Analisis Pengunaan *Merkle Patricia Tree* pada Sistem *Blockchain*

Samuel Christopher - 13520075
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520075@stei.itb.ac.id

***Abstrak – Di era teknologi yang maju ini, muncul sistem baru yaitu blockchain yang merubah pandangan orang tentang dunia digital. Sistem blockchain. Blockchain* adalah jenis struktur data yang memanfaatkan "*Merkle Patricia Tree*" untuk mewujudkan struktur data yang aman, kronologis, dan tidak berubah. *Blockchain* juga tidak memiliki biaya transaksi untuk biaya infrastruktur. Makalah ini akan membahas *Merkle Patricia Tree* sebagai struktur data pada *blockchain*, dan implementasinya.**

Kata kunci – Merkle Patricia Tree, Tree, Blockchain

I. PENDAHULUAN

Blockchain adalah jenis struktur data yang terus berkembang yang menyimpan catatan permanen dari semua transaksi yang telah terjadi dengan cara yang aman, kronologis, dan tidak berubah.

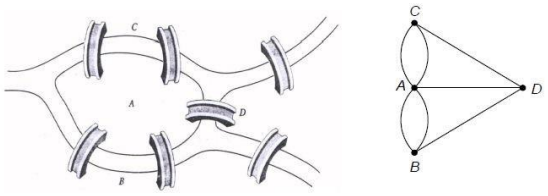
Selain itu, *blockchain* juga tidak memiliki biaya transaksi seperti biaya infrastruktur. Sehingga, blockchain adalah cara yang paling sederhana namun cerdik untuk bisa menyampaikan informasi dari A ke B secara otomatis dan aman. Blok yang ada di dalamnya diverifikasi oleh jutaan komputer dan didistribusikan dengan menggunakan internet. Blok yang diverifikasi ini lantas ditambahkan ke rantai dan disebar dalam suatu jaringan khusus, lalu membuat catatan dan juga riwayat yang unik.

Teknologi ini dapat diintegrasi untuk banyak industri, dan fungsi utamanya adalah untuk distribusi mata uang *crypto*. *Blockchain* sangatlah terkenal karena memiliki 6 keunggulan dibandingkan dengan *database* dengan struktur data yang lain. Dalam implementasinya, *Blockchain* menggunakan strukur data Pohon.

II. DASAR TEORI

A. GRAF

Dalam matematika dan ilmu komputer, graf adalah suatu objek dasar dalam pelajaran teori graf. Teori graf adalah cabang yang mempelajari tentang sifat-sifat graf. Teori Graf mulai dikenal pada saat seorang matematikawan bangsa Swiss, bernama Leonhard Euler, yang berhasil mengungkapkan Misteri Jembatan Konigsberg pada tahun 1736.



Gambar 2 Masalah Jembatan Konigsberg

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf

Di Kota Konigsberg (sekarang bernama Kaliningrad, di Uni Soviet) mengalir sebuah sungai bernama sungai Pregel.Di tengah sungai tersebut terdapat dua buah pulau. Dari kedua pulau tersebut terdapat jembatan yang menghubungi ke tepian sungai dan di antara kedua pulau. Jumlah jembatan tersebut adalah 7 buah seperti gambar di atas. Secara singkat, dalam tulisannya, Euler menyajikan keadaan jembatan Konigsberg tersebut terlihat pada gambar1 bagian kanan dalam masalah di atas, daratan (tepian A dan B, serta pulau C dan D) disajikan sebagai titik dan jembatan disajikan sebagai ruas garis. Euler mengemukakan teoremanya yang mengatakan bahwa perjalanan yang diinginkan di atas (yang kemudian dikenal sebagai perjalanan Euler) akan ada apabila graf terhubung dan banyaknya garis yang datang pada setiap titik (derajat simpul) adalah genap.

Pengertian lain dari graf adalah himpunan dari benda-benda yang disebut simpul (*Vertex* atau *Node)* yang terhubung oleh sisi-sisi (*Edge)* atau sudut dari sisi tersebut. Selain itu, graf digambarkan dengan adanya titik-titik yang merupakan simpul(*vertex)* dan terhubung oleh garis-garis yang melambangkan sisi (*Egde*). Dalam pengaplikasian graf, banyak sekali struktur yang dapat direpresentasikan dan banyak masalah yang dapat diselesaikan denganmenggunakan graf. Aplikasi teori graf antara lain pemodelan jaringan telepon, pemodelan jaringan listrik, pemodelan jaringan internet, dan pemodelan molekul di bidang ilmu kimia dan fisika.

