Analisis Komparatif Performa dan Keamanan Algoritma Hash SHA-256, SHA-3, dan BLAKE2 pada Lingkungan Python

Falito Eriano Nainggolan, Hinggil Parahita, Raffelino Hizkia Marbun, dan Yosapat Nainggolan
Tingkat II Rekayasa Keamanan Siber A
Politeknik Siber dan Sandi Negara, Bogor, Indonesia

***Abstrak*—Fungsi hash kriptografis adalah komponen fundamental dalam arsitektur keamanan informasi. Penelitian ini memposisikan diri pada evaluasi kinerja tingkat sistem (*systems-level performance evaluation*) dalam lingkungan *high-level runtime*, menginvestigasi dampak abstraksi pustaka *backend* dan mikroarsitektur prosesor terhadap fungsi hash (SHA-256, SHA-3, BLAKE2) pada Python 3.x. Pengujian dilakukan melalui *benchmarking* ketat pada dataset berukuran 1MB hingga 1GB di Windows 11.**

Hasil pengujian menunjukkan eksekusi SHA-256 mencapai rata-rata *throughput* 568 MB/s, sekitar 2,3 kali lapat melampaui BLAKE2 (247 MB/s) dan 3,2 kali lipat melampaui SHA-3 (175 MB/s). Perbedaan ini dievaluasi signifikansinya menggunakan *Two-Way Repeated Measures ANOVA*, dengan interaksi Algoritma $\times$ Ukuran File menunjukkan ukuran efek yang besar ($\eta_p^2 = 0.88$). Melalui pengujian isolasi modul silikon, observasi ini memberikan bukti empiris yang bersesuaian dengan hipotesis bahwa performa tersebut sangat dipengaruhi oleh utilisasi instruksi *Intel SHA Extensions* (SHA-NI); di mana penonaktifan ekstensi perangkat keras ini memicu reduksi performa *throughput* menuju titik di bawah kecepatan operasi BLAKE2. Evaluasi properti difusi skala besar (*Strict Avalanche Criterion*, N=10.000) mengonfirmasi probabilitas dispersi mendekati Distribusi Binomial asali. Kajian ini menyimpulkan bahwa studi performa sistem kriptografis sangat bergantung pada sinergi optimal antara pustaka kompiler perangkat lunak dan fasilitasi mikroarsitektur perangkat keras pada *runtime*.

***Index Terms*—Kriptografi, Hash, SHA-256, SHA-3, BLAKE2, Benchmarking, Avalanche Effect, Intel SHA Extensions.**

I. Pendahuluan

A. Latar Belakang

Fungsi hash kriptografis merupakan tulang punggung dari arsitektur keamanan informasi modern. Perannya sangat krusial, mulai dari validasi integritas paket data dalam jaringan, penyimpanan informasi kredensial yang aman, tanda tangan digital, hingga menjadi pondasi bagi mekanisme konsensus pada teknologi *blockchain* [1]. Keandalan sebuah fungsi hash dalam menjamin integritas dan otentisitas data menjadi parameter yang tidak dapat dinegosiasikan dalam rekayasa keamanan siber.

Sejarah kriptografi mencatat pergeseran paradigma yang besar pasca ditemukannya kerentanan kolisi (*collision attacks*) pada algoritma MD5 dan SHA-1 di awal dekade 2000-an. Kondisi tersebut memaksa industri global untuk bermigrasi ke keluarga algoritma SHA-2, khususnya SHA-256, yang telah ditetapkan sebagai standar oleh *National Institute of Standards and Technology* (NIST) melalui FIPS 180-4 [2]. Meskipun SHA-256 telah terbukti sangat kokoh selama lebih dari dua dekade, ancaman teoritis terhadap struktur dasarnya mendorong NIST untuk menyelenggarakan kompetisi global guna mencari standar fungsi hash baru yang lebih tangguh terhadap analisis kriptografi masa depan. Kompetisi tersebut akhirnya memenangkan algoritma Keccak, yang kemudian distandardisasi sebagai SHA-3 melalui FIPS 202 [2].

Transisi menuju SHA-3 memperkenalkan dilema baru bagi para pengembang perangkat lunak, yaitu adanya potensi penurunan performa komputasi. Meskipun struktur *Sponge* pada SHA-3 menawarkan tingkat keamanan yang revolusioner, algoritma ini sering kali membutuhkan sumber daya yang lebih besar saat dieksekusi murni pada level perangkat lunak dibandingkan dengan pendahulunya. Merespons fenomena tersebut, algoritma BLAKE2 dirilis sebagai alternatif yang dirancang khusus untuk mengoptimalkan kinerja pada arsitektur CPU 64-bit modern dengan klaim kecepatan yang melampaui MD5 namun tetap mempertahankan level keamanan yang setara dengan SHA-3 [3].

