# Penerapan Pohon dan Fungsi Hash pada Struktur Data Pohon Merkle

Louis Cahyadi 13517126
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*13517126@std.stei.itb.ac.id*

***Abstract*—Salah satu struktur data yang sangat penting yang banyak digunakan pada sistem *blockchain* dan *cryptocurrency* ialah pohon merkle. Pohon Merkle didasarkan pada struktur data pohon dan menerapkan metode fungsi hash untuk. Salah satu kegunaan utama dari pohon merkle merupakan kemampuannya yang dapat menjamin integritas data dengan mudah bagi pemilik data, serta aman selama fungsi hash yang digunakan merupakan fungsi hash yang baik tanpa ada kolisi. Algoritma traversal pohon merkle merupakan algoritma yang mampu memvalidasi integritas data, dan memiliki kompleksitas waktu O($\log_2$N) dan kompleksitas ruan O($\log_2$N).**

***Kata kunci*—pohon, fungsi hash, pohon merkle.**

## I. PENDAHULUAN

Struktur data merupakan sebuah cara mengatur atau mengelompokan suatu kumpulan data, yang biasanya diletakan pada memori, untuk dapat diakses dan dimodifikasi dengan efisien sesuai dengan algoritma yang digunakan. Ada sangat banyak struktur data yang sering digunakan tergantung persoalan atau masalah yang dihadapi. Suatu program atau aplikasi yang baik, membutuhkan waktu proses yang singkat dan memori yang tidak besar.

Selain itu, perkembangan dunia teknologi menuntut adanya keamanan dari data yang dimiliki pengguna. Hal tersebut melahirkan sebuah cabang ilmu kriptografi, yang didalamnya juga mempelajari fungsi hash. Fungsi hash memiliki berbagai kegunaaan dalam kehidupan nyata saat ini, diantaranya menjaga integritas data, mempersingkat waktu pengiriman, dan menormalkan panjang data yang berbeda-beda. Pohon merkle menjawab persoalan yang membutuhkan sistem penyimpanan yang efektif, dan memiliki tingkat keamanan yang tinggi, serta dapat memberikan informasi integritas data yang dibutuhkan.

Pohon merkle merupakan salah satu struktur data yang banyak digunakan pada teknologi *blockchain* dan *cryptocurrency*. Pohon merkle dipatenkan oleh Ralph Merkle pada tahun 1979 yang patennya berakhir pada tahun 2002. Pohon merkle juga sering disebut pohon hash karena merupakan penerapan dari penggunaan fungsi hash dan pohon biner.

Salah satu kegunaan

## II. DASAR TEORI

### *A. Graf*

#### *1. Definisi Graf*

Graf G merupakan sebuah pasangan himpunan

$$G = (V, E)$$

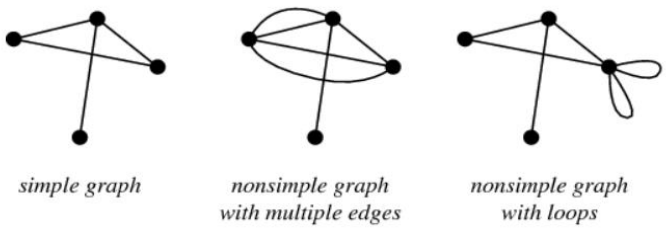
yang mana V merupakan himpunan titik – titik (simpul) dan E merupakan koleksi dari garis – garis (sisi) yang titik – titik ujungnya berada pada himpunan V. Sebuah graf mungkin memiliki tak berhingga banyaknya simpul dan sisi.

#### *2. Jenis – Jenis Graf*

Pada sebuah graf, gelang adalah sisi yang memiliki simpul awal dan simpul akhir sama. Selain itu, sisi ganda adalah dua buah sisi yang menghubungkan dua simpul yang sama.

