BAB II

TINJAUAN PUSTAKA

2.1 Disk Encryption

Disk Encryption adalah salah satu teknik enkripsi yang dilakukan pada data at rest. Istilah data at rest ditujukan pada data yang tersimpan pada media penyimpanan komputer. Dengan kata lain, enkripsi tidak dilakukan pada data yang sedang ditransmisi melalui jaringan. Media penyimpanan yang dimaksud dalam disk encryption merupakan peralatan/device yang memiliki sektor-sektor dan alamat untuk tiap sektor tersebut. Contohnya adalah hard disk atau flash card.

2.2 Hard Disk Encryption

Penerapan disk encryption pada hard disk dinamakan hard disk encryption. Satu sektor pada hard disk biasanya memiliki ukuran 512 byte. Istilah lain yang paling sering digunakan untuk menyebut hard disk encryption adalah volume encryption. Untuk penulisan dokumen Tugas Akhir ini, istilah disk encryption mengacu pada hard disk encryption.

2.2.1 Volume Encryption

Berkaitan dengan teknik enkripsi pada data at rest, teknik yang umum digunakan adalah enkripsi yang dilakukan pada file atau file encryption. Satu file yang diinginkan dienkripsi pada satu waktu. File encryption sudah dilakukan

bertahun-tahun dan masih memiliki manfaat dari segi keamanan. Berbeda dengan file encryption, disk encryption dalam implementasinya menggunakan suatu wadah/kontainer/volume. Wadah ini digunakan sedemikian rupa sehingga tiap file yang dimasukkan ke dalam wadah ini akan otomatis terenkripsi. Dengan alasan itu, salah satu istilah lain untuk disk encryption adalah volume encryption. Wadah/kontainer/volume ini bila dibuka dengan password yang tepat akan diperlakukan sebagai disk tersendiri (contohnya H:\ drive). Dengan kata lain, dengan password yang tepat, sistem enkripsi ini akan memunculkan sebuah virtual disk.

Dibandingkan dengan teknik file encryption, disk encryption memiliki beberapa keuntungan terutama bila jumlah file penting yang hendak dienkripsi amat banyak. Wadah yang disediakan oleh disk encryption memungkinkan file-file penting tersebut terkumpul dengan rapi dalam satu tempat (tidak tersebar). Penanganan kunci pun lebih mudah karena hanya diperlukan satu kunci untuk satu wadah. Dengan demikian, pengguna tidak perlu mengingat kunci mana yang digunakan untuk file apa. Selain itu, pengguna tidak perlu turun tangan untuk memicu proses enkripsi/dekripsi berulang kali karena sifatnya yang otomatis melakukan proses tersebut terhadap isi virtual disk.

Berdasarkan besarnya kontainer/volume tersebut, disk encryption dapat digolongkan menjadi dua yakni entire hard disk encryption (EHD encryption) dan virtual hard disk encryption (VHD encryption).

2.2.2 EHD Encryption dan VHD Encryption

Sesuai namanya, EHD encryption berarti enkripsi dilakukan pada seluruh area hard disk, termasuk faktor yang menyangkut perangkat keras hard disk itu sendiri. Dengan demikian, EHD encryption ini hanya dapat diimplementasikan secara khusus terhadap salah satu jenis hard disk tertentu. Jenis enkripsi ini tentu memiliki kelebihan karena seluruh area hard disk aman. Namun, jika dikaitkan dengan jaringan dan kegiatan sharing yang biasa dilakukan di dalamnya, EHD encryption menuntut penyesuaian yang tidak mudah.

VHD encryption, di sisi lain, tidak melakukan pengamanan terhadap seluruh area hard disk, tetapi hanya sebagian, yaitu sebuah virtual disk. Dengan demikian, sepanjang ada kesesuaian sistem enkripsi dengan sistem operasi tempat VHD encryption terinstal, semua akan berjalan lancar, tidak perlu memikirkan faktor yang berkaitan dengan perangkat keras hard disk.

2.2.3 Virtual Disk

Virtual disk berarti tidak ada disk yang nyata dalam bentuk fisik. Walaupun demikian, sebuah virtual disk dapat digunakan seperti physical disk lainnya. Virtual

disk dapat digunakan untuk menyimpan file dan file yang terdapat di dalamnya dapat dimodifikasi atau dihapus. Virtual disk merupakan disk yang muncul sementara, hanya selama dipanggil saja (istilahnya di-mount). Bentuk asli dari virtual disk dapat berupa memory. Hal tersebut berarti pada saat virtual disk di-mount, sebagian dari memory diperlakukan seperti layaknya physical disk. Selain untuk mengeksekusi program, data dapat disimpan pada bagian memory tersebut. Kondisi ini memungkinkan seluruh proses yang berkaitan dengan virtual disk yang tercipta berlangsung lebih cepat. Kelemahan cara ini adalah bila komputer dimatikan, seluruh data juga hilang (karena tersimpan di memory).

