Perbandingan Keoptimalan Fungsi Hash Untuk Membuat Audio Fingerprint Berdasarkan Ukuran Berkas Fingerprint Pada Shazam dan Echoprint

Diani Pavitri Rahasta 13509021[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13509021@std.stei.itb.ac.id

***Abstrak*—*Audio fingerprint* merupakan representasi sebuah data musik dalam bentuk rangakaian *string*. Dalam penerapannya, untuk bisa mengahsilkan sebuah *audio fingerprint*, dibutuhkan sebuah fungsi *hash* yang sesuai untuk membentuk rangkaian *string* yang bisa merepresentasikan data musik. Kinerja *audio fingerprint* yang paling disoroti pengguna adalah dari kecepatan proses pengaksesan. Salah satu faktor paling menentukan dalam kecepatan akses adalah ukuran *audio fingerprint*. Ada dua metode yang menggunakan metode pembuatan *audio fingerprint* yang paling populer namun menggunakan fungsi *hash* yang berbeda. Makalah ini akan menunjukkan hasil eksplorasi terhadap fungsi *hash* yang digunakan oleh kedua metode tersebut berdasarkan ukuran berkas *audio fingerprint* yang dihasilkan.**

Termin Indeks*— *hash*, *audio fingerprint

I. PENDAHULUAN

Kriptografi modern saat ini memiliki bermacam cara untuk mengubah tulisan biasa (*plain text*) menjadi tulisan kode (*cipher text*). Salah satu cara yang bisa digunakan adalah memanfaatkan fungsi *hash* untuk menyamarkan pesan yang sebenarnya. Fungsi *hash* adalah sebuah fungsi yang bisa mengubah rangkaian *string* dengan panjang yang berbeda-beda menjadi serangkaian *string* lain dengan panjang yang tetap.

Pada pemanfaatannya, fungsi *hash* banyak digunakan pada penyimpanan *password*. Beberapa fungsi seperti MD5 dan SHA adalah fungsi-fungsi *hash* yang paling populer. Mengapa fungsi *hash* dipilih untuk menyamarkan pesan? Karena fungsi *hash* sendiri bersifat unik. Jika pesan diubah satu huruf saja dan diberi fungsi yang sama, maka *cipher text* yang dihasilkan bisa jadi jauh berbeda. *Cipher text* bisa dikembalikan menjadi *plain text* asal ada kunci yang tepat dan sesuai.

Dalam perkembangannya, fungsi *hash* itu sendiri tidak hanya dimanfaatkan di dunia kriptografi saja, tapi juga dimanfaatkan untuk bidang-bidang yang lain. Salah satu bidang yang memanfaatkan fungsi *hash* untuk kerjanya adalah bidang penyimpanan dan akses langsung musik. Bagaimana caranya? Fungsi *hash* dimanfaatkan untuk mengubah rangkaian frekuensi pada musik menjadi rangkaian *string* tertentu. Mekanisme ini disebut sebagai *audio fingerprint*.

Audio fingerprint sendiri merupakan sebuah bentuk representasi dari musik dalam bentuk rangkaian *string*. Bentuk ini nantinya dimanfaatkan untuk mempermudah pengaksesan langsung dari musik itu sendiri. Pengaksesan langsung maksudnya antara lain pencarian, pencocokan, dan sebagainya.

Salah satu tolok ukur efektivitas dari *audio fingerprint* adalah kecepatan akses. Kecepatan akses ini salah satunya dipengaruhi oleh ukuran berkas *audio fingerprint* yang ada. Makalah ini akan menilai fungsi *hash* yang digunakan dalam pembuatan *audio fingerprint* berdasarkan ukuran dari berkas *audio fingerprint* yang dihasilkan. Perlu diketahui bahwa tidak seperti fungsi *hash* pada umumnya dalam kriptografi, fungsi *hash* yang digunakan untuk memberntuk *audio fingerprint* tidak memiliki panjang keluaran yang tetap.