Berdasarkan ada atau tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum sebuah graf dapat digolongkan menjadi dua jenis yaitu

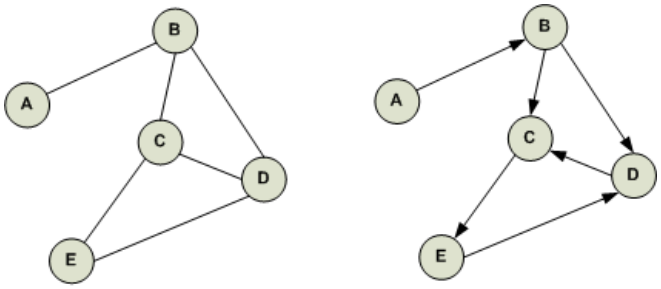
1) Graf sederhana
   Graf sederhana merupakan graf yang tidak mengandung gelang maupun sisi ganda.
2) Graf tak-sederhana
   Graf tak-sederhana merupakan graf yang mengandung gelang atau sisi ganda.




*Gambar 1 : Graf sederhana, graf tak sederhana dengan sisi ganda, dan graf tak sederhana dengan gelang.*
*(Sumber : http://mathworld.wolfram.com/SimpleGraph.html diakses pada 8 Desember 2018)*

Sisi pada sebuah graf bisa memiliki orientasi arah. Berdasarkan orientasi arah pada sisi, sebuah graf dapat dibedakan menjadi dua jenis :

1) Graf tak – berarah
   Graf tak – berarah merupakan graf yang sisi – sisinya tidak memiliki orientasi arah. Pada jenis graf ini urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan. (u,v) = (v,u) merupakan sisi yang sama.
2) Graf berarah
   Graf berarah merupakan graf yang setiap sisinya memiliki orientasi arah. Pada graf berarah urutan pasangan simpul yang dihubungkan oleh sisi diperhatikan. (u,v) merupakan sisi yang berbeda dari (v,u).




*Gambar 2 : Graf tak berarah dan graf berarah.*
*(Sumber https://algorithmsinsight.wordpress.com/graph-theory-2/ diakses pada 8 Desember 2018)*

*3. Terminologi Dasar Graf*

Terdapat beberapa terminologi atau istilah terkait graf, antara lain :

a) Bertetangga
Dua buah simpul v dan w dikatakan bertetangga jika terdapat sebuah sisi yang menghubungkan simpul v dan w.

b) Bersisian
Sebuah sisi e dan sebuah simpul v dikatakan bersisian jika v merupakan salah satu titik ujung dari e.

c) Derajat
Derajat dari sebuah simpul v didefinisikan sebagai banyaknya sisi yang bersisian dengan v.

d) Lintasan
Sebuah lintasan pada sebuah graf merupakan sebuah barisan berhingga dari simpul-simpul $v_0$, $v_1$, … , $v_t$ sedemikian sehingga $v_i$ bertetangga dengan $v_{i+1}$. Panjang dari sebuah lintasan didefinisikan sebagai banyaknya sisi yang ada pada lintasan

e) Sirkuit
Sirkuit adalah sebuah lintasan yang simpul pertama dan terakhirnya merupakan simpul yang sama.

f) Terhubung
Sebuah graf dapat dikatakan terhubung jika untuk sebarang pasangan dua simpul, terdapat sebuah lintasan yang menghubungkan keduanya.

g) Upagraf
Misalkan G = (V,E) adalah sebuah graf. $G_1$ = ($V_1$,$E_1$) adalah upagraf dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$

h) Upagraf merentang
Misal G1 = ($V_1$,$E_1$) merupakan upagraf dari G = (V,E). G1 merupakan upagraf merentang dari G jika $V_1$ = V atau dengan kata lain G1 mengandung semua simpul dari G.

i) Graf berbobot
Graf berbobot ialah graf yang mana setiap sisinya memiliki bobot (harga) tertentu.

B. Pohon

*1. Definisi Pohon*

Pohon adalah sebuah graf tak-berarah yang terhubung dan tidak mengandung sirkuit di dalamnya.

