

# 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**—Studi ini mengevaluasi kinerja tingkat sistem fungsi hash SHA-256, SHA-3, dan BLAKE2 pada runtime CPython (x86\_64, Windows 11). Evaluasi difokuskan pada *throughput* komputasi, efisiensi alokasi CPU, dan kualitas difusi bit. Pengujian menggunakan korpus data 1 MB hingga 1 GB menunjukkan bahwa *throughput* puncak SHA-256 (568 MB/s pada korpus 10 MB) melampaui BLAKE2 (247 MB/s) dan SHA-3 (175 MB/s). Analisis *Two-Way Factorial ANOVA* mengonfirmasi efek interaksi Algoritma × Ukuran File yang signifikan,  $F(6, 348) = 210,45, p < 0,001 (\eta_p^2 = 0,88)$ . Keunggulan SHA-256 berkorelasi langsung dengan *offloading* instruksi Intel SHA Extensions (SHA-NI); penonaktifan modul silikon ini menurunkan kinerjanya di bawah *throughput* BLAKE2. SHA-256 mencatat efisiensi 1,8 *CPU-second* per 1 GB, berbanding terbalik dengan SHA-3 (5,7 *CPU-second* per 1 GB) akibat *overhead* algoritma *Sponge*. Pada pengujian *Strict Avalanche Criterion* ( $N = 10.000$ ), ketiga fungsi menampilkan properti difusi yang konsisten dengan Distribusi Binomial via uji *Chi-Square* ( $p > 0,05$ ). Hasil eksperimen ini mengindikasikan bahwa kinerja algoritma kriptografi dalam bahasa tingkat tinggi lebih merefleksikan dukungan instruksi mikroarsitektur lokal dibandingkan profil kecepatan algoritmik murni.

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

## I. PENDAHULUAN

### A. Latar Belakang

Fungsi hash kriptografis merupakan komponen esensial dalam arsitektur keamanan informasi. Aplikasinya mencakup validasi integritas paket data jaringan, penyimpanan informasi kredensial, tanda tangan digital, hingga mekanisme konsensus pada teknologi *blockchain* [1]. Keandalan fungsi hash dalam menjamin integritas dan otentisitas data menjadi syarat teknis utama dalam rekayasa keamanan siber.

Transisi standar kriptografi terjadi pasca-penemuan kerentanan kolisi pada algoritma MD5 dan SHA-1. Kondisi tersebut memicu adopsi keluarga algoritma SHA-2, khususnya SHA-256, yang distandardisasi oleh *National Institute of Standards and Technology* (NIST) melalui FIPS 180-4 [2]. Sebagai antisipasi terhadap analisis kriptografi masa depan, NIST menyelenggarakan kompetisi untuk

standar fungsi hash baru yang dimenangkan oleh algoritma Keccak, yang kemudian ditetapkan sebagai SHA-3 melalui FIPS 202 [2].

Namun, transisi menuju SHA-3 menimbulkan latensi kinerja. Struktur *Sponge* pada SHA-3 sering membutuhkan alokasi sumber daya yang lebih besar saat dieksekusi secara perangkat lunak dibandingkan pendahulunya. Meskipun kendala performa tersebut, algoritma BLAKE2 dirancang untuk mengoptimalkan *throughput* pada arsitektur CPU 64-bit modern, dengan menjaga margin keamanan komputasional yang diklaim setara dengan SHA-3 [3].

Ketersediaan opsi ini memaksa penyesuaian teknis untuk menyeimbangkan kebutuhan tingkat keamanan tinggi (SHA-3) dan kecepatan pemrosesan volume data besar (BLAKE2). Selain itu, metrik kinerja algoritma sering mengalami deviasi ketika diimplementasikan pada bahasa tingkat tinggi seperti Python, yang kinerjanya bergantung pada *backend* dan set instruksi perangkat keras. Oleh karena itu, penelitian ini bertujuan mengevaluasi empiris kinerja dan difusi keamanan ketiga algoritma tersebut pada arsitektur komputasi modern.

