Estimasi Konsumsi Energi *Bitcoin Mining* Berdasarkan Analisis Kombinatorik Ruang Pencarian dan Kompleksitas Algoritma Proof-of-Work

Muhammad Fauzi Muharam - 13525091
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: mfauzim10@gmail.com , 13525091@std.stei.itb.ac.id

***Abstrak*—Bitcoin telah menjadi aset digital yang diperdebatkan secara global. Namun, pertanyaan mendasar tentang seberapa besar energi yang dibutuhkan untuk memproduksinya masih jarang dijawab dari perspektif matematika diskrit. Makalah ini menganalisis mekanisme Proof-of-Work (PoW) pada Bitcoin mining menggunakan dua alat matematika diskrit, yaitu kombinatorika dan analisis kompleksitas algoritma. Melalui kaidah perkalian dan distribusi geometrik, diturunkan bahwa ekspektasi jumlah iterasi yang dibutuhkan untuk menemukan satu blok Bitcoin bernilai D dikali 2 pangkat 32, di mana D adalah tingkat difficulty. Kompleksitas algoritma PoW dianalisis sebagai O(D), dan hasilnya digunakan untuk mengestimasi konsumsi energi per blok. Eksperimen simulasi mini PoW pada skala kecil dengan variasi difficulty menunjukkan bahwa distribusi jumlah iterasi yang diperoleh konsisten dengan prediksi teoritis distribusi geometrik, dengan galat rata-rata di bawah 6 persen, dan ekstrapolasi ke skala Bitcoin menghasilkan estimasi yang konsisten dengan data Cambridge Bitcoin Electricity Consumption Index sebesar 10–20 GW. Hasil akhir menunjukkan bahwa solo mining dengan GPU konsumen membutuhkan ekspektasi waktu sekitar 306 juta tahun, sebuah kesimpulan yang dapat ditarik langsung dari analisis matematis.**

Kata Kunci— Bitcoin, Proof-of-Work, Kombinatorika, Distribusi Geometrik, Kompleksitas Algoritma, Estimasi Energi.

I. PENDAHULUAN

Pada tahun 2023, *Bitcoin* menjadi pusat perhatian dunia maya Indonesia. Fenomena tersebut membuat Bitcoin yang awalnya hanya sebuah teknologi yang hanya dikenal kalangan terbatas menjadi aset digital yang didiskusikan di tingkat global. Bersamaan dengan naiknya popularitas Bitcoin, fenomena naiknya harga Bitcoin yang terjadi berulang kali juga menarik jutaan orang untuk berinvestasi dan berspekulasi di pasar kripto. Di sisi lain, tidak sedikit orang yang harus menelan kerugian di tengah volatilitas pasar yang ekstrem. Pengalaman merugi tersebut justru memantik sebuah pertanyaan yang lebih menarik: Daripada membeli Bitcoin, bagaimana jika kita memproduksi sendiri melalui *mining Bitcoin*?

Bitcoin adalah *cryptocurrency* terdesentralisasi yang diperkenalkan oleh Satoshi Nakamoto pada tahun 2008 [1]. Berbeda dengan mata uang konvensional yang dicetak oleh bank sentral, Bitcoin "dicetak" melalui proses komputasi yang disebut *mining*. Dalam proses ini, penambang (*miner*) bersaing untuk menemukan sebuah nilai yang disebut ***nonce*** yang sedemikian rupa sehingga hasil fungsi hash SHA-256 dari data blok memenuhi syarat tertentu yang ditentukan oleh jaringan. Proses inilah yang dikenal sebagai mekanisme **Proof-of-Work** (**PoW**).

Pertanyaan yang kemudian muncul adalah: Seberapa sulit sebenarnya proses *mining* ini secara matematis, dan berapa besar energi yang harus dikeluarkan? Berbagai penelitian telah dilakukan untuk mengestimasi konsumsi energi Bitcoin dari data empiris jaringan [7]. Namun, pendekatan dari analisis matematika diskrit, khususnya melalui kombinatorika dan analisis kompleksitas algoritma masih jarang dijadikan fokus utama. Makalah ini hadir untuk mengisi celah tersebut.