**Lemma (Karakteristik Pohon)** : Misalkan G merupakan graf tak berarah yang terhubung dengan n simpul. Maka pernyataan – pernyataan di bawah ini ekivalen :

a) Tidak terdapat sirkuit pada G
b) G memiliki tepat n-1 sisi
c) Untuk setiap 2 simpul, terdapat tepat satu lintasan yang menghubungkan keduanya.
d) Penhapusan sebuah sisi dari G akan menyebabkan G tak terhubung.
e) G merupakan sebuah pohon.

*2. Pohon Merentang*

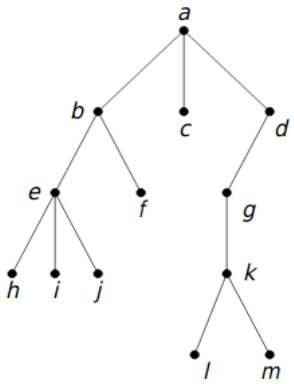
Pohon merentang dari sebuah graf yang terhubung adalah upagraf merentang yang berupa pohon. Pohon merentang dapat diperoleh dengan cara memutus sirkuit yang ada pada graf.

*3. Pohon Merentang Minimum*

Sebuah graf terhubung bisa saja memiliki lebih dari satu pohon merentang. Pohon merentang minimum adalah pohon merentang dari sebuah graf yang setiap sisinya mengandung nilai bobot tertentu dengan total bobot minimum.

*4. Pohon Berakar*

Pohon berakar adalah bohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi – sisinya diberi arah menjauhi akar.




*Gambar 3 : Pohon berakar*
*(Sumber : http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf diakses pada 8 Desember 2018)*

*5. Terminologi Pohon Berakar*

Pembahasan terminologi pohon berakar didasarkan pada gambar 3.

a) Anak dan Orangtua (Akar)
Simpul b, c, dan d, adalah anak dari simpul a dan a adalah orangtua dari b, c, maupun d.

b) Lintasan
Lintasan dari simpul a ke j adalah a, b, e, dan j. Panjang lintasannya adalah 3

c) Saudara kandung
f adalah saudara kandung dari e, tetapi g bukan saudara kandung dari e karena orang tua dari g dan e berbeda.

d) Upapohon
Setiap simpul dalam pohon berakar dapat dilihat sebagai akar dari sebuah pohon yang berisi anak – anaknya. Simpul b dengan simpul – simpul di bawahnya merupakan upapohon dari pohon dengan simpul a.

e) Derajat
Derajat sebuah simpul adalah jumlah upapohon atau jumlah anak pada simpul tersebut. Sebagai contoh derajat a adalah 3 dan derajat c adalah 0.

f) Daun
Daun merupakan simpul yang berderajat nol (atau tidak memiliki anak). Simpul h, i, j, f, l, dan m merupakan daun.

g) Simpul dalam
Simpul dalam adalah simpul yang mempunyai anak. Simpul b, d, e, g, dan k merupakan simpul dalam.

h) Aras (Level) atau Tingkat
Aras atau tingkat menyatakan kedalaman suatu simpul ditinjau dari akar. Simpul a berada pada aras 0, simpul b, c, d berada pada aras 1, dst.

i) Tinggi atau Kedalaman
Tinggi merupakan aras maksimum dari suatu pohon. Pohon pada gambar 2 memiliki tinggi 4.

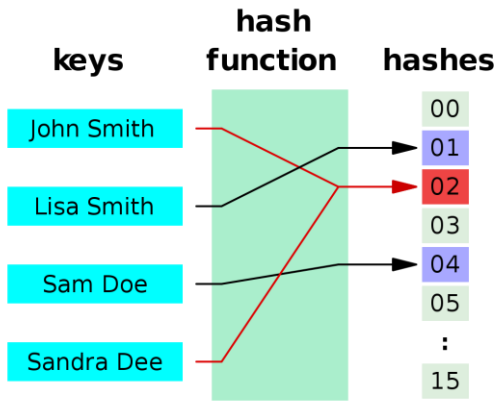
6. Pohon Biner

Pohon biner merupakan pohon yang setiap simpulnya memiliki paling banyak 2 buah anak. Kedua anak dari sebuah simpul dibedakan antara anak kiri *(left child)* dan anak kanan *(right child)*.

