Pemanfaatan Sumber Entropi Suara untuk Pembangkit Angka Acak

Yakobus Iryanto Prasethio – 13520104
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520104@std.stei.itb.ac.id

***Abstract*—Dalam kriptografi, penggunaan bilangan acak sangat penting untuk menjaga kerahasiaan, integritas, dan autentikasi. Salah satu metode yang digunaan untuk menghasilan bilangan acak adalah True Random Number Generator (TRNG). Fokus utama makalah ini adalah penggunaan sumber entropi dari suara untuk membangkitkan angka acak melalui TRNG. Berlandaskan pada keacakan alami yang dihasilkan dari sinyal suara, dibuat suatu sistem pembangkitan angka acak yang dapat digunakan dalam kriptografi. Pada makalah ini dipaparkan implementasi dari metode tersebut beserta analisis menggunakan metrik uji keacakan NIST.**

Keywords—True Random Number Generator, Entropi, Keamanan, NIST

I. INTRODUCTION

Dalam kriptografi, keacakan bilangan adalah salah satu kunci keamanan sebuah sistem komputer. Salah satu faktor keamanan yang ditawarkan dalam kriptografi adalah sifat ketidakpastian. Semakin tidak pasti sebuah bilangan, maka proses pemecahan akan menjadi lebih sulit dan memakan waktu lebih banyak. Sebagai contoh, apabila kunci sebuah algoritma enkripsi panjang dan tidak memiliki pola yang jelas, maka lebih sulit bagi seorang penyerang untuk memecahkan enkripsinya. Salah satu cara untuk membentuk bilangan yang acak adalah True Random Number Generator (TRNG).

TRNG secara umum menggunakan proses fisik untuk membentuk bilangan acak. TRNG ini disebut TRNG berbasis perangkat keras. Hal ini dikarenakan proses fisik sulit untuk direplikasi, sehingga hasilnya tidak dapat diprediksi. TRNG berbeda dengan Pseudo Random Number Generator (PRNG). PRNG membentuk bilangan acak menggunakan algoritma deterministik, sehingga kemungkinan terbentuknya pola dalam bilangan cukup tinggi.

Banyak aplikasi dari sebuah TRNG dalam dunia kriptografi. Salah satu aplikasi TRNG adalah pembangkitan kunci untuk algoritma enkripsi simetris dan asimetris, misalnya Advanced Encryption Standard (AES) atau Rivest – Shamir – Adleman (RSA). Aplikasi TRNG lain adalah Initialization Vector (IV) untuk mode dalam sebuah *block cypher*. Kedua aplikasi tersebut memanfaatkan keacakan bilangan yang dihasilkan oleh TRNG untuk mengurangi potensi serangan pola.

Terdapat beberapa contoh TRNG berbasis perangkat keras, salah satunya adalah penggunaan *radioactive decay*. Setiap kali sebuah peristiwa ionisasi di deteksi, informasi ini akan dikonversi menjadi bit yang dapat diproses oleh komputer. Melalui metode ini, bit – bit yang dihasilkan oleh proses radioaktif acak dan prosesnya tergolong cepat. Akan tetapi, perangkat keras yang digunakan cukup berbahaya untuk dipakai dalam aplikasi yang nyata. Diperlukan pengamanan cukup tinggi agar terlindung dari paparan radiasi yang dikeluarkan oleh materi radioaktif.

Di sisi lain, TRNG berbasis perangkat lunak biasanya menggunakan algoritma deterministik, sehingga lebih cocok untuk digunakan sebagai PRNG. Ketika diberikan sebuah *seed* yang sama, algoritma PRNG akan membentuk bilangan acak yang sama, sehingga apabila nilai *seed* ini diketahui, akan mudah untuk mendapatkan bilangan tersebut. Hal ini dikarenakan sifat RNG berbasis perangkat lunak yang menggunakan prinsip matematika dan bukan proses fisika.

Oleh karena itu, diperlukan metode pembentukan bilangan acak yang mampu menyelesaikan kelemahan metode yang ada. Salah satu alternatif yang cukup menjanjikan adalah penggunaan suara sebagai sumber pembentukan bilangan acak. Suara yang direkam menggunakan sebuah mikrofon memiliki tingkat ketidakpastian yang tinggi. Hal ini dikarenakan audio tersebut dapat mencakup suara latar, suara manusia, atau suara mekanis. Dengan gabungan suara – suara tersebut, hasil rekaman akan berbeda setiap kali.

