Pemanfaatan Konsep Kriptografi Visual untuk membangun *Java API* Pengamanan Perangkat Lunak

Yogi Adytia Marsal - 13508016
Teknik Informatika
Institut Teknologi Bandung

Dr. Ir. Rinaldi Munir, M.T - 132084796
Teknik Informatika
Institut Teknologi Bandung

Abstrak - Pembajakan merupakan pelanggaran hak cipta yang memiliki dampak negatif bagi pengguna maupun pengembang perangkat lunak. Dampak bagi pengguna adalah semakin mahalnya harga perangkat lunak yang asli, dan dampak bagi pengembang adalah kerugian materil. Oleh karena itu, diperlukan inovasi baru dalam keamanan perangkat lunak untuk melindungi perangkat lunak tersebut, maka pada kesempatan ini dirancang sebuah mekanisme perangkat lunak baru yang diimplementasikan kedalam sebuah *Java API*.

Mekanisme ini memanfaatkan konsep dasar dari kriptografi visual yang digunakan sebagai pengamanan file utama. Namun, mekanisme ini membutuhkan beberapa dukungan dari teknik kriptografi lainnya seperti, *Hash*, *AES* (*Advanced Encryption Standard*) dan *SSL* (*Secure Socket Layer*). Beberapa teknik tersebut dikombinasikan menjadi sebuah teknik pengamanan perangkat lunak yang kompleks yang bisa mengatasi beberapa masalah pembajakan.

Metoda tersebut diimplementasikan kedalam *Java API*. Setiap Java API tersebut akan diuji setelah diimplementasikan kedalam sebuah sistem perangkat lunak yang kompleks. Pengujian tersebut terbagi atas pengujian fungionalitas, faktor ekternal dan keamanannya. Dan pengujian tersebut ditujukan untuk memperlihatkan metode ini sudah bisa digunakan dengan baik sebagai pengamanan perangkat lunak.

Kata kunci : pembajakan, kriptografi visual, *Hash*, *AES*, *SSL*, *API*, *server*, *installer*, aplikasi.

I. PENDAHULUAN

Perkembangan zaman membuat kebutuhan manusia dari hari ke hari meningkat sehingga manusia memanfaatkan ilmu mereka untuk mencapai kebutuhan tersebut, dan salah satu implementasi dari ilmu yang dimiliki oleh manusia untuk mempermudah mereka memenuhi kebutuhan mereka adalah dengan memanfaatkan perangkat lunak. Dengan adanya perangkat lunak manusia bisa mempermudah dan mengefisiensikan pekerjaan mereka untuk bisa memenuhi kebutuhan yang semakin bertambah tersebut. Namun, keterbatasan kemampuan manusia khususnya materi, membuat manusia menemukan berbagai cara untuk mendapatkan perangkat lunak tersebut dengan cara yang lebih ekonomis, dan cara tersebut sering disebut sebagai "pembajakan".

Pembajakan membuat kerugian yang sangat besar di pihak pengembang perangkat lunak, dan demi mengatasi masalah tersebut pengembang perangkat lunak berusaha berbagai metode untuk menutup celah bagi para "*Hacker*" atau "*Cracker*" untuk melakukan pembajakan. Dampak buruk pembajakan perangkat lunak dipihak pengguna adalah harga perangkat lunak yang meningkat drastis (Altinkerner, 2003). Namun harga perangkat lunak tersebut bisa saja tidak mencekik pengguna jika perangkat lunak tersebut tidak bisa lagi dibajak oleh pihak yang tidak bertanggung jawab.

Banyak metode pada saat ini yang memiliki tingkat pengamanan yang cukup tinggi, namun ada beberapa faktor yang menyebabkan metode tersebut tidak digunakan.Selain itu metode tersebut tentunya masih memiliki celah bagi pihak yang tidak bertanggungjawab untuk melakukan pembajakan.