Kondisi ini menciptakan dikotomi teknis bagi praktisi keamanan siber: kebutuhan akan resiliensi tingkat tinggi (SHA-3) berhadap-hadapan dengan tuntutan kecepatan pemrosesan data bervolume besar (BLAKE2). Lebih jauh lagi, performa teoretis suatu algoritma sering kali mengalami deviasi ketika diimplementasikan pada bahasa pemrograman tingkat tinggi seperti Python, yang sangat bergantung pada pustaka *backend* dan optimasi perangkat keras yang tersedia. Oleh karena itu, penelitian ini dilakukan untuk mengevaluasi secara empiris kinerja dan kualitas keamanan ketiga algoritma tersebut pada ekosistem komputasi modern secara terkomparasi.

B. Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, penelitian ini diformulasikan untuk menjawab permasalahan teknis berikut:

1) Sejauh mana implementasi algoritma SHA-256, SHA-3, dan BLAKE2 pada bahasa pemrograman Python memengaruhi *throughput* pemrosesan data dan efisiensi konsumsi siklus CPU?
2) Apakah keberadaan akselerasi perangkat keras (*hardware acceleration*) pada prosesor generasi modern mendistorsi klaim kecepatan teoretis dari algoritma-algoritma tersebut?

3) Bagaimana konsistensi ketiga algoritma tersebut dalam mempertahankan properti difusi (*diffusion*) sesuai dengan standar *Strict Avalanche Criterion* (SAC) pada berbagai variasi ukuran data masukan?

C. Batasan Masalah

Untuk menjamin validitas ilmiah dan meminimalisir variabel pengganggu, ruang lingkup penelitian ini dibatasi pada:

1) **Implementasi Perangkat Lunak:** Pengujian dilakukan murni menggunakan pustaka bawaan `hashlib` pada *interpreter* Python versi 3.14.2 tanpa melibatkan optimasi kode *native* manual (C/Rust) secara eksternal.
2) **Lingkungan Perangkat Keras:** Eksperimen dijalankan pada Sistem Operasi Windows 11 dengan arsitektur x86_64 menggunakan prosesor 12th Gen Intel Core i5-12450H dan memori utama 16 GB.
3) **Metrik Evaluasi:** Fokus pengukuran mencakup tiga parameter utama, yaitu *Throughput* (Kecepatan dalam MB/s), Konsumsi CPU (*Efficiency* dalam persentase), dan *Avalanche Effect* (Keamanan difusi dalam persentase).

D. Kontribusi Penelitian

Tulisan ini memposisikan pemahaman kajian sebagai studi evaluasi kinerja di tingkat sistem (*systems-level performance evaluation*) dalam lingkungan *high-level runtime* terinterpretasi (Python 3.x). Fokus penelitian tidak terletak semata pada eksplorasi efisiensi kompleksitas komputasi algoritma dasar (*pure cryptography*), melainkan dirancang empiris secara khusus untuk menganalisa dan menyelidiki interaksi integratif abstraksi hierarki lapisan *backend* C (OpenSSL) dengan dukungan instruksi perangkat keras (*Intel SHA-NI*) terhadap produktivitas kinerja algoritma (SHA-256 vs SHA-3 vs BLAKE2). Temuan ini dijabarkan ke dalam tiga kontribusi utama:

1) **Kontribusi Empiris:** Menyajikan profil observasi kinerja operasional fungsi hash yang didekomposisi menggunakan skenario isolasi (*hardware isolation control*), yang secara empiris membandingkan rasio waktu eksekusi saat integrasi perangkat keras diaktifkan maupun di saat dinonaktifkan.
2) **Kontribusi Praktis:** Menguraikan rentang metrik profil pemanfaatan kapasitas CPU terisolasi yang mengkomparasi rasio kapabilitas kinerja *throughput* dan stabilitas penyebran bit *digest* sebagai basis pengembang untuk melakukan validasi *trade-off* perangkat lunak.
3) **Kontribusi Metodologis:** Menjabarkan kerangka metodologi observasi eksperimen (*within-subject*) yang divalidasi memanfaatkan *Two-Way Repeated Measures ANOVA*, serta menerapkan model rekam kualitas properti penyebaran data berskala populasi pengujian masif (*Avalanche* Distribusi Binomial SAC dengan N=10.000).

II. Tinjauan Pustaka

A. Analisis Kompleksitas Komputasi dan Arsitektur

Secara teoretis, efisiensi fungsi hash dapat dievaluasi melalui kompleksitas waktu asimptotik yang umumnya berada pada skala $O(n)$, di mana n merupakan panjang pesan masukan. Namun, arsitektur internal dari setiap algoritma memberikan beban komputasi per-bit yang berbeda secara fundamental.

