Analisis Penggunaan *Message Digest* pada Media *Sharing File* dalam Jaringan

Ginanjar Fahrul Muttaqin - 13507103
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
If17103@students.if.itb.ac.id

***Abstract*—Tukar menukar file dalam jaringan memang menguntungkan. Untuk beberapa kasus kita tidak perlu susah-susah mencari dimana kita harus mencari berkas atau data yang suatu saat kita butuhkan. Adakalanya sebuah berkas atau file sudah tidak asli lagi. Banyak pihak yang tidak bertanggung jawab yang menyalahgunakan penyimpanan berkas atau data untuk kepentingan pribadi. Perubahan berkas atau data tersebut akan mengubah isi file dan berarti berkas atau data tersebut sudah tidak asli. Untuk itulah diperlukan sebuah mekanisme untuk memvalidasi keaslian sebuah berkas atau data yang dipertukarkan.**

Index Terms—media sharing file, hash, message digest, jaringan, peer to peer, hosting, intruder, victims.

I. PENDAHULUAN

Baru-baru ini teknologi *sharing file* sangat banyak dimanfaatkan untuk mendapatkan berkas atau data yang dibutuhkan. Pengguna media *sharing file* ini bisa dengan mudah mencari berkas atau data yang dia butuhkan dari jaringan yang dia masuki, dengan tidak terlalu mengindahkan siapa orang yang mengunggah berkas atau data tersebut. Padahal pada kenyataannya banyak berkas atau data yang disisipi sesuatu yang berbahaya, seperti trojan, virus, dll. Namun apakah benar sepenting itu? Apakah benar perlu autentikasi yang jelas mengenai keaslian berkas atau data yang diunggah? Oleh karena itulah penulis hendak membahas secara dalam bagaimana sistem ini bekerja.

Pada kenyataannya penambahan sesuatu ke dalam berkas atau data yang di-*sharing* akan mengubah *binnary file* berkas tersebut. Fungsi hash dapat dimanfaatkan untuk mengetahui keaslian berkas. Pada implementasinya setiap file akan memiliki nilai *message digest* yang didapatkan dari fungsi hash. *Message digest* inilah yang dimanfaatkan untuk mengidentifikasi keaslian berkas atau data tersebut. Makalah ini juga akan membahas beberapa media *sharing file* yang memanfaatkan *checksum hash* ini, beserta sedikit eksplorasi dari penulis.

II. PRINSIP DASAR

A. File Sharing

File Sharing adalah sebuah konsep praktis dalam menyediakan atau mendistribusikan informasi digital yang tersimpan. Informasi digital yang dimaksud banyak, bisa aplikasi komputer, multimedia (audio maupun visual), dokumen, ataupun buku elektronik.

Sejak tahun 1990, *file sharing* mulai terkenal dan sering digunakan oleh pengguna komputer pada saat itu. Sebenarnya konsep *file sharing* yang diterapkan tidaklah memiliki legalitas, bahkan akan merugikan beberapa pihak, seperti peredaran MP3. Penyedia layanan *file sharing* pada saat itu, seperti Napster, Gnutella, eDonkey2000, dan BitTorrent banyak bertindak sebagai pengalih perhatian pengguna internet pada masanya.

Sejak tahun 2000, beberapa penyedia layanan *file sharing* dibekukan karena memang merugikan pihak yang memiliki hak cipta atas berkas yang dipertukarkan secara bebas. Namun pada kenyataannya media *file sharing* yang bersifat *peer to peer* masih merajalela sampai sekarang. Hingga perkembangannya sekarang bukan hanya MP3, tetapi juga hampir semua jenis berkas digital sudah beredar bebas di jaringan *file sharing.*

Tipe File Sharing

Menurut mekanisme teknis di jaringan, *file sharing* dikategorikan menjadi dua jenis:

- ***Peer to peer network***

  Pada jenis ini, berkas yang dipertukarkan akan dilakukan secara simultan antar dua *client* dalam jaringan. Kedua pihak tersebut membuat koneksi sendiri untuk mempertukarkan berkas dengan seizin masing-masing pihak. Layanan yang menyediakan *tools* seperti ini banyak yang sudah populer sejak dulu, seperti Gnutella, iMesh, dan eDonkey.