2.2.4 Cipher Blok

Disk encryption bekerja dengan cipher blok (block cipher). Rangkaian bit plainteks/cipherteks yang akan dienkripsi atau didekripsi dibagi menjadi blok-blok bit yang panjangnya sama dan sudah ditentukan sebelumnya. Cipher blok termasuk dalam tipe algoritma simetri, tipe algoritma kriptografi modern yang berarti proses enkripsi dan dekripsi memiliki struktur yang serupa. Salah satu jenis cipher blok yang dapat digunakan untuk disk encryption adalah algoritma SHA.

2.2.5 Menciptakan Virtual Disk

Untuk memunculkan sebuah virtual disk, diperlukan peran dari komponen user mode dan kernel mode. Aplikasi Pengguna yang bergerak dalam user mode

dapat menerima informasi seperti nama file volume, ukuran virtual disk, juga nama drive yang diinginkan. Informasi itu diberikan oleh pengguna Kemudian, aplikasi pengguna bertugas memberi tahu virtual disk driver untuk membuat virtual disk dengan spesifikasi yang telah diberikan dengan memanggil fungsi Windows API yang sesuai.

Windows API (Application Programming Interface) merupakan sekumpulan fungsi-fungsi eksternal yang terdapat dalam file-file perpustakaan Windows (library Windows) atau file library lainnya yang dapat digunakan oleh program. Fungsi ini dapat menangani semua yang berhubungan dengan Windows, seperti pengaksesan disk, interface printer, grafik Windows, kotak dialog (buka file, simpan file, memilih font, memilih warna, dll) Windows shell, setting sistem operasi, penanganan file, mengakses sitem registry, memainkan musik, dan sebagainya. Fungsi inni menyediakan banyak fitur-fitur standar untuk semua program yang berbasis Windows.

Hampir semua fungsi Windows API terdapat dalam direktori sistem milik Windows dan paling banyak berekstensi .DLL yang digunakan oleh sistem operasi Windows. Selain itu fungsi ini juga memastikan secara konsisten penggunaan semua sumber yang terdapat dalam Windows. File-file itulah yang disebut dengan Windows API.

Karena fungsi Windows API merupakan fungsi eksternal, maka untuk menggunakan fungsi tersebut terlebih dahulu dideklarasikan dengan perintah Declare didalam kode program. Setelah pendeklarasian fungsi selesai dilakukan, selanjutnya anda bebas untuk menggunakan fungsi tersebut layaknya bahasa Visual Basic didalam program.

Fungsi Windows API yang digunakan adalah DeviceIoControl. Parameter yang digunakan fungsi ini adalah sebagai berikut:

```
BOOL DeviceIOControl(

    HANDLE hDevice,              // handle to device of interest
    DWORD dwIoControlCode,       // control code of operation to
                                 perform
    LPVOID lpInBuffer,           // pointer to buffer to supply input
                                 data
    DWORD nInBufferSize,         // size of input buffer
    LPVOID lpOutBuffer,          // pointer to buffer to receive
                                 output data
    DWORD nOutBufferSize,        // size of output buffer
    LPDWORD lpBytesReturned,     // pointer to variable to receive
                                 output byte count
                                 // pointer to overlapped structure
    LPOVERLAPPED lpOverlapped    for asynchronous operation
);
```

Gambar 2.1 Device IO Control

Penanganan request yang berasal dari komponen user mode melalui Windows API merupakan salah satu fungsi utama driver. Untuk kepentingan request tersebut, driver memiliki control code yang spesifik untuk melakukan operasi-operasi yang menjadi fungsionalitasnya. Control code tersebut akan menjadi masukan

DeviceIoControl pada parameter dwIoControlCode. Pada kasus virtual disk, Aplikasi Pengguna akan mengirimkan kode untuk menciptakan virtual disk kepada driver pada saat menu mount dipilih.

Selain untuk memicu fungsi driver tertentu, fungsi DeviceIoControl juga dapat dimanfaatkan untuk mengirimkan informasi yang diperlukan untuk melakukan fungsi tersebut maupun menerima hasil fungsi driver. Parameter yang berkaitan adalah lpInBuffer dan nInBufferSize untuk masukan pada driver serta lpOutBuffer dan nOutBufferSize untuk menampung hasil dari driver.

Setelah mendapatkan kode untuk menciptakan virtual disk dan informasi yang dibutuhkan, driver melakukan pemeriksaan berkaitan dengan virtual disk yang akan dibuat, seperti kapasitas hard disk yang masih mencukupi atau tidak. Apabila file volume sukses dibuat, driver akan menciptakan sebuah device yang dihubungkandengan file volume tersebut. Untuk memunculkan virtual disk, device akandiemulasikan sebagai hard disk dengan memasukkan parameter tertentu pada fungspenciptaan device. Bersamaan dengan penciptaan device tersebut, sebuah thread akan diikutsertakan selama device tersebut ada.