Tentunya, karena keterbatasan sumber daya, penulis tidak dapat melakukan *mining* Bitcoin secara langsung. Sebagai gantinya, penulis membangun model matematis berbasis analisis kombinatorika, menganalisis kompleksitas algoritmik PoW, dan melakukan eksperimen simulasi pada skala kecil yang kemudian akan **diekstrapolasi** ke skala Bitcoin sesungguhnya untuk menghasilkan estimasi konsumsi energi. Secara spesifik, makalah ini berusaha menjawab pertanyaan berikut: Bagaimana pengaruh tingkat *difficulty* terhadap ekspektasi jumlah hash dan estimasi konsumsi energi yang diperlukan untuk menemukan satu blok Bitcoin?

II. LANDASAN TEORI

A. Bitcoin dan Blockchain

Bitcoin adalah *cryptocurrency* terdesentralisasi yang diperkenalkan oleh Satoshi Nakamoto pada tahun 2008 [1]. Berbeda dengan sistem keuangan konvensional yang bergantung pada otoritas terpusat seperti bank, Bitcoin beroperasi pada jaringan *peer-to-peer* yang memvalidasi transaksi secara kolektif. Fondasi dari sistem ini adalah struktur data yang disebut *blockchain*, yaitu sebuah rangkaian blok yang saling terhubung secara kriptografis.

Setiap blok dalam *blockchain* memiliki sebuah *block header* yang mengandung enam komponen utama: versi protokol, *previous block hash* (referensi ke blok sebelumnya), *merkle root* (ringkasan seluruh transaksi dalam blok), *timestamp*, *bits* (representasi ringkas dari nilai *target*), serta *nonce*. Struktur ini menjamin bahwa riwayat transaksi tidak dapat diubah tanpa mengubah seluruh rantai blok. Sifat inilah yang menjadi fondasi keamanan Bitcoin dan sekaligus yang membuat proses *mining* menjadi begitu krusial [1].

B. *Proof-Of-Work dan Nonce*

Untuk menambahkan blok baru ke *blockchain*, seorang *miner* harus membuktikan bahwa ia telah melakukan sejumlah besar pekerjaan komputasi. Mekanisme ini disebut *Proof-of-Work* (PoW) [1]. Yang secara konkret, *miner* harus menemukan nilai *nonce* n, yaitu bilangan bulat 32-bit yang dimasukkan ke dalam *block header* yang sedemikian rupa sehingga

$$SHA256(SHA256(block\_header \parallel n)) < target$$

Nilai *target* adalah sebuah bilangan bulat 256-bit yang ditentukan oleh jaringan secara berkala. Semakin kecil nilai *target*, semakin sedikit nilai hash yang memenuhi syarat, dan semakin sulit proses *mining*. Karena SHA-256 dirancang agar outputnya tidak dapat diprediksi dari inputnya, satu-satunya pendekatan yang mungkin adalah mencoba nilai *nonce* satu per satu hingga syarat terpenuhi. Proses pencarian inilah yang menjadi objek analisis kombinatorik dalam makalah ini.

C. *Fungsi Hash SHA-256*

SHA-256 (*Secure Hash Algorithm 256-bit*) adalah fungsi hash kriptografis yang menghasilkan output 256-bit dari input sembarang [2]. Dalam konteks *mining* Bitcoin, SHA-256 memiliki tiga properti krusial. Pertama, *pre-image resistance*: diberikan output y, secara komputasi tidak mungkin menemukan input x sehingga SHA256(x) = y. Kedua, *avalanche effect*: perubahan 1 bit pada input menghasilkan perubahan sekitar 50% bit pada output secara tidak terprediksi. Ketiga, *deterministic*: input yang sama selalu menghasilkan output yang sama.

Ketiga properti ini secara bersama-sama memastikan bahwa SHA-256 berperilaku seperti *random oracle*, yaitu sebuah model teoretis di mana setiap input yang berbeda menghasilkan output yang terdistribusi seragam dan tidak berkorelasi satu sama lain [2]. Konsekuensinya, tidak ada struktur aljabar yang dapat dieksploitasi untuk mengarahkan output hash menuju *target*. Dengan demikian, *brute-force* menjadi salah satu strategi yang valid, dan hal ini yang menjustifikasi penggunaan model probabilistik dalam analisis berikutnya.

