Proteksi *Content Scramble System* pada Video DVD

Diko Aldillah Patiwiri – NIM : 13503046

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13046@students.if.itb.ac.id

Abstraksi`

Makalah ini membahas tentang studi proteksi *Content Scramble System* (*CSS*) pada DVD video/film. *CSS* merupakan salah satu dari sekian banyak metode yang termasuk dalam *Digital Rights Management* (*DRM*) yang merupakan sebuah sistem proteksi terhadap konten digital, seperti gambar, audio (musik), video (film), dan lain-lain. Pada kebanyakan keping DVD video/film original saat ini telah dilekatkan sebuah mekanisme proteksi *Content Scrambling System* (*CSS*). *CSS* secara tidak langsung juga diimplementasikan pada kebanyakan DVD player dan *software* yang memutar DVD yang berperan dalam proses pembacaan/pemutaran (*playback*) sebuah keping DVD. Hal-hal tersebut dilakukan agar isi/konten berupa video/film dalam DVD tersebut tidak dapat dibajak ataupun diperbanyak oleh pihak-pihak tertentu, atau dengan kata lain konsumen hanya memiliki hak untuk melakukan *playback*, sedangkan isinya tidak menjadi hak milik pribadi. Selain itu, juga terdapat tujuan lain demi kebutuhan komersial di pasar.

Pada *CSS* terdapat proses enkripsi data-data yang akan disimpan pada keping DVD. Teknik Enkripsi yang digunakan dikenal dengan nama *Linear Feedback Shift Register* (*LFSR*). *LFSR* melakukan rekayasa terhadap kumpulan bit-bit stream suatu data, yaitu dengan mengkombinasikannya dengan berbagai macam cara matematis.

Kata Kunci: Kriptografi, *Content Scrambling System*, *Digital Rights Management*, DVD, enkripsi, *Linear Feedback Shift Register* (*LFSR*), *cipher*, kriptanalisis

1. Pendahuluan

Konten digital dapat didefinisikan sebagai objek-objek yang direpresentasikan dalam bentuk digital. Gambar, musik, film bisa direpresentasikan sebagai contoh dari konten digital. Konten digital telah mengalami perkembangan yang sangat pesat pada satu dekade terakhir. Hal ini dikarenakan konten digital mempunyai kelebihan dari segi konten dalam bentuk nyatanya, terutama dari sisi kompaknya ukuran, kemudahan distribusi, dan juga kemudahan dalam penggunaan. Namun hal ini juga membawa dampak buruk karena konten digital sangat mudah dibajak dan disalahgunakan.

Digital Rights Management (*DRM*) lahir untuk mencegah terjadinya dampak buruk terhadap konten digital ini. Lebih jauh, DRM juga berperan untuk memberikan opsi untuk melakukan pengontrolan atas penggunaan *rights* dari sebuah konten digital. Implementasi dari *DRM* tidak lain adalah dengan menggunakan berbagai macam teknik kriptografi.

DVD yang merupakan salah satu bentuk representasi media yang di dalamnya terdapat konten digital, dalam hal ini video/film, telah banyak dipakai di seluruh dunia bahkan telah menggantikan peran media sebelumnya yang lebih dulu dikenal, yaitu VCD yang menggunakan kepingan *compact disk* (CD). Seperti halnya konten digital yang lain, DVD juga memunginkan mengalami pembajakan dari sisi kontennya, dalam hal ini terancam diperbanyak tanpa izin dan dijual demi kebutuhan komersil.

Sebelumnya perusahaan pembuat film tidak mau untuk merilis sebuah film dalam bentuk digital hingga mereka merasa yakin bahwa film mereka aman karena telah mempunyai cara untuk

menghindari pembajakan. Pertama mereka menetapkan bahwa DVD sebagai sebuah media standar digital untuk mendistribusikan film. Kemudian mereka mencari sebuah mekanisme *DRM* berupa sebuah skema enkripsi untuk melindungi film dalam DVD. Pada tahun 1996, dengan bantuan dari banyak anggota *consumer electronics* dan industri komputer (selanjutnya lebih dikenal dengan **DVD Copy Control Association**), akhirnya lahirlah sebuah metode yang dikenal dengan nama *Content Scramble System* (*CSS*).