Maka dirancang sebuah *API (Application Programming Interface)* dengan menggunakan metode pengamanan perangkat lunak baru yang memanfaatkan prinsip kriptografi visual yang diterapkan pada berkas – berkas yang dianggap penting dalam eksekusi program. Awalnya pengguna harus melakukan validasi ke sebuah *server* untuk mendapatkan potongan *file*nya yang nanti digabungkan dengan konsep kriptografi visual. Jika perangkat lunak tersebut mengetahui terjadi pembajakan, maka file tersebut tidak bisa dijalankan.

II. LANDASAN TEORI

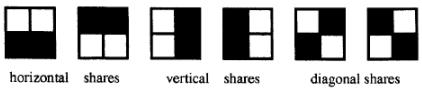
Dalam pembangunan *Java API* pengamanan perangkat lunak ini dibutuhkan beberapa landasan teori yang akan menjadi dasar perancangan metode

pengamanan perangkat lunak tersebut. Teori – teori yang terkait antara lain adalah :

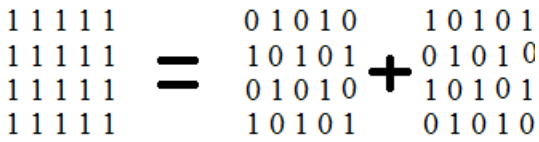
1. Visual Kriptografi

Kriptografi visual adalah salah satu teknik dalam kriptografi yang tidak membutuhkan komputasi kriptografi untuk membaca informasi yang tersembunyinya (Naor, 1994). Pada teknik ini gambar dipecah berdasarkan *pixel*-nya menjadi beberapa potongan. Pada potongan tersebut jika *pixel* merupakan *pixel* kosong maka diisi dengan warna transparan.Untuk pemotongannya bisa dilakukan secara horizontal, vertikal ataupun diagonal dan bisa dikombinasikan seperti pada Gambar II.1.

Jika gambar yang ingin dipecah digambarkan sebagai matriks yang melambangkan *pixel* yang dimilikinya. Untuk *pixel* yang memiliki gambar digambarkan dengan 1 dan untuk *pixel* yang tidak memiliki gambar atau transparan dilambangkan dengan 0 maka, jika dimisalkan ukuran gambarnya adalah 5x4 dan gambar dishare menjadi dua potongan maka bentuknya adalah seperti gambar II.2.



Gambar II-1 Beberapa Jenis Potongan yang bisa Dilakukan untuk Gambar



Gambar II-2 Matriks untuk Kriptografi Visual

Dengan memecah *pixel*-nya lagi menyebabkan hasilnya akan semakin besar daripada gambar aslinya dan jika didekripsi gambar akan terlihat bagus kembali setelah diperkecil. Kriptografi visual ini memiliki kelemahan (Munir, 2010), yaitu

a. Gambar hasil dekripsi tidak sama persis dengan gambar semula
b. Gambar hasil dekripsi mengandung *noise*
c. Gambar hasil enkripsi tidak mengandung makna sehingga mencurigakan mengandung pesan rahasia

2. AES (*Advanced Encryption Standard*)

Pada January 1997, *National Institute of Standards and Technology* (*NIST*) mengumumkan standar baru untuk enkripsi yaitu *AES* (Daemen, 2002). Garis besar Algoritma *Rijndael* yang beroperasi pada blok 128-bit dengan kunci 128-bit adalah sebagai berikut (di luar proses pembangkitanround key):

1. *AddRoundKey*: melakukan *XOR* antara state awal (plainteks) dengan cipher key. Tahap ini disebut juga initial round.
2. Putaran sebanyak Nr – 1 (Mod panjang data dengan chipernya kurang satu) kali. Proses yang dilakukan pada setiap putaran adalah:
   a. *SubBytes*: substitusi *byte* dengan menggunakan tabel substitusi (S-box).
   b. *ShiftRows*: pergeseran baris-baris *array state* secara *wrapping*.
   c. *MixColumns*: mengacak data di masing-masing kolom *array state*.
   d. *AddRoundKey*: melakukan *XOR* antara state sekarang *round key*.
3. *Final round*: proses untuk putaran terakhir:
   a. *SubBytes*
   b. *ShiftRows*
   c. *AddRoundKey*