D. *Difficulty dan Mekanisme Penyesuaiannya*

Tingkat *difficulty* D adalah bilangan positif yang merepresentasikan seberapa sulit proses *mining* relatif terhadap blok pertama Bitcoin (*genesis block*). Secara formal, D didefinisikan sebagai:

$$D = \frac{\text{target}_{\text{genesis}}}{\text{target}_{\text{current}}}$$

Jaringan Bitcoin menyesuaikan nilai *difficulty* setiap 2.016 blok, atau sekitar dua minggu, agar rata-rata waktu penemuan blok tetap mendekati 10 menit [1]. Mekanisme penyesuaiannya mengikuti relasi rekurens:

$$D\_baru \;=\; D\_lama \;\times\; (t\_aktual \;/\; 20.160\; menit)$$

Relasi ini merupakan bentuk relasi rekurens: nilai *difficulty* berikutnya bergantung pada nilai sebelumnya dan waktu yang dibutuhkan jaringan. Semakin banyak *miner* yang bergabung dan *hashrate* jaringan naik, *difficulty* ikut naik untuk menjaga keseimbangan sistem.

E. *Kombinatorika dan Distribusi Geometrik*

Kombinatorika adalah cabang matematika diskrit yang mempelajari jumlah penyusunan, pengaturan, dan pemilihan objek dari suatu himpunan, tanpa menjabarkan semua kemungkinannya [3]. Salah satu prinsip paling mendasar dalam kombinatorika adalah **kaidah perkalian**, yaitu, misalkan terdapat n percobaan yang saling bebas, di mana masing-masing percobaan ke-i memiliki p_i kemungkinan hasil. Berdasarkan kaidah perkalian (*rule of product*), total kemungkinan hasil dari keseluruhan n percobaan adalah:

$$p_1 \times p_2 \times \cdots \times p_n$$

Prinsip ini menjadi fondasi untuk menghitung ruang sampel dalam proses *mining*. Output SHA-256 adalah string sepanjang 256 bit, di mana setiap bit dapat bernilai 0 atau 1. Dengan kaidah perkalian, total kemungkinan output SHA-256 adalah:

$$|\,\mathcal{H}\,| = \underbrace{2 \times 2 \times \cdots \times 2}_{256\ kali} = 2^{256}$$

Selanjutnya, dari total 2^{256} kemungkinan output tersebut, hanya sebagian yang memenuhi syarat *mining*, yaitu hash yang nilainya lebih kecil dari target. Banyaknya output yang valid adalah sebesar nilai target itu sendiri. Dengan kaidah pembagian, peluang satu percobaan hash menghasilkan output yang valid adalah:

$$p = \frac{\text{banyak output valid}}{\text{total output}} = \frac{\text{target}}{2^{256}}$$

Dari nilai p ini, pertanyaan berikutnya adalah: berapa banyak percobaan yang dibutuhkan hingga pertama kali berhasil? Untuk menjawab ini, perlu dipahami terlebih dahulu konsep **percobaan Bernoulli**, yaitu eksperimen yang hanya memiliki dua kemungkinan hasil: berhasil dengan peluang p, atau gagal dengan peluang $1 - p$ [3]. Setiap percobaan hash dalam PoW memenuhi definisi ini karena hasilnya hanya dua: hash valid atau tidak valid, dan setiap percobaan bersifat independen satu sama lain.

Misalkan X adalah jumlah percobaan hingga pertama kali berhasil. Peluang bahwa percobaan pertama kali berhasil tepat pada percobaan ke-k berarti $k-1$percobaan sebelumnya gagal dan percobaan ke-k berhasil. Dengan kaidah perkalian untuk kejadian independen:

$$P(X=k) = \underbrace{(1-p)^{k-1}}_{\text{gagal sebanyak } k-1 \text{ kali}} \times \underbrace{p}_{\text{berhasil}}, \quad k = 1,2,3,\ldots$$

Distribusi ini dikenal sebagai **distribusi geometrik** [3]. Dari distribusi ini, dapat diturunkan ekspektasi dan variansi jumlah percobaan:

$$E[X] = \frac{1}{p}, \text{Var}[X] = \frac{1-p}{p^2}$$

F. Kompleksitas Algoritma

Kompleksitas algoritma adalah ukuran sumber daya komputasi, umumnya waktu atau ruang memori, yang dibutuhkan suatu algoritma sebagai fungsi dari ukuran inputnya [5]. Notasi $O()$ (*Big-O*) digunakan untuk mengekspresikan **batas atas** pertumbuhan kompleksitas tersebut.

Dalam analisis algoritma probabilistik seperti PoW, yang relevan adalah *expected complexity*, yaitu kompleksitas rata-rata yang diharapkan atas seluruh kemungkinan jalannya algoritma:

$$T_{\text{expected}} = E[X] \cdot C_{\text{hash}}$$

di mana C_{hash} adalah biaya komputasi per operasi hash. Dengan demikian, kompleksitas PoW tidak bersifat tetap, melainkan bergantung langsung pada nilai *difficulty* yang ditetapkan jaringan. Hal ini akan dianalisis lebih mendalam pada bagian berikutnya.