II. FUNGSI *HASH*

Seperti sudah disebutkan sebelumnya, fungsi *hash* adalah sebuah metode kriptografi modern yang saat inis edang cukup populer. Metode ini populer sebab kemampuannya yang luar biasa dalam mengubah sebuah *plain text* ke dalam sebuah *cipher text.* Kemampuannya untuk menghasilkan *cipher text* yang benar-benar unik, karena setiap bagian *plain text* diganti sedikit saja hasil *cipher text* nya akan jauh berbeda, serta sulitnya metode ini untuk dirusak oleh orang untuk mendapatkan *plain text* dari suatu *cipher text* yang tersedia, membuat fungsi *hash* menjadi salah satu metode kriptografi modern yang sulit digantikan. [5]

Secara konsep, fungsi *hash* adalah sebuah metode yang bisa digunakan untuk menciptakan *cipher text* yang bersifat unik dari *plain text* yang diberikan sebagai masukan. Fungsi *hash* akan mengganti atau mengubah teks yang ada untuk membentuk sebuah *cipher text* yang unik. Unik sebab *cipher text* yang dihasilkan akan berbeda apabila sedikit saja bagian dari *plain text* diganti. Dalam istilah fungsi *hash*, *cipher text* ini bisa disebut juga dengan *hash value*. *Hash value* ini biasanya berupa rangkaian *string* pendek yang terdiri atas huruf dan/atau angka yang bersifat acak. Seperti representasi data biner

yang kemudian ditulis dalam notasi heksadesimal.

Sebuah fungsi *hash* sebenarnya adalah sebuah fungsi matematis yang mengambil panjang untai dari masukan yang berupa *string*. Dalam istilah fungsi *hash*, masukan ini disebut sebagai *pre-image*. Fungsi *hash* yang digunakan tersebut lantas akan menghasilkan sebuah *string* keluaran dengan panjang tetap dan biasanya jauh lebih pendek dari panjang masukan. Ini disebut sebagai *message digest*.

Fungsi *hash* yang ada lantas digunakan untuk memberikan sebuah perlakuan kepada *pre-image* yang ada tersebut, yaitu untuk mengubahnya menjadi sebuah *hash value* yang unik. Selain unik, *hash value* yang dihasilkan haruslah benar-benar merepresentasikan *pre-image* yang ada tersebut.

Fungsi *hash* yang bersifat satu arah (*one-way hash function*) adalah fungsi *hash* yang bekerja satu arah. Maksud satu arah di sini adalah fungsi *hash* tersebut dapat dengan mudah menghitung *hash value* sebagai keluaran dari *pre-image* masukan, tapi akan sangat sulit untuk menghitung atau mengembalikan nilai *pre-image* dari *hash value* yang disediakan.

Jika digambarkan secara lebih jelas, sebuah fungsi satu arah $h(M)$, yang bekerja pada suatu *pre-image* pesan M dengan panjang sembarang akan mengembalikan sebuah nilai *hash* H yang panjangnya tetap. Jika dituliskan lebih jelas dalam notasi matematis, gambarannya kurang lebih sebagai berikut:

$H = h(M)$, dengan H pasti memiliki panjang l

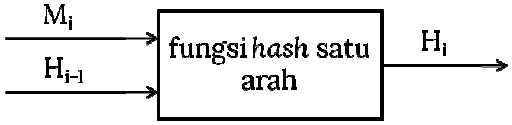
Jika kita membahas mengenai fungsi *hash*, bukan hanya fungsi *hash* satu arah yang mampu menerima masukan dengan panjang yang sembarang kemudian menghasilkan keluaran dengan panjang yang selalu tetap. Namun, fungsi *hash* satu arah memiliki beberapa kriteria penentu yang membuatnya berbeda dengan fungsi *hash* lain yang tidak bersifat satu arah. Kriteria tersebut antara lain adalah:

- Diberikan M, mudah menghitung H
- Diberikan H, sulit menghitung M agar $h(M) = H$.
- Diberikan M, sulit menemukan pesan lain, M', agar $h(M) = h(M')$.

Pada praktiknya, fungsi *hash* yang bersifat satu arah dikembangkan berdasarkan ide dari sebuah fungsi kompresi. Fungsi satu arah ini menghasilkan nilai *hash* dengan ukuran *n* jika diberikan sebuah masukan dengan ukuran *t*. Masukan untuk fungsi kompresi adalah sebuah blok pesan dan hasil dari blok teks yang sebelumnya. Sehingga *hash* dari suatu blok M adalah

$H_i = f(M_i, H_{i-1})$, di mana
H_i adalah *hash value* saat ini
M_i adalah blok pesan saat ini
H_{i-1} adalah blok teks sebelumnya

Jika digambarkan dalam bentuk bagan, kurang lebih akan terlihat seperti ini:



Gambar 1 - Fungsi Hash Satu Arah

Fungsi *hash* sangat berguna untuk menjaga kerahasiaan dan jug integritas data. Oleh karena itulah penggunaannya di bidang kriptografi menjadi semakin populer. Sudah banyak algoritma fungsi *hash* yang diciptakan untuk bisa memberikan kerahasiaan dan integritas data yang tinggi. Namun di antara banyaknya algoritma yang ada, ada dua algoritma yang umum digunakan untuk fungsi *hash*. Kedua algoritma tersebut adalah MD5 dan SHA (*Secure Hash Algorithm*). Selain bentuknya yang satu arah, algoritma untuk fungsi *hash* yang baik adalah yang menghasilkan sedikit *collision* sehingga tidak mengganggu *cipher text* yang sudah ada.

