Penerapan Pohon Biner dalam Proses Pengamanan *Peer to Peer*

Eka Yusrianto Toisutta - NIM : 13504116
Program Studi Teknik Informatika, Institut Teknologi Bandung
Jalan Ganesha 10, Bandung
email: if14116@students.if.itb.ac.id

Abstract *– Makalah ini membahas mengenai penerapan pohon biner dalam proses pengamanan dalam Peer to Peer. Hal ini sangat penting mengingat interkoneksi peer to peer sangatlah rentan terhadap ancaman-ancaman yang dilakukan oleh salah satu dari dua belah pihak yang terlibat di dalamnya.*
Pada dasarnya peer to peer sangat mementingkan kejujuran antar kedua belah pihak tersebut. Hal ini akan mudah terwujud apabila kedua belah pihak saling mengenal satu sama lainnya. Apabila kedua belah pihak tersebut tidak saling mengenal, diperlukan suatu metoda untuk dapat memastikan data yang disalurkan benar-benar data yan asli. Salah satu metode yang digunakan adalah metode hash tree yang menerapkan struktur pohon dalam melakukan fungsi hash.

Kata Kunci*: Peer to Peer, Pohon Biner, Keamanan, Hash Tree*

1. PENDAHULUAN

Perkembangan dunia digital saat ini membuat lalu lintas pengiriman data semakin ramai. Hampir setiap orang melakukan transaksi data setiap harinya. Data yang dipertukarkan pun semakin bervariasi mulai dari segi jenis data hingga kepada ukuran data tersebut. Selain itu interkoneksi yang dipergunakan pun semakin beragam. Salah satunya adalah interkoneksi *peer to peer*.

Peer to peer secara langsung melibatkan dua belah pihak yang dihubungkan oleh jaringan internet dalam upaya pertukaran data. Terkadang dalam melakukan pertukaran data, kedua belah pihak belum tentu saling mengenal satu dengan lainnya. Oleh karena itu diperlukan suatu metode yang dapat memastikan bahwa data yang dipertukarkan tersebut valid dan tidak merupakan data palsu yang sekiranya dapat membahayakan penerimanya.

Fungsi *hash* adalah salah satu metode untuk menjamin keaslian suatu paket data yang dikirimkan. Selain itu fungsi *hash* turut berperan penting dalam mengkompresi paket data tersebut sehingga mempercepat proses transfer data. Hingga saat ini telah banyak jenis fungsi *hash* yang berkembang di dunia ini. Salah satunya adalah *hash tree* yang menggunakan struktur data berupa pohon biner dalam melakukan tugasnya

2. PEER TO PEER

Gambar 1: Jaringan peer to peer

Jaringan peer to peer merupakan jaringan komputer yang menggunakan beragam konektivitas antar parsipan yang terdapat dalam jaringan tersebut. Peer to peer digunakan sebagai alternatif dari jaringan yang telah umum dengan mengandalkan beberapa *server* sebagai penyedia data maupun informasi yang diperlukan. Dalam peer to peer setiap penggunanya diibaratkan sebagai *node* yang dihubungkan melalui sebuah koneksi mandiri yang sangat besar. Dengan begitu setiap pengguna peer to peer turut berperan sebagi penyedia informasi yang dapat diakses oleh pengguna-pengguna lainnya. Umumnya teknologi ini digunakan dalam *file sharing* dan data *real-time* seperti *telephony* dan *media streaming*.

Gambar 2: Jaringan komputer berbasis server

Tujuan utama munculnya koneksi peer to peer adalah setiap penggunanya menyediakan sumber daya yang dibutuhkan dalam transfer data antara lain *bandwidth*,

storage dan kemampuan komputasi. Hal ini menyebabkan semakin banyak jumlah pengguna yang terhubung maka kapasitas sistem secara keseluruhan meningkat. Selain itu sifat peet to peer yang sangat terdistribusi membuat seseorang dapat mencari sumber alternatif terhadap suatu data apabila terjadi kegagalan transfer dari suatu sumber.