Sebagian besar perangkat yang digunakan sehari – hari memiliki mikrofon yang dapat digunakan untuk merekam suara. Metode suara memiliki aksesibilitas yang tinggi dan biaya yang rendah. Selain itu, proses rekaman suara tidak memerlukan daya yang besar. Keuntungan tersebut digabungkan dengan metode perekaman yang dapat digunakan di lingkungan apapun membuat suara menjadi alternatif yang layak untuk diuji.

II. TEORI DASAR

A. Pembangkitan Bilangan Acak

Bilangan acak adalah sebuah bilangan yang tiddak dapat diprediksi nilai dan kemunculannya. Bilangan acak sangat penting dalam dunia kriptografi, misalnya untuk membangkitkan nilai parameter kunci algoritma kriptografi, pembangkitan *initialization vector* (IV) untuk sebuah *block cipher*, atau pembuatan one – time pad.

Tidak ada prosedur komputasi yang dapat menghasilkan deret bilangan acak secara sempurna. Bilangan acak yang dapat dihasilkan oleh sebuah algoritma adalah bilangan acak semu. Dengan memberikan umpan atau *seed* yang sama ke dalam algoritma, bilangan acak dapat kembali dibentuk. Pembangkitan deret bilangan acak ini disebut *pseudo – random number generation* (PRNG).

Sebuah komputer secara desain dibuat untuk menjalankan instruksi yang diberikan untuk menghasilkan hasil yang dapat diprediksi. Hal ini menyulitkan proses pembentukan entropi yang baik menggunakan komputer. Untuk mendapatkan bilangan yang benar – benar acak, diperlukan sumber entropi yang baik. Beberapa kriteria yang mendefinisikan keacakan nyata adalah:

- Ketidakpastian bit yang muncul
- Distribusi seragam dari bit dalam bilangan
- Minimnya pola yang dibentuk

B. *Entropi*

Entropi adalah ukuran ketidakpastian atau kekacauan yang terkait dengan sebuah variabel acak. Entropi pertama kali diperkenalkan oleh Claude Shannon pada tahun 1948. Dalam konteks teori informasi, entropi mengkuantifikasi jumlah ketidakpastian atau konten informasi. Entropi untuk sebuah variabel acak X dapat dihitung menggunakan persamaan berikut:

$$H(X) = -\sum_{i} P_x(x_i) \log_b P_x(x_i)$$

$H(X)$ adalah entropi dari variabel acak X, $P_x(x_i)$ adalah probabilitas dari kejadian x_i, dan b adalah basis logaritma yang menentukan unit entropi. Persamaan ini menyatakan tingkat ketidakpastian rata – rata dari semua kejadian yang mungkin untuk variabel acak tersebut. Semakin tinggi entropi, maka semakin tidak dapat diprediksi atau acak variabel tersebut.

Dalam kriptografi, entropi sangat penting untuk mengevaluasi keamanan kunci kriptografi, karena entropi yang lebih tinggi berarti tingkat ketidakpastian lebih tinggi serta kunci yang lebih kuat dan lebih sulit untuk diprediksi. Konsep entropi juga fundamental dalam aspek kriptografi lainnya seperti pembangkitan bilangan acak.

C. *Entropi Berbasis Perangkat Keras: Radioactive Decay*

Proses fisika seperti *radioactive decay* terjadi secara acak dan tidak berpola. Sifat ini yang dimanfaatkan untuk menjadi basis dari sebuah TRNG. Proses ionisasi dapat dideteksi menggunakan tabung Geiger – Müller, yang kemudian dihubungkan ke sebuah Raspberry Pi untuk dikonversi menjadi bentuk digital. Setiap sinyal dari proses ionisasi akan dipetakan menjadi bit yang berbeda.

Radioactive decay digunakan untuk membentuk bit acak karena mengikuti distribusi Poisson. Waktu di antara dua peristiwa *decay* mengikuti distribusi eksponensial. Distribusi eksponensial ini *memoryless*, yang berarti waktu antara peristiwa tidak saling terhubung. Dalam kata lain, kemungkinan sebuah peristiwa terjadi tidak dipengaruhi oleh peristiwa sebelumnya. Sifat ini memastikan bahwa bit yang dibentuk acak dan tidak berpola.

Sifat entropi yang tidak memiliki pola ini sangat berguna untuk mengurangi serangan pola. Serangan pola biasa digunakan terhadap algoritma deteministik, sedangkan dalam kasus *radioactive decay*, tidak ada pola untuk proses fisika. Oleh karena itu, bilangan acak yang dihasilkan memiliki kualitas yang tinggi dan dapat digunakan untuk pembentukan one – time pad atau pembangkitan kunci.