Formalnya, suatu graf misalkan G dapat dinyatakan sebagai berikut:

$$G = < V,E >$$

Graf G terdiri atas himpunan V (*vertex)* yang berisikan simpul pada graf tersebut dan himpunan dari E (*edge)* yang berisi sisi pada graf tersebut. Himpunan E dinyatakan sebagai pasangan dari simpul yang ada dalam V.

V = Himpunan tidak-kosong dari simpul-simpul (*vertces)*
= {v1, v2, v3, …, vn}

E = Himpunan sisi (*egdes)* yang menghubungkan sepasang simpul
= {e1, e2, e3, …, en}

Sebagai contoh definisi dari graf pada pengertian di atas adalah sebagai berikut:

Adapun jenis-jenis graf berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf yaitu, digolongkan menjadi 2 jenis:

1. Graf sederhana(*simple graph)*
   Graf yang tidak mengandung gelang maupun sisi ganda biasa dinamakan graf sederhana.
2. Graf tak-sederhana (*unsimple graph)*

   Graf yang mengandung sisi ganda atau gelang dinamakan Graf tak-sederhana. Graf tak-sederhana dibedakan menjadi 2, yaitu :

   1. Graf ganda (*Multi-graph)*
      Graf yang mengandung sisi ganda.
   2. Graf semu (Pseudo-graph)
      Graf yang mengandung sisi gelang.

Berdasarkan jumlah simpul pada suatu graf, maka secaraumum graf dapat digolongkan menjadi dua jenis:

1. Graf berhingga (*limited graph)*
   Graf berhingga adalah graf yang jumlah simpulnya, n, berhinga.
2. Graf tak-berhingga (*unlimited graph)*
   graf tak-berhingga adalah graf yang jumlah simpulnya, n sampai tidak berhingga banyaknya disebut graf tak-berhingga.

Berdasarkan orientasi arah pada isi, maka secara umum graf dibedakan menjadi dua jenis:

1. Graf tak-berarah (*undirected graph)*
   Graf yang sisinya tidak mempunyai orientasi arah disebut graf yang tak-berarah.
2. Graf berarah (*directed graph)*
   Graf yang setiap sisinya diberika orientasi arah disebut graf yang berarah.

Peranan graf dapat di kembangan dengan memberikan bobot pada tiap sisi. Banyak konsep yang dapat dibuat dengan adanya graf berbobot ataupun pengaplikasiannya, seperti mencari lintasan terpendek, pengambilan keputusan dengan pohon keputusan. Selain itu, graf dapat merepresentasikan objek- objek diskrit dan hubungan antaraobjek-objek tersebut.

B. TERMINOLOGI GRAF

a. Ketetanggaan (Adjacent) Dua simpul bertetangga apabila terhubung langsung.

b. Bersisian (Incidency) Untuk sebuah sisi e = (vj, vk), e bersisian dengan simpul vj dan vk.

c. Simpul Terpencil (Isolated Vertex) Simpul yang tidak mempunyai sisi yang bersisian dengannya.

d. Graf Kosong (Null graph or empty graph)
Graf yang himpunan sisinya merupakan himpunan kosong.

e. Derajat (Degree)
Jumlah sisi yang bersisian dengan suatu simpul.

f. Lintasan (Path)
Sisi yang ditempuh dari simpul awal (vo) sampai simpul akhir (vn).

g. Sirkuit (Circuit)
Lintasan yang berawal dan berakhir pada simpul yang sama.

h. Keterhubungan (Connected)
Untuk setiap pasang simpul vi dan vj, terdapat lintasan dari vi ke vj.

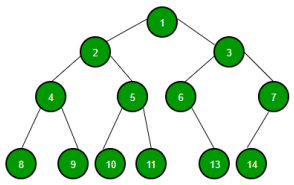
i. Upagraf (Subgraph)
Bagian dari graf yang simpul dan sisinya merupakan himpunan bagian dari sebuah graf.

j. Cut-set
Himpunan sisi yang dibuang dari sebuah graf yang menyebabkan graf tersebut tidak terhubung.

k. Graf Berbobot (Weighted Graph)
Graf yang setiap sisinya diberi sebuah harga.