## C. Fungsi Hash

Fungsi Hash adalah fungsi yang menerima masukan sebuah string yang panjangnya sembarang dan menghasilkan sebuah string lain yang panjangnya tetap untuk berapapun panjang string masukanya. Keluaran dari fungsi hash umumnya berukuran jauh lebih kecil dari ukuran string semula.

Fungsi hash bekerja satu arah karena tidak bisa mengembalikan output ke string awal. Output dari fungsi hash belum tentu unik, karena bisa saja input yang berbeda menghasilkan output yang sama. Hal tersebut disebut *collision*.




*Gambar 4 : Ilustrasi fungsi hash*
*(Sumber : https://en.wikipedia.org/wiki/Hash_function diakses pada 7 Desember 2018)*

Sebagai contoh algoritma hash yang sangat sederhana adalah penggunaan prinsip modulo

$$f(x) = x \pmod 5$$

Fungsi diatas akan mengembalikan nilai 0, 1, 2, 3, atau 4 untuk setiap masukan yang diterima. Secara umum fungsi hash dapat ditulis dengan persamaan :

$$h = H(M)$$

M = pesan dengan ukuran sembarang
h = nilai hash

Beberapa sifat dari algoritma Hash yang baik adalah :

a) Fungsi hash dapat menerima blok data dengan ukuran berapa saja.
b) Fungsi hash akan menghasilkan keluaran yang tetap untuk input yang sama.
c) Fungsi Hash menghasilkan keluaran dengan panjang yang tetap.
d) Fungsi hash memiliki kemungkinan kemunculan setiap keluaran yang sama besar.
e) Untuk setiap nilai h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga H(x) = h.
f) Untuk setiap nilai x yang diberikan, tidak mungkin ditemukan y (≠ x) sehingga H(y) = H(x).
g) Tidak mungkin mencari pasangan x dan y sedemikian sehingga H(x) = H(y)

Pada kehidupan nyata, berbagai aplikasi fungsi hash yang biasa digunakan ialah :

a) Menjaga integritas data
Karena fungsi hash sangat teliti dan sensitive terhadap perubahan 1 bit pada masukan. Pada fungsi hash yang baik, perubahan 1 bit akan menyebabkan perubahan nilai hash yang cukup signifikan. Sehingga dengan membandingkan nilai hash baru dengan nilai hash lama dapat diketahui apakah sebuah pesan masih sama atau sudah pernah diubah sebelumnya.

b) Menghemat waktu pengiriman
Untuk memverifikasi data, tidak perlu mengirimkan data yang sesungguhnya. Hanya perlu mengirimkan nilai hashnya (yang berukuran jauh lebih kecil) ke komputer pusat.

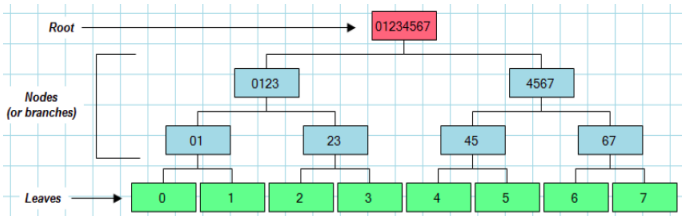
c) Menormalkan panjang data yang berbeda – beda
Misal kata sandi yang panjangnya bebas, dapat diseragamkan panjangnya di dalam basis data dengan menyimpan hasil hash dari kata sandi tersebut.