D. *Entropi Berbasis Perangkat Lunak: Linux /dev/random*

Ada dua jenis *file* random yang disediakan dalam Linux, yaitu /dev/random dan /dev/urandom. RNG di dalam kedua file ini mengambil informasi dari lingkungan dan sumber lainnya, yang kemudian disimpan dalam sebuah penyimpanan entropi. RNG-nya juga menyimpan informasi keacakan bit di dalam tempat penyimpanan entropi. Dari penyimpanan entropi inilah bilangan acak dibentuk.

Ketika /dev/random digunakan, maka RNG akan membentuk bilangan dari penyimpanan entropi. Apabila penyimpanan entropi habis, maka /dev/random akan melakukan *blocking* hingga kolam entropi terisi kembali. Oleh karena itu, /dev/random dapat dipakai untuk proses yang sangat membutuhkan keacakan dengan kualitas yang tinggi, seperti *one – time pad* atau pembangkitan kunci.

Ketika /dev/urandom digunakan, maka RNG tidak akan melakukan *blocking* apabila penyimpanan entropi habis. RNG akan membentuk bilangan menggunakan algoritma deterministik, sehingga tingkat keacakan tidak sebagus /dev/random dan terdapat potensi serangan. Oleh karena itu, penggunaan /dev/urandom tidak disarankan untuk proses yang membutuhkan keacakan dengan kualitas tinggi. /dev/urandom cocok digunakan untuk aplikasi yang membutuhkan bilangan acak tanpa memprioritaskan tingkat keamanannya.

III. Desain dan Implementasi

Pembuatan TRNG akan menggunakan suara sebagai masukan entropi. Hal ini dikarenakan suara yang direkam menggunakan sebuah mikrofon tidak akan sama dan tidak memiliki pola, sehingga cocok untuk digunakan sebagai basis entropi TRNG.

A. *Desain*

TRNG dirancang untuk membangkitkan angka acak dengan kualitas entropi yang tinggi menggunakan suara sebagai sumber masukan entropi. Tahapan yang akan dilakukan dalam TRNG adalah sebagai berikut:

1. Pengambilan suara: Menggunakan mikrofon untuk merekam sinyal suara. Mikrofon akan menangkap suara lingkungan, suara manusia, dan suara mekanis
2. *Pre-processing*: Melakukan penyaringan dan normalisasi sinyal suara
3. Ekstraksi entropi: Sinyal suara yang sudah dinormalisasi akan dikonversi menjadi bit – bit acak menggunakan teknik seperti FFT dan *hashing*

4. *Post-processing*: Menghilangkan bias dan melakukan validasi bit – bit acak yang dibentuk
5. Pembangkitan angka acak: Menggunakan entropi yang sudah di-ekstraksi untuk menghasilkan angka acak

B. Implementasi

1. Pengambilan suara

   Pengambilan suara dilakukan menggunakan mikrofon yang terhubung ke komputer. Mikrofon yang digunakan adalah bagian dari sebuah *headset*. *Headset* yang digunakan adalah Logitech G535 Lightspeed Wireless Gaming Headset. Suara akan direkam dengan *sampling rate* yang standar, yaitu 44,1 kHz dan resolusi 16 – bit.

   Pengambilan suara diambil dengan durasi yang berbeda dan lingkungan yang sama, yaitu pagi hari dan cuaca cerah. Dikumpulkan 3 buah rekaman suara dengan durasi 5 menit, 10 menit, dan 30 menit.

2. *Pre – processing*

   Rekaman suara yang didapatkan harus melewati tahapan penyaringan sebelum digunakan. Penyaringan yang dilakukan bertujuan agar tidak terjadi *aliasing*. *Aliasing* terjadi ketika *sampling rate* sebuah rekaman terlalu rendah untuk menangkap perubahan frekuensi yang terjadi, sehingga frekuensi tinggi dapat direpresentasikan sebagai frekuensi rendah. Untuk mengatasi *aliasing*, digunakan frekuensi Nyquist. Frekuensi Nyquist adalah frekuensi tertinggi yang dapat ditangkap pada *sampling rate* tertentu. Ukuran frekuensi Nyquist adalah setengah dari *sampling rate*, sehingga dalam rekaman pada tahapan pertama, frekuensi maksimal yang dapat ditangkap tanpa terjadinya *aliasing* adalah

   $$44.100\ Hz\ /\ 2\ =\ 22.050\ Hz$$

   Apabila terdapat frekuensi dalam rekaman yang lebih tinggi dari 22.050 Hz, ada kemungkinan terjadinya *aliasing*.