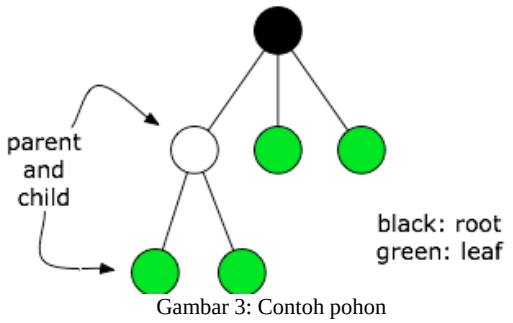
Jaringan peer to peer sering menjadi sarana yang digunakan oleh beberapa pihak untuk menyerang baik pihak lain maupun sistem itu sendiri. Beberapa ancaman terhadap peer to peer, antara lain:

- *Poisoning*, yaitu menyediakan data yang tidak sesuai dengan deskripsinya.
- *Polluting*, yaitu menyisipkan suatu bagian yang "jahat" ke dalam file yang valid
- *Freeloaders*, yaitu pihak-pihak yang menggunakan sistem tetapi tidak berkontribusi terhadap sistem tersebut
- Memasukkan virus ke dalam data yang ditransfer
- Serangan *denial of service*, membuat sistem berjalan sangat lambat atau merusak sistem secara keseluruhan
- *Filtering*, mengcegah terjadinya transfer data di dalam jaringan peer to peer
- *Identitiy attack*, memperoleh informasi tentang pengguna per to peer lalu digunakan untuk keperluan yang tidak sepantasnya.

3. POHON

Pohon merupakan salah satu pengembangan dari graf. Ciri khas dari pohon adalah sebuah graf tidak berarah yang terhubung dan tidak terdapat sirkuit di dalamnya. Diantara sekian banyak konsep mengenai graf, dapat dikatakan bahwa konsep pohon inilah yang paling penting. Hal ini dikarenakan konsep pohon merupakan konsep graf yang memiliki paling banyak penerapan di dunia nyata.

3.1. Pohon Bebas dan Pohon Berakar



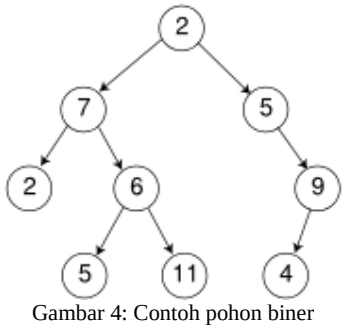
Gambar 3: Contoh pohon

Pada dasarnya pohon terbagi menjadi dua yaitu pohon bebas (*free tree*) dan pohon berakar (*rooted tree*). Pohon bebas merupakan sebuah graf yang seluruh simpul-simpul yang terdapat di dalamnya diperlakukan secara setara atau tidak dibedakan antara satu dengan lainnya. Sedangkan pada pohon berakar, terdapat satu buah simpul yang diperlakukan sebagai akar yang menjadi titik awal dari setiap penulusuran untuk mencapai simpul-simpul lainnya.

Jika dibandingkan, jumlah penerapan dari pohon berakar jauh lebih banyak jika dibandingkan dengan pohon bebas. Oleh karena itu muncullah beberpa terminology tambahan mengenai pohon berakar, antara lain:

- Anak dan Orang tua (*child and parent*), perlambangan keterhubungan antar simpul yang dihubungkan oleh sebuah sisi. Simpul yang diakses lebih dahulu akan menjadi orang tua terhadap simpul yang satu lagi. Begitu pula sebaliknya.
- Lintasan (*path*), urutan simpul yang harus dilalui untuk mencapai suatu simpul dari akar.
- Keturunan dan Leluhur (*descendant and ancestor*), pengembangan dari anak dan orang tua, hanya saja yang digunakan bukan sisi melainkan simpul.
- Saudara Kandung (*sibling*), hubungan antar simpul yang memiliki orang tua yang sama.
- Upapohon (*subtree*), bagian-bagian dari pohon yang apabila berdiri sendiri merupakan sebuah pohon dan tetap merupakan bagian dari pohon asalanya.
- Derajat (*degree*), jumlah anak yang dimiliki oleh suatu simpul merupakan derajat dari simpul tersebut.
- Daun (*leaf*), simpul yang tidak memiliki anak atau simpul yang berderajat nol.
- Simpul Dalam (*internal nodes*), simpul yang memiliki anak
- Aras atau Tingkat (*level*), jumlah simpul yang harus dilalui untuk mencapai suatu simpul (tidak termasuk dirinya sendiri) merupakan aras dari suatu simpul.
- Kedalaman (*depth*), aras maksimum yang dimiliki suatu pohon.

