Analisis Pembangkitan Angka *Random* dalam Kriptografi

Samuel Gondokusumo 18219024
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
vincentius.sameulgk2@gmail.com

***Abstract*—Kriptografi merupakan ilmu yang mempelajari cara membentuk komunikasi yang aman meski terdapat pihak ketiga pun. Dalam implementasinya pada bidang teknologi informasi, banyak sekali aspek kriptografi yang memanfaatkan angka *random* yang dihasilkan dari *random number generator* (RNG). Makalah ini akan melakukan eksplorasi akan jenis-jenis RNG yang digunakan dalam konteks kriptografi, kemudian membandingkannya dan menganalisisnya, untuk mencari tahu titik-titik lemah yang dapat diperbaiki dan dapat dikembangkan lebih lanjut untuk mendapatkan hasil yang paling mangkus dan aman.**

Keywords—Kriptografi; random number generator; analisis

I. Pendahuluan

Angka *random* memiliki peran yang penting dalam dunia teknologi informasi. Peran ini mencakup, namun tidak terbatas pada, enkripsi data, keamanan informasi, dan navigasi satelit. Dalam kriptografi, angka *random* berperan dalam algoritma-algoritma seperti fungsi *hash*, tanda tangan digital (*digital signatures*), dan *one-time passwords* (OTP) [1].

Dalam pembangkitan kunci kriptografi, seperti *secret key* pada kunci privat atau kunci publik, digunakan angka *random* untuk menghasilkan sebuah *string* yang panjang dan sulit ditebak, seperti pada algoritma RSA, yang membangkitkan dua angka prima p dan q secara acak [2]. Aspek *randomness* dan *unpredictability* ini sangat krusial dalam upaya memastikan bahwa kunci yang dibangkitkan tidak dapat ditebak ataupun direplikasi oleh pihak lain, alhasil menjaga keautentikan kunci tersebut.

Angka *random* dapat dibangkitkan melalui beberapa cara, seperti *true random number generators* (TRNG), yang membangkitkan angka benar-benar secara acak, contohnya sebuah *hardware random number generator*, atau *pseudo-random number generators* (PRNG), yang membangkitkan angka secara algoritmik, tetapi dapat mensimulasikan *randomness* sehingga angka yang dihasilkan terkesan dibangkitkan secara acak. Contoh dari pembangkit angka *random* PRNG adalah *Middle-square method* yang dikembangkan pada tahun 1946, hingga *Squares RNG* yang dikembangkan pada tahun 2020, sebagai salah satu perkembangan iteratif dari algoritma PRNG *Middle-square method* sebelumnya. [3]

Karena perannya yang krusial dan kegunaannya yang luas, maka konsep angka *random* dalam konteks kriptografi sangatlah penting untuk dipahami, sehingga untuk setiap jenis kriptografi, dapat digunakan angka *random* yang dibangkitkan dengan cara yang paling optimal dan menyediakan jaminan keamanan yang paling kuat.

II. Metodologi Penelitian

A. Metode Penelitian

Penelitian yang dilakukan pada penyusunan makalah ini memanfaatkan sumber-sumber yang berada dari internet, baik jurnal, studi ilmiah, buku, maupun sumber-sumber akademik lainnya yang terbuka untuk diakses oleh kalangan luas.

B. Batasan Penulisan

Angka *random* yang akan diteliti secara lebih lanjut pada penelitian ini hanyalah angka *random* dan aplikasinya dalam konteks kriptografi. Adapun pembahasan mengenai aplikasi angka *random* pada sarana lain, seperti gim, hanya akan digunakan sebagai pembanding.