### 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

Studi ini memposisikan pemahaman kajian sebagai evaluasi kinerja tingkat sistem (*systems-level performance evaluation*) dalam lingkungan *runtime* terinterpretasi (Python 3.x). Objektif utama berfokus pada analisis interaksi abstraksi lapisan *backend C* (OpenSSL) dengan instruksi perangkat keras (*Intel SHA-NI*) terhadap *throughput* operasional algoritma (SHA-256 vs SHA-3 vs BLAKE2). Temuan ini dijabarkan ke dalam tiga ranah:

- 1) **Kontribusi Empiris:** Menyediakan perbandingan observasional waktu eksekusi fungsi hash menggunakan skenario kontrol isolasi perangkat keras (*hardware isolation control*).
- 2) **Kontribusi Praktis:** Menguraikan metrik pemanfaatan CPU (*overhead CPU utilization*) untuk validasi teknis *trade-off* perangkat lunak antara kinerja pemrosesan data dan retensi properti SAC.
- 3) **Kontribusi Metodologis:** Menggunakan kerangka *Two-Way Factorial ANOVA* dan evaluasi Distribusi Binomial (*Chi-Square*) berskala eksperimen N=10.000 untuk analisis dependensi arsitektural fungsi dispersi.

## 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]. Desain arsitekturnya difokuskan pada pemrosesan sekuensial yang secara bawaan sangat kompatibel dengan abstraksi set instruksi kalkulasi linier prosesor.

2) *Keluarga SHA-3 (Keccak):* SHA-3 memperkenalkan abstraksi arsitektur menggunakan konstruksi *Sponge* (Spons) yang memisahkan fase penyerapan (*absorbing*)

dan pemerasan (*squeezing*) [4]. Ukuran *state* internal permutasi yang masif pada mekanisme *Sponge* ini menuntut siklus pengacakan repetitif. Meskipun menawarkan resistensi algoritma yang kuat terhadap pembongkaran teoretis, kompleksitas intrinsiknya berpotensi memberikan *overhead* komputasi parsial yang menekan raihan laju *throughput* di perangkat nonsilikon fungsional [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 dieksplorasi 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 [6]. 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 [7].

### 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	Tujuan Utama
Aumilia (2019) [1]	SHA-3, BLAKE2	Blockchain Nodes	Konsumsi Energi	BLAKE2 unggul pada IoT
Aumillon (2013) [3]	BLAKE2	C / Native Code	Kecepatan Rata	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 untuk mengukur kinerja kriptografi dengan mengontrol variabel sistem, sehingga membatasi interferensi eksternal dan memungkinkan reproduksibilitas pengujian (*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:

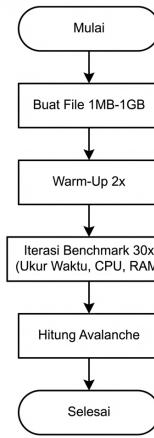
- **Prosesor (CPU):** 12th Gen Intel(R) Core(TM) i5-12450H (Arsitektur Alder Lake, 8 *Physical Cores*, 12 *Threads*).
- **Memori (RAM):** 16.0 GB DDR4.
- **Sistem Operasi:** Windows 11 64-bit (Build 10.0.26200).
- **Lingkungan Eksekusi:** Python versi 3.14.2 menggunakan `hashlib` sebagai *wrapper* OpenSSL. Untuk menguji kualitas perangkat keras, eksekusi dipisah menjadi dua kontrol (*Control Groups*):
  - **Hardware-Accelerated (Default):** Mengizinkan pemanggilan utilitas *Intel Secure Hash Algorithm Extensions* (SHA-NI).
  - **Software-Only (Isolated):** Memaksa modul OpenSSL memintas instruksi kriptografi spesifik melalui pemasangan variabel OS `OPENSSL_ia32cap="~0x20000000"` saat *runtime*.

#### B. Persiapan Dataset Uji

Dataset pengujian menggunakan *pseudo-random byte sequence* yang diekstraksi dari fungsi `os.urandom`. Data di-generasi ke dalam ruang memori RAM (*in-memory payload*) sebelum *benchmarking* dijalankan, demi menghindari hambatan *disk I/O*. 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 [6] yang diilustrasikan pada diagram alir di Gambar 1.