3.2. Pohon Biner



Gambar 4: Contoh pohon biner

Pohon biner merupakan salah satu jenis dari pohon

berakar yang sering digunakan. Pohon biner sebenarnya termasuk ke dalam golongan pohon n-*ary* dengan jumlah maksimal anak yang dimiliki oleh suatu simpul adalah dua. Hanya saja karena penggunaan dan penerapannya yang sangat luas, maka pohon biner diberikan kategori khusus di dalam pengelompokan pohon berakar.

Dikarenakan jumlah anak yang maksimal dua, pengaksesan suatu simpul oleh pohon biner ini menggunakan istilah anak kiri (*left child*) dan anak kanan (*right child*). Hal serupa juga diterapkan dalam penentuan upapohon, menggunakan istilah upapohon kiri (*left subtree*) dan upapohon kanan (*right subtree*).

Pohon biner merupakan struktur yang sangat penting dalam ilmu komputer. Penggunaannya antara lain:

- Pohon keputusan
- Pohon ekspresi
- Pohon pencarian biner
- Kode Huffman
- *Hash Tree*

4. FUNGSI HASH

Fungsi *Hash* pada dasarnya merupakan fungsi transformasi nilai masukan menjadi nilai keluaran yang berada dalam *range* nilai tertentu. Perbedaan mendasar antara fungsi *hash* dengan fungsi transformasi lainnya, yaitu hasil keluaran yang diharapkan harus selalu unik. Tujuannya adalah agar nilai keluaran tersebut dapat diproses kembali untuk mengembalikan nilai awal sebelum dilakukan proses *hash*.

Dalam prakteknya, fungsi *hash* banyak dipergunakan dalam bidang kriptografi dan basis data. Hanya saja pendekatan yang dilakukan sangatlah berbeda mengingat jenis masukan dan keluaran dari kedua bidang tersebut sangatlah berbeda.

4.1. Fungsi *Hash* dalam Basis Data

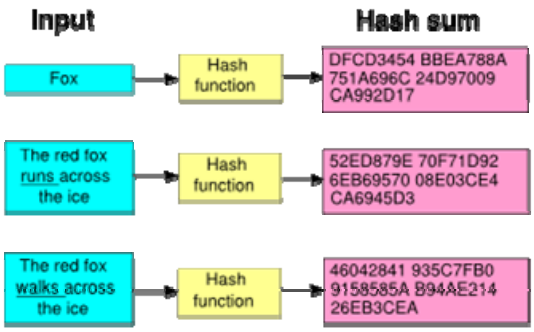
Pada basis data, Fungsi *hash* bertujuan untuk menentukan penempatan suatu data ke dalam memori utama. Tujuannya untuk membatasi jumlah transfer I/O menjadi hanya satu kali.

Dalam proses penyimpanan data, fungsi *hash* ini menerima masukan berupa nilai dari atribut yang menjadi *primary key*. Nilai tersebut lalu diolah dan menghasilkan blok yang merupakan lokasi penyimpanan data tersebut di dalam memori utama. Setelah lokasi penyimpanannya telah ditentukan, nilai-nilai dari data tersebut lalu dimasukkan ke dalam memori utama.

Selanjutnya dalam proses pengaksesan data, hasil dari pencarian data lalu dimasukkan ke dalam fungsi *hash* untuk menentukan blok data mana saja yang perlu diambil dari dalam memori utama. Setelah data tersebut berhasil diperoleh lalu ditampilkan ke hadapan pengguna.