III. Analisis Kombinatorik Ruang Pencarian

Pada bagian sebelumnya, telah dibahas bahwa setiap percobaan hash dalam mekanisme *Proof-of-Work* merupakan pencarian acak dalam ruang yang sangat besar. Bagian ini akan menganalisis secara matematis seberapa besar ruang pencarian tersebut, seberapa kecil peluang keberhasilannya, dan bagaimana kedua hal tersebut bersama-sama menentukan ekspektasi jumlah iterasi yang dibutuhkan untuk menemukan satu blok Bitcoin yang valid.

A. Ruang Pencarian Nonce

Dalam mekanisme *Proof-of-Work*, *miner* mencari nilai *nonce* n yang memenuhi syarat hash tertentu. Secara teknis, *nonce* pada header blok Bitcoin adalah bilangan bulat 32 bit, sehingga ruang pencariannya adalah:

$$|\mathcal{N}| = 2^{32} \approx 4{,}29 \times 10^{9}$$

Namun, output SHA-256 adalah string 256 bit, sehingga ruang kemungkinan nilai hash yang dihasilkan adalah:

$$|\mathcal{H}| = 2^{256} \approx 1{,}16 \times 10^{77}$$

Di sinilah letak fondasi keamanan kriptografis Bitcoin, yaitu ruang pencariannya begitu besar sehingga tidak ada jalan pintas yang mungkin.

B. Probabilitas Keberhasilan Suatu Hash

Syarat keberhasilan *mining Bitcoin* adalah hash output harus lebih kecil dari nilai target, yang dinotasikan sebagai berikut

$$\text{SHA256}(\text{SHA256}(\text{block\_header} \parallel n)) < \text{target}$$

Perhatikan bahwa pada SHA-256 berperilaku seperti *random oracle*, outputnya terdistribusi seragam pada rentang $[0, 2^{256})$. Oleh karena itu, probabilitas satu percobaan hash menghasilkan nilai yang memenuhi syarat adalah:

$$p = \frac{\text{target}}{2^{256}}$$

Difficulty didefinisikan oleh jaringan Bitcoin sebagai perbandingan antara *target* blok pertama dengan *target* saat ini:

$$D = \frac{\text{target}_{\text{genesis}}}{\text{target}_{\text{current}}}$$

Dengan substitusi, diperoleh hubungan antara *difficulty* dan probabilitas sukses:

$$p \approx \frac{1}{D \cdot 2^{32}}$$

Hubungan ini menunjukkan bahwa semakin besar D, semakin kecil p, dan semakin sulit proses *mining*.

C. Ekspektasi dan Variansi Jumlah Iterasi

Karena setiap percobaan hash merupakan percobaan Bernoulli independen dengan peluang sukses p, jumlah iterasi X hingga pertama kali berhasil mengikuti distribusi geometrik sebagaimana didefinisikan pada Bagian II.E. Dengan demikian:

$$E[X] = 1/p = D \cdot 2^{32}$$

$$Var[X] = (1-p)/p^2 \approx D^2 \cdot 2^{64}$$

$$\sigma[X] \approx D \cdot 2^{32} = E[X]$$

Hasil terakhir ini menunjukkan bahwa standar deviasi jumlah iterasi kira-kira sama dengan ekspektasinya. Artinya, ketidakpastian hasil *mining* sangat besar. Seorang *miner* bisa berhasil jauh lebih cepat atau jauh lebih lambat dari ekspektasi. Di sinilah secara matematis dapat dijelaskan mengapa *pool mining* muncul sebagai solusi: dengan menggabungkan *hashrate*, variansi dapat ditekan secara signifikan.

D. Pengaruh Difficulty terhadap Ruang Pencarian Efektif

Ruang pencarian efektif H_valid dapat didefinisikan sebagai himpunan nilai hash yang memenuhi *target*. Banyaknya elemen ruang ini adalah:

$$\boldsymbol{|H\_valid| = target = 2^{256} / D}$$

Maka rasio ruang valid terhadap total ruang hash adalah:

$$|H\_valid| / |H| = 1 / (D \cdot 2^{32}) = p$$

Tabel I. Pengaruh Difficulty terhadap p dan E[X]