- ***File Hosting service***

  Jenis yang satu ini menggunakan layanan penyimpanan berkas secara terpusat di sebuah *host. Host* inilah yang akan diakses oleh setiap pengguna yang membutuhkan berkas yang diinginkan, melakukan pencarian dan langsung mengunduh berkas yang mereka butuhkan. Selain itu pengguna juga dapat mengunggah berkas dari komputer mereka ke *host* tersebut.

Pada perkembangannya konsep *file sharing* mengalami pertentangan dari berbagai kalangan karena mengganggu hak cipta dari berkas yang dipertukarkan.

Perbandingan aplikasi *file sharing*

Berikut beberapa contoh aplikasi yang melayani *file sharing:*

Tabel 1 *File Sharing Sample*

Name	Programming language	Spyware/Adware
µTorrent	C++	No
Wuala	Java	No
Winny	C++	No
WinMX	C++	No
Warez P2P	?	No
Vuze (formerly Azureus)	Java	No
TrustyFiles	?	Yes
Transmission	C, ObjC	No
Thaw	Java	No
SymTorrent	C++	No
Stealthnet	C#	No
Soulseek	?	No
Shareaza	C++	No
Share	Delphi	No
RShare	C#	No
Qbittorrent	C++	No
Piolet	?	Yes
Opera	C++	No
Nodezilla	Java	No
MUTE	C++	No
Morpheus	?	No
MonoTorrent(client library)	C#	No
MLDonkey	OCaml	No
Manolito	?	Yes
LimeWire	Java	Yes
KTorrent	C++	No
KCeasy	C++, Delphi	No
Kazaa Lite	-	No
Kazaa	C++	Yes
iMesh (pre v6.0)	C++	No
gtk-gnutella	C	No
GNUnet	C	No
Gnucleus	C++	No
giFT	C	No
FrostWire	Java	No
Frost	Java	No
Freenet's FProxy	Java	No
Free Download Manager	C++	No
ExoSee	?	No
eMule	C++	No
eDonkey2000	C++	No
DC++	C++	No
Cabos	Java,REALbasic	No
Blubster	?	Yes
BitTorrent 6	C++	No
BitTorrent 5	Python	No
BitTornado	Python	No
BitComet	C++	No
BearShare (pre v6.0)	C++	Yes
Ares Galaxy	Delphi	No
ANts P2P	Java	No
aMule	C++	No
Acquisition	ObjC, Java	No

**sumber: forbes.com*

B. Fungsi Hash

Sebelum berkembangnya hash, enkripsi sebelumnya akan mengubah plaintext menjadi ciphertext dengan sebuah fungsi enkripsi, dan mengubah ciphertext menjadi plaintext kembali dengan sebuah fungsi dekripsi.

Lain dengan metode enkripsi biasanya, fungsi hash adalah sebuah fungsi yang menerima masukan plaintext, berapapun panjangnya. Fungsi ini lalu mentransformasikan plaintext tersebut menjadi sebuah string keluaran yang ukurannya *fixed.* String keluaran ini dinamakan *message digest.*

$$h = H(M) \qquad (1)$$

size(h) : constant

Dengan demikian dapat disimpulkan bahwa fungsi hash bersifat satu arah. *Message digest* tidak dapat dikembalikan mejadi plaintext semula (*irreversible*).