III. PENGGUNAAN FUNGSI *HASH* UNTUK PEMBUATAN *AUDIO FINGERPRINT*

Audio fingerprint merupakan sebuah bentuk penyederhanaan dari bentuk frekuensi sebuah data musik. Bentuk ini biasanya merupakan kumpulan atau rangkaian *string* acak yang dengan sedemikian rupa merupakan representasi dari frekuensi dari berkas musik. *Audio fingerprint* dibuat karena kebutuhan masa kini yang menuntut adanya akses langsung yang cepat untuk berkas musik. Akses langsung yang dimaksud di sini adalah pencarian, pencocokan, dan sebagainya yang tidak perlu menggunakan metadata dari berkas musik itu sendiri melainkan langsung dilakukan pada data musik yang ada. *Audio fingerprint* digunakan sebab jauh lebih mudah dan cepat untuk melakukan pengaksesan kepada data yang berbentuk rangkaian *string* dibandingkan dengan data yang berbentu musik, berupa frekuensi atau apapun representasinya yang sulit dilakukan pencocokan jika tidak menggunakan alat khusus. *Audio fingerprint* mempermudah proses tersebut dengan mengubah semua bentuk data musik ke dalam representasi rangkaian *string* sederhana.

Audio fingerprint dibuat dengan cara mengubah data musik, yang berupa frekuensi menjadi sebuah rangkaian *string* acak. Untuk melakukan hal ini, ada beberapa pendekatan yang dilakukan. Salah satu pendekatan yang terkenal dan banyak dimanfaatkan adalah pendekatan pencarian puncak spektogram dari frekuensi musik. Puncak-puncak ini dilihat posisinya, berada di mana saja, kemudian diubah menjadi sebuah rangkaian tertentu sebelum dikenai fungsi untuk diubah menjadi rangkaian *string*. Secara sederhana bisa dibilang prosesnya memang mengubah frekuensi musik ke dalam bentuk rangkaian *string* acak.

Jika digambarkan secara sederhana, berikut langkah-langkah pembuatan *audio fingerprint*:

1. Data audio diubah ke dalam representasi frekuensi sinyal.
2. Pendeteksian posisi puncak spektogram dari frekuensi sinyal yang ada.
3. Pemetaan posisi tersebut ke dalam sebuah peta frekuensi.
4. Pemberian sebuah fungsi ke dalam peta tersebut sehingga menghasilkan sebuah kumpulan *string* yang seolah-olah bersifat acak.

Jika ditinjau dari sudut pandang ilmu kriptografi, *audio fingerprint* sendiri bisa dibilang sebagai sebuah *cipher text* hasil dari sebuah fungsi *hash*. Di mana *plain text* nya adalah peta posisi puncak spektogram yang menjadi masukan dari fungsi. Selanjutnya fungsi yang digunakan itu sendiri bisa disebut sebagai fungsi *hash* karena fungsi tersebut mengubah masukan yang berupa data menjadi sebuah rangkaian *string* dengan susunan yang acak dengan ukuran tertentu. Perbedaan yang mungkin bisa dibilang sedikit krusial adalah bahwa fungsi yang digunakan tidak selalu menghasilkan rangkaian *string* dengan panjang yang tetap, meskipun mungkin memang ada sebuah rentang di mana panjang ukuran *string*nya pasti berada.

Pengukuran kemampuan *audio fingerprint* sebuah data musik sendiri, meskipun ada beragam bentuk dan cara, memiliki parameter yang relatif sama. Menurut (Deng, 2011), ada empat parameter untuk mengukur kemampuan *audio fingerprint* sebuah data musik:

1. Diskriminasi

Diskriminasi berarti bagaimana sebuah *audio fingerprint* mampu membedakan dengan tepat satu berkas musik dengan berkas lainnya. Sebuah *audio fingerprint* harus mampu membedakan dengan tepat, meskipun mungkin ada fitur-fitur yang mirip dari dua berkas musik.