2.3 Algoritma SHA

Dalam dunia kriptografi, hash function bukan merupakan suatu barang yang baru. Merupakan salah satu cabang dalam kriptografi, hash function memiliki daya

tarik tersendiri dikarenakan cukup banyak aplikasi yang menggunakan hash function dalam penerapannya. Hash function digunakan sebagai autentikasi, integritas dan digital signature, salah satu aplikasinya yaitu penggunaan password dalam aplikasi digital atau internet.

Cryptographic Hash Function adalah suatu fungsi dengan inputan yang berubah-ubah panjangnya (atau sangat panjang) dan memetakannya sehingga menghasilkan output yang pendek dan panjang nya tetap. Hash functions berawal dari ilmu komputer, dimana dibutuhkan sebuah fungsi yang berguna untuk mengkompresi sebuah string dengan panjang yang berubah-ubah menjadi sebuah string tetap yang lebih pendek. Hash functions digunakan untuk menentukan secara keseluruhan tempat penyimpanan yang mungkin dari sebuah file. Pada aplikasi kriptografi, hash function dibedakan menjadi unkeyed dan keyed hash functions.

1. Unkeyed Hash Function (Manipulation Detection Codes = MDCs)

   Hanya memerlukan satu parameter input, yaitu berita.

2. Keyed Hash Function (Message Authentication Codes = MACs)

Menggunakan dua parameter input, yaitu berita dan kunci.Selanjutnya, Unkeyed hash functions atau MDCs yang akan dikenal sebagai Hash Functions.

Hash functions juga dapat digunakan untuk keamanan pada autentikasi berita yaitu dengan melakukan autentikasi dari hasil hash berita tersebut. Contoh sederhana

misalnya proses komunikasi pengiriman file ukuran besar yang melalui jalur insecure, autentikasi yang dilakukan yaitu dengan mengirimkan hasil hash dari berita melalui jalur komunikasi biasa misalnya mail biasa atau melalui telefax.

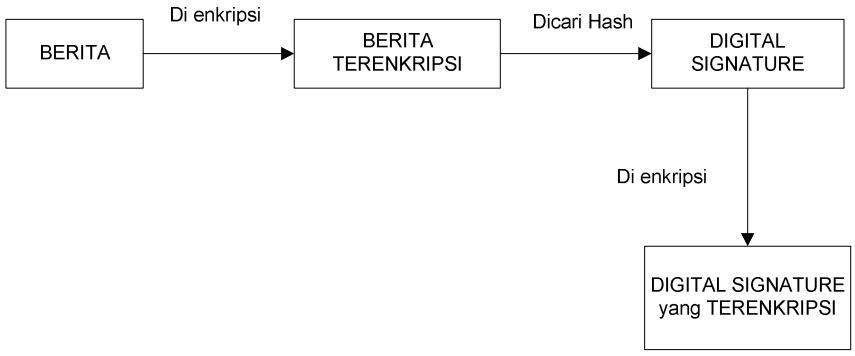
Aplikasi hash functions yang umum adalah digital signatures yaitu aplikasi untuk menandatangani hasil hash dan hal ini jauh lebih baik daripada menandatangani berita aslinya, dan akan mendapatkan keuntungan keamanan sekaligus performance. Dengan hash functions kita dapat membandingkan dua buah nilai tanpa harus membuka berita. Misalnya password dan passphrase.

A adalah seorang user pada suatu aplikasi yang memerlukan proses autentikasi dalam hal ini password. Aplikasi tersebut akan meminta input dari A kemudian dengan algoritma hash functions yang digunakan oleh aplikasi tersebut maka akan diubah inputan tersebut menjadi suatu output yang unik dan dengan suatu panjang tertentu. Tidak ada satupun output atau hasil hash functions dalam aplikasi tersebut yang nilainya sama. Selanjutnya, aplikasi tersebut hanya akan mencocokkan hasil output setiap kali user A login dengan database yang ia punya.

Sebuah digital signatures digunakan sebagai fungsi integritas suatu berita atau data yang dikirim. Apakah berita itu asli? Adakah kekurangan dalam berita tersebut? Inputan dalam algoritma Hash Function dapat berubah-ubah panjangnya (atau sangat

panjang) untuk kemudian menghasilkan output yang panjang nya tetap (misalnya 128 bit, 256 bit).

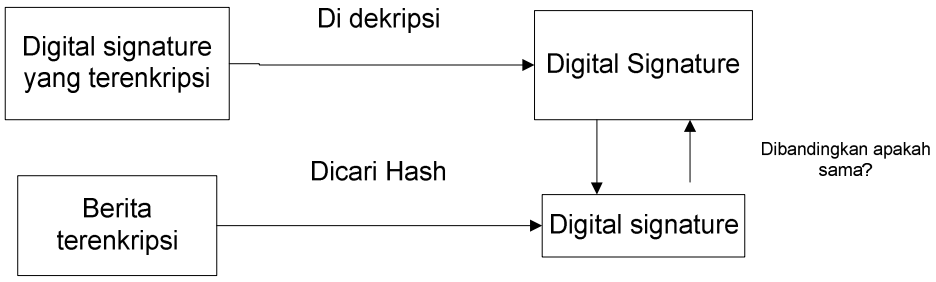
User A ingin mengirimkan sebuah berita kepada user B, sebelumnya user A mencari digital signatures berita yang telah terenkripsi dengan menggunakan suatu algoritma hash function. Dan mengenkripsi digital signature tersebut. Kemudian user A mengirimkan berita dan digital signature berita yang keduanya telah di enkripsi tersebut kepada user B.



