**TUGAS KELOMPOK**

**KRIPTOGRAFI**

**( MATERI TENTANG HASH )**

****

**GIHFAR PRAMADAN 201581098**

**ABIB RIYANSYAH 201581099**

**BAYU ABIE RASMANA 201581101**

**BAGAS LUTFIARTA 201581066**

**ANDRE KURNIAWAN 201581045**

**FANOLO GULO 201581006**

**IMAM MUJAHID 201581121**

**NUR AGUNG PATRIA 201581092**

**RIZKI RATNO PUTRO 201581053**

**DADING TINTON SETIONO 201581020**

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS ESA UNGGUL**

**JAKARTA**

**2018**

**Fungsi Hash** merupakan fungsi yang menerima masukan string yang panjangnya sembarang kemudian mentransformasikannya menjadi string keluaran yang panjangnya tetap (fixed) dan umumnya berukuran jauh lebih kecil daripada ukuran string semula. Di samping ini merupakan skema fungsi hash di mana ukuran dari masukan sembarang, namun menghasilkan output yang ukurannya tetap.

Persamaan fungsi hash :

h = H(M)

M = pesan ukuran sembarang

h = nilai hash atau pesan singkat (message digest)

h <<<< M

Fungsi Hash juga memiliki nama-nama lain seperti :

* Fungsi kompresi (compression function);
* Cetak-jari (fingerprint);
* Cryptographic checksum;
* Message Integrity Check (MIC);
* Manipulation Detection Code (MDC).

Penggunaan fungsi hash sangat mudah dijumpai, sebagai contoh adalah pencocokan kunci antara client dan server. Sistem akan membandingkan nilai hash kunci yang dimasukkan pengguna dengan nilai hash yang tersimpan pada server. Jadi, informasi yang dipertukarkan tidak berupa pesan asli.

Pada fungsi hash terdapat istilah fungsi hash satu arah (one way function) yang merupakan fungsi hash yang bekerja dalam satu arah. Artinya pesan yang sudah diubah menjadi message digest tidak dapat dikembalikan lagi menjadi pesan semula (irreversible). Ada pun sifat-sifat dari fungsi hash satu arah, yaitu :

1. Fungsi H dapat diterapkan pada blok data berukuran berapa saja.

2. H menghasilkan nilai (h) dengan panjang tetap (fixed length output).

3. H(x) mudah dihitung untuk setiap nilai x yang diberikan.

4. Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga H(x)=h. Itulah sebabnya fungsi H dikatakan fungsi hash satu arah(one way hash function).

5. Untuk setiap x yang diberikan, tidak mungkin mencari y tidak sama dengan x sedemikian sehingga H(y)=H(x).

6. Tidak mungkin mencari pasangan x dan y sedemikian sehingga H(x)=H(y).

Ada pun beberapa fungsi hash satu arah yang pernah dibuat, antara lain :

* MD2(Message Digest2), MD4(Message Digest4), MD5(Message Digest5);
* Secure Hash Function (SHA);
* Snefru;
* N-has;
* RIPE-MD, dan lain sebagainya.

Ada pun manfaat dari fungsi hash satu arah, antara lain :

1. Menjaga integritas data

Fungsi hash sangat peka terhadap perubahan 1 bit pada pesan. Pesan berubah 1 bit maka nilai hash berubah sangat signifikan.

2. Menghemat waktu pengiriman

Misalnya untuk memverifikasi sebuah salinan arsip dengan arsip asli. Di sini salinan dokumen berada di tempat yang jauh dari basis data arsip asli. Daripada mengirim salinan arsip tersebut secara keseluruhan ke komputer pusat (yang membutuhkan waktu transmisi lama), lebih baik mengirimkan message digestnya. Jika message digest salinan arsip sama dengan message digest arsip asli, berarti salinan arsip tersebut sama dengan arsip master.