Fungsi hash memiliki karakteristik tersendiri yang berbeda dengan fungsi enkripsi pada umumnya:

a. Fungsi H dapat diterapkan pada blok data berukuran berapa saja
b. H menghasilkan nilai (h) dengan panjang yang tetap
c. H(x) mudah dihitung untuk setiap nilai x yang diberikan
d. Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga H(x) = h
e. Untuk setiap x yang diberikan, tidak mungkin mencari y ≠ x sedemikian sehingga H(y) = H(x)
f. Tidak mungkin mencari pasangan x dan y sedemikian sehingga H(x) = H(y)

Jenis-jenis fungsi hash

Sesuai dengan mekanisme kerja setiap bloknya, dan cara memperoleh *message digest*, ada beberapa jenis fungsi hash yang berkembang dan sering digunakan:

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

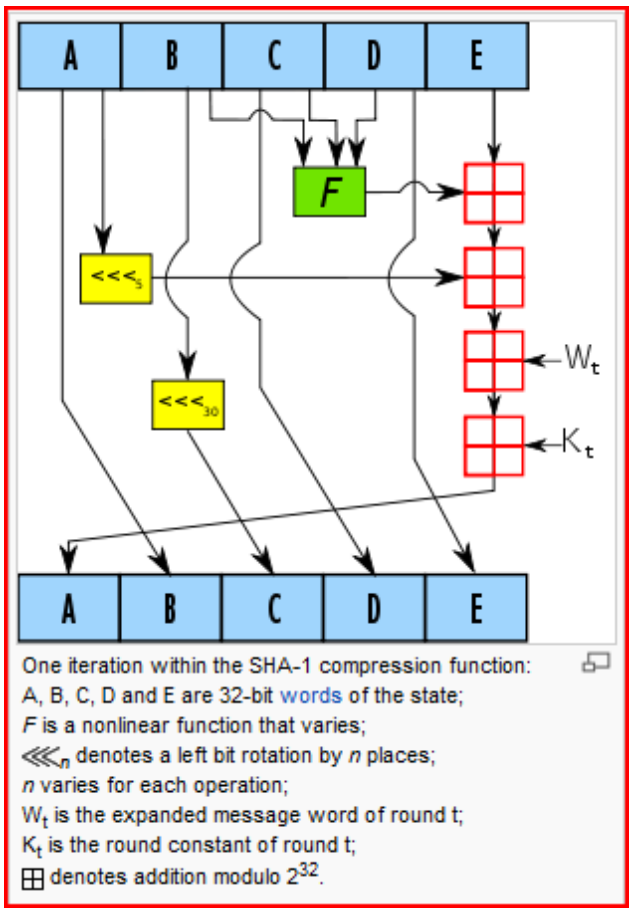
untuk melihat perbedaan secara detail ada pada tabel di bawah ini:

Tabel 2 Jenis-jenis Fungsi Hash

Algoritma	Ukuran *message digest* (bit)	Ukuran blok pesan	Kolisi
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
RIPEMD	128	512	Ya
RIPEMD-128/256	128/256	512	Tidak
RIPEMD-160/320	160/320	512	Tidak
SHA-0	160	512	Ya
SHA-1	160	512	Ada cacat
SHA-256/224	256/224	512	Tidak
SHA-512/384	512/384	1024	Tidak
WHIRLPOOL	512	512	Tidak

SHA1

SHA1 adalah salah satu jenis fungsi hash yang memiliki karakteristik khusus dan belum terpecahkan. SHA1 didesain oleh *National Security Agency* pada tahun 1995. Panjang *message digest* yang dihasilkan adalah 160 bits dengan jumlah pengulangan 80 kali untuk menghasilkan *message digest* tersebut.



Gambar 1 Skema SHA1

Message Digest

Output dari sebuah fungsi hash tentunya sebuah *message digest*. Pada umumnya *message digest* disajikan dalam bentuk string heksadesimal. Setiap *message digest* dari sebuah fungsi hash yang sama memiliki panjang yang konstan, contoh:

```
de9f2c7f d25e1b3a fad3e85a 0bd17d9b 100db4b3
```

III. ANALISIS

A. Keuntungan dan Kerugian penggunaan media sharing file

Membahas keuntungan penggunaan *file sharing* memang sudah tidak perlu dibahas kembali. Setiap pengguna yang memanfaatkan fitur ini akan dengan mudah mencari, dan mendapatkan berkas atau data yang dia butuhkan di dalam sebuah jaringan. Baik itu metode *peer to peer* maupun melalui *hosting file.* Penyebaran berkas tersebut akan saling menguntungkan ketika seseorang memiliki kebutuhan yang beririsan.