2. Ketahanan

Ketahanan di sini berarti bagaimana *audio fingerprint* bisa bertahan hanya mencetak *audio fingerprint* dari data musik. Ketahanan sebuah *audio fingerprint* akan dibilang semakin tinggi jika *noise* yang mungkin ada pada saat *query* dibuat tidak tercetak dalam *audio fingerprint* dan tidak mengganggu pencarian.

3. Granularitas

Granularitas menyatakan durasi yang dibutuhkan agar sampel musik dari *query* bisa dibandingkan dengan data yang ada. Hal ini penting sebab tidak ada orang yang mau merekam sampai semenit untuk mengetahui apa judul lagu yang mereka rekam itu. Semakin singkat durasi yang dibutuhkan, maka semakin tinggi granularitas dari suatu fingerprint.

4. Waktu pencarian

Hal yang bisa dibilang paling penting dan paling diperhatikan oleh pengguna adalah waktu pencarian. Percuma membangun *audio fingerprint* yang mampu membuang *noise* sama sekali jika waktu pencariannya lama. Pengguna akan lebih bisa mentoleransi jika diminta mengulang memberikan *query* daripada harus menunggu lama dan belum tentu berhasil.

Seperti sudah disebutkan, dari sudut pandang pengguna, yang paling penting dalam sebuah *audio fingerprint* tentunya adalah kecepatan pencarian. Dan salah satu faktor yang menentukan dalam kecepatan pencarian adalah berapa besar ukuran *audio fingerprint* yang harus diakses selama pencarian. Oleh karena itulah ukuran dari sebuah berkas *audio fingerprint* bisa dibilang sebagai salah satu penentu utama dalam mengukur kualitas sebuah *audio fingerprint*.

Dalam pembentukan *audio fingerprint* itu sendiri, terdapat perbedaan dari berbagai metode yang ada. Meskipun langkah pembentukan *audio fingerprint*nya sama, namun ada fungsi-fungsi yang berbeda yang digunakan yang menyebabkan *audio fingerprint* yang dihasilkan pun berbeda.

IV. EKSPLORASI YANG DILAKUKAN

Dalam melakukan eksplorasi ini, ada beberapa tahapan yang dilakukan. Setiap tahapan yang dilakukan tentunya memiliki tujuan yang sudah terdefinisikan sebelumnya, yaitu untuk mengetahui mana fungsi *hash* pembentuk *audio fingerprint* yang lebih efektif jika dilihat dari ukuran berkas *audio fingerprint*.

1. Studi literatur
   Sebelum dilakukan eksplorasi, perlu dilakukan studi literatur terhadap bidang terkait. Langkah ini dilakukan untuk lebih bisa mengetahui lebih jauh mengenai bidang yang dimaksud. Selain itu perlu juga dilakukan agar eksplorasi yang dilakukan tidak salah langkah maupun salah sasaran.
   Hasil dari langkah ini telah dituliskan pada bagian II dan III dari makalah ini.
2. Eksplorasi fungsi *hash* masing-masing metode
   Untuk mengetahui dengan pasti bagaimana bentuk dari fungsi *hash* yang diterapkan oleh masing-masing metode, perlu juga dilakukan eksplorasi terhadap fungsi-fungsi tersebut. Eksplorasi akan dilakukan dengan cara mencari seperti apa bentuk dari fungsi *hash* dari masing-masing metode. Dari sini akan dilakukan analisis awal sebagai bekal analisis tahapan selanjutnya.
3. Analisis penerapan fungsi *hash* masing-masing metode
   Dari setiap metode yang diterapkan, akan dianalisis hasilnya berdasarkan ukuran berkas yang ada. Hasil-hasil ini kemudian akan dianalisis lebih jauh mengenai bagaimana kinerja fungsi *hash* terhadap hasil *audio fingerprint*. Untuk eksplorasi kali ini, analisis hasil hanya akan dilakukan pada ukuran berkas *audio fingerprint*.
4. Perbandingan hasil dari kedua metode
   Dari kedua metode yang dihasilkan, akan ditinjau seberapa efektif fungsi *hash* yang digunakan berdasarkan ukuran berkas *audio fingerprint* yang dihasilkan. Perbandingan metode ini dilakukan untuk menghasilkan sebuah kesimpulan dari eksplorasi yang dilakukan.

V. ANALISIS HASIL EKSPLORASI

Pada bagian ini akan dijelaskan hasil dari setiap tahapan ekplorasi yang ada serta analisis dari hasil tersebut. Namun untuk tahapan 1. tidak akan ditulis hasilnya di sini karena sudah dituliskan dengan lebih jelas pada bagian II dan III.