3. Menormalkan panjang data yang beraneka ragam

Misalnya password yang panjangnya bebas (minimal 8 karakter). kemudian password disimpan di komputer host (server) untuk keperluan otentikasi pemakai komputer. Untuk menyeragamkan panjang field password di dalam basis data, password disimpan dalam bentuk nilai hash (panjang nilai hash tetap).

**Kelebihan dan Kekurangan**

· Binary search dan interpolation search memberikan hasil pencarian dalam waktu yang cukup singkat. Kedua algoritma pencarian ini mengharuskan data (record) telah diurutkan, sehingga cukup lama.

· Hashing bertujuan untuk mendapatkan posisi (lokasi, alamat) record secara langsung (immediate, direct) pada waktu dicari.

· Suatu hashing function yang baik harus memenuhi dua syarat:

1. penghitungannya cepat

2. menghasilkan nilai yang terdistribusi merata

Hashing merupakan salah satu struktur data yang digunakan dalam penyimpanan data sementara. Tujuan dari hashing adalah untuk mempercepat pencarian kembali dari banyak data yang disimpan. Hashing menggunakan suatu teknik penyimpanan sehingga waktu yang dibutuhkan untuk penambahan data(insertions), penghapusan data(deletions), dan pencarian data(searching) relatif sama dibanding struktur data atau algoritma yang lain.

Kelebihan dari hashing:

· Hashing relatif lebih cepat

· Kecepatan dalam insertions, deletions, maupun searching relatif sama

Kekurangan dari hash table di bahas pada halaman berikutnya.

Hashing menggunakan memori penyimpanan utama berbentuk array dengan tambahan algoritma untuk mempercepat pemrosesan data. Pada intinya hashing merupakan penyimpanan data menggunakan key value yang didapat dari nilai data itu sendiri. Dengan key value tersebut didapat hash value. Jadi hash function merupakan suatu fungsi sederhana untuk mendapatkan hash value dari key value suatu data

Yang perlu diperhatikan untuk membuat hash function :

· ukuran array/table size(m),

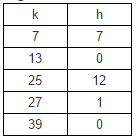
· key value/nilai yang didapat dari data(k),

· hash value/hash index/indeks yang dituju(h).

Berikut contoh penggunaan hash table dengan hash function sederhana yaitu memodulus key value dengan ukuran array : h = k (mod m)

Misal kita memiliki array dengan ukuran 13, maka hash function : h = k (mod 13).

Dengan hash function tersebut didapat :



Perhatikan range dari h untuk sembarang nilai k.

Maka data 7 akan disimpan pada index 7, data 13 akan disimpan pada index 0, dst..

Untuk mencari kembali suatu data, maka kita hanya perlu menggunakan hash function yang sama sehingga mendapatkan hash index yang sama pula.

Misal : mencari data 25 → h = 25 (mod 13) = 12

Namun pada penerapannya, seperti contoh di atas terdapat tabrakan (collision) pada k = 13 dan k = 39. Collision berarti ada lebih dari satu data yang memiliki hash index yang sama, padahal seperti yang kita ketahui, satu alamat / satu index array hanya dapat menyimpan satu data saja.

Untuk meminimalkan collision gunakan hash function yang dapat mencapai seluruh indeks/alamat. Dalam contoh di atas gunakan m untuk me-modulo k. Perhatikan bila kita menggunakan angka m untuk me-modulo k maka pada indeks yang lebih besar dari dan sama dengan m di hash table tidak akan pernah terisi (memori yang terpakai semakin kecil), kemungkinan terjadi collision juga semakin besar.

Karena memori yang terbatas dan untuk masukan data yang belum diketahui tentu collision tidak dapat dihindari.