3. RIPEMD-160

RIPEMD-160 adalah fungsi *hash* yang merupakan evolusi dari *MD4* yang diperkenalkan oleh Ron Rivest pada tahun 1990 (preneel, 1997). *RIPEMD* merupakan fungsi *hash* yang bisa menerima *input*-an dalam ukuran yang tidak terbatas dan menghasilkan *output* dengan ukuran tertentu tergantung dengan tipe dari *RIPEMD*-nya, salah satunya adalah *RIPEMD*-160 akan menghasilkan data dengan ukuran 160 *bit*.

RIPEMD-160 mengecilkan ukuran input dengan membaginya menjadi 512 *bit*. Setiap blok tersebut dibagi menjadi 16 *string* yang ukuran tiap stringnya adalah 4 *bytes*. Setiap string tersebut yang memiliki ukuran 4 *bytes* dikonversi menjadi 32 *bit* dengan menggunakan *Little-Endian*.

Proses pada *RIPEMD* diawali dengan inisialisasi dengan lima kata yang terdiri dari 32 *bit*. Fungsional utama dalam fungsi ini adalah fungsi kompresi. Fungsi ini memproses state awal dengan state

berikutnya yang terdapat dalam 16 *string* yang dipisahkan untuk setiap blok. Proses yang terjadi dalam setiap putaran adalah :

a. *Operation in One Step*. Merupakan fungsi utama dalam fungsi hash ini yang mengubah nilai setiap lima kata yang terdiri dari 32 *bit*.

b. *Ordering of the Message words*. Merupakan fungsi untuk memutar posisi 16 string untuk setiap putaran.

4. SSL (*Secure Socket Layer*)

Sebuah *server* biasanya menyediakan sumber daya untuk jaringan *client*. Program pada *client* mengirimkan permintaan ke program *server*, dan *server* akan merespon permintaannya. Untuk meningkatkan keamanan agar tidak bisa dideteksi data yang dikirim oleh server, maka dikembangkan koneksi *socket* dengan menggunakan algoritma kriptografi. Pada saat ini telah berkembang teknologi *SSL* yang bisa digunakan untuk keamanan koneksi *socket*. Untuk *SSL* ini dibutuhkan kunci yang bisa di-*generate* dengan menggunakan *Toolkit java*. Setelah memiliki *file .jks* dan *file .csr* maka *server* dan *client* bisa melakukan komunikasi. Untuk koneksi menggunakan *SSL*, *port* yang bisa digunakan adalah 8888.

III. PENELITIAN TERKAIT

Pengamanan perangkat lunak sudah lama dilakukan oleh para *programmer* untuk melindungi perangkat lunak mereka dari pembajakan. Pada bagian ini akan dijelas tiga penelitian yang menjadi inspirasi dalam pembangunan metoda pengamanan perangkat lunak ini, yaitu :

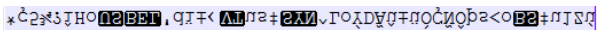
1. Penelitian *Hiding Program slices for software security*, penelitian ini mengatasi pembajakan dengan menyembunyikan bagian tertentu dari perangkat yang diletakkan disuatu tempat rahasia, sehingga tidak bisa dipindahkan secara manual (Zhang, 1998).

2. Penelitian *Method for securing software to Decrease Software Piracy*, penelitian ini mengatasi pembajakan dengan menggunakan nomor serial dan pengecekan berkala oleh server (Colvin, 2004).