   Rentang frekuensi yang dapat didengar oleh manusia adalah 20 – 20.000 Hz. Oleh karena itu, penyaringan akan dilakukan untuk membatasi suara ke frekuensi yang dapat didengar oleh manusia dan dibawah frekuensi Nyquist untuk mencegah *aliasing*. Digunakan sebuah filter Butterworth sebagai filter *band – pass* yang akan menyaring suara. Filter Butterworth digunakan karena sifatnya yang tidak memperkuat atau meredam frekuensi dalam rentang yang diinginkan (20 – 20.000 Hz). Filter Butterworth juga memiliki proses penghalusan untuk frekuensi yang berada di luar rentang ini, sehingga tidak terbentuk artefak yang diakibatkan oleh penyaringan.

3. Ekstraksi entropi

   Suara yang sudah dilakukan penyaringan dan normalisasi akan dikonversi dari *time domain* menjadi *frequency domain*. Hal ini bertujuan agar dapat dilakukan analisis untuk frekuensi berbeda yang membentuk sebuah sinyal suara. Konversi ini akan menggunakan *Fast Fourier Transform* (FFT). Konversi dari *time domain* menjadi *frequency domain* dilakukan karena *frequency domain* memiliki tingkat keacakan dan kompleksitas yang lebih tinggi dibandingkan *time domain*, sehingga lebih cocok untuk digunakan sebagai sumber entropi.

4. *Post – processing*

   Sinyal suara yang sudah dikonversi kemudian akan dilakukan *hashing*. *Hashing* bertujuan untuk memastikan distribusi seragam frekuensi yang didapatkan. Dalam *frequency domain*, distribusi nilai mungkin tidak seragam, misalnya ada frekuensi yang mendominasi. Hal ini akan menyebabkan bias dalam data, karena ada nilai yang lebih banyak muncul. Oleh karena itu, digunakan fungsi *hash* untuk mengatasi permasalahan ini. Fungsi *hash* seperti SHA – 256 dirancang untuk menghasilkan keluaran yang seimbang, yang berarti setiap bit memiliki kemungkinan yang sama untuk menjadi 0 atau 1.

   Fungsi *hash* dirancang agar *collision – resistant*, artinya secara komputasi akan sulit untuk menemukan dua masukan berbeda yang menghasilkan keluaran yang sama. Hal ini memastikan integritas dan ketidakpastian nilai yang dihasilkan. Walaupun fungsi *hash* sendiri adalah algoritma deterministik (masukan yang sama akan menghasilkan keluaran yang sama), tetapi perubahan kecil dalam masukan akan memberikan perubahan yang besar dan tidak pasti pada keluaran.

5. Pembangkitan angka acak

   Bit entropi yang sudah didapat dari tahapan sebelumnya dapat digunakan untuk membangkitan bilangan acak. Bit entropi sendiri sudah memiliki keacakan sejati dan dapat langsung digunakan dengan melakukan konversi bentuk menjadi integer. Penggunaan lain dari bit entropi ini adalah sebagai umpan untuk sebuah PRNG, tetapi ini berada di luar batasan makalah.

IV. ANALISIS

Bilangan acak yang dibentuk oleh TRNG akan diuji menggunakan National Institute of Standards and Technology (NIST) Statistical Test Suite. *Test suite* ini terdiri dari 15 uji statistik yang menilai tingkat keacakan sebuah bilangan. 15 uji statistik ini adalah *frequency, block frequency, cumulative sums, runs, longest run of ones, rank, discrete fourier transform, nonperiodic template matchings, overlapping template matchings, universal statistical, approximate entropy, random excursions, random excursions variant, serial*, dan *linear complexity*. Dari pengujian ini, *test suite* akan menghasilkan nilai P, yang mengindikasikan keacakan sebuah bilangan. Semakin dekat nilai P ke 1, semakin baik dan acak sebuah bilangan.

Hasil pembangkitan bit acak untuk setiap durasi rekaman suara adalah sebagai berikut:

1. Durasi 5 menit menghasilkan 3307264 bit
2. Durasi 10 menit menghasilkan 6614784 bit

3. Durasi 30 menit menghasilkan 19844864 bit

NIST Statistical Test Suite merekomendasikan panjang bit 1.000.000 untuk melakukan pengujian yang optimal. Proses pengujian akan dilakukan dengan panjang 100.000 per segmen. Hal ini dikarenakan panjang bit yang tidak mencukupi untuk memenuhi rekomendasi 1.000.000 bit. Sebagai contoh, untuk rekaman suara dengan durasi 30 menit, jumlah bit *stream* yang dipakai adalah 198, karena perlu dilakukan pembulatan ke bawah untuk melakukan pengujian.

Untuk memvisualisasikan hasil pengujian NIST, akan diambil 3 pengujian sebagai dasar, yaitu *frequency*, *block frequency*, dan *cumulative sum*.