Gambar 2.2 User A

User B akan menerima berita yang telah terenkripsi dan digital signature dari user A. Langkah pertama, user B akan mendekripsi digital signature tersebut. Kemudian ia mencari nilai hash dari berita yang telah terenkripsi untuk kemudian dicocokkan dengan digital signature berita tersebut. Jika hasil nya sama, maka berita tersebut asli, dan berita tersebut terjamin keutuhannya. Jika hasilnya berbeda maka integritas berita tersebut patut dipertanyakan. Karena perbedaan satu huruf pada

inputan hash function akan menghasilkan nilai hash yang jauh berbeda. Sehingga kita dapat melihat integritas atau keaslian berita tersebut tanpa harus membuka berita tersebut.



Gambar 2.3 User B

Jika sama, maka berita tersebut asli, user B dapat mendekripsi berita tersebut

Hash function adalah sebuah fungsi h yang :

Konsep One-Way Hash Function diperkenalkan pertama kali oleh Diffie dan Hellman pada papernya "New Directions in Cryptography".

Definisi 1 Sebuah One-Way Hash Function (OWHF) adalah fungsi h yang memenuhi kondisi :

Sebuah X dapat mempunyai panjang yang bervariasi dan hasil dari h(x) hanya mempunyai sebuah panjang yang tetap yaitu n bit.Hash Function akan bersifat one-way yaitu jika diberikan sebuah Y hasil dari h, maka akan tidak mungkin (secara perhitungan) untuk menemukan sebuah berita X dimana h(X) = Y (preimage

resistant) dan diberikan X dan h(X) maka akan tidak mungkin (secara perhitungan) untuk menemukan sebuah berita X' ? X dimana h(X') = h(X) (second preimage resistant).Sebuah fungsi yang bersifat preimage resistant dikenal dengan one-way function (namun preimage resistance biasanya digunakan untuk hash function). Beberapa penulis menyebutkan second preimage resistantce sebagai weak collision resistance. Untuk beberapa aplikasi (seperti fungsi pseudo-random dan algoritma MAC yang berdasarkan hash functions) sebagian besar dari input dapat diketahui namun sulit untuk menemukan bagian yang tidak diketahui pada input tersebut. Hal seperti ini disebut dengan partial preimage resistance.

Definisi dari collision resistance hash functions (CRHF) secara formal

di perkenalkan oleh Yuval pada papernya "How to swindle Rabin".

Definisi 1 Sebuah Collision Resistance Hash Functons (CRHF) adalah fungsi h yang memenuhi kondisi :

1. Sebuah X dapat mempunyai panjang yang bervariasi dan hasil dari h(x) hanya mempunyai sebuah panjang yang tetap yaitu n bit
2. Hash Function tersebut harus merupakan OWHF, yaitu memenuhi preimage resistance dan second preimage resistance.

3. Hash function tersebut harus bersifat collision resistant, yaitu dimana tidak mungkin (secara perhitungan) untuk menemukan dua berita yang mempunyai nilai hash yang sama.

Jelas dilihat bahwa menemukan sebuah second preimage tidak akan lebih mudah dibanding menemukan sebuah collision. Bagaimanapun juga untuk menentukan hubungan yang pasti dari kedua kondisi diatas memerlukan definisi khusus. Pada kondisi tertentu, collision resistance menyebabkan preimage resistance dan second preimage resistance.

MD5 Message Digest Algorithm (RFC 1321) ditemukan oleh Ron Rivest pada MIT.Sampai beberapa tahun ketika brute-force dan kriptanalisa berkembang pesat, MD5 adalah algoritma hash function yang paling banyak digunakan. Input algoritma ini adalah sebuah berita dengan panjang yang bervariasi dan menghasilkan output sebuah 128-bit message digest.

SHA Family adalah merupakan algoritma hash function yang dibuat oleh National Security Agency (NSA) dan dipublikasikan sebagai standar oleh pemerintah USA.Algoritma SHA family yang paling banyak digunakan adalah SHA-1 dan telah banyak diaplikasikan pada berbagai macam aplikasi keamanan dan protokol keamanan seperti SSL, PGP, SSH, S/MIME dan IPSec.

Tabel 2.1 Perbandingan SHA family

Algoritma Hash	Ukuran Digest Hash (bits)	Ukuran Internal State (bits)	Ukuran Blok (bytes)
SHA-0	160		
SHA-1	160	160	64
SHA-224	224	256	64
SHA-256	256	256	64
SHA-384	384	512	128
SHA-512	512	512	128

Catatan : Internal state adalah “sum (digest) internal hash“ setelah tiap kompresi dari satu blok data.