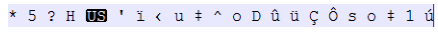
3. Penelitian *Preventing Piracy With Crypto-Microprocessors*, penelitian ini mengatasi pembajakan dengan menggunakan mikroprosessor yang menjadi sebuah kunci untuk melakukan instalasi ataupun untuk menjalankan perangkat lunak tersebut (Best, 1980).

IV. RANCANGAN SOLUSI

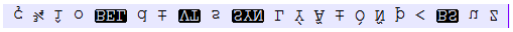
Konsep dasar kriptografi visual adalah membagi sebuah gambar menjadi beberapa buah *share* dimana untuk menggabungkan *share* tersebut tidak memerlukan proses komputasi yang kompleks. Konsep tersebutlah yang akan dimanfaatkan untuk metode pengamanan perangkat lunak. Pada metode yang dirancang ini, konsep kriptografi visual tidak akan membuat *share* dari sebuah gambar tetapi dari *file* yang dipilih oleh pengembang perangkat lunak yang nantinya dijadikan kunci untuk menjalankan perangkat lunak tersebut.



(a)



(b)



(c)

Gambar IV-1 (a) Isi *File* Utuh, (b) *Share* Pertama, (c) *Share* Kedua.

Pemotongan *file* menjadi beberapa buah share menggunakan konsep kriptografi visual terdapat perbedaan dengan pemotongan gambar yang harus ditentukan pada metode ini. Permasalahan itu adalah, pada pemotongan gambar terdapat *pixel* kosong atau transparan yang menyatakan isi sebenarnya ada pada share lainnya, sedangkan pada pemotongan *file* harus ditentukan karakter apa yang akan digunakan untuk mengisi bagian kosong tersebut. Contoh pemotongan pada *file* dapat dilihat pada gambar IV-1. pemotongan ini mengunakan karakter spasi sebagai pengganti *pixel* kosong yang terdapat pada pemotongan gambar.

File yang akan dipotong ada dua jenis, yaitu *file* yang *share*-nya dibuat sebelum perangkat lunak didistribusikan dan *file* yang *share*-nya dibuat saat proses instalasi perangkat lunak di-*client*.

Beberapa *File share* yang dibuat sebelum perangkat lunak didistribusikan diletakkan disebuah *server*. *User* bisa memperoleh *share* tersebut pada proses

instalasi perangkat lunak. Proses instalasi tersebut akan membutuhkan koneksi internet yang akan menghubungi *server* untuk memperoleh *share*-nya. Tujuan dari proses ini adalah untuk memastikan proses instalasi benar-benar menghubungi *server* agar bisa didata oleh *server* kunci mana saja yang telah digunakan dan data *user*yang diperoleh selama proses instalasi.

Untuk mencegah terjadinya pembajakan tersebut *file share* yang disimpan di-*client* harus dilengkapi dengan proses pengamanan yang baik. Proses pengamanan yang akan digunakan adalah dengan memanfaatkan AES yang memiliki tingkat keamanan yang tinggi pada saat ini.

Rancangan dasar yang jelaskan sebelum digabungkan menjadi sebuah metoda yang komplek dan diimplementasikan pada Java API. Implementasinya dibagi menjadi tiga bagian, yaitu :

a. Server

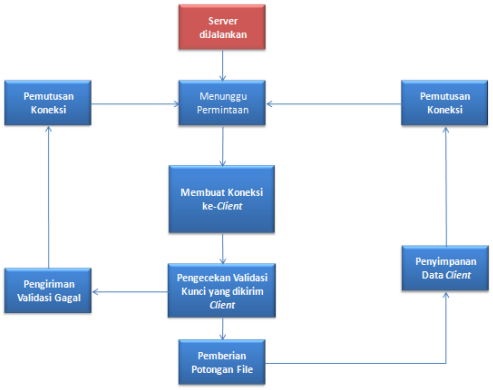
Server memiliki potongan yang tidak terdapat di *installer*, oleh karena itu *installer* akan meminta permintaan ke *server* untuk melengkapi potongan yang tidak dimilikinya untuk digabungkan menjadi *file* yang utuh.