C. POHON

Pohon merupakan graf terhubung yang tidak mengandung sirkuit dan tak-berarah.

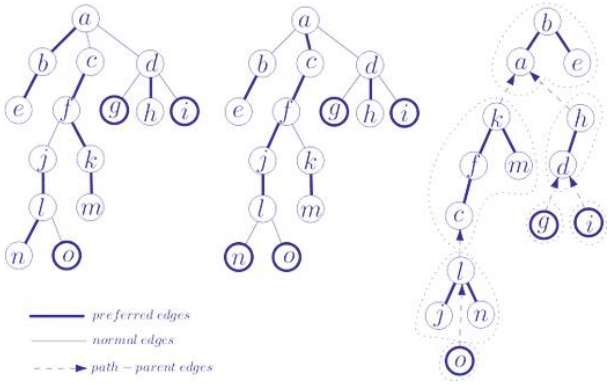


Gambar 2.1 Contoh Pohon

Sumber: https://www.geeksforgeeks.org/binary-tree-data-structure/

D. HUTAN

Hutan terdiri dari sekumpulan pohon yang saling lepas. Dengan kata lain hutan merupakan graf tidak terhubung yang tak-berarah dan tidak mengandung sirkuit. Komponen-komponen graf tersebut adalah pohon.



Gambar 2.2 Contoh Hutan

Sumber: https://www.w3schools.in/data-structures-tutorial/forests-and-orchards/

E. SIFAT POHON

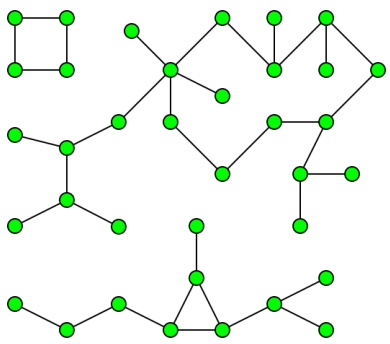
Jika terdapat G = (V, E), merupakan graf sederhana dengan simpul n yang tak-berarah. Maka pernyataan di bawah ini adalah benar.

1) G adalah pohon
2) Tiap pasang simpul G terhubung oleh lintasan tunggal
3) G terhubung dengan sisi sebanyan n-1
4) Tidak terdapat sirkuit di G dan jumlah sisi n-1
5) Tidak terdapat sirkuit di G, penambahan satu sisi akanmembuat hanya satu sirkuit
6) G terhubung dan semua sisisnya adalah jembatan

F. POHON MERENTANG

Pohon merentang sebuah graf terhubung merupakan upagraf merentang yang berupa pohon. Pohon merentang atau *spanning tree* dapat diperoleh dengan memutus sirkuit di dalam graf.

Pada setiap graf terdapat paling sedikit satu pohon merentang. Graf tak-terhubung dengan jumlah komponen sebanyak k, memiliki k buah pohon merentang, yang disebut hutan merentang atau *spanning forest.*

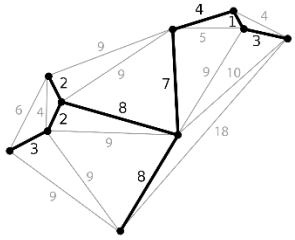


Gambar 2.3 Contoh Pohon Merentang

Sumber: https://stackoverflow.com/questions/43252588/spanning-tree-vs-spanning-forest

G. POHON MERENTANG MINIMUM

Graf terhubung yang berbobot bisa jadi memiliki lebih dari satu pohon merentang. Pohon merentang dengan bobot terkecildisebut *minimum spanning tree* atau pohon merentang minimum.



Gambar 2.4 Contoh Pohon Merentang Minimum

Sumber: https://en.wikipedia.org/wiki/Minimum_spanning_tree#/media/File:Minimum_spanning_tree.svg

H. POHON MERENTANG

Pohon merentang (*spanning tree*). Pohon merentang dari graf terhubung adalah upagraf merentang yang berupa pohon. Pohon merentang diperoleh dengan memutus sirkuit di dalam graf.

Setiap graf terhubung mempunyai paling sedikit satu buah pohon merentang. Graf tak terhubung dengan *k* komponen mempunyai *k* buah pohon merentang yang disebut hutan merentang (*spanning forest*).