Berikut ini merupakan daftar fungsi hash yang sering digunakan :

a) MD2, MD4, MD5
b) Secure Hash Function (SHA)
c) Snefru
d) N – hash
e) RIPE-MD
f) dll

## III. PENERAPAN POHON DAN FUNGSI HASH PADA POHON MERKLE

### *1. Definisi Pohon Merkle*




*Gambar 5 : Struktur pohon merkle*
*(Sumber : https://www.codeproject.com/Articles/1176140/Understanding-Merkle-Trees-Why-use-them-who-uses-t diakses pada 7 Desember 2018 pukul 13.35 WIB)*

Pohon merkle merupakan sebuah pohon biner yang mana setiap daun dari pohon tersebut diberi label berupa hasil hash dari suatu blok data dan setiap simpul dalam dari pohon biner tersebut diberi label berupa hasil hash dari penggabungan nilai hash dari kedua anaknya. Proses hashing penggabungan nilai hash dari kedua anak suatu simpul dilakukan terus menerus hingga puncak dari pohon *(root hash)* tercapai.

*2. Membangun Pohon Merkle*

Untuk mensimulasikan pembentukan pohon merkle akan digunakan empat buah file a.txt, b.txt, c.txt, dan d.txt. Keempat file tersebut akan menjadi informasi yang diletakkan pada daun pohon merkle. Misalkna keempat file tersebut berisi string sebagai berikut :

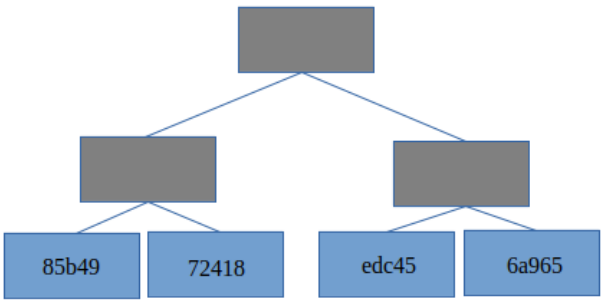
a.txt : “Mengerjakan makalah”
b.txt : “Matematika diskrit”
c.txt : “Sangat menyenangkan”
d.txt : “Semoga nilai saya A”

Sebelum meletakan keempat file diatas, setiap file harus melalui fungsi hash terlebih dahulu, pada simulasi ini digunakan fungsi hash md5 yang telah terinstalasi bawaan dari sistem operasi ubuntu 18.04 lts, dengan menjalankan perintah md5sum pada terminal seperti gambar berikut ini :

```
louis@louis-ThinkPad:~/merkletrees$ echo "Mengerjakan makalah" >> a.txt && md5sum a.txt &&
> echo "Matematika diskrit" >> b.txt && md5sum b.txt &&
> echo "Sangat menyenangkan" >> c.txt && md5sum c.txt &&
> echo "Semoga nilai saya A" >> d.txt && md5sum d.txt
85b4920c15d5f6f3bc2a91f4e1fbdc47  a.txt
72418c3091bac0eb9a4bc7ad2d802b80  b.txt
edc45141fbb4793fa4e577dac6b6c829  c.txt
6a965d8894a6f984cb09ba2ab13089f6  d.txt
louis@louis-ThinkPad:~/merkletrees$
```

*Gambar 6 : Hashing a.txt, b.txt, c.txt, dan d.txt*

Nilai hash dari empat file di atas dijadikan nilai dari daun pohon mekle yang akan dibangun. Pohon merkle ini hanya terdiri dari empat daun sehingga hanya memiliki tinggi sebesar 3. Untuk memudahkan, hanya 5 digit pertama dari nilai hash yang akan ditampilkan pada pohon merkle (pada kenyataannya seluruh string yang diperoleh berdasarkan fungsi hash diatas dijadikan nilai dari daun). Pohon merkle saat ini dapat digambarkan sebagai berikut :