Berikut ini cara-cara yang digunakan untuk mengatasi collision :

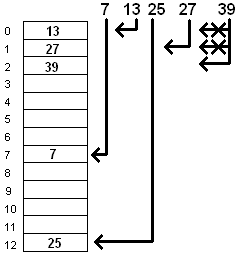
1. Closed hashing (Open Addressing)

Close hashing menyelesaikan collision dengan menggunakan memori yang masih ada tanpa menggunakan memori diluar array yang digunakan. Closed hashing mencari alamat lain apabila alamat yang akan dituju sudah terisi oleh data. 3 cara untuk mencari alamat lain tersebut :

· Linear Probing

Apabila telah terisi, linear probing mencari alamat lain dengan bergeser 1 indeks dari alamat sebelumnya hingga ditemukan alamat yang belum terisi data, dengan rumus

(h+1) mod m.



Penerapannya pada contoh di atas :

Data 7, 13, 25, 27 dapat dimasukan ke dalam array tanpa masalah.

Collison yang terjadi pada data 39 di indeks 0 diselesaikan dengan bergeser ke indeks 1. Karena indeks 1 juga telah terisi maka data 39 bergeser ke indeks 2. Karena indeks 2 masih kosong maka data 39 dapat dimasukkan.

· Quadratic Probing

Quadratic Probing mencari alamat baru untuk ditempati dengan proses perhitungan kuadratik yang lebih kompleks. Tidak ada formula baku pada quadratic probing ini,anda dapat menentukan sendiri formula yang akan digunakan.

Contoh formula quadratic probing untuk mencari alamat baru:

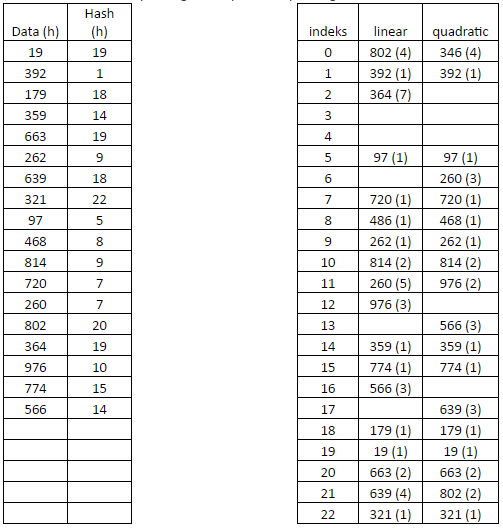
h,(h+i2)mod m,(h-i2)mod m, … ,(h+((m-1)/2)2)mod m, (h-((m-1)/2)2)mod m

dengan i = 1,2,3,4, … , ((m-1)/2)

Mksud formula di atas adalah jika alamat h telah terisi, maka alamat lain yang digunakan adalah (h+1)mod m, jika telah terisi gunakan alamat (h-1)mod m, jika telah terisi gunakan alamat (h+4)mod m, jika telah terisi gunakan alamat (h-4)mod m, dan seterusnya.

Jadi jika m=23,maka nilai maksimal i adalah : ((23-1)/2)=11.

Berikut ini contoh penyimpanan sejumlah data menggunakan hash table dengan ‘collision solver’ ‘linear probing’ dan ‘quadratic probing’



· Double hashing

Sesuai dengan namanya, alamat baru untuk menyimpan data yang belum dapat masuk ke dalam table diperoleh dengan menggunakan hash function lagi. Hash function kedua yang digunakan setelah alamat yang dihasilkan oleh hash function awal telah terisi tentu saja berbeda dengan hash function awal itu sendiri.