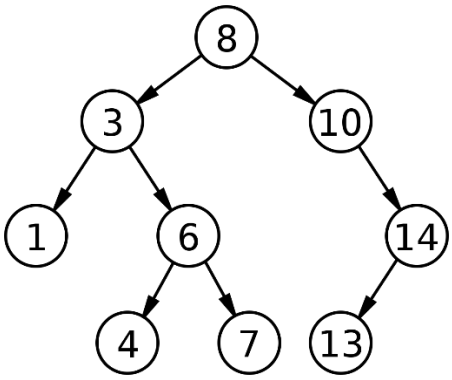
Sedangkan, pohon berakar (*rooted tree*) merupakan pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah. Adapun terminologi pohon berakar adalah sebagai berikut:

1. Anak (*child/children*)
   Simpul yang derajat masuknya tidak nol dengan sisi yang menghubungkan simpul anak dengan simpul orangtua.
2. Orangtua (*parent*)
   Simpul yang terhubung dengan simpul anak.
3. Lintasan (*path*)
   Jalan atau kumpulan simpul yang dilalui dari simpul asal ke simpul tujuan.
4. Saudara kandung (*sibling*)
   Satu simpul dikatakan sebagai saudara kandung dari simpul lain jika berasal dari orangtua yang sama.
5. Upapohon (*subtree*)
   Pohon yang terdapat di dalam pohon sebagai bagian dari pohon yang lebih besar.

6. Derajat (*degree*)
   Banyaknya anak dari suatu simpul.
7. Daun (*leaf*)
   Simpul yang berderajat nol.
8. Simpul dalam (*internal nodes*)
   Simpul selain daun
9. Aras (*level*) atau tingkat
   Aras suatu simpul dimulai dari nol, yaitu dari akar dan terus bertambah satu untuk tiap anak.
10. Tinggi (*height*) atau kedalaman (*depth*)
    Aras terbesar dalam suatu pohon.
11. Anak Kanan
    Anak pada pohon biner yang terhubung dengan sisi yang ke arah kanan
12. Anak kiri
    Anak pada pohon biner yang terhubung dengan sisi yang ke arah kiri

I. POHON TERURUT

Pohon terurut adalah pohon yang mempertimbangkan urutan anaknya. Urutan anak-anak tersebut penting.

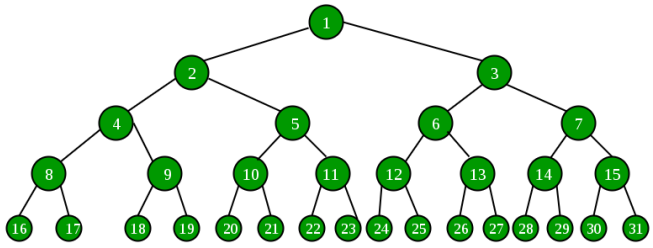


Gambar 2.6 Contoh Pohon Terurut

Sumber: https://en.wikipedia.org/wiki/Binary_search_tree#/media/File:Binary_search_tree.svg

J. POHON BINER

Pohon biner adalah pohon yang jumlah anak maksimal setiap simpul adalah 2. Pohon ini sangat sering digunakan, dan sangat penting kegunaanya. Anak pohon biner dibedakan menjadi anak kiri atau *left child* dan anak kanan atau *right child*. Karena urutan anak dibedakan, pohon biner juga merupakan pohon terurut.



Gambar 2.7 Contoh Pohon Berakar

Sumber: https://www.geeksforgeeks.org/print-middle-level-perfect-binary-tree-without-finding-height/

K. TERAPAN POHON BINER

1. Pohon Ekspresi
   Merupakan pohon yang terdiri dari operan di bagiandaun dan operator di simpul dalam.
2. Pohon Keputusan
   Pohon keputusan terdiri dari perbandingan suatu hal pada akar, dan hasil perbandingan pada sisi. Daun merupakan hasil akhir keputusan.
3. Kode Awalan
   Merupakan pohon dengan bobot sisi 0 dan 1. Daun pada pohon merupakan lintasan yang menuju daun tersebut.
4. Kode Huffman
   Kode Huffman digunakan untuk memampatkan rangkaian pesan.