III. Dasar Teori

A. Kriptografi

Kriptografi didefinisikan sebagai alat yang memiliki peran integral dalam konteks keamanan informasi digital. Kata "*cryptography*" sendiri memiliki asal usul dari bahasa Yunani, yang merupakan gabungan antara *cryptos* yang berarti

tersembunyi, dan *graphein* yang berarti tulisan, sehingga kriptografi memiliki arti sebagai "tulisan yang tersembunyi", atau dengan kata lain, ilmu untuk menjaga kerahasiaan dan keamanan pesan.[4]

Dalam konteks kriptografi, "keamanan" yang dimaksud mencakup beberapa aspek, yang juga dapat dijuluki sebagai layanan kriptografi. Aspek-aspek tersebut ialah kerahasiaan pesan (*Confidentiality*), keaslian pesan (*Data Integrity*), keaslian baik pengirim maupun penerima pesan (*Authentication*), dan anti penyangkalan (*Non-repudiation*).

1. *Confidentiality*

   Confidentiality, atau kerahasiaan, merupakan aspek yang menjelaskan mengenai penjagaan isi informasi dari sebuah pesan dari semua pihak lain yang tidak berkepentingan dengan pesan tersebut.

   Dalam konteks kriptografi, contoh dari penerapan aspek *confidentiality* adalah penerapan *cipher* seperti Vigenere Cypher yang merupakan teknik penyamaran pesan menggunakan suatu fungsi atau algoritma tertentu yang hanya diketahui oleh pihak yang berkepentingan, sehingga pihak yang tidak berwenang hanya akan melihat pesan yang telah dienkripsi oleh *cipher* sebagai kumpulan teks *gibberish* yang tidak berarti apa-apa.

2. *Data Integrity*

   Integritas data merupakan jaminan bahwa pesan yang dikirim tidak memiliki cacat atau kekurangan dalam bentuk apapun, baik dalam proses pengiriman, penyimpanan, maupun penerimaan, baik secara sengaja atau tidak.

   Dalam konteks kriptografi, integritas data dapat dijaga melalui beberapa upaya. Salah satu contohnya adalah menggunakan *hash functions*, yang mengubah sebuah pesan M menjadi sebuah *message digest* h(M). Pada umumnya, fungsi *hash* akan menghasilkan *message digest* h(M) berupa sebuah string yang bergantung pada isi pesan awal M. Akan tetapi, perubahan yang hanya berupa 1 huruf saja pada M dapat menyebabkan perubahan hasil h(M) yang drastis, sehingga mudah mencocokkan apabila terdapat pelanggaran integritas data pada pesan M.

3. *Authentication*

   Autentikasi merupakan proses verifikasi dari identitas pihak-pihak yang terlibat dalam sebuah proses komunikasi. Aspek autentikasi sangat penting untuk diperhatikan ketika hendak menjalin hubungan komunikasi yang aman, terutama melalui jaringan internet.

   Dalam konteks kriptografi, contoh upaya melakukan autentikasi adalah kriptografi kunci publik (*public key cryptography*), atau kriptografi asimetris, dimana sepasang kunci, yakni kunci publik dan kunci privat, dimanfaatkan untuk memverifikasi bahwa penerima layak membaca pesan yang telah dienkripsi.

4. *Non-Repudiation*

   Non-repudiasi dapat diartikan sebagai prevensi terjadinya penyangkalan atau keraguan akan sumber suatu pesan yang telah dikirim.

   Dalam konteks kriptografi, skenario yang dapat menggambarkan aspek ini adalah ketika seorang pengirim S hendak mengirimkan sebuah pesan M kepada penerima R, maka M akan dibubuhi oleh identitas S, dan apabila pada kemudian hari S menyangkal bahwa S bukan penulis pesan M, maka R dapat menunjukkan identitas S sebagai bukti bahwa benar S yang telah mengirimkan pesan tersebut. Hal tersebut merupakan proses yang terjadi dalam penandatanganan dan verifikasi berupa *digital signature*, yang memanfaatkan kunci privat dan kunci publik dari pengirim. [5]

B. *Random Number Generation*

Random Number Generation didefinisikan sebagai sebuah proses dimana serangkaian angka dibangkitkan secara acak, dengan sifat bahwa angka ini sulit ditebak dan tidak memiliki pola yang jelas antara angka hasil pembangkitan satu dengan lainnya, sehingga layak disebut *random*.