Difficulty D	p	E[X]
1	$\approx 2{,}3 \times 10^{-10}$	$\approx 4{,}3 \times 10^{9}$
10^{6}	$\approx 2{,}3 \times 10^{-16}$	$\approx 4{,}3 \times 10^{15}$
10^{12}	$\approx 2{,}3 \times 10^{-22}$	$\approx 4{,}3 \times 10^{21}$
$12{,}49 \times 10^{13}$ *	$\approx 0{,}186 \times 10^{-23}$	$\approx 53{,}7 \times 10^{22}$

** Difficulty Bitcoin saat makalah ini ditulis (sumber: https://www.coinwarz.com/mining/bitcoin/difficulty-chart?)*

E. Mengapa Tidak Ada Solusi Analitis?

Lalu, mengapa pencarian harus dilakukan secara *brute-force*? Apakah tidak ada pendekatan matematis yang lebih efisien?

Jawabannya terletak pada sifat SHA-256 yang telah dibahas pada Bagian II.C. Pendekatan matematis yang umumnya powerful, seperti *Chinese Remainder Theorem* (CRT) yang membutuhkan struktur kongruensi linear, atau metode aljabar linear, semuanya gagal karena SHA-256 bukan fungsi linear. Operasi internalnya penuh dengan XOR, rotasi bit, dan penjumlahan modular yang saling berinteraksi secara non-linear, sehingga tidak ada representasi aljabar yang dapat dieksploitasi untuk mengarahkan output menuju *target*.

Dengan demikian, model kombinatorik pada bagian ini bukan hanya sekadar pendekatan yang nyaman digunakan, melainkan merupakan sebuah model yang paling natural untuk menganalisis proses brute-force pada mekanisme Proof-of-Work Bitcoin karena secara langsung merepresentasikan ruang pencarian, peluang keberhasilan, dan jumlah percobaan yang diperlukan hingga solusi ditemukan.

IV. Analisis Kompleksitas Algoritma Proof-Of-Work

Setelah menganalisis ruang pencarian, akan dianalisis kompleksitas algoritma Proof-Of-Work untuk dapat mengestimasi energi yang dikonsumsi ketika melakukan *mining bitcoin.*

A. Algoritma Proof-Of-Work

Sebelum menganalisis kompleksitasnya, perlu dirumuskan terlebih dahulu algoritma PoW secara formal. Algoritma ini menerima *block_header* dan *target* sebagai input, dan mengembalikan *nonce* n yang valid sebagai output. Berikut algoritma Proof-Of-Work disajikan dalam kode Python:

```python
import hashlib

def proof_of_work(header_bytes, target_value):
    n = 0
    while (True):
        nonce_bytes = n.to_bytes(4, byteorder='big')
        block_input = header_bytes + nonce_bytes
        first_hash = hashlib.sha256(block_input).digest()
        second_hash = hashlib.sha256(first_hash).digest()
        hash_numeric = int.from_bytes(second_hash, byteorder='big')
        if hash_numeric < target_value:
            break
        n += 1
    return n
```

(Sumber: Arsip pribadi penulis)

(Sumber: Arsip pribadi penulis)

Setiap iterasi *loop* melakukan satu operasi hash dengan biaya komputasi konstan C_hash. Lalu, akan diestimasi berapa kali loop terjadi sampai akhirnya berhenti.

B. Kompleksitas Worst-Case vs Expected

Dalam analisis algoritma, terdapat dua perspektif kompleksitas yang relevan [5]. Pertama, *worst-case complexity*, yaitu jumlah operasi maksimum yang mungkin terjadi. Untuk PoW, *worst-case* adalah ketika *nonce* valid ditemukan pada percobaan terakhir:

$$T\_worst = O(2^{32})$$

Angka ini secara praktis tidak bermakna karena tidak ada komputer yang dapat menyelesaikannya dalam waktu wajar. Kedua, *expected complexity* yang jauh lebih relevan. Karena jumlah iterasi X mengikuti distribusi geometrik dengan E[X] = 1/p, maka:

$$T\_expected = E[X] \cdot C\_hash = D \cdot 2^{32} \cdot C\_hash$$

Dalam notasi *Big-O* terhadap *difficulty* D:

$$T\_expected = O(D)$$

Artinya, kompleksitas PoW tumbuh linear terhadap *difficulty*. Maka setiap kali *difficulty* naik dua kali lipat, ekspektasi waktu komputasi juga naik dua kali lipat.