1. Eksplorasi fungsi hash masing-masing metode

Pada eksplorasi kali ini dilakukan pengamatan pada dua macam metode pembuatan *audio fingerprint*. Metode pertama adalah metode pembentukan *audio fingerprint* yang dilakukan oleh aplikasi Shazam, yang merupakan aplikasi pengenal musik yang cukup populer. Metode kedua adalah metode pembentukan *audio fingerprint* yang dilakukan oleh Echoprint, sebuah sistem yang bersifat *open source* yang bisa mengenali dan mengakses langsung data musik. Kedua metode ini dipilih sebab kedua metode ini yang paling mudah ditemukan sumbernya secara bebas di internet.

Sebelum mengetahui bagaimana bentuk fungsi *hash* untuk masing-masing metode, dilakukan eksplorasi untuk mengetahui apakah fungsi *hash* memiliki peran yang sama untuk kedua metode. Ditemukan bahwa kedua metode menggunakan langkah-langkah sebagai berikut dalam membuat *audio fingerprint* dari suatu data musik:

i. Menampilkan musik dalam bentuk spektogram untuk mengawali proses pembuatan *audio fingerprint*.
ii. Mencari posisi puncak-puncak spektogram dari data musik tersebut.
iii. Memetakan posisi puncak-puncak spektogram ke dalam sebuah peta gambar.
iv. Mengubah peta yang ada menjadi *audio fingerprint* dengan memanfaatkan fungsi *hash* yang ada.

Dari langkah-langkah di atas dapat dilihat bahwa fungsi *hash*, baik pada Shazam maupun pada Echoprint menempati posisi yang sama dalam proses pembuatan *audio fingerprint*. Hal ini berarti perbandingan atas keduanya bisa dilakukan.

a. Fungsi *Hash* pada Shazam[3]

Pada metode ini, *audio fingerprint* dibentuk dari sebuah peta konstelasi yang menunjukkan hubungan kombinatorial dari pasangan titik-titik poin frekuensi. Pada proses pembentukannya, setiap titik yang menjadi acuan dipilih, kemudian titik itu akan memiliki daerah target yang berisi poon-poin yang berhubungan dengan titik tersebut. Setiap titik acuan ini nantinya akan dihubungkan dengan setiap poin yang berada dalam area targetnya, setiap pasangan frekuensi serta perbedaan waktu dari setiap poin. Fungsi ini bisa digunakan lagi meskipun ada *noise* yang terlibat maupun kompresi codec dari suara musik. Lebih jauhnya lagi, setiap bagian hasil dari fungsi *hash* ini bisa dikelompokkan dalam *unsigned* integer berukuran 32 bit. Setiap hasil dari fungsi *hash*ini juga bisa dihubungkan dengan waktu *offset* ddari awal mulainya berkas tersebut ke titik acuannya, meskipun waktu absolut bukanlah bagian dari *hash* itu sendiri.

Ukuran dari hasil fungsi *hash* untuk setiap detik dari musik yang direkam yang diproses adalah mendekati jumlah kepadatan dari titik konstelasi per detik dikalikan dengan faktor kedekatan dari area target. Sebagai contoh, apabila setiap titik konstelasi dijadikan titik acuan, dan area target memiliki kedekatan sebesar F = 10, maka ukuran dari hasil fungsi *hash* yang ada adalah mendekati 10 kali jumlah titik konstelasi yang diekstrak dari berkas. Dengan membatasi jumlah titik yang dipilih untuk setiap area target, bisa dilakukan pembatasan dari kombinatorial pasangan. Faktor kedekatan bisa mengarah kepada faktor biaya untuk penyimpanan *audio fingerprint* nantinya.

b. Fungsi *Hash* pada Echoprint[4]

Untuk mendapatkan ketahanan yang tinggi terhadap pemrosesan modifikasi spektral dan penghindaran *noise* dalam perekaman di udara bebas, Echoprint mengutamakan hanya pada waktu relatif di antara onset pada audio yang terdeteksi secara sukses sebagai serupadengan *beat*. Deteksi onset dilakukan secara terpisah dan tidak saling berkaitan pada 8 frekuensi berbeda, berdasarkan pada 8 *band* terendah pada penyaring MPEG-Audio 32 *band* (yang secara nominal berada antara 0 sampai 5512.5 Hz). Puncak dari sinyal *band-pass complex* di setiap *band* kemudian dibandingkan dengan suatu *threshold* yang berubah secara eksponensial, pada titik tertentu *threshold* ditingkatkan menjadi 1.05 kali puncak sinyalyang baru. Sebuah algoritma yang adaptif digunakan untuk mengambil sebuah target IOI dan mengurangi *threshold* jika IOI lebih pendek, dan sebaliknya. Target tingkat *onset* untuk Echoprint adalah 1 onset per detik per *band*.