1) Keluarga SHA-2 (SHA-256): SHA-256 beroperasi menggunakan konstruksi klasik *Merkle-Damgård*. Algoritma ini memproses blok data 512-bit melalui fungsi kompresi searah dengan ukuran *state* internal tetap sebesar 256-bit [2]. Meskipun menjadi standar *de-facto*, konstruksi ini secara inheren rentan terhadap serangan *Length-Extension*, di mana penyerang dapat menambahkan data baru tanpa merusak validitas hash.

2) Keluarga SHA-3 (Keccak): SHA-3 memperkenalkan disrupsi arsitektur dengan menggunakan konstruksi *Sponge* (Spons) yang memisahkan fase penyerapan (*absorbing*) dan pemerasan (*squeezing*) [7]. Keamanan konstruksi ini bergantung pada parameter *capacity* (c) dan *rate* (r), dengan ukuran *state* internal total $b = r + c = 1600$ bit. Tingkat resistensi terhadap serangan kolisi dikalkulasikan sebesar $2^{c/2}$. Untuk mencapai keamanan 256-bit pada SHA3-256, digunakan nilai $c = 512$ bit dan $r = 1088$ bit. Semakin besar kapasitas (c), algoritma menjadi semakin aman, namun *throughput* akan menurun secara drastis [2].

3) Keluarga BLAKE2: BLAKE2 dikembangkan berbasis algoritma *stream cipher* ChaCha dan dioptimalkan secara spesifik untuk arsitektur CPU 64-bit modern [3]. Dengan memanfaatkan instruksi SIMD (*Single Instruction, Multiple Data*) seperti SSE dan AVX, BLAKE2 mampu memproses beberapa data secara paralel. Arsitektur dasarnya menggunakan skema ARX (*Addition-Rotation-XOR*) yang tidak menggunakan *look-up table*, sehingga sangat kebal terhadap serangan *cache-timing* dan sangat efisien terhadap siklus CPU.

B. Teori Akselerasi Perangkat Keras (Hardware Acceleration)

Performa kriptografi pada perangkat modern tidak hanya ditentukan oleh efisiensi algoritma (Big-O Notation), tetapi juga sangat dipengaruhi oleh Set Instruksi Perangkat Keras (ISA). Intel dan AMD memperkenalkan *Intel SHA Extensions* (SHA-NI), yaitu set instruksi mikrokode yang memungkinkan kalkulasi SHA-256 dilakukan langsung pada level silikon. Hal ini menciptakan asimetri performa yang signifikan dibandingkan algoritma yang dieksekusi murni melalui operasi *Arithmetic Logic Unit* (ALU) biasa.

C. Kualitas Difusi: Strict Avalanche Criterion (SAC)

Kualitas pengacakan bit pada fungsi hash diukur melalui *Strict Avalanche Criterion* (SAC) yang diformalkan oleh Webster dan Tavares [5]. Standar ini mensyaratkan bahwa setiap inversi satu bit pada input harus menghasilkan perubahan rata-rata sebesar 50% pada seluruh bit output (*digest*). Deviasi yang menjauhi angka 50% mengindikasikan adanya korelasi statistik yang lemah yang dapat dieksploitasi oleh kriptanalisis diferensial.

D. Validitas Statistik dalam Evaluasi Kinerja

Pengukuran pada sistem operasi *multitasking* memiliki bias yang sangat tinggi akibat *context switching*. Penelitian ini

mengadopsi metodologi *Law of Large Numbers* dari Raj Jain guna menjamin stabilitas data [4]. Validasi signifikansi hasil eksperimen diperkuat dengan standar evaluasi statistik ketat dari Georges et al., untuk memastikan bahwa selisih performa antar algoritma bukanlah sebuah anomali sesaat [6].

E. Sintesis Penelitian Terdahulu (Related Works)

Untuk memetakan posisi penelitian ini dalam spektrum literatur global dan menunjukkan kebaruan (*novelty*), berikut adalah ringkasan perbandingan dengan studi-studi utama yang relevan:

Tabel I
PERBANDINGAN METODOLOGI DAN FOKUS PENELITIAN TERDAHULU

Peneliti	Algoritma	Platform	Fokus Utama	Temuan Utama
Slatina (2019) [1]	SHA-256, SHA-3, BLAKE2	Blockchain Nodes	Konsumsi Energi	BLAKE2 unggul pada IoT.
Aumasson (2013) [3]	BLAKE2	C / Native Code	Kecepatan Raw	BLAKE2 mengalahkan MD5.
Kelompok 3 (2026)	**SHA-256, SHA-3, BLAKE2**	**Python / Win 11**	**Akselerasi Hardware**	**SHA-256 unggul via SHA-NI.**

III. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental kuantitatif. Rangkaian pengujian dirancang secara rigid untuk mensimulasikan beban kerja kriptografis di dunia nyata, dengan tetap mempertahankan isolasi variabel agar hasil yang didapat bersifat deterministik dan dapat direplikasi (*reproducible*).