Banyak peneliti yang mencoba melakukan cryptanalysis terhadap algoritma hash functions yang ada. Dan perkembangan cryptanalysis tersebut sangat mengejutkan.Pada konferensi CRYPTO 98, dua peneliti asal Perancis mempresentasikan sebuah attack terhadap SHA-0 [3] dimana collisions dapat ditemukan dengan kompleksitas 261; lebih rendah dari 280, kompleksitas ideal suatu hash functions.

Pada tahun 2004, Biham dan Chen menemukan near-collisions untuk SHA-0 [8] , yaitu menemukan dua berita yang hampir mempunyai nilai hash yang sama,

dimana dari 160 bit output, 142 bitnya sama. Mereka juga menemukan full collisions pada SHA-0 dengan 62 round dari total 80 round.

Berikutnya, pada 12 Agustus 2004, sebuah collision untuk full SHA-0 diumumkan oleh Joux, Carribault, Lemuet dan Jalby. Dengan menggunakan Chabaud dan Joux Attack[10]. Yaitu menemukan collisions dengan kompleksitas 251 dan memerlukan 80.000 jam dengan menggunakan superkomputer yang didalamnya terdapat 256 buah prosesor Itanium 2.

Pada 17 Agustus 2004, pada Rump Session CRYPTO 2004, sebuah hasil pendahuluan telah diumumkan oleh Wang, Feng, Lai dan Yu, mengenai attack terhadap MD5, SHA-0 dan hash functions lainnya [5]. Kompleksitas attack mereka terhadap SHA-0 adalah sekitar 240, jauh lebih baik dibandingkan attack yang dilakukan oleh Joux dan yang lainnya.

Pada Februari 2005, sebuah attack kembali dilakukan oleh Xiaoyun Wang, Yiqun Lisa Yin, dan Hongbo Yu, dan diumumkan bahwa mereka dapat menemukan collision pada SHA-0 dalam 239 operasi [6].

Berdasarkan beberapa kriptanalisis terhadap SHA-0, beberapa ahli menyarankan untuk menggunakan SHA-1 sebagai kriptosistem baru. Setelah hasil tersebut dipublikasikan, NIST mengumumkan bahwa mereka berencana menghapus

setahap demi setahap penggunaan SHA-1 sampai 2010, dan menggantikan nya dengan SHA-2 variant.

Pada tanggal 17 Agustus 2005, perkembangan attack terhadap SHA-1 diumumkan oleh Xiaoyun Wang, Andrew Yao dan Frances Yao pada rump session CRYPTO 2005,dan dapat menemukan collision dengan kompleksitas 263. Karena telah ditemukannya full-collision pada beberapa algoritma hash functions (selain SHA-1 dapat juga diterapkan pada MD5, RIPEMD) oleh para peneliti dari China tersebut, NIST (National Institute of Standards and Technology) mengadakan suatu workshop khusus mengenai hash functions di Santa Barbara pada Oktober 2005. Dalam workshop tersebut mereka mengeluarkan issue mengenai algoritma hash function yang dapat menggantikan SHA-1. Issue tersebut sangat penting mengingat SHA-1 telah dijadikan standard algoritma hash function oleh pemerintah USA (NIST). Sampai saat ini belum ada algoritma hash yang dijadikan standard (dalam hal ini oleh USA dan indonesia sendiri belum mempunyai standard tersendiri mengenai hal ini), padahal aplikasi yang menggunakan hash functions sebagai digital signature, autentikasi banyak dipakai di perbankan atau aplikasi online lainnya. Hal ini akan menjadi masalah tersendiri bagi keamanan informasi.

2.3.1 Algoritma SHA-1

Fungsi Hash SHA adalah lima fungsi hash kriptografis yang dirancang oleh National Security Agency (NSA) dan yang diterbitkan oleh NIST sebagai U.S.

Federal Information Processing Standard. SHA adalah kepanjangan dari Secure Hash Algorithm. Algoritma-algoritma hash menghitung suatu representasi digital dengan panjang yang pasti atau telah ditentukan (yang yang dikenal sebagai suatu message digest) dari suatu urutan data masukan (pesan) yang panjangnya bermacam-macam. Mereka disebut aman ketika, dapat dikomputasi

untuk:

1. Menemukan suatu pesan yang berpasangan dengan suatu message digest yang diberi, atau
2. Menemukan dua pesan yang berbeda yang menghasilkan message digest yang sama. Setiap perubahan akan, dengan kemungkinan yang sangat tinggi, menghasilkan suatu message digest yang berbeda.

SHA-1 menghasilkan suatu message digest dengan panjang 160 bit; nomor di dalam nama-nama empat algoritma yang lainnya menandakan panjangnya bit dari digest yang mereka hasilkan. SHA-1 digunakan secara luas di dalam beberapa aplikasi-aplikasi keamanan dan protokol-protokol, termasuk TLS dan SSL, PGP, SSH, S/MIME, dan IPsec. SHA-1 dianggap sebagai pengganti lanjutan MD5,

algoritma yang telah populer sebelumnya.