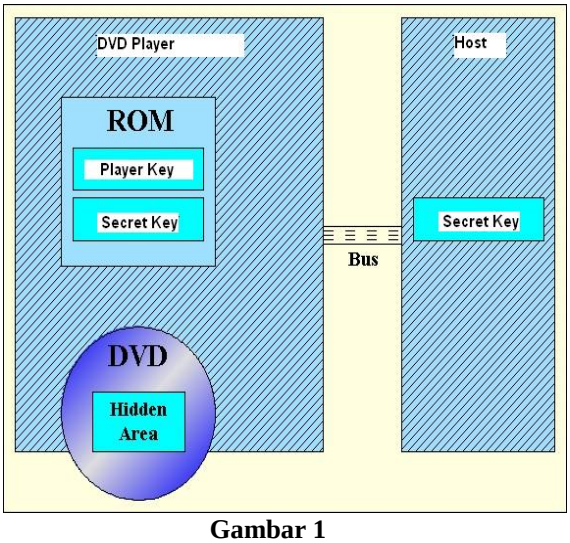
2. CSS

2.1 Definisi

CSS adalah sebuah skema enkripsi yang ditujukan pada DVD agar isi yang terkandung di dalamnya menjadi sesuatu yang tidak dapat dikenali. *CSS* melakukan pengacakan pada konten sebuah DVD hingga menjadi suatu format yang tidak dikenali. *CSS* digunakan untuk memastikan bahwa materi yang terkandung dalam format yang tidak dikenali ini hanya dapat dipakai/dibaca oleh DVD *player* yang telah memiliki lisensi untuk melakukan pemutaran (*playback*).

2.2 Sistem *CSS*

Sistem *CSS* terdiri dari kepingan DVD (selanjutnya disebut dengan DVD), pemutar DVD (DVD *player*), dan perangkat lunak (*software*) yang memutar DVD tersebut, atau dalam hal ini lebih dikenal sebagai *host*.

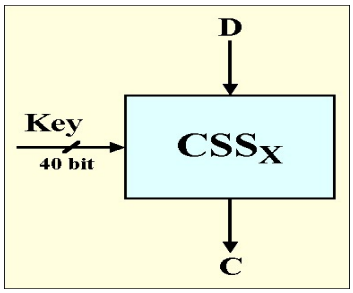


Gambar 1

Di sistem ini juga terdapat :

1. Protokol/transmisi autentikasi pada *bus* dengan menggunakan *bus key*, antara DVD *player* dan *host*. DVD diputar di dalam DVD player, lalu keduanya akan berhubungan langsung dengan *host* melalui sebuah protokol/transmisi yang disebut dengan *bus*. Protokol ini berguna dalam autentikasi DVD dan DVD *player* dengan *host* dengan menggunakan kunci-kunci yang akan dibahas nanti.
2. Enkripsi dengan *CSS*. Persamaan enkripsi secara umum:

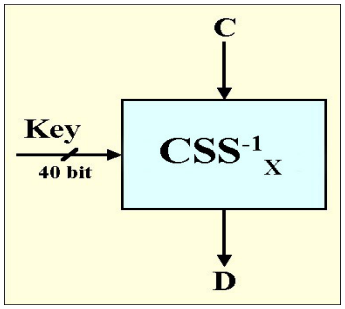
   $C = CSS_x(D, Key)$



Gambar 2

3. Dekripsi dengan *CSS*. Persamaan umum:

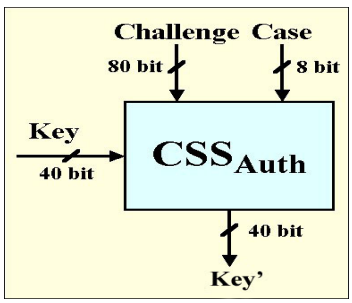
   $D = CSS^{-1}_X(C, Key)$



Gambar 3

4. Proses Autentikasi pada protokol *bus* di atas. Persamaan umum:

   $Key' = CSS_{Auth}(Challenge, Case, Key)$



Gambar 4

Dalam hal ini :