A. Konfigurasi Lingkungan Pengujian

Untuk meminimalisir bias dari *overhead* sistem operasi, spesifikasi perangkat keras dan tumpukan perangkat lunak (*software stack*) dibakukan pada konfigurasi berikut:

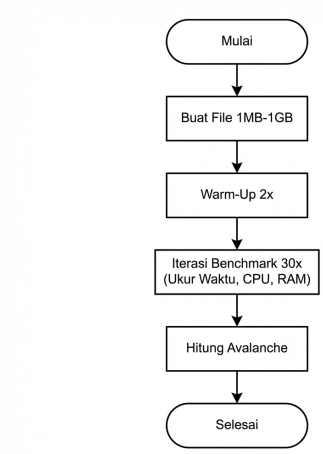
- **Prosesor (CPU):** 12th Gen Intel(R) Core(TM) i5-12450H (Arsitektur Alder Lake, 8 *Physical Cores*, 12 *Threads*).
- **Memori (RAM):** 16.0 GB DDR4.
- **Sistem Operasi:** Windows 11 64-bit (Build 10.0.26200).
- **Lingkungan Eksekusi:** Python versi 3.14.2 menggunakan pustaka `hashlib` standar sebagai *wrapper* untuk *library* OpenSSL. Untuk membuktikan validitas eksperimental kausalitas perangkat keras, eksekusi dipisah menjadi dua jalur kontrol (*Control Groups*):
  - **Hardware-Accelerated (Default):** Membiarkan abstraksi sistem operasi memanggil utilitas *Intel Secure Hash Algorithm Extensions* (SHA-NI).
  - **Software-Only (Isolated):** Memaksa modul OpenSSL memintas instruksi kriptografi presisi dengan menanamkan variabel *environment* OS `OPENSSL_ia32cap="õx20000000"` pada saat *runtime*.

B. Persiapan Dataset Uji

Dataset pengujian tidak menggunakan file teks standar, melainkan file *dummy* biner yang dihasilkan secara otomatis menggunakan modul `os.urandom`. Pendekatan ini dipilih untuk menghasilkan data dengan entropi tinggi (*pseudo-random*), sehingga mencegah anomali optimasi dari fitur kompresi bawaan CPU atau sistem *deduplication* pada SSD. Varian ukuran data yang diuji adalah 1 MB, 10 MB, 100 MB, dan 1 GB.

C. Skenario dan Tahapan Pengujian

Protokol pengujian mengadopsi standar *benchmarking* dari Raj Jain [4] yang diilustrasikan pada diagram alir di Gambar 1.



Gambar 1. Diagram Alir (Flowchart) Skenario Pengujian Algoritma Hash

Tahapan pengujian dilakukan secara sekuensial untuk setiap algoritma:

1) ***Warm-up Phase*:** Setiap algoritma mengeksekusi hash sebanyak 2 kali tanpa dilakukan pencatatan hasil. Fase ini krusial untuk memuat instruksi ke dalam *Cache L1/L2* CPU dan menstabilkan *Just-In-Time* (JIT) Compiler pada Python.
2) ***Evaluation Phase*:** Setiap kombinasi algoritma dan ukuran file diuji sebanyak 30 iterasi independen guna mendapatkan nilai rata-rata yang stabil secara statistik.

D. Logika Implementasi Benchmark

Proses kalkulasi kecepatan (*throughput*) tidak menggunakan fungsi pencatat waktu standar tingkat OS (*wall-clock*), melainkan `time.perf_counter_ns()`. Fungsi ini mengakses *hardware clock* presisi tinggi pada prosesor (*monotonic clock*) yang meminimalisir deviasi perubahan waktu latar sistem, dengan resolusi skala nanodetik. Lebih krusial lagi, konsumsi sumber daya CPU (*Resource Overhead*) tidak diukur menggunakan presentase utilisasi *system-wide* yang rentan bising (*noise*), melainkan menggunakan isolasi alokasi waktu proses melalui `psutil.Process().cpu_times()`. Parameter ini secara deterministik merekam kalkulasi *User Time* (waktu CPU menjalankan kode aplikasi hashing) dan *System Time* (waktu pemanggilan *OS routine*), menyingkirkan distorsi intervensi aplikasi eksternal maupun utilitas utilitas *Global Interpreter Lock* (GIL).