Terdapat beberapa jenis *random number generators* yang dapat menghasilkan angka *random* dengan metode yang berbeda-beda. Contoh dari metode-metode ini adalah *true random number generators*, dan *pseudo-random number generators*.

1. *True Random Number Generators*

   True number random generators (TRNG) dapat didefinisikan sebagai pembangkit angka acak yang membangkitkan angka yang secara acak dengan cara menangkap atau memroses fenomena alam yang terjadi secara acak. Salah satu contoh dari atmospheric noise, yakni sinyal elektromagnetis yang terjadi pada atmosfer Bumi akibat fenomena seperti radiasi termal atau sinar kosmik. [6]

2. *Pseudo-Random Number Generators*

Pseudo-random number generators (PRNG) dapat didefinisikan sebagai algoritma atau program yang membangkitkan angka acak dengan bantuan sebuah algoritma dan bersifat deterministik, berbeda dengan TRNG yang membangkitkan angka dengan bantuan fenomena alam yang benar-benar *random*. Pada umumnya, terdapat empat periode dalam proses pembangkitan angka melalui PRNG, yakni inisialisasi, dimana PRNG diinisialisasi dengan bantuan sebuah *seed*, yang kemudian diaplikasikan kepada sebuah algoritma matematis, yang menghasilkan angka *random* pertama. Angka *random* ini kemudian akan digunakan kembali sebagai *seed* untuk pembangkitan angka-angka *random* berikutnya.

Dalam kriptografi, jenis-jenis PRNG yang dapat membangkitkan angka *random* antara lain adalah *block ciphers* dan *stream ciphers*. [7]

IV. Pembahasan

Pada bab ini, akan dikumpulkan hasil-hasil penelitian mengenai angka *random* dan RNG dalam konteks kriptografi, kemudian akan dilakukan analisis lebih lanjut bagi hasil-hasil tersebut.

A. Pembangkitan *Pseudo-Random* untuk *Digital Signature Standard* (DSS)

Digital Signature Standard (DSS) merupakan spesifikasi untuk tanda tangan digital yang dicetuskan oleh National Institute of Standards and Technology (NIST) yang menspesifikasikan algoritma dan protokol yang digunakan untuk membangkitkan dan memverifikasi tanda tangan digital.

Pada penelitiannya, Mihir Bellare et.al. [8] menganalisis implementasi DSS dengan pembangkitan angka *random* menggunakan PRNG bernama *Linear Congruential Generator* (LCG), yang bersifat mudah diimplementasi dan efisien secara komputasional, namun memiliki tingkat keamanan yang tidak terlalu tinggi. Peneliti tersebut melakukan kriptanalisis pada sistem DSS yang memakai algoritma LCG untuk membangkitkan angka *random* nya untuk melihat apabila angka *random* hasil algoritma LCG tidak aman, dengan ciri memiliki pola yang mudah ditebak.

Security concern yang terdapat pada LCG adalah aspek *unpredictability* nya yang rendah. Pada dasarnya, LCG memiliki rumus dasar sebagai berikut:

$$X_{n+1} = (aX_n + c) \mod m$$

where X is the sequence of pseudorandom values, and

$m,\ 0 < m$ – the "modulus"
$a,\ 0 < a < m$ – the "multiplier"
$c,\ 0 \leq c < m$ – the "increment"
$X_0,\ 0 \leq X_0 < m$ – the "seed" or "start value"

Gambar 1. Rumus LCG (stockexchange.com)

Karena fungsi LCG yang sederhana, LCG tidak memerlukan daya komputasi yang kuat, dan memiliki sifat-sifat statistik yang cukup bagus apabila nilai a,b, dan m yang dipilih sesuai. Namun, apabila beberapa nilai Xi diketahui, penelitian dari Plumstead telah menemukan bahwa pola bilangan yang dihasilkan dapat diketahui [9]. Salah satu solusinya adalah modifikasi LCG sebagai bentuk *truncated*, namun modifikasi tersebut masih memiliki banyak lubang keamanan yang membuatnya tidak layak digunakan dalam konteks kriptografi.