CSS_x = fungsi enkripsi dengan *CSS*
CSS^{-1}_x = fungsi dekripsi dengan *CSS*
CSS_{Auth} = fungsi autentikasi dengan *CSS*
x = jenis mode enkripsi, total ada 3 mode (dijelaskan nanti)
P = plainteks, dalam hal ini data yang terdapat dalam DVD
C = chiperteks, hasil enkripsi
Challenge = konstanta, berukuran 80 bit
Case = konstanta, berukuran 8 bit
Key = kunci yang dipakai, berukuran 40 bit
Key' = kunci keluaran pada autentikasi

Di dalam DVD *player* terdapat beberapa kunci, yaitu *player key* dan *secret key*. Begitu pula di dalam *host*, juga terdapat *secret key*. Mengenai kunci-kunci apa saja yang ada akan dibahas di bagian selanjutnya, yaitu bagian kunci-kunci.

Terdapat 1 fungsi autentikasi dan 3 Jenis Mode Enkripsi atau dekripsi pada *CSS* :

- CSS_A/CSS^{-1}_A, adalah enkripsi/dekripsi pada *disk key*
- CSS_B/CSS^{-1}_B, adalah enkripsi/dekripsi pada *title key*
- CSS_D/CSS^{-1}_D, adalah enkripsi/dekripsi pada *sector* pada DVD
- CSS_{Auth}, adalah fungsi autentikasi antara DVD *player* dan *host*

2.2.1 Struktur DVD

Struktur dalam DVD secara garis besar terdiri dari dua bagian, yaitu:

1. *Filesystem*

Gambar 5

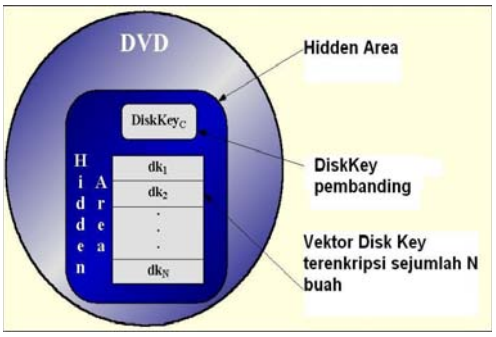
Filesystem adalah organisasi/sistem file-file di dalam DVD. Terdapat tiga jenis file, yaitu video (VIDEO_TS), audio (AUDIO_TS), dan jenis file lainnya. Di dalam VIDEO_TS terdapat tiga jenis format file, yaitu:

a. File ".VOB", yaitu file utama yang berisi data video dan audio yang terkompres dalam format MPEG-2.
b. File ".IFO", yaitu file yang berisi informasi untuk masing-masing file ".VOB".
c. File ".BUP", merupakan file *backup* dari file ".IFO".

Dua jenis file pertama diatas mempunyai format nama awal "VTS_xx_y". (lihat gambar dibawah)

Gambar 6

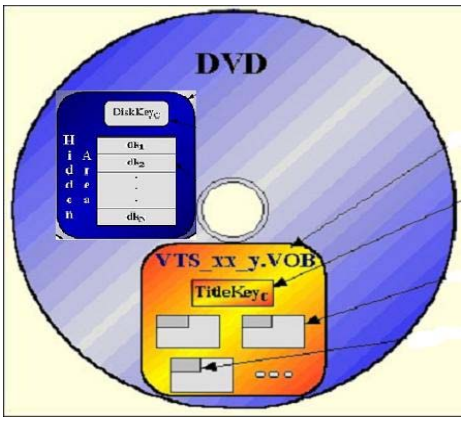
2. *Hidden area*



Gambar 7

Hidden area hanya ada secara logika. Di dalam DVD terdapat sebuah *hidden area* yang di dalamnya terdapat *disk key* yang telah terenkripsi. *Disk key* yang dimaksud terdiri dari kumpulan berupa vektor *disk key* yang berjumlah N buah dan *disk key* pembanding ($DiskKey_c$). Setiap dari *disk key* ini nantinya akan didekripsi oleh *player key* dari DVD *player*. Salah satunya nanti akan menjadi *final disk key*, yang akan dipakai untuk melakukan dekripsi terhadap *title key* yang juga terdapat dalam DVD.