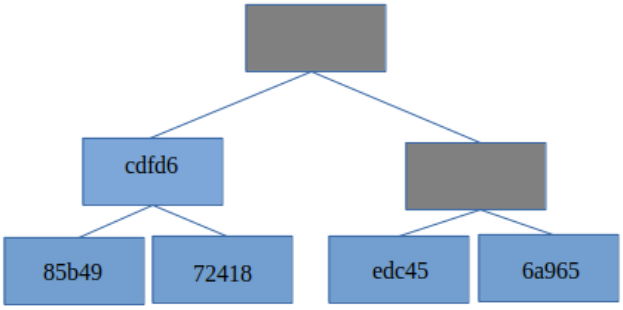

*Gambar 7: Pohon merkle 1*

Untuk membentuk orangtua dari simpul 85b49 dan 72418 dilakukan dengan cara menggabungkan nilai hash dari file a.txt dan nilai hash dari file b.txt seperti berikut **85b4920c15d5f6f3bc2a91f4e1fbdc47**72418c3091bac0eb9a4bc7ad2d802b80. Kemudian string tersebut dijadikan masukan untuk fungsi hash md5 sehingga menghasilkan keluaran cdfd67c3bc0e963898c03a5a06c75804.

```
louis@louis-ThinkPad:~/merkletrees$ md5sum<<<85b4920c15d5f6f3bc2a91f4e1fbdc4772418c3
091bac0eb9a4bc7ad2d802b80
cdfd67c3bc0e963898c03a5a06c75804  -
louis@louis-ThinkPad:~/merkletrees$
```

*Gambar 8 : Hashing H(a.txt) + H(b.txt)*

String diatas menjadi nilai dari orangtua simpul 85b49 dan 72418. Sehingga pohon merkle yang terbentuk sekarang ialah :



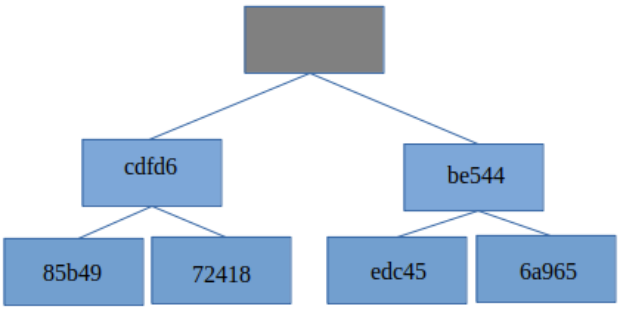

*Gambar 9: Pohon merkle 2*

Dengan cara yang sama, untuk mendapatkan nilai dari orangtua simpul edc45 dan simpul 6a965 dilakukan penggabungan nilai hash dari file c.txt dan d.txt yaitu **edc45141fbb4793fa4e577dac6b6c829**6a965d8894a6f984cb09ba2ab13089f6. Kemudian dijalankan fungsi hash md5 dengan masukan string tersebut sehingga diperoleh nilai be5446bbb391f675f9933f67b8152c96 yang akan menjadi nilai dari orangtua simpul edc45 dan 6a965

```
louis@louis-ThinkPad:~/merkletrees$ md5sum<<<edc45141fbb4793fa4e577dac6b6c8296a965d8
894a6f984cb09ba2ab13089f6
be5446bbb391f675f9933f67b8152c96  -
louis@louis-ThinkPad:~/merkletrees$
```

*Gambar 10 : Hashing H(c.txt) + H(d.txt)*

Sekarang sudah didapatkan nilai dari orangtua simpul edc45 dan 6a965 sehingga keadaan pohon merkle saat ini ialah :




*Gambar 11: Pohon merkle 3*

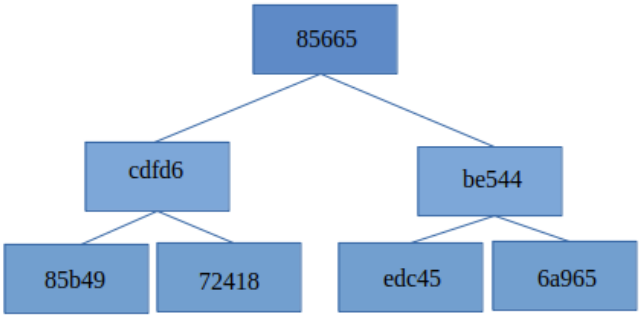
Selanjutnya, dengan cara yang sama untuk mencari nilai orangtua dari simpul cdfd6 dan be544, kedua nilai hash tersebut digabung menjadi :
**cdfd67c3bc0e963898c03a5a06c75804**be5446bbb391f675f9933f67b8152c96