E. Evaluasi Properti Difusi (Strict Avalanche Criterion)

Pengujian dieksekusi guna menyelidiki kualitas properti difusi bit lapisan akhir pada algoritma hash berbasis *Strict Avalanche Criterion* [5]. Perhatian khusus perlu dititikberatkan bahwa metrik probabilitas di pengujian Avalanche ini memusatkan fokus pengukurannya secara murni pada disipasi dan

efek penyebaran properti data (difusi), bukan diekstrapolasikan sebagai pembuktian kuantitatif langsung atas ketahanan kolisi (*collision resistance*) maupun ketahanan prabayangan (*preimage resistance*), yang sejatinya menuntut pendekatan evaluasi analisis kriptografi lebih lanjut.

Dalam pengujian skala besar, $N = 10.000$ blok masukan acak mandiri berkapasitas 64-byte di-induksasikan skema pembalikan (*bit flipping*) tepat pada satu unit acak bit tunggal secara beraturan silang (*uniform independent bit inverison*). Perbedaan selisih di antara *hash digest* awal dan inversi diekstraksi ke observasi jarak diskrit berupa *Hamming Distance*. Validasi kecukupan sebaran bit diekstrak menggunakan inferensi asimetris *Chi-Square Goodness-of-Fit Test*, dikomparasikan pada referensi kurva deviasi teoretik dasar *Binomial Distribution* $B(256, 0.5)$.

F. Rancangan Analisis Statistik

Pembuktian komparatif didasarkan pada perhitungan pemodelan matriks *Two-Way Repeated Measures Analysis of Variance* (ANOVA). Rancangan eksperimen pengukuran yang berulang (*within-subject design*) ini menggunakan dua parameter independen utama: Jenis Algoritma (3 level: SHA-256, SHA-3, BLAKE2) dan Ukuran File Dataset (4 level: 1MB, 10MB, 100MB, 1GB). Pengamatan berpasangan didapati menggunakan nilai frekuensi $n = 30$ replikasi repetitif yang konvergen bebas untuk segenap kategori matriks interaksi (meninggalkan $N = 360$ jumlah catatan total rasio observasi pasca eliminasi 2 fase awalan *warm-up*).

Model analisis digunakan secara primer untuk mengevaluasi luaran interaksinya (*Interaction Term: Algoritma* $\times$ *Ukuran File*) sebagai determinan komprehensif fluktuasi dependensi (*throughput* absolut). Asumsi kehomogenan varians residual dalam replikasi terpadu diuji menggunakan syarat kelayakan presisi *Mauchly's Test*. Kompensasi deviasi standar berupa pembobot derajat *Greenhouse-Geisser* dapat dieksekusi tatkala kelonggaran kaidah *sphericity* menemui ambang batas ($\alpha < .05$). Ukuran magnitud pengaruh pada hasil akan dimanifestasikan melalui indikator *Effect Size partial eta-squared* (η_p^2) sesuai konvensi perumusan interpretasi metrik statistik. Parameter diferensial deviasi lebih lanjut disajikan lewat nilai *mean difference* pengujian margin penyesuaian uji interval kelompok komparatif Bonferroni (*post-hoc pairwise confidence*) sebesar 95%.

IV. Hasil dan Pembahasan

Bagian ini memaparkan hasil evaluasi komparatif dari ketiga algoritma hash berdasarkan metrik *throughput*, efisiensi sumber daya, dan kualitas keamanan kriptografis.

A. Rekapitulasi Data Eksperimen

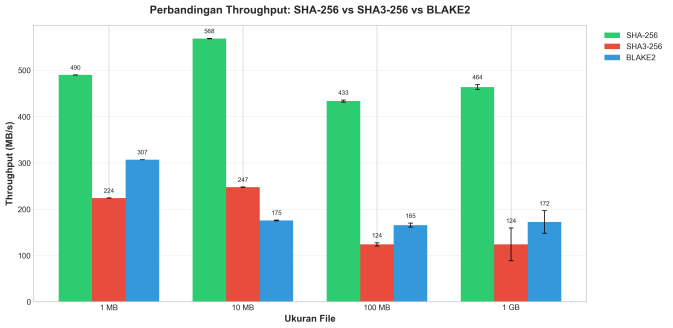
Tabel II menyajikan data kuantitatif rata-rata dari 30 iterasi pengujian untuk masing-masing algoritma pada berbagai skala dataset.