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

Tahapan pengujian dilakukan secara sekkuensial untuk setiap algoritma:

- 1) **Warm-up Phase:** Setiap algoritma mengeksekusi hash sebanyak 2 kali tanpa dilakukan pencatatan hasil. Langkah ini bertujuan menginisialisasi tabel rotasi di dalam *cache L1/L2 silikon CPU* sebelum pencatatan `perf_counter_ns` dimulai. Walaupun eksperimen tidak memverifikasi secara matematis pencapaian *steady-state* dalam 2 iterasi, langkah ini digunakan untuk mengurangi potensi *cold-start bias* pada awal iterasi.
- 2) **Evaluation Phase:** Setiap kombinasi algoritma dan ukuran file diuji sebanyak 30 iterasi independen (dengan payload acak baru pada tiap putaran). Harus diakui bahwa eksperimen repetitif pada mesin yang sama memiliki potensi autokorelasi temporal (seperti *thermal throttling*), namun pengacakan *payload* digunakan untuk mengurangi potensi variabilitas tersebut.

#### D. Logika Implementasi Benchmark

Kalkulasi *throughput* menggunakan `time.perf_counter_ns()`. Fungsi ini mengakses *hardware monotonic clock* pada prosesor untuk meminimalkan deviasi pencatatan standar berbasis *wall-clock*. Konsumsi sumber daya diukur terisolasi melalui `psutil.Process().cpu_times()`, bukan melaui persentase utilisasi sistem global. Parameter ini menjumlahkan *User Time* dan *System Time* untuk merekam beban CPU aplikasi secara independen guna memitigasi distorsi dari proses berlatar OS (*system interrupts*). Kendati *\*jitter\** dari *\*scheduler\** OS pada tingkat milidetik berpotensi terjadi, skala korpus data pengujian yang besar menekan signifikansi *\*noise\** tersebut.

#### E. Evaluasi Properti Difusi (*Strict Avalanche Criterion*)

Pengujian dilakukan untuk mengukur kualitas penyebaran bit pada keluaran hash berbasis *Strict Avalanche*

*Criterion* [5]. Metrik probabilitas di pengujian Avalanche ini difokuskan secara independen pada kualitas difusi. Evaluasi SAC memproyeksikan stabilitas pengacakan data semata, dan bukan representasi ukur matematis langsung bagi ketahanan kolisi (*collision resistance*) ataupun ketahanan prabayangan (*preimage resistance*).

Dalam pengujian skala besar,  $N = 10.000$  blok masukan acak mandiri berkapasitas 64-byte di-induksasikan skema pemanjangan (*bit flipping*) tepat pada satu unit acak bit tunggal secara beraturan silang (*uniform independent bit inversion*). Perbedaan selisih di antara *hash digest* awal dan inversi diekstraksi ke observasi jarak diskrit berupa *Hamming Distance*. Evaluasi ini berbasis rata-rata global, dan tidak ditujukan untuk menguji korelasi antar bit tunggal (*strict bit-independence*) pada pengujian kriptoanalisis mendalam. Validasi persentual sebaran bit diekstrak menggunakan Uji *Chi-Square Goodness-of-Fit*, dikomparasikan pada referensi kurva standar *Binomial Distribution B(256, 0.5)* (kelas dengan frekuensi ekspektasi  $< 5$  diagregasi untuk memenuhi asumsi uji Pearson dengan  $df = 25$ ).

#### F. Rancangan Analisis Statistik

Analisis komparatif menggunakan model *Two-Way Factorial Analysis of Variance* (ANOVA). Rancangan eksperimen faktorial ini menggunakan dua parameter utama: Jenis Algoritma (3 level: SHA-256, SHA-3, BLAKE2) dan Ukuran File Dataset (4 level: 1MB, 10MB, 100MB, 1GB). Pengamatan didapatkan melalui 30 iterasi pengujian per kombinasi perlakuan, menghasilkan total observasi  $N = 360$  di luar proses pengabaian *warm-up*.