Mengetahui hal-hal tersebut, meskipun LCG memiliki sifat yang cukup *predictable*, hal tersebut tidak langsung mengindikasikan bahwa algoritma kriptografi yang didasarkan atas LCG selalu mudah dipecahkan, karena mungkin saja bagi angka-angka yang digunakan dalam algoritma tidak dipublikasikan dan dijaga kerahasiaannya. Implementasinya pada DSS memenuhi kondisi tersebut.

Untuk menguji ketahanan (*robustness*) dari sistem, dilakukan penyerangan pada titik lemah LCG yakni persamaan *sk - rx = m mod q* untuk setiap tanda tangan digital *(r, s) = DSA(x, k, m)* yang dihasilkan dari algoritma DSA. Peneliti mengasumsikan, bahwa apabila penyerang menerima dua pesan m1 dan m2, dengan tanda tangan digital masing-masing berupa *(rl, st) = DSA(x, k1, m1)* dan *(r2, s2) = DSA(x, k2, m2)*, maka diketahui bahwa *s1k1 -r1x =m1 mod q* and *s2k2 -r2x = m2 mod q*. Sehingga, diketahui m1, r1, s1, m2, r2, dan s2. Karena diketahui pula parameter publik *p, q*, dan *g*, aspek yang tidak diketahui hanyalah kunci rahasia penulis *z*, serta k1 dan k2 yang digunakan untuk menghasilkan tanda tangan tersebut.

Dengan menggabungkan persamaan-persamaan di atas, terdapat tiga buah persamaan yang muncul dan menghasilkan sistem persamaan modular bagi variabel-variabel yang tidak diketahui, berupa:

$$\begin{cases} s_1 k_1 - r_1 x = m_1 & (\text{mod } q) \\ s_2 k_2 - r_2 x = m_2 & (\text{mod } q) \\ -a k_1 + k_2 = b & (\text{mod } M) \end{cases}$$

Gambar 2. Persamaan modular untuk memecahkan algoritma LCG.

Persamaan-persamaan tersebut dapat diselesaikan dengan beberapa cara untuk mendapatkan variabel terpenting yakni *secret key*. Metode-metode yang dapat digunakan adalah *integer programming* yang memanfaatkan algoritma waktu polinomial, akan tetapi metode ini cukup kompleks dan membutuhkan waktu komputasional yang tinggi. Alternatif lain adalah menggunakan *nearest lattice vector problem* (NLVP), yang mencari vektor *lattice* terdekat dengan vektor *lattice* target. Peneliti dapat memecahkan algoritma LCG dengan bantuan konsep NLVP.

Hasil tersebut tidak terbatas saja pada LCG, namun algoritma-algoritma PRNG yang memanfaatkan persamaan linier, seperti *Truncated LCG* dan *LCG with Concatenation*. Maka, dapat disimpulkan bahwa metode pembangkitan RNG menggunakan PRNG dengan algoritma yang hanya berbasiskan persamaan linier tidak cocok untuk digunakan dalam kriptografi karena tingkat keamanannya yang rendah dan mudah diretas. Algoritma LCG kini lebih sering diimplementasikan pada situasi dimana dibutuhkan tingkat *randomness*, namun aspek keamanan tidak terlalu penting, seperti simulasi atau gim.

B. Perbandingan PRNG dan Algoritma Kriptografi menggunakan *General Evaluation Pattern*

Pada penelitiannya, Abdolrasoul Mirghadri, et.al. [10], akan dilakukan analisis kepada angka *random* pada kriptografi, yang dikerucutkan pada aspek angka *pseudo-random*. Jenis pembangkit angka *random* yang dipilih adalah PRNG, *Block Cipher*, dan *Stream Cipher*.

Untuk menentukan algoritma yang paling mangkus dan aman, dilakukan beberapa tahap penelitian yang bersifat eliminatif, yang berarti akan terdapat metode yang tereliminasi untuk setiap tahap percobaan, hingga didapatkan satu metode terakhir yang dinilai paling efisien.