dan dilakukan hashing dengan fungsi md5 untuk memperoleh nilai dari akar pohon merkle yang dibangun yaitu 856655e77cb9170e2db60b2b22dbcb6c.

```
louis@louis-ThinkPad:~/merkletrees$ md5sum<<<cdfd67c3bc0e963898c03a5a06c75804be5446b
bb391f675f9933f67b8152c96
856655e77cb9170e2db60b2b22dbcb6c  -
louis@louis-ThinkPad:~/merkletrees$
```

*Gambar 12: Root Hash*

Jadi pohon merkle yang telah selesai terbentuk dari empat buah file a.txt, b.txt, c.txt dan d.txt ialah :




*Gambar 13: Pohon merkle*

*3. Validasi Data*

Salah satu kegunaan penting dari pohon merkle adalah dapat membuktikan (memverifikasi) bahwa sebuah data benar berada pada sistem dengan efisien. Dengan mengacu pada gambar 12, misalkan seorang pengguna yang memiliki file b.txt ingin mengetahui apakah benar data yang ia punya berada pada pohon tersebut.

Asumsikan pengguna memiliki nilai hash root dari pihak yang berwenang, yang dalam simulasi ini bernilai 856655e77cb9170e2db60b2b22dbcb6c

Ketika pengguna meminta server untuk membuktikan kebenaran bahwa file b.txt benar berada pada pohon, server akan memberikan nilai 85b4920c15d5f6f3bc2a91f4e1fbdc47 sebagai hash dari file a.txt (saudara dari b.txt), dan nilai be5446bbb391f675f9933f67b8152c96 yang merupakan saudara dari orangtua simpul b.txt.

Berdasarkan informasi yang diberikan dari server, pengguna dapat melakukan perhitungan – perhitungan berikut :

a) Menggabungkan nilai hash dari b.txt dengan nilai hash dari a.txt dan melakukan hashing kembali dari nilai tersebut.
b) Menggabungkan nilai yang diperoleh dari poin a dengan string kedua yang diberikan server, lalu lakukan kembali hashing dengan nilai tersebut.
c) Jika hasil keluaran dalam poin b sama dengan root hash yang dimiliki pengguna maka terbukti bahwa file b.txt yang dimiliki pengguna benar berada pada pohon.

Selain itu, cara diatas juga membuktikan bahwa sistem yang mana pengguna mengajukan permintaan bukti merupakan sistem yang benar karena sistem lain tidak dapat memberikan data yang akurat mengenai nilai hash dari simpul – simpul lain yang ada pada pohon.

4. Kompleksitas Algoritma Traversal Pohon Merkle

Algoritma traversal pohon merkle merupakan algoritma yang digunakan untuk menjalankan penjabaran validasi data yang dijelaskan di poin 3. Algoritma ini akan menghasilkan nilai – nilai yang perlu dikirimkan server kepada pengguna yang meminta pembuktian kebenaran data kepada sistem. Ada beberapa algoritma yang pernah diusulkan oleh beberapa ilmuwan.

Salah satu algoritma yang paling efektif mengenai algoritma traversal pohon merkle ialah algoritma yang diajukan oleh Michael Syzdlo dalam karyanya yang berjudul *"Merkle Tree Traversal in Log Space and Time"* yang memakan waktu $2\log_2(N)$ dan ruang kurang dari $3\log_2(N)$, yang mana N merupakan jumlah daun pada pohon merkle. Michael Syzdlo juga membuktikan tidak akan ada algoritma traversal pohon merkle yang lebih baik dari kompleksitas ruang $O(\log_2 N)$ dan kompleksitas waktu $O(\log_2 N)$.