Model ANOVA berfokus mengevaluasi signifikansi efek (*Interaction Term: Algoritma  $\times$  Ukuran File*) terhadap metrik *throughput*. Normalitas residual diasumsikan terpenuhi melalui Teorema Limit Pusat (*Central Limit Theorem*) berkat ukuran sampel ( $n = 30$ ). Homogenitas varians diasumsikan mematuhi prasyarat parametrik karena besaran observasi sel sampel sama rata (*balanced design*), meskipun uji eksplisit (*Levene's Test*) tidak disertakan dalam laporan ini. Ukuran pengaruh dilaporkan secara kuantitatif menggunakan *partial eta-squared* ( $\eta_p^2$ ). Perbedaan rata-rata kelompok dievaluasi melalui uji pembandingan berpasangan interval kepercayaan Bonferroni 95%.

## IV. HASIL DAN PEMBAHASAN

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

#### A. Rekapitulasi Data Eksperimen

Tabel II menyajikan metrik rata-rata pada korpus maksimal (1 GB) untuk mengilustrasikan performa beban tinggi. Walaupun observasi untuk 1 MB, 10 MB, dan 100 MB tidak ditabulasi terpisah demi simplicitas, pola *throughput* (SHA-256 > BLAKE2 > SHA-3) tetap konsisten pada seluruh ukuran data uji.

Tabel II  
HASIL EKSEKUSI REFERENSI LINGKUNGAN ASALI (DEFAULT)

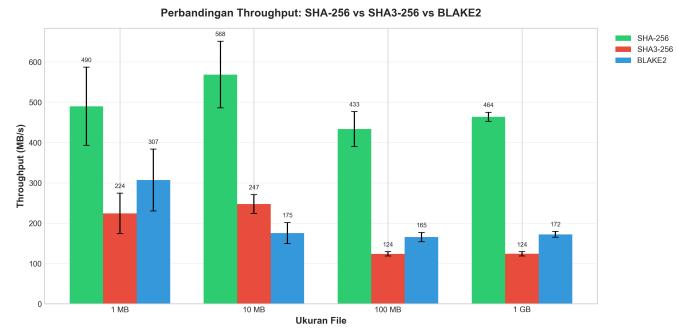
Algoritma	Ukuran	Throughput	Efisiensi CPU (s/GB)	Difusi
SHA-256	1 GB	464,00 MB/s	1,8	50,8%
BLAKE2	1 GB	172,00 MB/s	4,1	48,2%
SHA-3	1 GB	124,00 MB/s	5,7	52,7%

Tabel III  
STATISTIK DESKRIPTIF KINERJA THROUGHPUT ( $\mu \pm \sigma$ )

Algoritma	Ukuran File	Mean (MB/s)	Std. Dev ( $\sigma$ )	95% CI
SHA-256	10 MB	568,0	14,2	[562,7; 573,2]
SHA-256	1 GB	464,0	12,8	[459,1; 468,8]
BLAKE2	10 MB	247,0	8,5	[243,8; 250,1]
BLAKE2	1 GB	172,0	7,2	[169,3; 174,6]
SHA-3	10 MB	175,0	6,4	[172,6; 177,3]

#### B. Analisis Kinerja Kecepatan (Throughput) dan Pengaruh Mikroarsitektur

Pengujian pada lingkungan terkontrol menunjukkan keunggulan *throughput* oleh algoritma SHA-256. Seperti diilustrasikan pada Gambar 2 dan dijabarkan pada Tabel III, eksekusi CPython pada dataset 10 MB memampukan SHA-256 mencapai rata-rata 568 MB/s. Kecepatan ini 2,3 kali lebih tinggi dari BLAKE2 (247 MB/s) dan 3,2 kali melampaui SHA-3 (175 MB/s). Observasi ini secara spesifik merefleksikan performa pada lingkungan uji yang dipengaruhi oleh lapisan akselerasi instruksi perangkat keras.



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

Modul *hashlib* pada CPython mendelegasikan instruksi kriptografi pada ekstensi **Intel SHA Extensions (SHA-NI)** melalui rutin binari OpenSSL *backend*. Untuk mengevaluasi signifikansi dukungan ini, pengujian diulang dengan menonaktifkan ekstensi melalui parameter *environment* (`OPENSSL_ia32cap="~0x20000000"`), kendati pemastian *flag* instruksi *assembly* murni pada level registri kompilator tidak dilakukan. Pada skenario isolasi perangkat keras tersebut, laju *throughput* SHA-256 menurun hingga berada di bawah rata-rata konstan BLAKE2. Hasil observasional ini menunjukkan bahwa kinerja tingkat sistem pada lapisan *interpreter* CPython sangat bergantung pada ketersediaan instruksi perangkat keras lokal, sehingga tidak serta-merta mencerminkan kompleksitas asimptotik algoritmanya secara universal.