Pada tahap pertama, pengujian yang akan dilakukan adalah terhadap tingkat kecepatan komputasi algoritma. Hal ini merupakan aspek yang penting untuk diperiksa, namun terdapat beberapa konsiderasi juga dalam melakukan penelitian seperti medium pengujian, yakni *hardware* atau *software*. Sebagai contoh, algoritma DES memiliki kecepatan yang lebih tinggi pada *hardware* daripada *software*.

Berikut merupakan hasil perbandingan kecepatan algoritma hasil penelitian tahap pertama.

TYPE of ALGORITHM	NAME of ALGORITHM	SPEED QUALITY	PASS or REJECT
Block Cipher	AES256	Good	PASS
Block Cipher	DES	Good	PASS
Block Cipher	CAST-256	Slow	FAIL
Block Cipher	SKIPJAC	Slow	FAIL
Stream Cipher	F-FCRS-8	Good	PASS
Stream Cipher	Frogbit	Good	PASS
Stream Cipher	RC4-like	Good	PASS
Stream Cipher	Salsa20	Good	PASS
Stream Cipher	Trivium	Good	PASS
PRNG	CCCBG	Good	PASS
PRNG	MT19937	Good	PASS
PRNG	PHP-MT	Good	PASS
PRNG	Standard C LCG	Good	PASS

Gambar 3. Perbandingan kecepatan algoritma.

Seperti yang dapat dilihat pada tabel, terdapat beberapa algoritma yang gagal melalui tes kecepatan tersebut, seperti algoritma CAST-256 dan SKIPJAC, dengan kualitas kecepatan yang rendah. Alhasil, algoritma-algoritma tersebut tereliminasi.

Kemudian, dilakukan pengujian terhadap ketahanan (*robustness*) akan algoritma terhadap serangan berupa *search attack*. *Search attack* sendiri merupakan serangan berupa *brute force*, dimana penyerang akan mencoba semua input yang mungkin untuk dapat memecahkan algoritma terkait. Dalam kriptografi, ketahanan sebuah algoritma kriptografi dibasiskan atas asumsi bahwa secara komputasional, akan tidak mungkin bagi seorang penyerang untuk mencoba semua kombinasi kunci yang mungkin dalam waktu yang masuk akal. Dalam *search attack*, asumsi tersebut diabaikan dan ketahanan algoritma akan diuji secara sistematis dan menyeluruh.

Proses *search attack* memiliki sebuah ketergantungan, yakni *keyspace* atau jumlah input yang mungkin. Apabila *keyspace* bernilai kecil, maka akan lebih mungkin bagi *search attack* untuk sukses. Akan tetapi, semakin besar nilai *keyspace*, maka akan semakin sulit dilakukan serangan *search attack*. Berikut merupakan hasil pengujian untuk ketahanan algoritma.

TYPE of ALGORITHM	NAME of ALGORITHM	Key Space	PASS or FAIL
Block Cipher	AES256	2^{256}	PASS
Block Cipher	DES	2^{56}	FAIL
Stream Cipher	F-FCRS-8	2^{128}	PASS
Stream Cipher	Frogbit	2^{128}	PASS
Stream Cipher	RC4-like	variable > 2^{128}	PASS
Stream Cipher	Salsa20	2^{256}	PASS
Stream Cipher	Trivium	2^{80}	PASS
PRNG	CCCBG	variable > 2^{128}	PASS
PRNG	MT19937	variable > 2^{128}	FAIL
PRNG	PHP-MT	variable > 2^{128}	PASS
PRNG	Standard C LCG	variable > 2^{128}	PASS

Gambar 4. Ketahanan algoritma terhadap *search attack*.

Pengujian dilakukan secara iteratif terhadap berbagai aspek lainnya, seperti *Iterative Algorithm Law Test*, *The First, Second, and Third Level of NIST*, dan *LIL Test*.