## IV. KELEBIHAN POHON MERKLE

Kelebihan – kelebihan yang ditawarkan struktur data pohon merkle ialah

a) Pohon merkle dapat membuktikan (memverifikasi) data dan dapat memastikan bahwa data disimpan secara konsisten dengan data pada versi sebelumnya.

b) Pohon merkle dapat mengurangi ukuran dari data secara signifikan ketika diperlukan pembuktian data. Misalkan pada teknologi bitcoin, seorang pengguna hanya perlu mendownload blok – blok data yang diperlukan sesuai dengan algoritma yang telah dijelaskan sebelumnya, tanpa perlu mendownload seluruh blok.

c) Proses verifikasi data terpisah dari data – data yang digunakan. Proses berjalan tanpa harus mengirimkan data yang dimiliki dan tidak perlu mendownload semua data – data yang ada pada server sehingga proses dapat berjalan dengan aman.

d) Algoritma traversal pohon merkle dalam proses validasi data merupakan algoritmaa yang efisien karena memiliki kompleksitas waktu dan ruang yang logaritmik.

## V. KESIMPULAN

Teori graf dan pohon serta pengetahuan mengenai fungsi hash merupakan dasar dari penggunaan pohon merkle. Struktur data pohon merkle sangat berguna dalam proses verifikasi data yang dapat dilakukan dengan efisien dan aman. Selain itu proses verifikasi data pada pohon merkle hanya menghabiskan waktu dan memori dalam bilangan logaritmik.

## VI. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih yang sebesar – besarnya kepada seluruh pihak yang telah membantu penulis dalam menyelesaikan makalah ini. Terutama kepada seluruh dosen mata kuliah IF2120 Matematika Diskrit yaitu Bapak Dr. Judhi Santoso M.Sc. , Bapak Dr.Ir.Rinaldi Munir, M.T. , dan Ibu Dra. Harlili M.Sc. yang telah berkenan membagikan ilmunya kepada seluruh peserta kuliah.

## REFERENCES


[1] Ederov, Boris.2007.*Merkle Tree Traversal Techniques*.Jerman:Darmstadt University of Technology, Department of Computer Science, Cryptography and Computer Algebra.

[2] Munir, Rinaldi. 2006. *Diktat Kuliah IF2120 Matematika Diskrit*. Bandung:Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.

[3] Satyanegara, Biyan. *Penerapan Fungsi Hash pada URL Shortening Service*. Bandung:Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.

[4] Supendi, Kevin. *Aplikasi Teori Bilangan pada Fungsi Hash dan Enkripsi dalam Sistem Keamanan Kata Sandi*. Bandung:Program Studi Teknik Informatika Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung.

[5] Szydlo, Michael.2004.*Merkle Tree Traversal in Log Space and Time*.RSA Laboratories, Bedford.

[6] Tang, Adrian. 2008. *IMO Training 2008 : Graph Theory*.Canada

[7] http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf diakses 8 Desember 17.42 WIB.

[8] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2017-2018/Fungsi-Hash-(2018).pdf diakses 8 Desember 17.43 WIB.

[9] https://www.codeproject.com/Articles/1176140/Understanding-Merkle-Trees-Why-use-them-who-uses-t, diakses pada 7 Desember 2018 pk 13.16 WIB.

[10] https://medium.com/coinmonks/merkle-trees-the-backbone-of-distributed-software-4fa0805132c6 diakses pada 7 Desember 2018 pk 13.15 WIB.

[11] https://hackernoon.com/merkle-trees-181cb4bc30b4 diakses pada 7 Desember 2018 pk 13.16 WIB.

[12] https://medium.com/byzantine-studio/blockchain-fundamentals-what-is-a-merkle-tree-d44c529391d7 diakses pada 7 Desember 2018 pk 13.16 WIB.

[13] https://hackernoon.com/merkle-tree-introduction-4c44250e2da7 diakses pada 7 Desember 2018 pk 13.15 WIB.


## 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, 9 Desember 2018

Louis Cahyadi
13517126