Perlu ditambahkan catatan penting mengenai ukuran input (*input size*) yang digunakan. Notasi O(D) di atas mengukur kompleksitas terhadap nilai numerik *difficulty* D itu sendiri, bukan terhadap ukuran representasi bit dari input tersebut sebagaimana lazimnya standar analisis algoritma. Jika tingkat kesulitan diukur berdasarkan jumlah bit atau jumlah karakter *leading zeros* (d) yang disyaratkan pada output hash — sebagaimana digunakan pada simulasi di Bagian VI — maka kompleksitas *expected* dari algoritma ini sebenarnya adalah:

$$T\_expected = O(2^{d})$$

Karakteristik ini mengategorikan algoritma PoW ke dalam kelas *pseudo-polynomial time*, yaitu algoritma yang kompleksitasnya terlihat polinomial (linear) terhadap nilai parameter numerik D, namun sebenarnya tumbuh secara eksponensial terhadap panjang representasi bit dari inputnya. Hal ini konsisten dengan ruang pencarian efektif pada Bagian III.D, di mana penurunan |H_valid| sebesar satu bit (penambahan satu *leading zero*) melipatgandakan ekspektasi jumlah iterasi.

C. Konversi Jumlah Hash ke Waktu Komputasi

Jumlah operasi hash dapat dikonversi ke waktu nyata menggunakan *hashrate* perangkat keras, yaitu jumlah hash yang dapat dilakukan per detik:

$$E[waktu] = E[X] / hashrate = D \cdot 2^{32} / hashrate$$

Tabel II. Estimasi Waktu per Hardware

Hardware	Hashrate	E[waktu] (D = 9×10^{13})
GPU RTX 3080	4×10^{7} H/s	≈ 306 juta tahun

ASIC Antminer S21	2×10^{14} H/s	≈ 61 tahun

Hasil ini menunjukkan bahwa *solo mining* dengan *hardware* konsumen tidak mungkin secara praktikal. Kesimpulan ini dapat ditarik langsung dari analisis kompleksitas, bukan sekadar intuisi.

D. Pool Mining

Dari Bagian III.C, diketahui bahwa σ[X] ≈ E[X], yang menunjukkan bahwa waktu yang diperlukan dalam *mining bitcoin* memiliki varian yang sangat tinggi. Jika dikombinasikan dengan *expected time* yang ratusan juta tahun untuk *solo miner*, secara matematis muncul dua masalah sekaligus, yaitu *expected time* yang terlalu lama dan varian yang sangat tinggi sehingga waktu keberhasilan tidak terprediksi. *Pool mining* menyelesaikan kedua masalah ini. Dengan menggabungkan k *miner* dengan *hashrate* masing-masing h:

$$E[waktu\_pool] = E[waktu\_solo] / k$$

Diperoleh bahwa waktu turun linear terhadap jumlah *miner* dan *reward* dibagi proporsional. Inilah mengapa perlu adanya *pool mining*.

V. Estimasi Konsumi Energi

Setelah mendapat estimasi waktu yang diperlukan untuk *mining bitcoin*, maka langkah terakhir, yaitu mengestimasi energi yang diperlukan untuk *mining bitcoin.*

A. Model Estimasi Energi

Dari analisis kompleksitas pada Bagian IV, diketahui bahwa ekspektasi jumlah hash yang diperlukan untuk menemukan satu blok adalah $E[X] = D \cdot 2^{32}$. Berdasarkan hal tersebut, energi yang dikonsumsi dapat dimodelkan sebagai:

$$\boldsymbol{E[energi] = E[X] \cdot \varepsilon\_hash}$$

di mana ε_hash adalah energi yang dikonsumsi per operasi hash, diturunkan dari spesifikasi *hardware*:

$$\boldsymbol{\varepsilon\_hash = TDP\ (Watt) / hashrate\ (H/s) \quad [Joule/hash]}$$

Model ini mengasumsikan *hardware* beroperasi pada kapasitas penuh selama proses *mining*, sebuah asumsi yang wajar mengingat *mining* adalah *workload* komputasi yang sangat intensif.