Pseudocode untuk SHA-1 adalah sebagai berikut:

Note 1: All variables are

unsigned 32 bits and wrap

modulo 232 when calculating

Note 2: All constants in this

pseudo code are in big endian.

Within each word, the most

significant bit is stored in

the leftmost bit position

Initialize variables:

h0 = 0x67452301

h1 = 0xEFCDAB89

h2 = 0x98BADCFE

h3 = 0x10325476

h4 = 0xC3D2E1F0

Pre-processing:

append the bit '1' to the message append k bits '0', where k is the minimum number ≥

0 such that the resulting message length (in *bits*) is congruent to 448 (mod 512)

append length of message (before pre-processing), in *bits*, as 64-bit big-endian integer

Process the message in successive 512-bit chunks:

break message into 512-bit

chunks

for each chunk

break chunk into sixteen 32-bit big-endian words w[i], $0 \leq i \leq 15$

Extend the sixteen 32-bit

words into eighty 32-bit words:

for i **from** 16 to 79

w[i] = (w[i-3] **xor** w[i-

8] **xor** w[i-14] **xor** w[i-16])

leftrotate 1

Initialize hash value for

this chunk:

a = h0

b = h1

c = h2

d = h3

e = h4

Main loop:

for i **from** 0 to 79

if $0 \leq i \leq 19$ **then**

f = (b **and** c) **or**

((**not** b) **and** d)

k = 0x5A827999

else if 20 ≤ i ≤ 39

f = b **xor** c **xor** d

k = 0x6ED9EBA1

else if 40 ≤ i ≤ 59

f = (b **and** c) **or** (b

and d) **or** (c **and** d)

k = 0x8F1BBCDC

else if 60 ≤ i ≤ 79

f = b **xor** c **xor** d

k = 0xCA62C1D6

temp = (a **leftrotate** 5)

+ f + e + k + w[i]

e = d

d = c

c = b **leftrotate** 30

b = a

a = temp

Add this chunk's hash to

result so far:

h0 = h0 + a

h1 = h1 + b

h2 = h2 + c

h3 = h3 + d

h4 = h4 + e

Produce the final hash value

(big-endian):

digest = hash = h0 **append** h1

append h2 **append** h3 **append** h4

Initialize hash value for

this chunk:

a := h0

b := h1

c := h2

d := h3

e := h4

f := h5

g := h6

h := h7

Main loop:

for i **from** 0 to 63

s0 := (a **rightrotate** 2)

xor (a **rightrotate** 13) **xor** (a

rightrotate 22)

maj := (a **and** b) **xor** (a

and c) **xor** (b **and** c)

t2 := s0 + maj

s1 := (e **rightrotate** 6)

xor (e **rightrotate** 11) **xor** (e

rightrotate 25)

ch := (e **and** f) **xor**

((**not** e) **and** g)

t1 := h + s1 + ch +

k[i] + w[i]

h := g

g := f

f := e

e := d + t1

d := c

c := b

b := a

a := t1 + t2

Add this chunk's hash to

result so far:

h0 := h0 + a

h1 := h1 + b

h2 := h2 + c

h3 := h3 + d

h4 := h4 + e

h5 := h5 + f

h6 := h6 + g

h7 := h7 + h

Produce the final hash value

(big-endian):

digest = hash = h0 **append** h1

append h2 **append** h3 **append** h4

append h5 **append** h6 **append** h7

2.3.1.1 Kriptanalisis SHA-1

Pada tahun 2005, Rijmen dan Oswald mempublikasiknan serangan pada versi SHA-1 yang direduksi (hanya menggunakan 53 putaran dari 80 putaran) dan

menemukan kolisi dnegan kompleksitas sekitar 280 operasi. [1] Pada bulan Februari 2005, Xiayoun Wang,Yiqun Lisa Yin, dan Hongbo Yo mempublikasikan serangan yang dapat menemukan *collision* pada versi penuh SHA- yang membutuhkan sekitar 269 operasi. Di dalam ilmu pengetahuan komputer, *hash collision* adalah suatu situasi yang terjadiketika dua masukan yang berbeda pada suatu fungsi hash menghasilkan keluaran yangidentik. Semua fungsi hash berpotensial untuk menghasilkan hash collision, meskipundemikian fungsi hash yang dirancang denganbaik dapat mengurangi intensitas terjadinya hash collision (bandingkan dengan suatufungsi dengan kurang baik merancang) ataulebih sulit untuk ditemukan. Bagaimanapun,banyak sekali fungsi hash, termasuk fungsi hash yang sering digunakan dalam kriptografi, menghasilkan suatu keluaran ukuran yang ditetapkan dari satu pesan yang panjang. Dalam desain yang demikian, akan selalu ada *hash collision*, karena setiap fungsi hash yang diberi harus berpasangan dengan satu bilangan tak hingga dari masukan-masukan yang mungkin.Contoh terjadinya *hash collision* adalah sebagai berikut:f(x) = f(y) ; x != y (untuk nilai masukan yang berbeda x dan y,fungsi hash f menghasilkan nilai keluaran yang sama).