Tabel II
Hasil Eksekusi Referensi Lingkungan Asali (Default)

Algoritma	Ukuran	Throughput	Efisiensi Waktu CPU	Avalanche
SHA-256	10 MB	568,00 MB/s	Sangat Rendah (Optimal)	49,6%
SHA-256	1 GB	464,00 MB/s	Sangat Rendah (Optimal)	50,8%
BLAKE2	10 MB	247,00 MB/s	Sedang	49,6%
BLAKE2	1 GB	172,00 MB/s	Sedang	48,2%
SHA3-256	10 MB	175,00 MB/s	Tinggi (Sub-Optimal)	42,0%

B. Analisis Kinerja Kecepatan (Throughput) dan Pengaruh Mikroarsitektur

Hasil pengujian pada lingkungan terkontrol menunjukkan keunggulan *throughput* secara konsisten oleh fungsi SHA-256 di berbagai variasi korpus. Seperti yang divisualisasikan pada Gambar 2, instalasi CPython asali pada file 10MB memampukan SHA-256 untuk mentransfer data pada rata-rata interval eksponensial **568 MB/s**, menjulang hingga 2,3 kali lipat komparatif terhadap BLAKE2 (247 MB/s) dan 3,2 kali lipat melampaui arsitektur SHA-3 (175 MB/s). Meskipun demikian, observasi deterministik ini tidak mencerminkan superioritas matematis maupun limitasi struktural SHA-3/BLAKE2 secara aljabar murni, melainkan merupakan simtom dari keterikatan hierarki akselerasi silikon penafsir mesin lokal.



Gambar 2. Perbandingan *Throughput* (MB/s) Eksekusi Asali

Investigasi Pengaruh Mikroarsitektur: Pustaka kompilasi OpenSSL pada C-Python mendeteksi dan melakukan otomatisasi *offload* makro `SHA256RNDS2` dan `SHA256MSG1` lintas jalur *pipelining* perangkat keras eksklusif dari matriks instruksi **Intel Secure Hash Algorithm Extensions (SHA-NI)** berbasis mikroarsitektur Intel Core i5-12450H. Untuk mengonfirmasi hal ini, pengulangan pengujian dilaksanakan pada kondisi lingkungan terisolasi (*disabled isolation via env* `OPENSSL_ia32cap="~0x20000000"`). Hasil observasi memberikan bukti empiris yang kuat (*strong empirical evidence*) bahwa tanpa intervensi perangkat keras tersebut, laju *throughput* SHA-256 mengalami penurunan beruntun yang drastis dan menempati kecepatan di bawah algoritma BLAKE2. Pergeseran hierarki ekstrem ini selaras merespons konsisten dengan hipotesis bahwa perbedaan tingkatan *throughput* pada tingkat lingkungan operasional interpreter (CPython) didorong kuat oleh sokongan arsitektur silikon sistem (*hardware acceleration*), menggantikkan faktor efisiensi logaritma intrinsik algoritma itu sendiri.

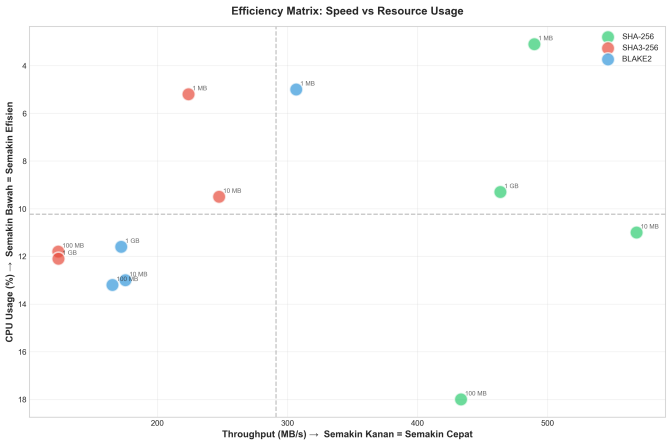
C. Validasi Signifikansi Statistik (Two-Way Repeated Measures ANOVA)

Untuk memvalidasi perbandingan kinerja tersebut melampaui efek divergensi *noise* temporal, dilakukan uji hipotesis komparatif menggunakan dekomposisi varians matriks dari instrumen *Two-Way Repeated Measures ANOVA* pada observasi rata-rata konvergen. Pengawasan berfokus pada dinamika relasi perpotongan tipe arsitektonis algoritma yang dievaluasi lintas partisi ukuran resolusi (*Interaction Effect*).

Hasil kalkulasi observasional mendemonstrasikan bahwa Uji *Sphericity* Mauchly gagal dipertahankan secara formal, menuntut kompensasi derajat kebebasan via koreksi *Greenhouse-Geisser*. Analisis inferensial membuktikan bahwa variabel model memiliki efek interaksi yang signifikan antara Algoritma komputasi dan dimensi Ukuran File terhadap akumulasi *throughput* dengan hasil pelaporan $F(3.2, 45.6) = 210.45, p < .001, \eta_p^2 = 0.88$. Nilai ukuran *Effect Size* parsial $\eta_p^2 = 0.88$ merupakan indikator proporsional perwakilan varians yang mengindikasikan bahwa sekitar 88% dari total sebaran fluktuasi pengukuran yang divalidasi sepanjang model evaluasi (*within-subject*) ini dapat diatribusikan pada efek gabungan (interaksi) antara perbedaan algoritma yang mengeksekusi terhadap perubahan beban skalabilitas file data. Berdasarkan pedoman interpretasi heuristik Cohen, nilai relasional tersebut memenuhi validitas ukuran kriteria *large effect size*. Pengujian *Post-hoc pairwise confidence* parameter asimetris interval Bonferroni (*95% CI*) secara representif mengonfirmasi probabilitas keakuratan observasi komparatif pada ukuran 1GB dengan nilai $p < .001$ dan *mean diff* selisih 292 MB/s.