B. Estimasi Energi

Dengan *difficulty* Bitcoin saat ini $D \approx 9 \times 10^{13}$ dan spesifikasi *hardware* dari [6], estimasi energi per blok untuk masing-masing skenario Adalah:

Tabel III. Ringkasan Estimasi Konsumsi Energi

Skenario	Hardware	Waktu	Daya
Solo miner	GPU RTX 3080	306 jt tahun	320 W
Solo miner	ASIC S21	61 tahun	3.500 W
Selesai 10 menit	GPU RTX 3080	10 menit	≈ 97,9 GW
Selesai 10 menit	ASIC S21	10 menit	≈ 11,2 GW
Jaringan Bitcoin	ASIC S21	≈ 8 menit	≈ 14 GW

Angka-angka di atas menjadi lebih bermakna ketika dibandingkan dengan konteks nyata. Agar satu blok Bitcoin dapat ditemukan dalam satu tahun menggunakan GPU, dibutuhkan sekitar 306 juta unit GPU dengan total konsumsi daya sekitar 97,9 GW. Sebagai perbandingan, total kapasitas listrik Indonesia adalah sekitar 105 GW. Artinya, dibutuhkan hampir seluruh listrik Indonesia hanya untuk menambang satu blok Bitcoin dalam setahun menggunakan GPU.

C. Validasi dengan Data Jaringan Bitcoin

Sebagai validasi independen, model diterapkan pada kondisi jaringan Bitcoin nyata. Dengan *hashrate* total jaringan saat ini sekitar 8×10^{20} H/s, model memprediksi waktu penemuan blok sekitar 8 menit, konsisten dengan target jaringan 10 menit. Estimasi daya jaringan yang dihasilkan model, yaitu sekitar 14 GW, juga konsisten dengan rentang 10 hingga 20 GW yang dilaporkan Cambridge Bitcoin Electricity Consumption Index [7]. Hal ini mengkonfirmasi bahwa pendekatan kombinatorik dari *first principles* menghasilkan estimasi yang valid dan dapat dipertanggungjawabkan secara akademik.

VI. Simulasi Dan Validasi

A. Desain Eksperimen

Seperti yang telah disebutkan, *mining* Bitcoin secara langsung tidak *feasible* untuk tujuan eksperimen akademik. Oleh karena itu, penulis merancang simulasi *mini Proof-of-Work* yang mereplikasi mekanisme PoW pada skala yang dapat dikomputasi, dengan tetap menggunakan fungsi hash SHA-256 yang sesungguhnya. Variabel independen adalah *difficulty* d, yaitu jumlah *leading zeros* heksadesimal yang dibutuhkan. Variabel dependennya adalah jumlah iterasi hingga hash valid ditemukan. Setiap level *difficulty* dijalankan sebanyak N = 1.000 kali untuk D = 1, 2, 3, 4, 5.

B. *Implementasi*

Simulasi diimplementasikan dalam Python menggunakan *library* hashlib untuk operasi SHA-256, statistics untuk menghitung rata" waktu komputasi, daj uuid untuk membangkitkan *header* unik.

```python
import hashlib
import statistics
import uuid

def bukti_kerja_sederhana(difficulty):
    sasaran = "0" * difficulty
    blok_awal = str(uuid.uuid4())

    nilai_nonce = 0
    langkah = 0

    while True:
        teks = blok_awal + str(nilai_nonce)
        hasil_hash = hashlib.sha256(teks.encode()).hexdigest()

        langkah += 1

        if hasil_hash.startswith(sasaran):
            return langkah

        nilai_nonce += 1
```

Gambar 2 potongan kode eksperimen
(Sumber: Arsip pribadi penulis)

```python
JUMLAH_PERCOBAAN = 1000

print("=" * 90)
print(f"{'d':<3} {'E[X] Teoritis':>15} {'Mean Empiris':>15} {'Std Empiris':>15} {'Galat (%)':>12}")
print("=" * 90)

for difficulty in range(1, 6):

    daftar_hasil = [bukti_kerja_sederhana(difficulty) for _ in range(JUMLAH_PERCOBAAN)]

    rata_teori = 16 ** difficulty
    rata_empiris = statistics.mean(daftar_hasil)
    deviasi_empiris = statistics.stdev(daftar_hasil)

    persentase_galat = (
        abs(rata_empiris - rata_teori)
        / rata_teori
        * 100
    )

    print(
        f"{difficulty:<3}"
        f"{rata_teori:>15,.0f}"
        f"{rata_empiris:>15,.2f}"
        f"{deviasi_empiris:>15,.2f}"
        f"{persentase_galat:>12.2f}"
    )

print("=" * 90)
print(f"Jumlah trial per difficulty = {JUMLAH_PERCOBAAN}")
```

Gambar 3 potongan kode eksperimen
(Sumber: Arsip pribadi penulis)