1. *Frequency* test

   Pengujian ini menghitung distribusi nilai bit dalam sebuah bilangan. Sebuah bilangan acak yang baik memiliki distribusi bit yang seragam antara bit 0 dan 1.

2. *Block Frequency* test

   Pengujian ini menghitung distribusi nilai bit per blok dalam bilangan. Misalkan ukuran blok adalah M, maka sebuah bilangan acak yang baik memiliki distribusi bit M/2 untuk bit 0 dan 1.

3. *Cumulative sum* test

   Pengujian ini menghitung penjumlahan bit dalam bilangan. Sebagai contoh, untuk bilangan 1, 0, 1, 1, 0 (dengan mengasumsikan 0 dipetakan ke -1 dan 1 dipetakan ke +1), perhitungan *cumulative sum* adalah sebagai berikut.

   - Mulai dari 0
   - Setelah (1): 0 + 1 = 1
   - Setelah (0): 1 - 1 = 0
   - Setelah (1): 0 + 1 = 1
   - Setelah (1): 1 + 1 = 2
   - Setelah (0): 2 - 1 = 1

   Hasil *cumulative sum* untuk bilangan acak yang baik adalah 0 atau mendekati 0.

Tujuan utama dari NIST Statistical Test Suite adalah distribusi yang seragam antara setiap segmen dalam pengujian, ditandai dengan C1 hingga C10. Semakin seragam distribusi yang dihasilkan, maka keacakan sebuah bilangan semakin baik.

1. Durasi 5 menit

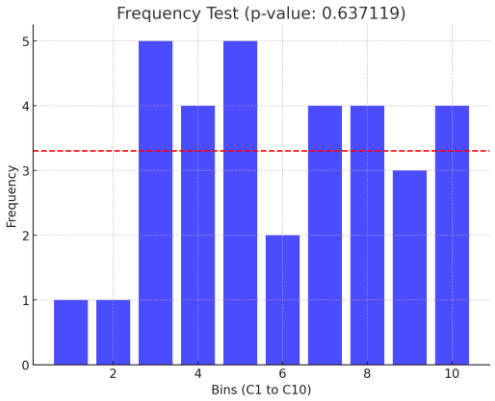

Fig 1. Frequency Test (5 min)

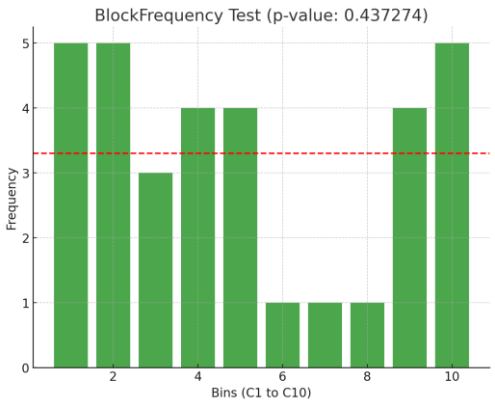

Fig 2. Block Frequency Test (5 min)

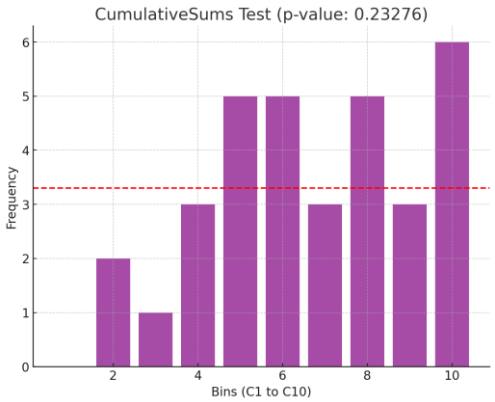

Fig 3. Cumulative Sums Test (5 min)

Terlihat bahwa distribusi untuk semua pengujian sangat tidak seragam. Hal ini menunjukkan bahwa bilangan yang dihasilkan oleh rekaman suara dengan durasi 5 menit memiliki tingkat keacakan yang buruk.

2. Durasi 10 menit

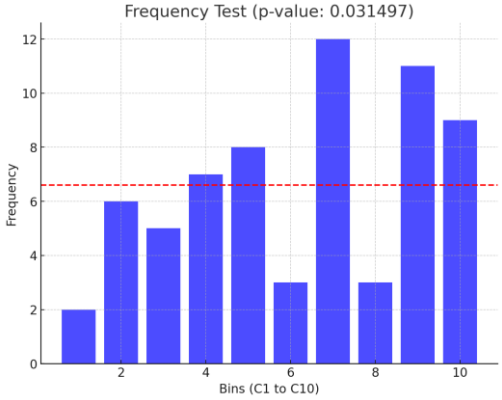

Fig 4. Frequency Test (10 min)