Contoh dengan hash function utama h1= k mod m kita dapat menggunakan hash function kedua h2 = (k+s) mod m

dengan s = c – (k mod c), c = konstanta

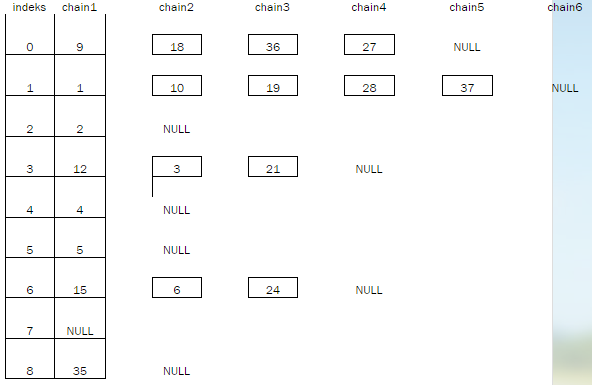
Kelemahan dari closed hashing adalah ukuran array yang disediakan harus lebih besar dari jumlah data. Selain itu dibutuhkan memori yang lebih besar untuk meminimalkan collision.

2. Open hashing (Separate Chaining)

Pada dasarnya separate chaining membuat tabel yang digunakan untuk proses hashing menjadi sebuah array of pointer yang masing-masing pointernya diikuti oleh sebuah linked list, dengan chain (mata rantai) 1 terletak pada array of pointer, sedangkan chain 2 dan seterusnya berhubungan dengan chain 1 secara memanjang.

Jika hash value menunjukkan index pada tabel (chain 1) telah terisi maka program akan membuat chain selanjutnya (ingat malloc!) dan menyimpan nilai data pada chai tersebut. Jika ternyata chain selanjutnya pun telah terisi maka program kembali membuat chain baru dam menyimpan data pada chain yang baru dibuat tersebut.

Contoh, data yang aka dimasukkan ke dalam hash table berukuran 9 dengan separate chaining adalah : 1,2,5,10,15,12,35,3,4,6,21,19,28,9,18,37,24,36,27 secara berurutan,maka (hash function → h = k mod 9) :



Kelemahan dari open hashing adalah bila data menumpuk pada satu/sedikit indeks sehingga terjadi linked list yang panjang.

Contoh program hash :

<?php

$p = empty($\_POST['p']) ? null : $\_POST['p'];

?>

<html>

<head><title>Tugas Hash</title></head>

<style type="text/css">

table {border-collapse: collapse;}

table, td, th {border: solid 1px #ccc;}

th {background: #e1e1e1;border-color: #999;}

td, th {padding: 0.25em;}

td.algo {font-weight: bold;}

tr.on td {background: #f0f0f0;}

</style>

<body>

<h1>String hashing</h1>

<div class="box-body">

Kelompok : <br />

1. Bayu Abie Rasmana (201581101)<br />

2. Gihfar Pramadan (201581098)<br />

3. Imam Mujahid Assalafy (201581121)<br />

4. Habib Riyansyah (201581099)<br />

5. Fanolo Gulo (201581006)<br />

6. Bagas Lutfiarta (201581066)<br />

7. Andre Kurniawan (201581045)<br />

8. Agung Patria (201581092)<br />

9. Rizki Ratno (201581053)<br />

</div><!-- /.box-body -->

</div>

<form method="post" action="<?php echo basename(\_\_FILE\_\_) ?>">

<p><label for="p">Enter a string to hash:</label><br /><input id="p" type="text" name="p" value="<?php echo $p ?>" /></p>

<p><input type="submit" name="submit" value="Hash It" /></p>

</form>

<?php /\* If there is a posted value use it \*/ ?>

<?php if ($p): ?>

<hr />

<h2>Table of hash values for <em><?php echo $p ?></em> based on algorithm</h2>

<table>

<tr>

<th>Algorithm</th>

<th>Hashed value of <em><?php echo $p ?></em></th>

</tr>

<?php /\* Loop through each hash algorithm, colorizing every other row \*/ ?>

<?php $on = false; foreach (hash\_algos() as $algo): ?>

<tr<?php if ($on): ?> class="on"<?php endif; ?>>

<td class="algo"><?php echo $algo ?></td>

<td class="hash"><?php echo hash($algo, $p) ?></td>

</tr>

<?php $on = !$on; endforeach; ?>

</table>

<?php endif; ?>

</body>

</html>