C. *Hasil Eksperimen*

```
PS C:\Users\HP\OneDrive\Desktop> & C:\msys64\ucrt64\bin\python.exe c:/Users/HP/OneDrive/Desktop/code/MiniEksperimen.py
==========================================================================
d     E[X] Teoritis    Mean Empiris     Std Empiris    Galat (%)
==========================================================================
1                16           15.89           15.59         0.66
2               256          252.06          254.02         1.54
3             4,096       4,015.04       4,106.37         1.98
```

Gambar 4 hasil eksperimen
(Sumber: Arsip pribadi penulis)

```
[Running] python -u "c:\Users\HP\OneDrive\Desktop\code\MiniEksperimen.py"
==========================================================================
d     E[X] Teoritis    Mean Empiris     Std Empiris    Galat (%)
==========================================================================
1                16           15.70           15.37         1.86
2               256          260.20          256.36         1.64
3             4,096       4,188.25       4,268.86         2.25
```

Gambar 5 hasil eksperimen
(Sumber: Arsip pribadi penulis)

Hasil eksperimen ini menunjukkan bahwa perilaku jumlah iterasi pada simulasi mengikuti prediksi distribusi geometrik dengan sangat baik. Dengan kata lain, model probabilistik yang digunakan mampu merepresentasikan mekanisme brute-force pada simulasi mini PoW secara akurat.

D. *Ekstrapolasi ke Skala Bitcoin*

Karena simulasi menunjukkan kesesuaian yang baik (galat < 6%) dengan prediksi distribusi geometrik pada skala kecil, model kemudian digunakan sebagai dasar untuk melakukan ekstrapolasi analitik ke skala Bitcoin yang sesungguhnya.

VII. KESIMPULAN

Makalah ini telah menunjukkan bahwa mekanisme *Proof-of-Work* Bitcoin dapat dianalisis secara rigorus menggunakan dua alat matematika diskrit: kombinatorika dan analisis kompleksitas algoritma. Dari analisis kombinatorik, diperoleh bahwa probabilitas keberhasilan satu percobaan hash adalah $p = 1/(D \cdot 2^{32})$, dengan ekspektasi iterasi $E[X] = D \cdot 2^{32}$ dan standar deviasi yang sama besarnya dengan ekspektasi. Dari analisis kompleksitas, diperoleh bahwa kompleksitas algoritma PoW adalah $O(D)$, yang berarti tumbuh linear terhadap *difficulty*. Dari simulasi, model distribusi geometrik terbukti valid secara empiris dengan galat di bawah 2%, dan ekstrapolasi ke skala Bitcoin menghasilkan estimasi yang konsisten dengan data jaringan nyata.

UCAPAN TERIMA KASIH

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga makalah ini dapat diselesaikan dengan baik. Penulis juga mengucapkan terima kasih kepada Pak Rinaldi, dosen pengampu mata kuliah IF1220 Matematika Diskrit, atas bimbingan dan ilmu yang telah diberikan selama satu semester ini. Juga, terima kasih kepada bibik dan keluarga penulis yang telah memberikan dukungan sehingga makalah ini dapat terselesaikan.

LAMPIRAN

Berikut pranala *source code* eksperimen yang penulis lakukan:
https://drive.google.com/file/d/1brqQx0ZXJA_keLdS0Z7EIh7D0flzdUnE/view?usp=sharing

REFERENSI

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. Tersedia pada: https://bitcoin.org/bitcoin.pdf. Diakses: 19 Juni 2026.

[2] National Institute of Standards and Technology, "Secure Hash Standard (SHS)," FIPS PUB 180-4, 2015. Tersedia pada: https://csrc.nist.gov/publications/detail/fips/180/4/final. Diakses: 19 Juni 2026.

[3] K. H. Rosen, Discrete Mathematics and Its Applications, 8th ed. New York: McGraw-Hill, 2019.

[4] A. M. Antonopoulos, Mastering Bitcoin: Programming the Open Blockchain, 2nd ed. Sebastopol: O'Reilly Media, 2017.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, dan C. Stein, Introduction to Algorithms, 4th ed. Cambridge: MIT Press, 2022.

[6] Bitmain Technologies, "Antminer S21 Product Specification," 2024. Tersedia pada: https://www.bitmain.com/product/. Diakses: 19 Juni 2026.

[7] A. de Vries, "Bitcoin's Growing Energy Problem," Joule, vol. 2, no. 5, hal. 801–805, 2018. [Daring]. Tersedia pada: https://www.repository.cam.ac.uk/items/fd416409-1aef-49da-b373-3db366738fc5. Diakses: 19 Juni 2026.

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.

Bogor, 19 Juni 2025

Muhammad Fauzi Muharam, 13525091