Meskipun *file sharing* masih berstatus tidak legal, sampai saat ini kepopuleran *file sharing* tetap memiliki peranan penting dalam penyebaran arus informasi.

Selain keuntungan yang didapatkan ada banyak kerugian yang mungkin bisa didapatkan oleh pengguna:

a. Penggunaannya ilegal
b. Ada kemungkinan penyusupan lewat file (virus, trojan, adware, malware)
c. Transaksi file adalah penyebab utama kemacetan *bandwidth* dalam jaringan

Kerugian yang ingin penulis soroti adalah kemungkinan penyusupan lewat berkas oleh pihak yang tidak bertanggung jawab.

Virus

Virus adalah aplikasi komputer yang dapat menggandakan diri otomatis dan menginfeksi komputer. Selain karakter utama yang seperti itu, virus juga dapat menjadi malware, adware, dan spyware dalam komputer victim.

Trojan Horse

Trojan lebih berbahaya lagi dari sekedar virus biasa, trojan akan mengaktifkan jaringan koneksi ke pihak pembuat program sehingga dia bisa mengendalikan komputer victim tanpa disadari secara langsung.

Malware

Secara umum malware tidak berbahaya, akan tetapi sifatnya yang menyebalkan dan mengganggu pengguna. Kadang-kadang menggandakan file, atau membuka aplikasi secara tidak terkendali.

Adware

Adware akan terus memenuhi misinya, yaitu iklan dalam komputer victim. Ketika komputer terinfeksi adware, iklan-iklan untuk suatu produk kadang-kadang mengganggu aktifitas pengguna dan cenderung *abusif.*

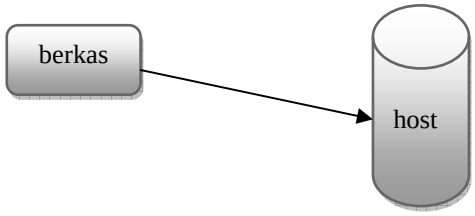
Sebetulnya masih banyak aplikasi lain yang bisa mengintervensi komputer pengguna, namun intinya satu yang berkitan dengan *file sharing.*

Dari semua sifat *intruder* di atas ada kesamaan, yaitu: ***intruder* menggunakan file *executable* sebagai media penyebarannya.** Inilah yang sering dimanfaatkan oleh *intruder* untuk dapat masuk ke komputer calon *victims.* Beberapa *file sharing* memang efektif menjadi inang para *intruder* yang siap mencari mangsa.

B. Pemanfaatan fungsi hash dalam mekanisme autentikasi berkas atau data

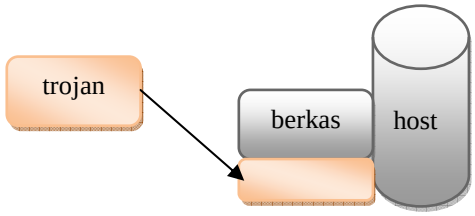
Untuk memanfaatkan fungsi hash dalam menanggulangi *intruder* lewat file sharing, terlebih dahulu kita pelajari bagaimana berkas atau data yang terdapat di jaringan bisa terinfeksi.