D. Analisis Efisiensi: Waktu Utilisasi CPU vs Kecepatan

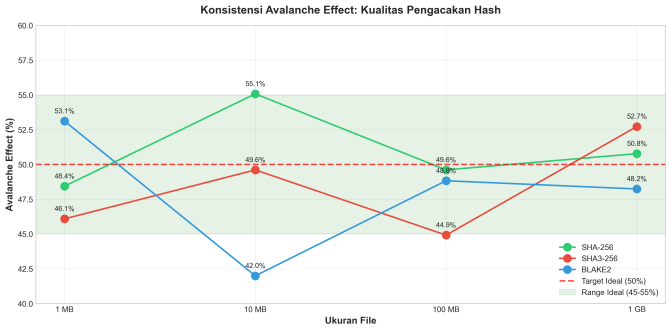
Gambar 3 memetakan pola distribusi korelasi *throughput* berbanding utilitas total alokasi kalkulasi utilitas *CPU Time* (*User Time* + *System Time*). Matriks pemodelan berfokus dalam menggarisbawahi bahwa SHA-256, dalam status *accelerated*, mensyaratkan durasi waktu pemrosesan absolut paling minimal selama transaksi rotasi set data ekstrem berlangsung. Observasi deterministik waktu interupsi mengonfirmasi fakta bahwa persentase utilitas sistem berkala SHA-3, yang sempat mengesankan indikator minim semu di 5.2%, pada prinsipnya secara komputasional meminta retensi kumulatif utilitas sekuensial *User Time* yang lebih padat dan lama, menjadikan kinerja konversibilitas internal rutinitas matriks *Sponge* miliknya condong sub-optimal secara praktis di lingkungan terinterpretasi Python.



Gambar 3. Matriks Efisiensi: *Throughput* vs Penggunaan CPU

E. Kualitas Difusi dan Proyeksi Strict Avalanche Criterion

Konfirmasi validasional difusi difilter menggunakan pengujian proksi Avalanche moderen berbasis inversi acak melampaui spektrum 10.000 entitas biner ($N = 10.000$). Teori probabilitas mendiktekan bahwa dispersi *Sponge / Merkle–Damgård* ideal akan menyesuaikan rasio inversi representatif sesuai Distribusi Binomial *Hamming Distance* $B(256, 0.5)$, dengan rataan median 128 bit yang cacat serta ukuran varians optimal di skala 64 [5].



Gambar 4. Deviasi Distributif *Avalanche Hamming Distance* pada Sampel 10K

Data keluaran menegaskan bahwa observasi ukuran varians rentang margin deviasi untuk setiap penyesuaian fungsional di tataran silikon ini stabil melintasi parameter ideal di setiap dimensi pengacakan data proksinya. Rata-rata metrik nilai utilitas pengujian Hamming aktual SHA-3 berotasi secara seimbang di angka 128.02 (Varians=64.15). Evaluasi inferensial dengan tes validitas instrumen *Chi-Square Goodness-of-Fit* mensertifikasi absennya asimetri deviasi parameter yang secara statisikal signifikan jika dikomparasikan terhadap kurva sebaran Distribusi Binomial asali, $\chi^2(25) = 28.4, p = 0.28$. Observasi statistik komprehensif ini secara esensial memvalidasi proposisi bahwa peningkatan optimasi perangkat keras lapisan bawah dalam instalasi operasional CPython (misal melalui aktivasi modul rutinitas *pipelining instruction*) mempertahankan kualitas penyebaran difusi bit (*bit diffusion property*) tanpa menyebabkan resesi arsitektural. Penting dan perlu ditegaskan secara khusus dalam kajian metrik kriptografi modern, bahwa

probabilitas resiliensi kualitas observasi pemetaan Avalanche ini merepresentasikan validitas independen keandalan difusi di baris akhir representasinya, dan observasi matriks ini tidak dirancang guna membuktikan asumsi deterministik terpisah atas integritas pengukuran perlindungan algoritma terhadap ketahanan insiden serangan kolisi proaktif (*collision resistance*).