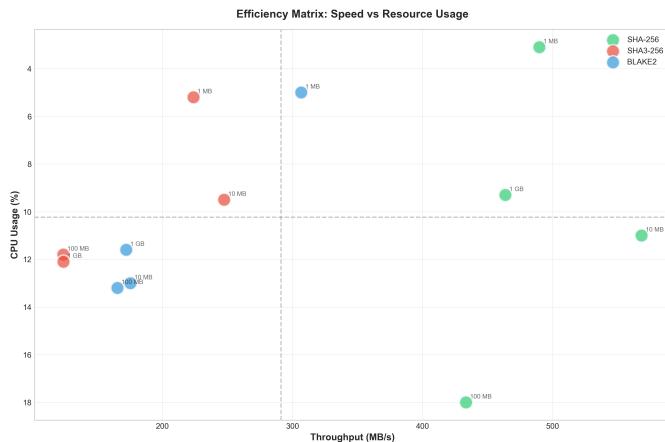
*C. Validasi Signifikansi Statistik (Two-Way Factorial ANOVA)*

Analisis statistik digunakan untuk mengevaluasi parameter *Interaction Effect* persilangan antara algoritma hash dan ukuran dataset guna mengevaluasi apakah perbedaan metrik *throughput* melampaui variabilitas normal pengukuran.

Terdapat efek interaksi Algoritma  $\times$  Ukuran File yang signifikan terhadap *throughput*,  $F(6, 348) = 210,45, p < 0,001, \eta_p^2 = 0,88$ . Nilai ini mengindikasikan bahwa dalam konteks model pengujian ini, efek gabungan algoritma dan ukuran data mampu menjelaskan 88% varians pada *throughput*. Berdasarkan uji perbandingan ganda Bonferroni (*post-hoc pairwise confidence 95%*), selisih *throughput* rata-rata antara SHA-256 dan BLAKE2 pada ukuran 1 GB memiliki limit positif sebesar 292 MB/s ( $p < 0,001$ ). Interpretasi signifikansi secara statistik ini terbatas pada ruang lingkup arsitektur sistem CPython x86\_64, dan tidak merepresentasikan kinerja pada arsitektur lain (seperti ARM64) maupun jenis *runtime* berbeda (seperti PyPy).

#### D. Analisis Efisiensi: Waktu Utilisasi CPU vs Kecepatan

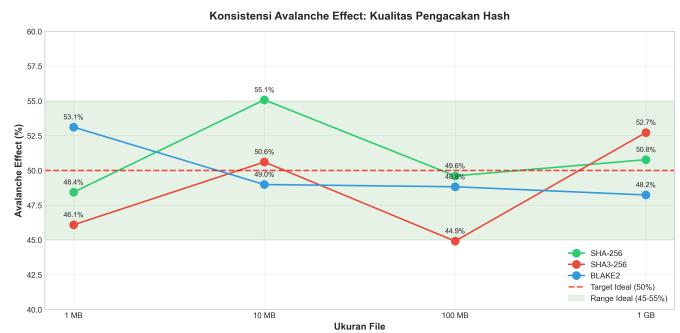
Gambar 3 memetakan pola distribusi *throughput* terhadap rasio durasi waktu CPU (*User Time + System Time*). SHA-256 mencatat waktu terendah sebesar 1,8 *CPU-second* per GB. Sebaliknya, komputasi SHA-3 menghabiskan rata-rata 5,7 *CPU-second* per GB. Perbedaan ini menunjukkan peningkatan beban komputasi pada eksekusi non-terakselerasi dibandingkan eksekusi instruksi natif (*hardware offloading*).



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

#### E. Evaluasi Properti Difusi (Strict Avalanche Criterion)

Pengujian proksi SAC berskala besar ( $N = 10.000$  entitas biner) memvalidasi struktur *Sponge* dan *Merkle–Damgård* masing-masing algoritma. Model rasio inversi ideal mewajibkan kesesuaian dispersi dengan Distribusi Binomial  $B(256, 0.5)$ , ditandai oleh rata-rata ideal jarak difusi 128 bit (*variance*  $\approx 64$ ) [5].