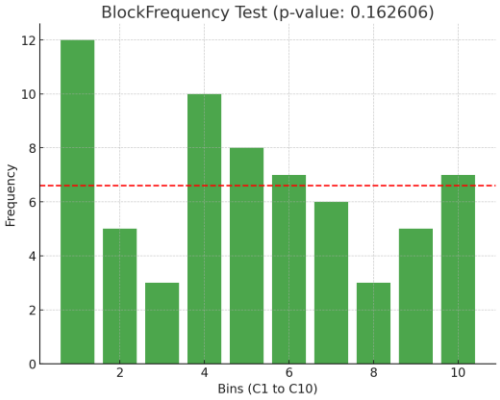

Fig 5. Block Frequency Test (10 min)

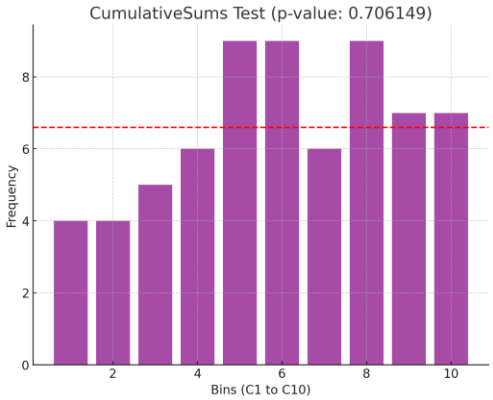

Fig 6. Cumulative Sums Test (10 min)

Terlihat bahwa distribusi untuk semua pengujian masih kurang seragam. Hal ini menunjukkan bahwa bilangan yang dihasilkan oleh rekaman suara dengan durasi 10 menit memiliki tingkat keacakan yang buruk. Nilai distribusi *frequency test* yang tidak seragam menunjukkan ketimpangan jumlah bit 0 dan 1 dalam bilangan. Akan tetapi, jika dibandingkan terhadap hasil rekaman suara dengan durasi 5 menit, terdapat peningkatan terhadap distribusinya, sehingga tingkat keacakan dari rekaman suara durasi 10 menit lebih baik.

3. Durasi 30 menit

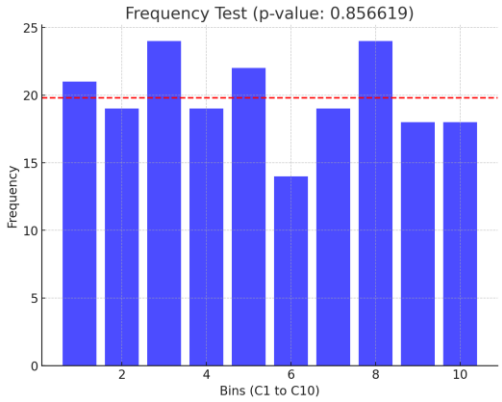

Fig 7. Frequency Test (30 min)

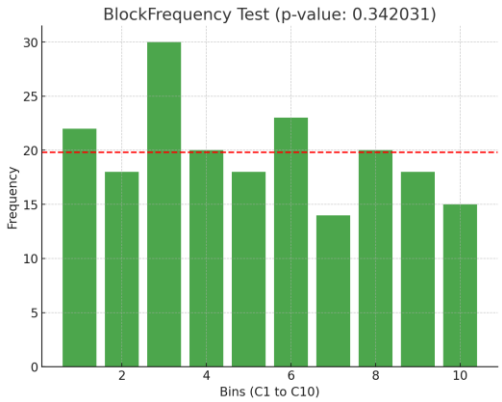

Fig 8. Block Frequency Test (30 min)

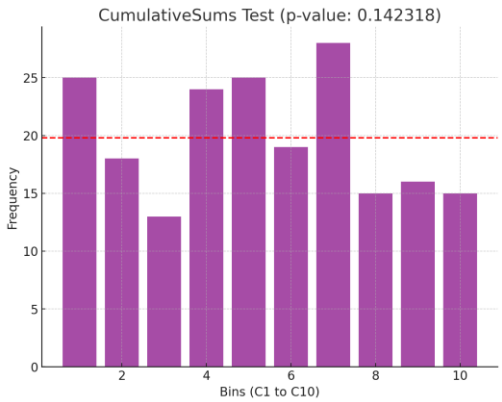

Fig 9. Cumulative Sums Test (30 min)

Terlihat bahwa distribusi frekuensi untuk pengujian *frequency* dan *block frequency* hampir seragam, tetapi untuk *cumulative sum*, distribusinya lebih rendah. Hal ini menunjukkan bahwa bilangan yang dihasilkan oleh rekaman suara dengan durasi 30 menit memiliki tingkat keacakan yang cukup baik, tetapi masih ada ruang untuk peningkatan. Jika dibandingkan dengan rekaman

suara durasi 5 dan 10 menit, peningkatan tingkat keacakan bilangan sangat tinggi.