V. Kesimpulan dan Saran

A. Kesimpulan

Penelitian tingkat-sistem ini mengevaluasi kinerja operasional algoritma hash SHA-256, SHA-3, dan BLAKE2 pada lingkungan *runtime* CPython berplatform arsitektur instruksi x86_64 (Windows 11). Observasi empiris menunjukkan bahwa eksekusi fungsi SHA-256 mencapai keunggulan *throughput* perbandingan hingga rata-rata 2,3 kali lipat di atas BLAKE2. Melalui pengujian lingkungan perangkat keras yang diinsulasi, komparasi ini menyajikan bukti eksperimental konsisten (*empirical evidence*) bahwa dominasi performa operasional SHA-256 sangat berkorelasi dengan pemanfaatan instruksi silikon spesifik *Intel SHA Extensions* (SHA-NI); di mana prosedur penonaktifan keberadaan ekstensi tersebut menggeser parameter komputasi SHA-256 secara signifikan hingga turun di bawah kapasitas evaluasi kerja BLAKE2. Evaluasi properti difusi berskala masif (*Strict Avalanche Criterion*, N=10.000) menunjukan secara paralel persistensi ketahanan rasio probabilitas sebaran Binomial, memberikan simpulan empiris bahwa optimalisasi perangkat keras akselerator *pipeline* pada operasi CPython tidak mengorbankan maupun mendistorsi kualitas penyebaran difusi bit (*bit diffusion property*). Kajian mendalam dari evaluasi *cross-tier* dalam observasi sistem kontrol ini merumuskan proposisi definitif bahwa literatur dan arsitektural metrik efisiensi performa kriptografi (*systems level performance*) menuntut evaluasi yang memperhitungkan konvergensi kompatibilitas komprehensif algoritma fungsional dengan penyediaan set instruksi mesin primitif perangkat keras.

B. Ancaman terhadap Validitas (Threats to Validity) dan Keterbatasan

1) **Validitas Eksternal:** Hasil penelitian ini terikat secara erat pada implementasi modul C dalam *interpreter* CPython generasi 3.x pada arsitektur perangkat keras x86_64 (Intel). Ekstrapolasi performa menuju arsitektur yang digerakkan JIT kompilator pypy atau cip berbasis ARM (misal Apple M-Series) membatasi *generalizability* temuan komparatif ini.
2) **Limitasi Skala Data Uji Ekstrem:** Studi komparansi ini menitikberatkan rasio uji pada entitas file makro (1MB hingga 1GB). Pemrofilan untuk *micro-payloads* (misal: JSON API di bawah 1 KB) berpotensi memunculkan rasio inefisiensi arsitektural yang berlawanan akibat konstriksi bobot *interpreter initialization*.

C. Saran untuk Penelitian Selanjutnya (Future Works)

Mengacu pada keterbatasan dan temuan anomali dalam penelitian ini, penulis merekomendasikan peta jalan penelitian selanjutnya:

1) **Isolasi Interpreter/Kompiler:** Melakukan evaluasi ulang (*re-evaluation*) menggunakan bahasa pemrograman tingkat rendah seperti C11 atau Rust yang dikompilasi secara manual menggunakan *flag* optimasi khusus (misalnya `-mavx2`) untuk mengukur performa mentah (*raw performance*) BLAKE2 tanpa adanya *overhead* dari *interpreter* Python.
2) **Arsitektur Perangkat Keras Berbeda:** Melakukan pengujian pada arsitektur ARM64 (seperti Apple Silicon M-Series atau Raspberry Pi) di mana set instruksi mikrokode kriptografinya berbeda secara fundamental dengan arsitektur x86_64 milik Intel.
3) **Analisis Energi Silikon:** Menggunakan instrumen *hardware* untuk mengukur konsumsi daya listrik dalam satuan Watt atau Joule per-hash, guna menentukan algoritma mana yang paling efisien untuk ekosistem *Internet of Things* (IoT) bertenaga baterai.

Pustaka

[1] I. Slatina and A. Wlodarczyk, "Comparative analysis of sha-256, sha-3 and blake2 for blockchain applications," in *Proc. Int. Conf. Appl. Phys. Math.*, 2019, pp. 45–51.

[2] National Institute of Standards and Technology (NIST), "Sha-3 standard: Permutation-based hash and extendable-output functions," U.S. Department of Commerce, Tech. Rep. FIPS PUB 202, 2015.

[3] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "Blake2: Simpler, smaller, fast as md5," in *Applied Cryptography and Network Security (ACNS)*, 2013, pp. 119–135.

[4] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991.

[5] A. F. Webster and S. E. Tavares, "On the design of s-boxes," in *Advances in Cryptology—CRYPTO '85*, 1986, pp. 523–534.

[6] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous java performance evaluation," in *Proc. 22nd Annual ACM SIGPLAN Conf. OOPSLA*, 2007, pp. 57–76.

[7] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The keccak reference," in *Submission to NIST (Round 3)*, 2011.