4.2. Fungsi *Hash* dalam Kriptografi

Fungsi *hash* pada kriptografi melakukan transformasi terhadap input lalu kemudian mengambalikan keluaran berupa *string* yang ukurannya tetap. Nilai string inilah yang menjadi *hash value* yang selanjutnya akan merepresentasikan masukan awalnya tadi. Selain itu *hash value* ini juga berperan sebagai tanda tangan digital terhadap dokumen-dokumen yang ukurannya lebih besar.



Gambar 5: Contoh penggunaan fungsi *hash*

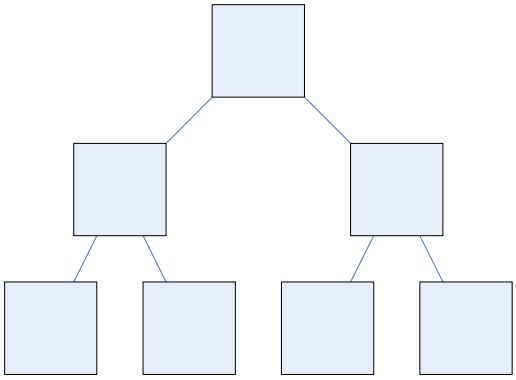
Dalam kriptografi tujuan penggunaan fungsi *hash* adalah untuk memastikan kesesuaian sebuah dokumen terhadap *hash value*-nya. Hal ini mungkin dilakukan karena apabila dokumen dasar mengalami perubahan, baik itu banyak maupun sedikit, maka nilai keluaran yang akan diperoleh akan jauh berbeda dengan nilai awalnya seperti yang ditunjukkan pada gambar 5 di atas.

Selain itu fungsi *hash* juga berfungsi untuk menyamarkan informasi yang bersifat rahasia agar tidak dapat dipecahkan oleh pihak lain yang tidak berhak.

5. HASH TREE

Hash tree merupakan salah satu contoh penerapan struktur pohon dalam ilmu komputer. *Hash tree* ini umumnya digunakan dalam jaringan peer to peer untuk memastikan bahwa data yang diterima seseorang dari orang lain merupakan data yang valid dan tidak berbahaya.

5.1. Struktur *Hash Tree*



Gambar 6: Contoh *hash tree*

Dalam penerapannya, *hash tree* menggunakan pohon biner sebagai struktur yang digunakan. Akan tetapi pada kenyataannya hampir seluruh jenis pohon berakar dapat digunakan sebagai stukturnya. Akar dari pohon ini dikenal juga dengan istilah *top hash*, *root hash*, maupun *master hash*. Pada akar inilah akan dilakukan proses *hash* terakhir.

5.2. Cara Kerja *Hash Tree*

Pertama-tama pengunduh data mencari data yang ingin dia unduh dari jaringan peer to peer yang dia masuki. Setelah dia menemukan data yang dicari, lalu dilakukan koneksi antar kedua peer yang terlibat. Selama proses transfer data inilah dilakukan *hash tree*.

Paket-paket data yang diterima lalu dimasukkan secara berurutan ke dalam daun-daun dari *hash tree* teresebut. Lalu di dalam setiap daun itu dilaksanakan fungsi *hash* baik itu SHA-1, *Whirlpool*, *Tiger*, maupun CRC. Hasil *hash* yang diperoleh dari daun-daun tersebut lalu diteruskan ke pada parentnya masing-masing. Sebagai contoh pada gambar 6 di atas, hasil *hash* dari simpul 0 merupakan hasil melakukan *hash* terhadap hasil *hash* dari simpul 0-1 dan 0-0 yang merupakan anak dari simpul 0. Atau dapat dituliskan sebagai berikut *hash* 0 = *hash* (*hash* 0-0 | *hash* 0-1).

Hal tersebut terus dilakukan hingga mencapai *root hash* yang merupakan proses *hash* terakhir. Setelah melalui tahapan ini lalu akan diperoleh hasil akhir dari rangkaian proses yang terjadi dalam *hash tree*. Selanjutnya tinggal membandingkan hasil yang diperoleh tersebut dengan hasil *root hash* yang disediakan sebelumnya. *Root hash* yang dijadikan acuan biasanya dapat diperoleh dari kawan maupun dari situs-situs internet yang terpercaya.