Pasangan IOI yang bagus di setiap *band* dikuantifikasi ke dalam unit yang berukuran 23.32 ms, kemudian dikombinasikan untuk membuat *hash value*. Untuk mempertahankan ketahanan, setiap onset dipertimbangkan bersama empat penerusnya. Enam *hash value* berbeda dibuat dengan memilih pasangan yang mungkin dari semua pasangan yang ada.

Dari sana dapat disimpulkan bahwa keseluruhan tingkat *hash value* yang ada adalah kira-kira 8 (*band*) x 1 (*onset*per detik) x 6 (*hash value* per detik) ≈ 48 hashes/detik.

Karena *onset* kira-kira terpisah sejauh 1 detik, IOI yang dikuantifikasikan berada pada 1/.0232 = 43 atau sekitar 5-6 bit, dan satu pasang *onset*

memiliki sekitar 12 bit informasi. Hasil ini kemudian dikombinasikan dengan 3 bit indeks *band* untuk menghasilkan *raw hash*, yang kemudian disimpan bersama dengan waktu kemunculan pada berkas.

2. *Analisis penerapan fungsi hash masing-masing metode*

Kedua metode yang ada masing-masing sudah memiliki penerapannya sendiri sehingga tidak perlu lagi dibuat sebuah metode penerapan untuk merealisasikan konsep fungsi *hash* yang dimiliki masing-masing metode. Berikut adalah analisis dari penerapan untuk setiap metode beserta contoh dari *audio fingerprint* yang dihasilkan.

a. Fungsi *Hash* pada Shazam

Berdasarkan informasi yang diperoleh dari situr resmi Shazam[1], tidak dapat ditemukan lebih jauh mengenai pengaplikasian fungsi *hash* yang digunakan. Penelusuran lebih jauh terhadap dokumen-dokumen pengembangan yang ada di internet pun tidak dapat memberikan hasil yang diinginkan. Selain itu, usaha untuk mencari bentuk dari *audio fingerprint* yang dimiliki oleh Shazam pun tidak memberikan hasil yang diinginkan. Oleh karena itu, tidak dapat dilakukan pengamatan lebih jauh dari fungsi *hash* yang dimiliki oleh Shazam.

b. Fungsi *Hash* pada Echoprint

Seperti bisa dilihat pada situs resmi Echoprint[2], fungsi *hash* ini diterapkan dengan menggunakan bahasa pemrograman Python. Didukung dengan sifatnya yang *open source,* situs ini juga menyediakan data contoh hasil pembuatan *audio fingerprint*.

Berikut adalah contoh *audio fingerprint* yang merupakan hasil penerapan fungsi *hash* berdasarkan metode Echoprint.

3. *Perbandingan hasil dari kedua metode*

Karena tidak bisa ditemukan data dari Shazam sebagai pembanding, maka perbandingan terhadap kedua metode tidak bisa dilakukan. Hanya diketahui bahwa ukuran *fingerprint* dari Echoprint berkisar antara belasan *kilobyte.*

VI. KESIMPULAN

1. Fungsi *hash* merupakan sebuah metode yang bisa digunakan untuk membuat *audio fingerprint*.
2. Perbandingan akhirnya tidak dapat dilakukan karena tidak terdapat data yang bisa diakses secara terbuka. Penerapan pun tidak dapat dilakukan karena ada bagian fungsi yang tidak dapat diterjemahkan.

REFERENSI

[1] Shazam Entertainment Ltd. (2013) Shazam. [Online]. http://www.shazam.com/

[2] Echoprint. (2012, February) Echoprint - Open source music identification. [Online]. http://echoprint.me

[3] Daniel P. W. Ellis, Brian Whitman, and Alastair Porter, "Echoprint - An Open Music Identification Service," 2011.

[4] Avery Li-Chun Wang, "An Industrial Strength Audio Search Algorithm," *Proc. 4th ISMIR*, pp. 7-13, 2003.

[5] Rowan Latuconsina, "Fungsi Hash: Tiger," Institut Teknologi Bandung, Bandung, Tugas Kuliah 2006.

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, 20 Mei 2013

Diani Pavitri R 13509021