Uploader* mengunggah berkas asli ke *host



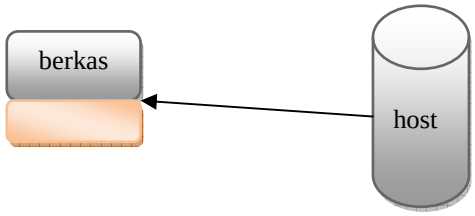
Gambar 2 Transaksi Unggah Berkas

Intruder* menginfeksi berkas di *host



Gambar 3 Transaksi Infeksi Berkas

***User* mengunduh berkas yang terinfeksi**



Gambar 3 Transaksi Unduh Berkas

Pada kenyataannya berkas asli dan berkas yang telah disisipi *intruder* akan berbeda struktur *binnary file*-nya. Hal inilah yang akah menjadi kunci utama pengecekan keaslian sebuah berkas yang tersebar di jaringan.

C. Analisis Implementasi message digest pada media sharing file

Sesuai dengan pembahasan tentang *intruder*, pada bagian ini akan dibahas bagaimana fungsi hash dimanfaatkan untuk mengidentifikasi keaslian sebuah berkas atau data.

Konsep yang dimanfaatkan adalah perubahan struktur *binnary file* berkas asli dan berkas yang sudah terinfeksi oleh *intruder*.

Sesuai karakteristiknya, fungsi hash akan menghasilkan nilai h yang berbeda untuk setiap M yang berbeda. Hal inilah yang dimanfaatkan untuk mengidentifikas keaslian sebuah berkas.

Berkas asli

Berkas terinfeksi

Pada saat dilakukan pengecekan kedua *message digest* akan dibandingkan. Berdasarkan ketentuan fungsi hash,

Dengan inilah, keaslian sebuah berkas dapat diidentifikasi dengan mudah.

Ide implementasi penulis

Pada implementasinya konsep ini tidak mudah untuk dijalankan, harus ada kesepakatan yang harus selalu diikuti oleh para pengguna *file sharing*.

a. Pada saat mengunggah berkas, setiap *uploader* diharuskan mencantumkan *message digest* berkas tersebut
b. Setiap *message digest* berkas asli harus dapat diakses oleh setiap pengguna *file sharing*
c. *Message digest* generator tidak disimpan di halaman server, melainkan aplikasi yang digunakan di desktop, karena proses pembangkitan *message digest* membutuhkan pembacaan keseluruhan isi berkas
d. Setiap pengguna dapat dengan mudah mengakses *message digest generator*, dan bisa dengan langsung melakukan *checksum* terhadap keaslian berkas.
e. Dengan ini kesempatan dari seorang *intruder* akan menjadi sempit. Pada saat melakukan intervensi dan mengubah *binnary file* berkas yang tersimpan di *host*, *intruder* akan tetap mengubah struktur berkas yang mengakibatkan perubahan *message digest*. Hal ini dengan mudah diidentifikasi oleh pengguna.
f. Pada kenyataannya ada kesempatan dimana *intruder* tetap bisa menyisipkan berkas terinfeksi, yaitu dengan berpura-pura menjadi *uploader*. Bagaimanapun usahanya, setidaknya dengan cara ini kesempatan dan peluang penyusupan lewat berkas atau data masih bisa dikurangi

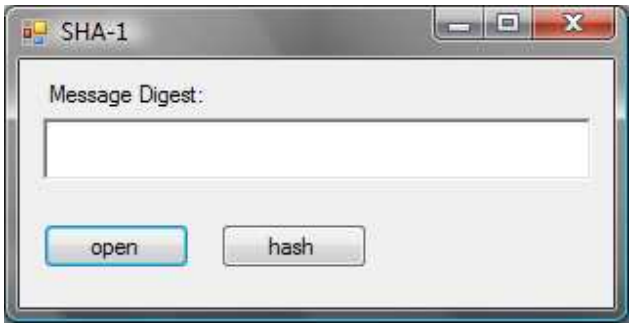
Message Digest Generator

Berikut penulis membuat program kecil sederhana yang bersifat sebagai pembangkit *message digest* dari sebuah berkas atau data dalam bahasa C#. Pembangkitan *message digest* menggunakah fungsi hash SHA1.