Server akan selalu hidup untuk menunggu permintaan dari *installer*. Saat permintaan diperoleh *server* akan melakukan pengecekan beberapa data dari *installer*, seperti *serial number*. Pada proses dipengguna, nomor serial bukan diinput dari pengguna, tapi sudah terdapat pada *installer*. Jika nomor tersebut telah digunakan dengan data pengguna yang berbeda, maka potongan tidak akan dikirimkan.

Namun, jika validasi cocok, maka pengiriman potongan akan dilakukan oleh *server* dan data client akan disimpan disebuah database *server*. Dan setelah semua data terkirim, maka koneksi akan diputus oleh *server*. Gambaran proses di *server* ini bisa dilihat pada gambar IV-2.

b. Installer

Pada saat proses instalasi, pada awalnya perangkat lunak tersebut akan meminta tujuan direktor. Jika perangkat lunak tersebut perangkat lunak yang cukup besar, bisa melengkapi *installer* dengan *license agreement* sebelum memasuki tahapan pemilihan direktori. Namun dalam implementasi tugas akhir ini, proses tersebut tidak dilakukan.

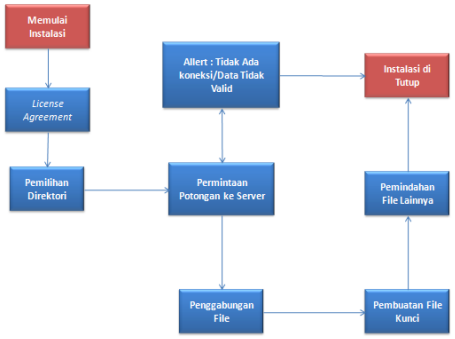


Gambar IV-2 Proses di *Server*

Setelah melakukan pemilihan direktori maka perangkat lunak akan menghubungi *server* untuk meminta *share* dari *file* yang akan dieksekusi yang disimpan di-*server*. Saat *file* tersebut telah diperoleh dari *server*, maka tahapan yang dilakukan adalah proses penggabungan menjadi sebuah *file* yang akan dieksekusidalam bentuk yang utuh dan siap untuk dijalankan.

Tahapan berikutnya yang terjadi adalah membuat sebuah *file* kunci yang akan mengambil data unik yang ada dalam komputer. Proses ini bisa dikombinasikan dengan data unik lainnya yang bersifat statik, sehingga tidak perlu menyimpan pembanding dari *file* kunci tersebut.

Dan tahapan terakhir adalah memindahkan beberapa *file* yang dibutuhkan perangkat lunak untuk menjalankan fungsionalitasnya. Dengan dipindahkannya *file-file* pendukung untuk menjalankan perangkat lunak tersebut, maka berakhirlah proses instalasi perangkat lunak tersebut dan perangkat lunak tersebut siap untuk digunakan. Prosesnya dapat dilihat pada gambar IV-3 dan IV-4.



Gambar IV-3 Proses Saat Instalasi

Gambar IV-4 Proses Pembuatan Kunci

c. Aplikasi

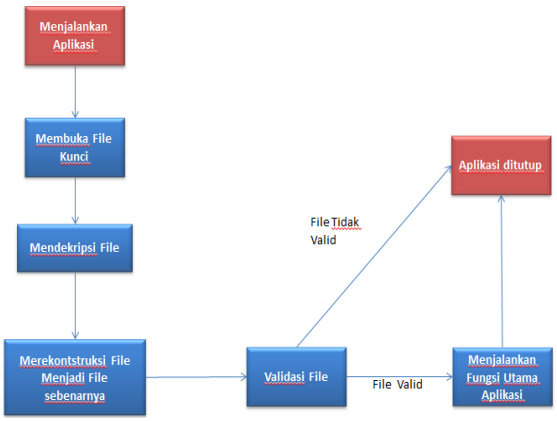
Setelah program melalui tahapan instalasi, maka pengamanan selanjutkan akan dilakukan pada *Client`s Side*. Pengamanan tersebut dilakukan setiap kali perangkat lunak tersebut dijalankan. Saat perangkat lunak tersebut dijalankan, ada beberapa tahapan yang harus dilalui oleh perangkat lunak tersebut sebelum melakukan fungsi utamanya. Beberapa tahapan ini dilakukan untuk validasi kunci perangkat lunak sebelum perangkat lunak tersebut menjalankan fungsi utamanya.

