasus uji perangkat lunak untuk menghasilkan deret bilangan acak dan menuliskan ke file eksternal.

Untuk kasus uji ini, perangkat lunak dijalankan seperti biasanya. Generate dipanggil / dieksekusi. Setelah proses keseluruhan selesai, akan ditemukan file output yang letaknya telah ditentukan oleh user. Ketika dibuka, file tersebut berisikan deret bit keluaran perangkat lunak. Setiap kali proses generate dipanggil, isi file digantikan dengan keluaran perangkat lunak (kecuali dispesifikasikan lokasi baru untuk menyimpan file).

iii. Kasus uji perangkat lunak untuk menghasilkan deret bilangan acak dengan panjang berbeda sesuai dengan masukan dari pengguna.

Untuk kasus uji ini, sebelum proses generate dipanggil, panjang output bit dispesifikasikan. Input yang akan diuji coba adalah nilai-nilai tertentu, serta apabila textbox tempat menerima masukan panjang dikosongkan. Hasil keluaran bit untuk nilai-nilai tertentu menghasilkan panjang bit yang sesuai dengan masukan pada textbox.

VI. PENGUJIAN DAN ANALISIS DERET BILANGAN ACAK

Tujuan pengujian ini adalah untuk mengetahui tingkat keacakan deret bit random yang dihasilkan menggunakan tes statistik NIST.

Dari sekian banyak kriteria, akan diambil beberapa kriteria yang dapat cukup menunjukkan tingkat keacakan algoritma yang dibuat. Kriteria yang akan digunakan sebagai dasar penilaian algoritma adalah :

i. *Frequency*

Frequency test melihat kemunculan 0 dan 1.

ii. *Cumulative Sums*

Melihat kemuncul jumlah kemunculan 0 dan 1 pada awal deret.

iii. *Longest runs of ones*

Mengetahui panjang run terpanjang dari bit 1 (run adalah kemunculan deret bit yang sama tanpa berganti bit)

iv. *Runs*

Mengetahui jumlah total run (jumlah total berapa kali terjadi pergantian bit pada deret).

v. *Non-overlapping template matching*

Mengetahui jumlah kemunculan sub-string tertentu. Jika bertemu pola, pengujian bergeser sebanyak jumlah blok.

vi. *Overlapping template matching*

Mengetahui jumlah substring tertentu. Jika menemukan pola, pengujian tetap hanya akan bergeser satu bit.

Pengujian dilakukan terhadap hasil keluaran generator sebanyak 40000 bit. Blok sample yang digunakan adalah sebanyak 5000 bit, jadi terdapat 8 blok. Untuk non-overlapping template test dan overlapping template test digunakan m (block length) : 9.

i. *Frequency*

Terdapat FAILURE pada blok ke-6, dimana perbedaan frekuensi antara 0 dan 1 cukup besar, yakni mencapai 190. Pada blok ke-2 juga hampir tidak memenuhi karena beda frekuensi antara 0 dan 1 cukup besar, namun nilai p-value masih masuk kedalam nilai yang dapat diterima.

ii. *Cumulative Sums*

Terdapat FAILURE pada blok ke-6, dimana cumulative sum forward terlalu besar. Menunjukkan adanya bit yang menumpuk jumlahnya di bagian depan deret. Mengakibatkan tingkat keacakan deret berkurang.

iii. *Longest run of ones*

SUCCESS untuk seluruh blok.

iv. *Runs*

Terdapat FAILURE pada blok ke-6, dimana jumlah run terlalu besar. Menunjukkan perubahan nilai yang terlalu cepat pada blok.

v. *Non-overlapping template matching*

Terdapat FAILURE pada bit 33, 44, 48, 50, 70, 90, 100, 107, 142. Terjadi kemunculan deret bit tersebut berkali-kali pada blok tertentu. Menunjukkan adanya template yang berulang, yang mana mengurangi tingkat keacakan.

vi. *Overlapping template matching*

SUCCESS untuk seluruh blok.

Berdasarkan hasil keluaran yang didapatkan di berbagai kriteria menunjukkan bahwa deret bit *random* sudah memiliki tingkat keacakan yang cukup.

VII. KESIMPULAN DAN SARAN

A. Kesimpulan

Kesimpulan yang didapat dari pengerjaan makalah ini adalah :

1. Deret bilangan acak telah melewati tes statistik NIST dengan berhasil melewati berbagai kriteria tes keacakan.
2. Sound dan video merupakan sumber keacakan yang mudah untuk digunakan untuk proses pembangkitan bilangan acak. Namun untuk penggunaannya sebagai sumber keacakan, diperlukan pemrosesan tambahan karena ada keadaan dimana bit-bit dari suara dan video adalah monoton (semua 1 atau semua 0).
3. Untuk algoritma ini, dibutuhkan waktu yang cukup lama untuk menghasilkan deret bit ukuran besar. Hal ini dikarenakan pemrosesan bit yang satu-per satu.

B. Saran

Saran untuk pengembangan perangkat lunak ini adalah masih diperlukan pemrosesan lebih lanjut pada file sound dan video agar bit – bit yang digunakan adalah bit – bit yang seluruhnya memberikan pengaruh pada proses pembangkitan.

REFERENSI

A. Menezes, P. van Oorschot, and S. Vanstone ,1996. Handbook of Applied Cryptography. CRC Press, ch 5.

LEWIS, P A W, GOODMAN, A.S., AND MILLER, J M A ,1969. *pseudo-random number generator* for the system/360. IBM Syst. J. 8, 2, 136-146.

Schrage, L ,1979. A More Portable FORTRAN *Random number generator*. ACM Transactions on Mathematical *Software* 5 (2), 132-138.

Jun, Benjamin and Kocher, Paul ,1999. The Intel *Random number generator*. Cryptography Research, Inc. White Paper Prepared for Intel Corporation

R. McEvoy, J. Curran, P. Cotter and C. Murphy ,2006. Fortuna : *Cryptographic*ally *SecurePseudo-RandomNumber* Generation In *Software* And *Hardware*. Department of Electrical and Electronic Engineering, University College Cork.

Soto, Juan (2008). Statistical Testing of *Random number generator*s. National Institute of Standards & Technology,

N. Anyanwu, Lih-Yuan Deng, and D. Dasgupta ,2009. Design of *Cryptographic*ally Strong *Generator* By Transforming Linearly Generated Sequences. Department of Computer Science, The University of Memphis.

https://ccrma.stanford.edu/courses/422/projects/WaveFormat/ (terakhir diakses 3 Mei 2012)

http://csrc.nist.gov/groups/ST/toolkit/rng/index.html (terakhir diakses 27 Mei 2012)