Akan tetapi, jika distribusi bit 0 dan 1 untuk setiap rekaman audio dipetakan sebagai berikut

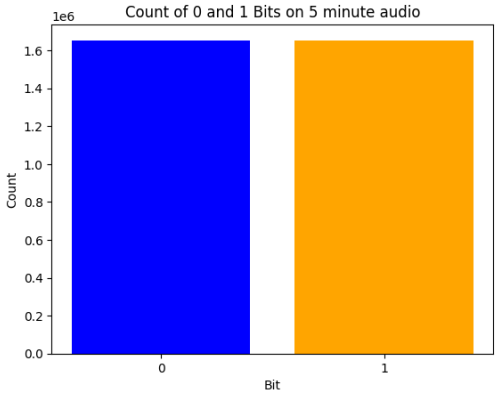

Fig 10. Distribusi Bit untuk rekaman 5 menit

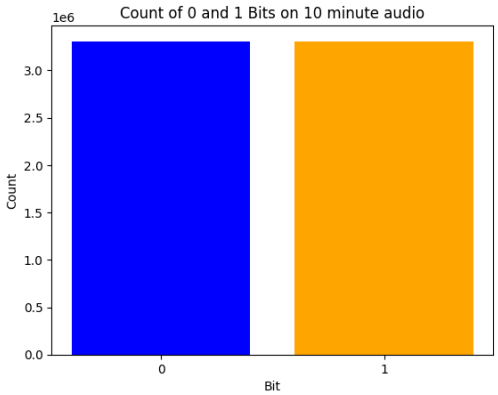

Fig 11. Distribusi Bit untuk rekaman 10 menit

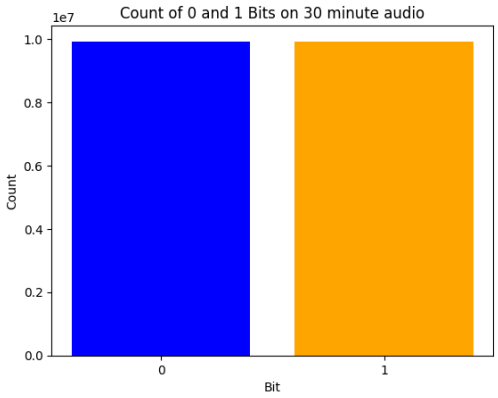

Fig 12.Distribusi Bit untuk rekaman 30 menit

Terlihat bahwa distribusi bit untuk ketiga rekaman sama, sehingga hasil dari NIST mungkin kurang akurat akibat penggunaan 100.000 bit (bukan nilai 1.000.000 bit yang direkomendasikan). Oleh karena itu, rekaman 30 menit diuji menggunakan NIST dengan ukuran segmen 1.000.000 bit.

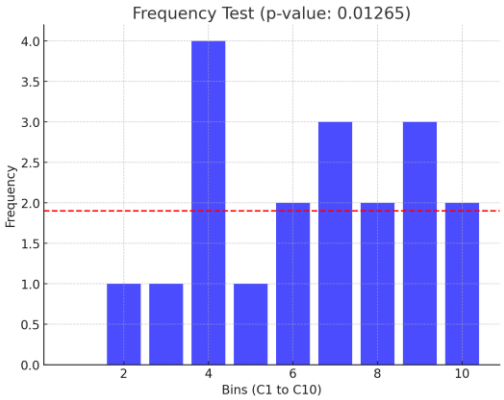

Fig 13. Frequency Test (30 min - 1 mil segment)

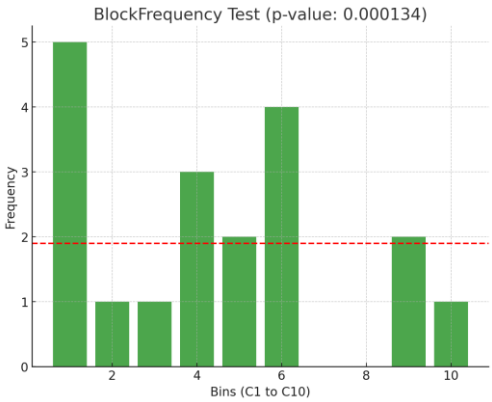

Fig 14. Block Frequency Test (30 min - 1 mil segment)