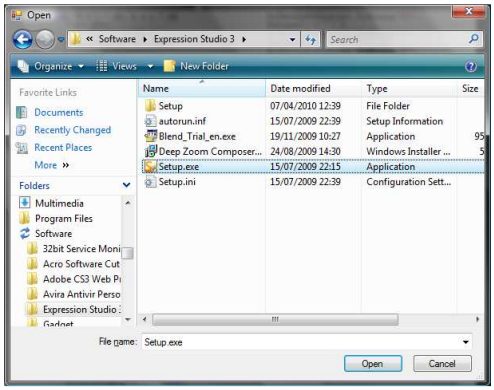
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace SHA1
{
    public class SHA1
    {
        BitArray bitContent, h0, h1, h2, h3, h4;
        string MessageDigest;
        int lenContent;
        public SHA1(){}

        public BitArray getBitContent();

        public string getMessageDigest();

        public BitArray strToBitArray(string);

        public BitArray intToBitArray(uint);

        public uint bitArrayToUint(BitArray);

        public BitArray reverseBitArray(BitArray);

        public byte[] reverseByte(byte[]);

        public uint leftrotate(uint, int);

        public BitArray plus(BitArray,BitArray);

        public void initHashConst()
        {
            h0 = intToBitArray(0x67452301);
            h1 = intToBitArray(0xEFCDAB89);
            h2 = intToBitArray(0x98BADCFE);
            h3 = intToBitArray(0x10325476);
            h4 = intToBitArray(0xC3D2E1F0);
        }
        public BitArray paddingBitArray(BitArray);

        public BitArray addLengthBit(BitArray,
        long);

        public void initBitArray(string text)
        {
            bitContent = strToBitArray(text);
            int len = bitContent.Length;
            bitContent =
```

```
            paddingBitArray(bitContent);
        bitContent =
            addLengthBit(bitContent, len);
        lenContent = bitContent.Length;
    }

    public void extractMD()
    {
        int nBlock = lenContent / 512;
        int i, j, k;
        BitArray a, b, c, d, e, f, kt, temp;
        f = new BitArray(32);
        kt = new BitArray(32);
        initHashConst();

        for (i = 0; i < nBlock; i++)
        {
            BitArray[] w = new BitArray[80];
            for (j = 0; j < 80; j++)
                w[j] = new BitArray(32);

            for (j = 0; j < 16; j++)
                for (k = j * 32; k < j * 32
                  + 32; k++)
                    w[j][k - j * 32] =
                  bitContent[k + i * 512];

            for (j = 16; j < 80; j++)
                w[j] =
                  intToBitArray(leftrotate(b
                  itArrayToUint(new
                  BitArray(new BitArray(w[j
                  - 3]).Xor(w[j -
                  8]).Xor(w[j - 14]).Xor(w[j
                  - 16]))), 1));

            a = new BitArray(h0);
            b = new BitArray(h1);
            c = new BitArray(h2);
            d = new BitArray(h3);
            e = new BitArray(h4);

            for (j = 0; j < 80; j++)
            {
                if (j >= 0 && j < 20)
                {
                    f = new BitArray(new
                  BitArray(b).And(c).Or(new
                  BitArray(b).Not().And(d)))
                  ;
                    kt =
                  intToBitArray(0x5A827999);
                }
                else if (j >= 20 && j < 40)
                {
                    f = new BitArray(new
                  BitArray(b).Xor(c).Xor(d))
                  ;
                    kt =
                  intToBitArray(0x6ED9EBA1);
                }
                else if (j >= 40 && j < 60)
                {
                    f = new BitArray(new
                  BitArray(b).And(c).Or(new
                  BitArray(b).And(d)).Or(new
                  BitArray(c).And(d)));
                    kt =
                  intToBitArray(0x8F1BBCDC);
                }
                else if (j >= 60 && j < 80)
                {
                    f = new BitArray(new
                  BitArray(b).Xor(c).Xor(d))
                  ;
                    kt =
                  intToBitArray(0xCA62C1D6);
                }
                temp =
                    plus(intToBitArray(leftrot
                ate(bitArrayToUint(a), 5)),new
                BitArray(plus(plus(f, e),
                plus(kt, w[j]))));