2.3.2 Algoritma SHA-256

Untuk menghindari kriptanalisis yang disebabkan oleh Collision pada SHA-1, maka dikembangkan algoritma-algoritma selanjutnya. Algoritma SHA generasi selanjutnya disebut juga algoritma SHA-2. Pada makalah ini akan dibahas salah satu dari

algoritma SHA-2 yaitu SHA-256. Algoritma SHA-256 dapat digunakan untuk menghitung nilai message digest dari sebuah pesan, dimana pesan tersebut memiliki panjang maksimum 264 bit. Algoritma ini menggunakan sebuah message schedule yang terdiri dari 64 elemen 32-bit word, delapan buah variabel 32-bit, dan variable penyimpanan nilai hash 8 buah word 32-bit. Hasil akhir dari algoritma SHA-256 adalah sebuah message digest sepanjang 256-bit. Preprocessing dilakukan dengan menambahkan bit pengganjal, membagi-bagi pesan dalam block berukuran 512-bit, dan terakhir menginisiasi nilai hash awal. Proses penambahan bit pengganjal adalah sama dengan aturan penambahan bit pengganjal pada SHA-1. Dalam proses komputasinya, SHA-256 menggunakan enam fungsi lojik, dimana setiap fungsi beroperasi mengguankan tiba buah word 32-bit (x,y, dan z) dan keluarannya berupa sebuah word 32-bit. Berikut ini adalah fungsi-fungsi SHA-256:

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$$
$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\sum_0^{256}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$
$$\sum_1^{256}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$
$$\sigma_0^{256}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$$
$$\sigma_1^{256}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

Gambar 2.4 Fungsi-Fungsi SHA-256

Nilai hash awal pada algoritma SHA-256

adalah sebagai berikut:

h0 := 0xc1059ed8

h1 := 0x367cd507

h2 := 0x3070dd17

h3 := 0xf70e5939

h4 := 0xffc00b31

h5 := 0x68581511

h6 := 0x64f98fa7

h7 := 0xbefa4fa4

Dan konstanta dalam SHA-256 adalah sebagai berikut (64 buah):

```
428a2f98 71374491 b5c0fbcf e9b5dba5
3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3
72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc
2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7
c6e00bf3 d5a79147 06ca6351 14292967

27b70a85 2e1b2138 4d2c6dfc 53380d13
650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3
d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5
391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208
90befffa a4506ceb bef9a3f7 c67178f2
```

Gambar 2.5 Konstanta SHA

Berikut ini adalah pseudo-code dari algoritma SHA-256.

Note 1: All variables are

unsigned 32 bits and wrap

modulo 232 when calculating

Note 2: All constants in this

pseudo code are in big endian

Initialize variables

(first 32 bits of the

fractional parts of the square

roots of the first 8 primes

2..19):

h0 := 0x6a09e667

h1 := 0xbb67ae85

h2 := 0x3c6ef372

h3 := 0xa54ff53a

h4 := 0x510e527f

h5 := 0x9b05688c

h6 := 0x1f83d9ab

h7 := 0x5be0cd19

Initialize table of round constants

(first 32 bits of the *fractional parts* of the cube roots of the first 64 primes 2..311):

k[0..63] :=

0x428a2f98, 0x71374491,

0xb5c0fbcf, 0xe9b5dba5,

0x3956c25b, 0x59f111f1,

0x923f82a4, 0xab1c5ed5,

0xd807aa98, 0x12835b01,

0x243185be, 0x550c7dc3,

0x72be5d74, 0x80deb1fe,

0x9bdc06a7, 0xc19bf174,

0xe49b69c1, 0xefbe4786,

0x0fc19dc6, 0x240ca1cc,

0x2de92c6f, 0x4a7484aa,

0x5cb0a9dc, 0x76f988da,

0x983e5152, 0xa831c66d,

0xb00327c8, 0xbf597fc7,

0xc6e00bf3, 0xd5a79147,

0x06ca6351, 0x14292967,

0x27b70a85, 0x2e1b2138,

0x4d2c6dfc, 0x53380d13,

0x650a7354, 0x766a0abb,

0x81c2c92e, 0x92722c85,

0xa2bfe8a1, 0xa81a664b,

0xc24b8b70, 0xc76c51a3,

0xd192e819, 0xd6990624,

0xf40e3585, 0x106aa070,

0x19a4c116, 0x1e376c08,

0x2748774c, 0x34b0bcb5,

0x391c0cb3, 0x4ed8aa4a,

0x5b9cca4f, 0x682e6ff3,

0x748f82ee, 0x78a5636f,

0x84c87814, 0x8cc70208,

0x90befffa, 0xa4506ceb,

0xbef9a3f7, 0xc67178f2

Pre-processing:

append the bit '1' to the message append k bits '0', where k is the minimum number >= 0 such that the resulting message length (in *bits*) is congruent to 448 (mod 512)

append length of message (before pre-processing), in *bits*, as 64-bit big-endian integer