III. MERKLE PATRICIA TREE

A. PENGERTIAN

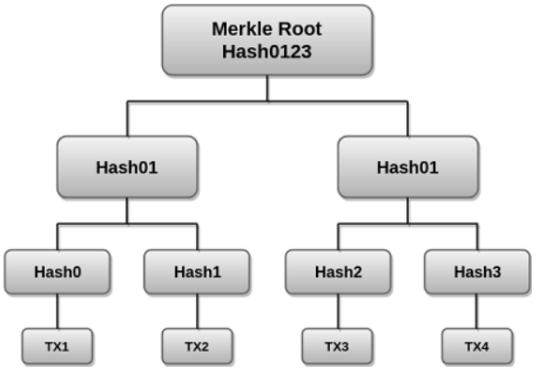
Merkle Tree diusulkan pada awal tahun 80-an oleh Ralph Merkle, seorang ilmuwan yang dikenal karyanya pada kriptografi.

Merkle Tree adalah struktur data yang sangat penting dalam sistem blockchain. Itu adalah struktur data yang dibuat dari beberapa *hash* dari blok data yang berbeda, dan bertujuan untuk melayani transaksi di dalam blok. *Merkle Tree* membantu efisiensi dan keamanan verifikasi dari konten di dalam data yang besar. Selain itu, dapat juga membantu konsistensi konten dari data. *Bitcoin* dan *Ethereum* menggunakan *Merkle Tree* Structures. *Merkle Tree* bisa disebut juga sebagai *Hash* Tree.

B. CARA KERJA

Merkle Tree menyimpan seluruh transaksi di dalam blok dengan cara memproduksi sidik jari digital dari seluruh bagian dari transaksi yang terjadi. *Merkle Tree* dapat membuat setiap penggunanya memverifikasi apakah transaksinya dapat di-*include* ke dalam blok atau tidak.

Merkle Tree terbuat dari pengulangan kalkulasi *hashing* dari *node*s sampai hanya terjadi satu *hash* lagi. *Hash* ini disebut sebagai *Merkle Root* atau *Root Hash. Merkle Tree* terbuat dari pendekatan yang *Bottom-Up.*

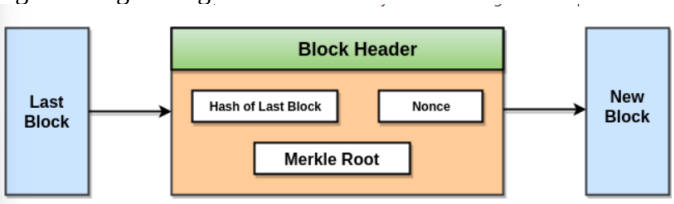


Gambar 3.1 Visualisasi *Merkle Tree*
Sumber: https://www.javatpoint.com/blockchain-merkle-tree

Setiap daun pada *node* adalah *hash* dari data transaksi, dan *node* yang bukan daun adalah *hash* dari sebelumnya. *Merkle Tree* adalah *Binary Tree*, ia membutuhkan daun berjumlah genap. Jika ada transaksi yang berjumlah ganjil, maka *hash* yang terakhir akan diduplikasi sekali untuk membuat daun yang genap.

Dari gambar 3.1 dapat dilihat bahwa itu adalah visualisasi dari *Merkle Tree*. Terdapat empat transaksi di dalam blok, yaitu TX1, TX2, TX3, dan TX4. Di situ kita dapat melihat bahwa di *root node* terdapat *Hash* yang disebut sebagai *Root Hash*. Tiap pengulangan *hash* yang terjadi akan disimpan di tiap daun *node*, dan menghasilkan *Hash01* dan akan dipecah lagi menjadi *Hash0, Hash1, Hash2,* dan *Hash3.*

Merkle Root disimpan di dalam sebuah *block header*. *Block Header* adalah sebuah tempat block yang berguna untuk mengambil *hash* di proses *mining*. *Block header* berisi *hash* dari blok terakhir, dan *Root Hash* dari semua transaksi di blok *hash* yang sekarang sedang diakses.



Gambar 3.2 Visualisasi Block Header
Sumber: https://www.javatpoint.com/blockchain-merkle-tree

Block header ini menyebabkan seluruh transaksi di dalam blok menjadi tahan terhadap kerusakan eksternal. Karena *Root Hash* ini terisi dengan seluruh *hash* yang ada di dalam transaksi di dalam blok.