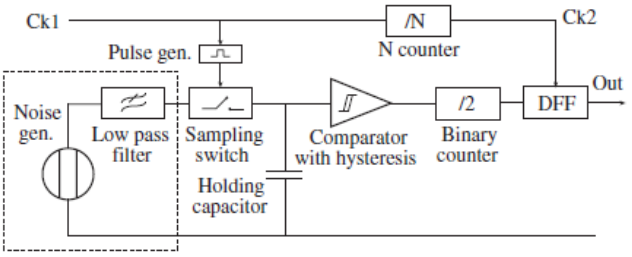
Dari 13 jenis algoritma yang diuji, hasil akhir yang didapatkan adalah satu algoritma dari masing-masing kategori, yakni AES256 untuk *Block Cipher*, Salsa20 untuk *Stream Cipher*, dan MT19937 untuk PRNG, yang berhasil melalui semua aspek yang diuji pada penelitian. Dapat disimpulkan, bahwa terdapat banyak sekali aspek yang dapat menentukan ketahanan, efisiensi, maupun keamanan dari sebuah algoritma kriptografi, dan semua aspek tersebut harus diperiksa secara menyeluruh, untuk mendapatkan algoritma-algoritma dengan kualitas dan performa terbaik.

C. Pengaplikasian TRNG yang dapat diandalkan bagi Kriptografi

Pada dua penelitian sebelumnya, jenis RNG yang diteliti cenderung lebih mengarah pada PRNG. Secara konsep, PRNG memang merupakan jenis pembangkitan angka *random* yang bersifat deterministik dan menggunakan algoritma untuk mensimulasikan efek *randomness*, sehingga tidak sebenarnya bernilai acak. Pada bagian ini, akan dilakukan analisis bagi salah satu penelitian yang memanfaatkan *true random number generator* (TRNG), yang seharusnya secara konsep memiliki nilai acak yang sebenarnya, dalam konteks kriptografi.

Pada umumnya, TRNG jarang digunakan dalam konteks kriptografi karena beberapa alasan. Pertama, karena TRNG bergantung pada fenomena alam, kecepatan algoritma-algoritma berbasis TRNG rata-rata lebih rendah daripada PRNG, yang bergantung pada algoritma yang sudah ditentukan. Karena sifatnya yang non-deterministik pula, angka hasil TRNG sulit untuk dihasilkan kembali (*reproduce*), yang akan berpengaruh pada konteks kriptografi, dimana reproduktifitas dibutuhkan untuk fungsi seperti pembangkitan kunci dan verifikasi. Kurangnya standarisasi akan TRNG yang dibasiskan atas fenomena alam dan vulnerabilities terhadap serangan fisik juga berkontribusi akan kurangnya penggunaan TRNG dalam konteks kriptografi. Untuk saat ini, TRNG memiliki peran khusus pada kriptografi sebagai pelengkap mekanisme kriptografi, seperti untuk menambah lapisan keamanan dimana angka yang benar-benar *random* dibutuhkan ketika ingin membangkitkan beberapa jenis kunci yang bersifat kriptografis. [6]

Pada penelitiannya, Fondazione Bordoni, et.al. [11] mencoba untuk mendesain sebuah TRNG yang layak digunakan pada konteks kriptografi. Mereka juga mengakui pentingnya pembangkitan angka yang benar-benar bersifat *random* dari aspek keamanan, sekaligus mengakui kekurangan TRNG dari aspek-aspek seperti yang sudah dijelaskan sebelumnya. Salah satu kekurangan tambahan yang dijelaskan adalah dari segi *bit sequence*, dimana sumber yang bersifat *truly random* sering menghasilkan *bit sequence* yang ciri-ciri statistiknya bergantung pada cara implementasi algoritma.



Gambar 5. Desain blok sirkuit yang diusulkan untuk menguji TRNG.

Penelitian yang dilakukan tersebut memproposisikan sebuah sirkuit yang menghasilkan *bit sequence* yang bersifat *unbiased*, serta memiliki ketahanan yang tinggi pada komponen-komponen sirkuitnya akan fluktuasi-fluktuasi yang mungkin terjadi. Setelah dilakukan pengujian, prototipe dianggap berhasil dari segi *bit sequence* yang dihasilkan, dimana hasil *bit sequence* tersebut juga dapat dan mudah untuk diatur lebih lanjut.