Process the message in successive 512-bit chunks:

break message into 512-bit

chunks

for each chunk

break chunk into sixteen 32-bit big-endian words

w[0..15]

Extend the sixteen 32-bit words into sixty-four 32-bit words:

for i **from** 16 to 63

s0 := (w[i-15]

rightrotate 7) **xor** (w[i-15]

rightrotate 18) **xor** (w[i-15]

rightshift 3)

s1 := (w[i-2]

rightrotate 17) **xor** (w[i-2]

rightrotate 19) **xor** (w[i-2]

rightshift 10) w[i] := w[i-16] **+** s0 **+** w[i-7] **+** s1

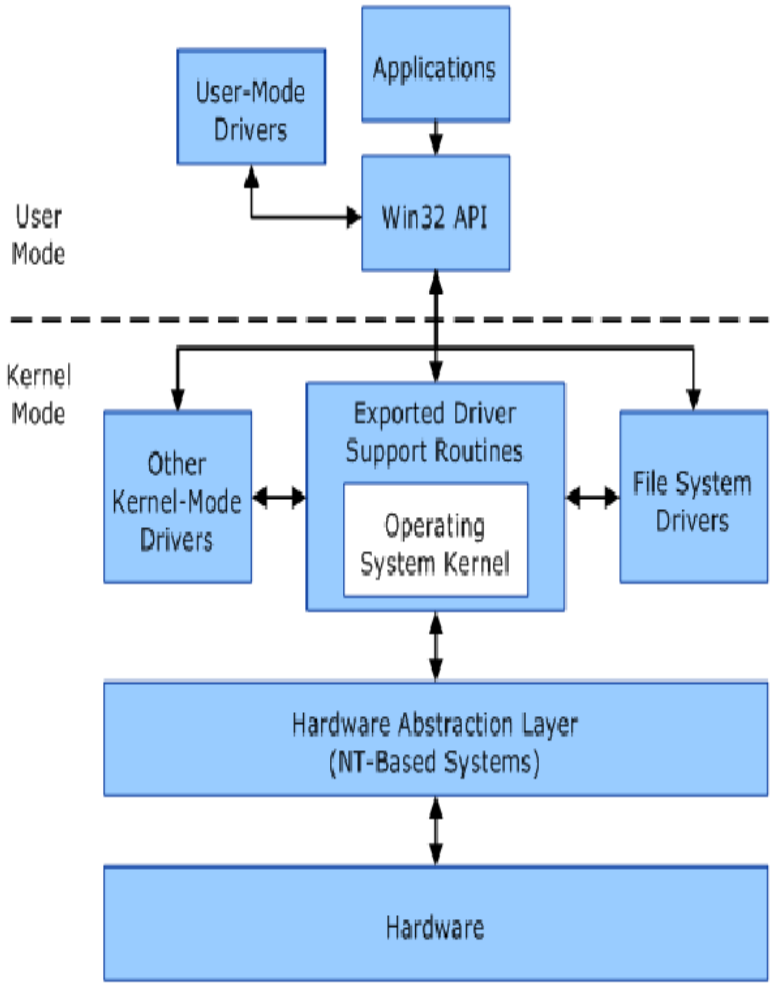
2.4 Device Driver

Seperti yang dipaparkan pada bagian 2.4, salah satu cara untuk memunculkan virtual disk adalah dengan memanfaatkan sebuah file sebagai kontainer (file volume).

Hal ini dimungkinkan dengan membuat sebuah driver. Driver merupakan wadah koleksi fungsi dan prosedur yang sistem operasi panggil untuk menjalankan berbagai operasi yang berkaitan dengan perangkat. Dalam kasus ini (virtual disk driver) perangkat keras yang dimaksud adalah hard disk.

Seperti yang dapat dilihat pada gambar, sistem operasi memiliki komponen user mode dan kernel mode. User mode driver dan aplikasi yang termasuk dalam user mode tidak memiliki akses langsung terhadap perangkat keras dan memori. Komponen ini harus memanggil Application Programming Interface atau API dari setiap kali mengakses perangkat keras dan memori. Isolasi semacam itu memberikan perlindungan sehingga kerusakan yang ditimbulkan pada komponen user mode lebih mudah untuk ditanggulangi dibandingkan dengan pada komponen kernel mode.

Komponen kernel mode mampu melakukan operasi dan mengakses struktur sistem yang terlarang untuk komponen user mode, misalnya operasi yang menyangkut Input/Output (I/O), konfigurasi sistem, Plug and Play, pengaturan memori dan fitur- fitur sistem operasi lainnya.



Gambar 2.6 Komponen Sistem Operasi