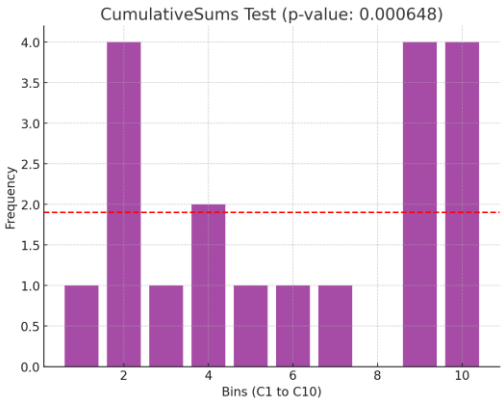

Fig 15. Cumulative Sums Test (30 min - 1 mil segment)

Terlihat bahwa distribusi untuk ketiga pengujian tidak seragam. Hasil tersebut mengindikasikan data sinyal suara yang kurang panjang untuk melakukan pengujian secara penuh. Dengan hanya 19 segmen yang mampu diuji oleh *test set*, hasil yang didapatkan menjadi kurang akurat.

V. KESIMPULAN

Dalam makalah ini, telah berhasil dikembangkan sebuah *True Random Number Generator* (TRNG) menggunakan sinyal suara sebagai sumber entropi. Sinyal suara sebagai sumber entropi memiliki potensi pengembangan untuk membentuk

bilangan acak yang sejati. Pemilihan suara sebagai sumber entropi didasari oleh sifatnya yang acak dan kompleks. Sebuah rekaman suara dapat memiliki suara lingkungan, suara manusia, dan suara mekanis, sehingga tingkat keacakannya baik.

Hasil pengujian menggunakan NIST Statistical Test Suite terhadap beberapa rekaman suara memberikan hasil yang bervariasi. Rekaman suara dengan durasi 5 dan 10 menit menunjukkan hasil yang kurang baik untuk tingkat keacakan, sedangkan untuk durasi 30 menit, hasil yang didapatkan cukup baik, dan masih ada ruang untuk peningkatan. Dapat disimpulkan bahwa algoritma yang dipakai kurang dapat memenuhi standar yang ditetapkan oleh NIST.

Kualitas dari entropi yang diambil dari sinyal suara sangat bergantung pada kualitas mikrofon dan lingkungan rekaman. Inkonsistensi pada hasil dari NIST Statistical Test Suite menunjukkan perlunya algoritma yang lebih kompleks untuk melakukan konversi dari sinyal suara menjadi bit entropi. Diperlukan juga data rekaman yang lebih banyak dan beragam untuk menguji berbagai kemungkinan. Proses rekaman juga perlu memperhitungkan lingkungan yang berbeda – beda.

SOURCE CODE

Kode algoritma yang digunakan beserta hasil yang didapatkan dapat dilihat di Github berikut

https://github.com/YakobusIP/audio-entropy-trng

UCAPAN TERIMA KASIH

Segala puji syukur penulis panjatkan bagi Tuhan Yang Maha Esa karena telah memberikan penulis kelancaran dalam menulis makalah yang berjudul “Pemanfaatan Sumber Entropi Suara untuk Pembangkit Angka Acak”. Penulis juga ingin menyampaikan terima kasih kepada orang tua penulis atas dukungan yang diberikan kepada penulis selama proses pembuatan makalah. Penulis ingin menyampaikan rasa terima kasih sebesar – besarnya kepada Rinaldi Munir selaku dosen pengampu IF4020 Kriptografi yang telah membimbing penulis selama proses perkuliahan. Penulis menyadari bahwa makalah ini jauh dari kata sempurna. Oleh karena itu, penulis berharap bahwa makalah ini dapat digunakan semaksimal mungkin dan dikembangkan lebih lagi agar memberi dampak yang besar ke masyarakat luas.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. Pembangkit Bilangan Acak (Bahan Kuliah IF4020 Kriptografi). 2024
Diakses pada 10 Juni 2024 (https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/32-Pembangkit-bilangan-acak-2024.pdf)

[2] Linux Manual. Random(4) – Linux Manual Page. 2023.
Diakses pada 10 Juni 2024
(https://www.man7.org/linux/man-pages/man4/random.4.html)

[3] D. Ruschen, M. Schrey, J. Freese, dan I. Heisterklaus, Generation of true random numbers based on radioactive decay. 2017.

[4] B. Zolfaghari, K. Bibak, dan T. Koshiba. The oddysey of entropy: cryptography. 2022.

[5] A. Vassilev dan T. A. Hall. The importance of entropy to information security. 2014.

[6] A. Rukhin, et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications. 2010.

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.

Bandung, 12 Juni 2024

Yakobus Iryanto Prasethio 13520104