Apabila hasil *hash tree* yang diperoleh tidak sesuai dengan *root hash* yang menjadi acuan, maka sistem akan mencari *hash tree* lain di dalam jaringan peer to peer yang menghasilkan hasil *hash tree* yang sama persis dengan nilai *root hash* yang diacu. Setelah terbukti bahwa hasil yang diperoleh sama persis dengan nilai *root hash* yang diacu, maka dapat dipastikan bahwa data yang diunduh tersebut merupakan data yang valid.

5.3. Keunggulan *Hash Tree*

Walaupun strukturnya tidak jauh berbeda jika dibandingkan dengan *hash list*, terdapat perbedaan yang sangat besar dalam hal efisiensi pelaksanaannya. Pada *hash tree*, data yang akan diunduh akan dibagi-bagi ke dalam blok-blok data yang berukuran relatif kecil. Blok-blok data tersebut lalu langsung diperiksa pada daunnya masing-masing. Hal ini membuat kita dapat mendeteksi lebih awal apabila ada blok data yang rusak. Selain itu jika sebuah blok data tersebut rusak, maka hanya blok data tersebut saja yang di unduh ulang.

5.2. Kekurangan *Hash Tree*

Kelemahan utama yang dimiliki oleh sebuah struktur pohon apabila diterapkan dalam ilmu komputer adalah adanya fungsi-fungsi maupun prosedur-prosedur yang bersifat rekursif. Hal ini membuat waktu untuk melakukan komputasi menjadi lebih lama jika dibandingkan dengan struktur lainnya seperti *list*. Selain itu akan sangat berbahaya apabila ternyata *root hash* yang dijadikan acuan ternyata telah rusak. Selain itu seluruh struktur pohon yang digunakan mulai dari jumlah simpul hingga lintasan-lintasannya harus sama persis dengan struktur pohon yang digunakan untuk membuat *root hash* yang diacu. Hal ini membuat *hash tree* terkesan kaku karena harus mengikuti yang manjadi acuannya.

6. KESIMPULAN DAN SARAN

Dari pembahasan di atas maka dapat ditarik kesimpulan sebagai berikut:

1. Peer to peer banyak digunakan dalam proses transfer data antar penggunanya. Oleh karena itu faktor keamanan baik sistem maupun penggunanya menjadi sangat penting untuk diperhatikan.
2. Penerapan *hash tree* dalam peer to peer akan lebih *robust* dan efektif jika dibandingkan dengan penggunaan *hash list*.
3. Pohon merupakan representasi graf yang banyak diterapkan tidak hanya di dalam ilmu komputer tetapi juga di dalam kehidupan yang sebenarnya

Adapun saran yang dapat diutarakan yaitu:

1. Ukuran blok data yang akan dimasukkan ke dalam daun-daun dalam *hash tree* bergantung pada besarnya ukuran file dan banyaknya daun yang tersedia. Akan lebih baik jika yang menjadi relatif adalah jumlah daun yang tersedia pada pohon tersebut.
2. Struktur pohon yang berbeda dapat menghasilkan keluaran yang berbeda oleh karena itu perlu disamakan bentuk pohon yang digunakan sesuai dengan struktur pohon yang digunakan penyedia *root hash*.

3. Untuk memperrumit pemecahan *hash tree*, dapat pula digunakan metode *hash* yang berbeda pada setiap simpulnya.

DAFTAR REFERENSI

[1]Munir, Rinaldi, *Diktat Kuliah IF2151 Matematika Diskrit*, Departemen Teknik Informatika Institut Teknologi Bandung, 2004.

[2]Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2006.

[3]http://en.wikipedia.org/wiki/Cryphtographic_*hash*_function

[4]http://en.wikipedia.org/wiki/Peer to peer

[5]http://en.wikipedia.org/wiki/Tree data structure

[6]http://www.nist.gov.dads/HTML/tree.html