Validasi perangkat lunak diketahui dari sebuah kunci yang akan dirancang menggunakan konsep visual kriptografi dan didukung dengan pengamanan enkripsi *AES* dan *Hash RIPEMD*. Pada bagian ini ada beberapa hal yang harus dipertimbangkan, yaitu :

1. Implementasi visual kriptografi pada *file* eksternal.
2. Pemilihan kunci yang digunakan untuk enkripsi dan dekripsi dengan algoritma *AES*

Beberapa tahapan yang akan dilalui perangkat lunak sebelum menjalankan fungsi utama perangkat lunak tersebut dapat dilihat pada gambar IV-5.

Pada metode ini *file* eksternal yang dijadikan kunci untuk pengecekan validasinya, file tersebut dienkripsi dengan *AES* dan selanjutnya akan digabung data dari tiap file menjadi sebuah kunci menggunakan konsep visual kriptografi. Lalu program akan mengambil nomor *Motherboard* dan nomor *Hardisk* untuk digabungkan dan dihash dengan *RipeMD*. Data yang diperoleh dari file eksternal dan data yang dihash tersebut dicocokkan dan bisa ditentukan apakah data tersebut valid atau tidak. Setelah melakukan validasi, maka akan diketahui perangkat lunak adalah asli atau tidak. Saat validasi gagal, maka perangkat lunak tidak bisa dijalankan dan akan tertutup kembali.



Gambar IV-5 Aksi Saat Perangkat Lunak Dijalankan

Rancangan ketiga *API* tersebut diimplementasikan menjadi tiga buah *Java API* yang bisa digunakan sebagai dalam perangkat lunak. Pengamanan ini ditujukan pada program *Java* dikarenakan *API* yang dibangun menggunakan bahasa pemrograman *Java*.

Setiap *Java API* yang dibangun tersebut diimplementasikan kedalam perangkat lunak sehingga terdapat tiga perangkat lunak yang saling terkait. Perangkat lunak tersebut yaitu, *server*, *installer* dan aplikasi.

V. PENGUJIAN

Setelah implementasi dilakukan, maka diperlukan pengujian terhadap Java API dan implementasi dari Java API tersebut. Pengujian yang dilakukan untuk dibagi menjadi tiga bagian yaitu :

1. Pengujian Internal, bertujuan untuk menguji fungsi – fungsi yang dibangun pada perangkat lunak tersebut. Hal – hal yang diuji adalah :
   a. *Server* bisa melakukan koneksi *multiclient*.
   b. *Server* bisa mengakses *database*. *Server* berhasil mengakses *database*
   c. *Server* bisa mengirim data ke *client*
   d. *Installer* bisa merekonstruksi aplikasi.
   e. Aplikasi bisa melakukan validasi kunci.

2. Pengujian Eksternal, bertujuan untuk menguji faktor – faktor luar yang mempengaruhi berjalan dengan baiknya fungsi – fungsi yang terdapat pada *Java API*. Hal – hal yang diuji adalah :

a. Hasil rekonstruksi bisa dieksekusi.

b. Kunci terbentuk pada direktori yang telah ditentukan *developer*.

c. Kunci yang dibangun terenkripsi.

d. *Serial number* bisa digunakan untuk komputer yang sama.