Merkle Tree menjada integritas dari data. Jika ada sedikit sekali detail transaksi yang berubah, maka perubahan yang terjadi akan memberi efek pada seluruh *hash* di dalam transasksi tersebut. Perubahan ini dapat mengubah seluruh susunan *Merkle Tree* sampai *Merkle Root*, merubah nilai pada *Merkle Root*. Sehingga semua orang dapat melihat bahwa *Merkle Tree* adalah cara yang cepat dan simpel untuk menguji apakah sebuah transaksi ada di dalam blok atau tidak.

C. PENGUNAAN PADA BLOCKCHAIN

Merkle Tree sangat penting dalam data yang mempunyai sistem blockchain seperti Bitcoin dan banyak mata uang kripto lainnya. Merupakan komponen integral dari setiap blok, dapat ditemukan di header blok. Demi mendapatkan daun untuk pohon kita, kita menggunakan *hash* transaksi (TXID) dari setiap transaksi yang dimasukkan ke dalam blok.

1. Penambangan

Blok Bitcoin terdiri dari dua bagian. Bagian pertama adalah header blok, segmen berukuran tetap berisi metadata blok. Bagian kedua adalah daftar transaksi yang ukurannya bervariasi, tetapi cenderung jauh lebih besar dari header.

Penambang perlu berulang kali melakukan *hashing* data untuk menghasilkan output yang cocok dengan persyaratan tertentu dalam menambang blok yang valid. Mereka dapat melakukan triliunan upaya sebelum menemukannya. Dalam setiap upaya, mereka mengubah angka acak di header blok (yaitu nonce) untuk menghasilkan output yang berbeda. Tetapi banyak blok yang masih tetap sama. Mungkin sudah ada ribuan transaksi, tetapi penambang masih harus melakukan *hashing*.

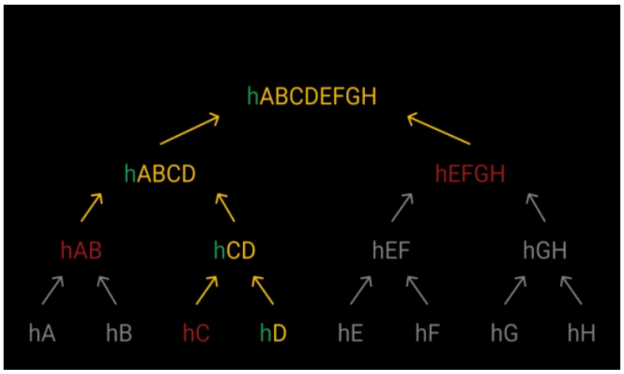
Merkle root menyederhanakan proses ini secara signifikan. Sebelum menambang, Anda mengatur semua transaksi yang ingin Anda masukkan dan membangun struktur *Merkle Tree*. Anda menempatkan root *hash* yang dihasilkan (32 byte) di header blok. Lalu, saat Anda menambang, Anda tidak perlu melakukan *hashing* terhadap semua blok, tetapi cukup terhadap header blok.

Proses ini akan berhasil karena memiliki sifat tamper-proof. Anda secara efektif mempersingkat semua transaksi blok dalam format yang ringkas. Anda tidak dapat menemukan header blok yang valid dan kemudian mengubah daftar transaksi, karena itu akan mengubah Merkle root. Ketika blok dikirim ke *node* lain, root dihitung dari daftar transaksi. Jika tidak cocok dengan yang ada di header, blok akan ditolak.

2. Verifikasi

Terdapat sifat *Merkle Tree* yang menarik lainnya yang dapat kita manfaatkan. Yang ini menyangkut klien ringan (*node* yang tidak memiliki salinan lengkap blockchain). Jika menjalankan *node* pada perangkat dengan sumber daya terbatas, Anda tidak akan mengunduh dan melakukan *hashing* terhadap semua transaksi blok. Sebaliknya, yang dapat Anda lakukan adalah hanya meminta Merkle proof – bukti yang diberikan oleh full *node* yang menunjukkan bahwa transaksi Anda berada di blok tertentu. Ini lebih sering disebut sebagai Verifikasi Pembayaran Sederhana atau Simplified Payment Verification (SPV), dijelaskan secara terperinci oleh Satoshi Nakamoto dalam whitepaper Bitcoin.