                e = new BitArray(d);
                d = new BitArray(c);
                c =
                  intToBitArray(leftrotate(b
                  itArrayToUint(b), 30));
                b = new BitArray(a);
                a = new BitArray(temp);
             }
             h0 = plus(h0, a);
             h1 = plus(h1, b);
             h2 = plus(h2, c);
             h3 = plus(h3, d);
             h4 = plus(h4, e);
        }
        MessageDigest =
            bitArrayToUint(h0).ToString("X").
            ToLower() + " " +
            bitArrayToUint(h1).ToString("X").
            ToLower() + " " +
            bitArrayToUint(h2).ToString("X").
            ToLower() + " " +
            bitArrayToUint(h3).ToString("X").
            ToLower() + " " +
            bitArrayToUint(h4).ToString("X").
            ToLower() + " ";
    }

    public string Hash(string inputText)
    {
        initBitArray(inputText);
        extractMD();
        return getMessageDigest();
    }
  }
}
```

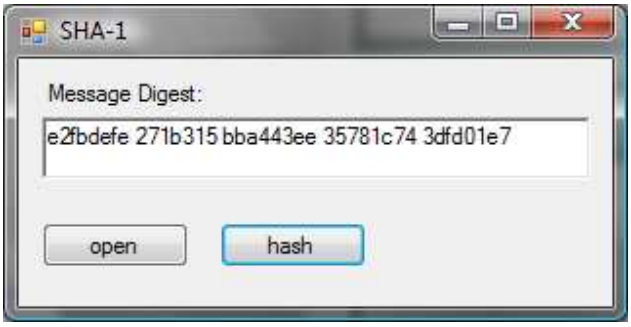
Antarmuka



Gambar 4 Halaman Utama



Gambar 5 Halaman Buka Berkas



Gambar 6 Halaman Keluaran *Message Digest*

IV. KESIMPULAN

Sharing file merupakan perkembangan dunia IT yang memiliki kelebihan dan kekurangan. Meskipun banyak ditentang beberapa kalangan, *sharing file* tetap *eksis* di dunia maya. Beberapa *intruder* memanfaatkan *media sharing file* untuk tindakan yang tidak bertanggung jawab dengan menyisipkan program *intruder.*

Dengan menggunakan *checksum file* fungsi hash, keaslian berkas atau data yang tersimpan di *host* jaringan dapat diidentifikasi dengan mudah. Meskipun pada kenyataannya, praktek untuk hal ini hanya menyebabkan sedikit perubahan.

VII. *ACKNOWLEDGMENT*

Terimakasih kepada semua pihak yang telah terlibat dalam pembuatan makalah ini. Kepada Pak Rinaldi sebagai dosen yang telah memberi kesempatan saya mempelajari fungsi hash. Terimakasih pula kepada pembaca yang telah menyempatkan diri membaca makalah ini, semoga ada manfaatnya. Saran dan kritik silakan disampaikan melalui *email.*

DAFTAR PUSTAKA

[1] Munir, Rinaldi, *Kriptografi*, Institut Teknologi Bandung, 2006.
[2] http://www.itl.nist.gov/fipspubs/fip180-1.htm
[3]http://delivery.acm.org/10.1145/rfc_fulltext/RFC1321/rfc1321.txt?key1=RFC1321&key2=4748052721&coll=GUIDE&dl=GUIDE&CFID=88339381&CFTOKEN=30390422
[4] Siu Man Lui and Sai Ho Kwok, *Interoperability of Peer-To-Peer File Sharing Protocols*, ACM, 2002
[5] Karen Kent and Murugiah Souppaya, *Guide to Computer Security Log Management*, U.S, 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, 16 Mei 2010

Ginanjar Fahrul Muttaqin - 13507103