3. Pengujian Keamanan, bertujuan untuk menguji kasus – kasus pembajakan yang mungkin terjadi. Hal – hal yang diuji adalah:

   a. Penyalinan data ke komputer lain.

   b. Pencurian data saat pengiriman *file*.

   c. Perekonstruksian eksekusi *file* secara manual.

   d. Penginstalasian ulang *installer* dengan *serial number* yang sama pada komputer yang berbeda.

Java API yang dibangun setelah diujicoba dengan beberapa kasus diatas menunjukkan bahwa Java API tersebut bisa berjalan dengan baik dan bisa mengatasi beberapa masalah pembajakan. Fungsi – fungsi bisa berjalan sesuai dengan harapan. Faktor – faktor luar yang mempengaruhi kerja fungsi tersebut juga memberikan dampak kepada fungsi sesuai dengan harapan. Selain itu kasus- kasus keamanan yang rencanakan pada pengujian keamanan juga bisa diatasi oleh *Java API* yang dibangun.

VI. KESIMPULAN

Kesimpulan dari pelaksanaan tugas akhir ini adalah:

a. Konsep *visual kriptografi* yang biasa digunakan untuk memecah sebuah sebuah gambar menjadi beberapa *share*

b. Pada proses instalasi, menggunakan *SSL* (*Socket Secure Layer*) untuk menjaga keamanan jaringan.

c. *File* eksekusi, keamanannya dijaga oleh jaringan karena potongannya terdapat pada *server*, sedangkan *file kunci* menggunakan *AES* yang kuncinya tidak disimpan pada client, tapi di-*generate* setiap kali melakukan pengecekan.

d. Pada tugas akhir ini, kunci diperoleh dari kombinasi nomor unik yang terdapat pada komputer pengguna.

e. Agar metoda yang dirancang bisa digunakan oleh *developer*, maka dibangun sebuah *Java API*.

VII. SARAN

Adapun saran terkait pelaksanaan tugas akhir ini adalah :

a. Implementasi rancangan metoda keamanan ini hanya pada *java*, oleh karena itu untuk diimplementasikan pada perangkat lunak sekala besar, diperlukan *porting* ke bahasa pemrograman yang lain.

b. *Java API* yang dibangun pada tugas akhir ini masih perlu beberapa peningkatan fungsionalitas agar semakin mudah untuk digunakan dan tingkat kemananannya semakin meningkat.

c. Perlu dikembangkan fungsi penggenerate kunci yang lebih kompleks dari pada yang digunakan pada tugas akhir ini, agar keamanan dari perangkat lunak yang dilindungi semakin baik.

DAFTAR PUSTAKA

[1]. Altinkerner, Kemal, dan Guan, Junwei(2003). *Analizing Protection Strategis for Online Software Distributin*. Krannert Graduate School of Management, Purdue University.

[2]. Best, Robert M.(1980). *Preventing Sofware Piracy With Crypto-Microprocessor*. Proc. IEEE Spring COMPCON, San Francisco.

[3]. Colvin, David S.(2004). *Method For Securing Software to Decrease Software Piracy*. Z4 Technologis, inc., Commerce Township, MI(US)

[4]. Daemen, Joan dan Rijmen, Vincent(2002). *The Design of Rijndael*. Springer-Verlag Berlin Heidelberg, German.

[5]. Hill, Charles E. (1998).*Software Piracy Ditection System*. Associates, Inc. indianopolis, Ind.

[6]. Munir, Rinaldi. *Kriptografi Visual*(2010). Institut Teknologi Bandung, Bandung.

[7]. Naor, Moni dan Shamir, Adi(1994).*Visual Cryptography*. Depertement of Applied Math and Computer Science, Weizmann Institute, Rehovot.

[8]. Preneel, Bart dkk(1997). *The Cryptographic Hash Function RIPEMD-160*. Katholieke University Leuven, Belgium.

[9]. Zhang, Hiangyu dan Gupta,Rajiv (2003). *Hiding Program Slices for Software Security*. Depertement of Computer Science.The University of Ariz