KESIMPULAN

Angka *random* memiliki peran yang sangat penting dalam kriptografi, terutama dalam proses-proses seperti pembangkitan kunci. Semakin tinggi nilai *unpredictability* dari sebuah algoritma RNG, maka tingkat keamanannya akan semakin baik, karena akan semakin sulit bagi calon penyerang untuk mengetahui bagaimana angka tersebut terbentuk. Contoh algoritma dengan nilai *unpredictability* rendah adalah algoritma berbasis PRNG dengan fungsi linier seperti LCG

atau *Truncated LCG*, sedangkan algoritma dengan nilai *unpredictability* yang tinggi adalah algoritma TRNG.

Meskipun TRNG memiliki nilai *unpredictability* yang tinggi, TRNG sendiri memiliki banyak kekurangan ketika digunakan dalam konteks kriptografi. Kekuatan terbesarnya juga menghambatnya dalam aspek lain, secara spesifik reproduktifitas, yang merupakan aspek yang penting ketika ingin melakukan pembangkitan kunci pada kriptografi.

Tak hanya aspek *unpredictability* yang penting dalam pembangkitan sebuah angka *random*, tetapi terdapat aspek-aspek lain pula yang harus diteliti, seperti kemampuan algoritma untuk menghasilkan nilai acak secara statistik dan kecepatan komputasi algoritma.

Maka dari itu, tidak ada satu algoritma RNG yang dapat memenuhi semua kriteria sebagai algoritma yang efisien dan fungsional, dan harus dilakukan analisis bersifat *case-by-case* untuk mengetahui jenis algoritma RNG yang dapat digunakan ketika ingin mengimplementasikan sistem kriptografi. Untuk itu, dibutuhkan pemahaman mendasar yang kuat akan RNG dan kegunaannya.

Penghargaan

Penulis mengucapkan syukur kepada Tuhan yang Maha Esa, karena telah memberikan kesempatan bagi penulis untuk menyusun makalah ini. Penulis juga ingin mengucapkan terima kasih kepada keluarga dan teman yang telah memberi dukungan selama proses penyusunan makalah.

Penulis ingin mengucapkan syukur pula kepada Bapak Dr. Ir. Rinaldi Munir, M.T. atas ilmunya yang telah diberikan selama proses belajar mengajar pada mata kuliah II4031 Kriptografi dan Koding selama satu semester.

Referensi

[1] P. Panagiotou, et.al. "*Cryptographic system for data applications, in the context of internet of things*.". Microprocess Microsyst, 2020.

[2] Milanov, Evgeny. "*The RSA Algorithm*". 2009.

[3] Widynski, Bernard. "*Squares: A Fast Counter-Based RNG*". 2022.

[4] Munir, Rinaldi. "Pengantar Kriptografi". Institut Teknologi Bandung, 2023.

[5] Lindell, Yehuda dan Katz, Jonathan. "*Introduction to Modern Cryptography*". Washington DC, 2007.

[6] Stipcevic, Mario. "*True Random Number Generators*". 2014.

[7] Oishi, Shin'ichi dan Inoue, Hajime. "*Pseudo-Random Number Generators and Chaos*". 1990.

[8] Bellare, Mihir, et.al. "*"Pseudo-Random" Number Generation Within Cryptographic Algorithms: The DDS Case*". MIT.

[9] Plumstead, J. "*Inferring a sequence generated by a linear congruence*". 1982.

[10] Mirghadri, Abdolrasoul, et.al. "*Comparing Some Pseudo-Random Number Generators and Cryptography Algorithms Using a General Evaluation Pattern*". 2016.

[11] Bordoni, Fondazione, et.al. "*A Design of Reliable True Random Number Generator for Cryptographic Applications*".

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, 22 Mei 2023

Samuel Gondokusumo 18219024