Bayangkan skenario di mana kita ingin mengetahui informasi mengenai transaksi yang TXID-nya hD. Jika hC diberikan, kita dapat memeriksa hCD. Kemudian, kita memerlukan hAB untuk menghitung hABCD. Terakhir, dengan hEFGH, kita dapat memeriksa apakah Merkle root yang dihasilkan cocok dengan yang ada di header blok. Jika ya, itu membuktikan bahwa transaksi telah dimasukkan ke dalam blok – hampir mustahil untuk membuat *hash* yang sama dengan data yang berbeda.



Gambar 3.3 Visualisasi *Hash*
Sumber: https://academy.binance.com/id/articles/merkle-trees-and-merkle-roots-explained

Dalam contoh di atas, kita hanya perlu melakukan *hashing* sebanyak tiga kali. Tanpa Merkle proof, kita harus melakukannya tujuh kali. Karena blok-blok saat ini mengandung ribuan transaksi, menggunakan Merkle proof menghemat banyak waktu dan sumber daya komputasi.

D. PERBANDINGAN ANTARA BLOCKCHAIN DAN DATABASE

Blockchain menggunakan algoritma *hash* SHA256, yang mengenkripsi informasi sehingga data menjadi lebih aman, sedangkan database adalah arsitektur jaringan yang hanya dapat diubah datanya oleh seorang admin yang mempunyai kunci akses pada database tersebut.

No	*Blockchain*	*Database*
1	Desentral sehinga tidak membuthkan admin	Membutuhkan admin untuk merubah data
2	Tidak membutuhkan izin akses, karena semua orang dapat memiliki akses	Membutuhkan izin akses
3	Mempunyai sejarah data dan kepunyaan digital seseorang	Tidak mempunyai sejarah data dan kepunyaan digital seseorang
4	Hanya memliki operasi pemasukkan	Mempunyai operasi membuat, membaca, memperbarui, dan menghapus. (CRUD)
5	Sepenuhnya rahasia	Tidak sepenuhnya rahasia
6	Semua orang mempunyai hak untuk menulis blockchain	Hanya beberapa orang yang dapat memanipulasi data

Dapat dilihat dari tabel di atas, *Blockchain* merupakan sebuah tempat penyimpanan data yang rahasia, aman, dan desentral, sehingga semua orang dapat mengaksesnya. Selain itu, kita tidak perlu ragu akan *Hacker* karena data yang sudah masuk ke dalam sistem *blockchain* tidak dapat dimanipulasi sama sekali, karena hanya memiliki operasi pemasukkan (*insert operation).*

IV. KESIMPULAN

Dapat disimpulkan bahwa struktur data pohon dapat digunakan dalam banyak hal contohnya pada sistem teknologi yang baru, *Blockchain.* Pemanfaatan struktur data pohon membuat seluruh sistem pada *Blockchain* menjadi aman dan konsisten, sebab dalam pengunaannya, struktur data pohon merkle menggunakan sistem *hash* pada setiap daunnya, di mana setiap daun menyatakan *hash* untuk setiap transaksi yang terjadi pada sistem *blockchain* ini.

Ada pula pemanfaat struktur data pohon sehingga struktur ini mempunyai beberapa keunggulan yaitu:

1. Menjaga integritas dan validitas dari data.
2. Mengurangi memori, karena dapat diolah secara cepat dan mudah
3. Manajemen dari struktur data ini hanya membutuhkan sedikit informasi untuk dapat disalurkan ke seluruh jaringan

V. UCAPAN TERIMAKASIH

Dengan selesainya makalah ini, saya mengucapakan terima kasih yang sebesar-besarnya kepada Tuhan, karena-Nya saya dapat mebuat makalah ini. Saya juga berterimakasih kepada orang tua dan teman-teman saya yang telah membantu saya dalam mengerjakan makalah ini. Terakhir, saya berterimakasih kepada ibu Dra. Harlili, M.Sc. dan bapak Dr. Ir. Rinaldi Munir MT. selaku dosen yang membimbing saya dalam mata kuliah Matematika Diskrit ini.

REFERENCES

[1] https://www.javatpoint.com/blockchain-merkle-tree, diakses pada 13 Desember 2021.
[2] https://academy.binance.com/id/articles/merkle-trees-and-merkle-roots-explained, diakses pada 13 Desember 2021.
[3] https://www.javatpoint.com/blockchain-vs-database, diakses pada 14 Desember 2021.
[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2010-2011/Makalah2010/MakalahStrukdis2010-091.pdf, diakses pada 12 Desember 2021.

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, 14 Desember 2020

Samuel Christopher, 13520075