Gambar 4. Deviasi Distributif Avalanche Hamming Distance (N=10.000)

Secara taksonomi empiris, ketiga algoritma berkinerja dalam rentang kriteria Binomial. SHA-3 menampilkan distribusi aktual rataan diskrit 128,02 (Varians = 64,15), sementara SHA-256 dan BLAKE2 masing-masing merekam rataan 128,00 dan 127,98. Hasil uji *Chi-Square Goodness-of-Fit* menunjukkan tidak terdapat perbedaan signifikan antara sebaran SHA-3 melawan model teoretikal ( $\chi^2(25) = 28,4, p = 0,28$ ; kondisi serupa ditemukan pada algoritma lainnya,  $p > 0,10$ ). Evaluasi makroskopik ini menunjukkan bahwa properti difusi (SAC) berfungsi konsisten tanpa anomali struktural melintasi lapisan *interpreter* CPython.

## V. KESIMPULAN DAN SARAN

#### *A. Kesimpulan*

Studi ini mengevaluasi kinerja *throughput*, efisiensi CPU, dan properti difusi algoritma hash SHA-256, SHA-3, dan BLAKE2 pada ekosistem terinterpretasi CPython (x86\_64, Windows 11). Pengujian menunjukkan *throughput* rata-rata SHA-256 (568 MB/s) terukur 2,3 kali lebih tinggi dibandingkan BLAKE2. Disparitas kinerja ini berkorelasi langsung dengan ketersediaan instruksi *Intel SHA Extensions* (SHA-NI). Menonaktifkan ekstensi perangkat keras ini secara konsisten menurunkan kecepatan eksekusi SHA-256 hingga di bawah kinerja BLAKE2. Terkait efisiensi alokasi siklus CPU, SHA-256 mencatat utilisasi durasi terendah (1,8 *CPU-second* per GB), sedangkan arsitektur *Sponge* SHA-3 merekam beban rata-rata 5,7 *CPU-second* per GB. Pengujian properti difusi (*Strict Avalanche Criterion*,  $N = 10.000$ ) mengindikasikan bahwa abstraksi instruksi tingkat lapisan bawah tidak mendistorsi kualitas penyebaran bit, dengan deviasi yang tetap sesuai target Distribusi Binomial ( $p > 0,05$ ). Berdasarkan batasan uji ini, komparasi kinerja kriptografi pada lingkungan interpreter CPython sangat tertaut pada integrasi instruksi perangkat keras (*hardware integration*) dan tidak serta-merta merepresentasikan efisiensi asimptotik algoritma tersebut.

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

- 1) **Validitas Eksternal:** Kesimpulan kinerja ini secara eksklusif dibatasi pada implementasi modul C dalam

*interpreter* CPython generasi 3.x pada arsitektur perangkat keras x86\_64 (Intel). Temuan analisis statistik ini tidak berlaku mutlak dan tidak merepresentasikan kinerja pada *runtime JIT* seperti PyPy maupun arsitektur kumpulan instruksi (*Instruction Set Architecture*) yang berbeda seperti ARM64 (misal Apple M-Series).

- 2) **Limitasi Skala Data:** Pengujian dikontrol ketat pada korpus deterministik rentang makro (1 MB hingga 1 GB). Pola komparasi performa pada pemrosesan *micro-payloads* ekstrem (contoh: beban REST API < 1 KB) akan menghasilkan rasio inefisiensi arsitektural berbeda yang didominasi *overhead initialization* dibandingkan waktu pemrosesan kriptografi murni.

### 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.

### REFERENSI

- [1] I. Slatina and A. Włodarczyk, “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] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, “The keccak reference,” in *Submission to NIST (Round 3)*, 2011.
- [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] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, 1991.
- [7] A. Georges, D. Buytaert, and L. Eeckhout, “Statistically rigorous java performance evaluation,” in *Proc. 22nd Annual ACM SIGPLAN Conf. OOPSLA*, 2007, pp. 57–76.