Sehingga struktur DVD dapat digambarkan menjadi seperti dibawah ini (*filesystem* diwakili oleh file ".VOB"):



Gambar 8

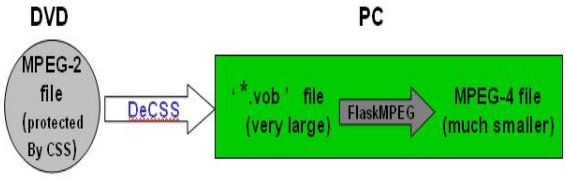
Saat ini, setiap DVD *player* di pasaran telah ditanamkan kode tertentu, dalam hal ini yaitu sebuah kumpulan *player key*. Setiap keping DVD di pasaran juga telah ditanamkan kode-kode tertentu yaitu dengan sebuah *disk key* untuk mengidentifikasi kepingan tersebut. Dekripsi dalam *CSS* terjadi ketika terjadi pemutaran keping DVD pada DVD *player*. DVD *player* harus memiliki *player key*, tanpa kunci ini DVD *player* tidak dapat mengakses isi dari sebuah DVD. Ketika sebuah DVD *player* memulai pembacaan pada sebuah keping DVD, *player* tersebut menggunakan *player key*-nya dan memeriksa daftar *disk key* yang terenkripsi pada kepingan. Dengan *player key* dan *disk key* ini sebuah DVD dapat diputar dan ditonton di televisi atau layar komputer, namun konsumen tidak dimungkinkan untuk mengkopi film tersebut ataupun memanipulasi isi DVD itu.

Perusahaan pembuat film (DVD) membangun kerja sama dengan produsen DVD *player*, yaitu produsen harus terlebih dahulu membeli DVD *license* untuk memproduksi DVD *player* yang legal secara hukum. Perusahaan pembuat film (DVD) memberikan *player key* dan informasi lain yang terdapat pada skema *CSS* yang dipakai. Sebaliknya produsen DVD *player* harus menjaga kerahasiaan *player key*. Produsen ini juga diwajibkan untuk menyertakan kemampuan pada DVD *player*, yaitu mencegah perpidahan data *CSS* dari sebuah *drive* DVD ke *internal recording device* seperti *hard disk* komputer. Total terdapat 409 *player key*, dan setiap produsen mempunyai *player key* yang berbeda.

DVD dapat dimainkan di dalam lingkungan Windows dan Machintosh dengan menggunakan DVD *player* (dapat berupa DVD ROM). Kita tidak dapat memutar DVD di dalam lingkungan Linux. Yaitu dikarenakan sifatnya yang *open source*, maka akan sangat mudah untuk melihat cara kerja sistem *CSS* karena *source code*-nya akan tersebar dan mudah dilihat.

Seseorang dapat menggunakan DVD *writer* untuk mengkopi konten di dalam sebuah keping DVD itu secara langsung, namun itu berarti diperlukan juga sebuah DVD RAM kosong yang sangat mahal dan jauh melebihi harga sebuah harga keping DVD. Di dalam setiap keping DVD, data disimpan sebagai sebuah file MPEG-2 yang ukurannya sangat besar dan sangat tidak mungkin disimpan di dalam sebuah keping CD-ROM standar.

File MPEG-4 sebenarnya mempunyai kualitas yang hampir sama dengan file MPEG-2, dan ukurannya hanya sekitar 10% dari ukuran file MPEG-2. Dikarenakan data di dalam DVD telah diproteksi dengan CSS, kita hanya tinggal mencari cara bagaimana mengkopi data dari kepingan DVD tersebut. Di bawah ini adalah gambaran bagaimana membuat sebuah file MPEG-4.



Gambar 9

Dari gambar di atas, diketahui bahwa kita dapat membuat sebuah file MPEG-4 dengan mudah. Kemudian karena ukuran file MPEG-4 yang kecil, maka akan sangat mudah disebarkan lewat jaringan internet.

2.3 Kunci-kunci

Setelah data-data dalam DVD telah dapat ditransmisikan ke dalam DVD *player*, terdapat beberapa langkah lebih lanjut yang harus dilakukan dalam melakukan enkripsi terhadap data-data tersebut agar bisa dimainkan. Namun sebelum kita membahas tentang proses kerja *CSS* dan enkripsi terhadap data-data di dalam DVD tersebut, terlebih dahulu kita harus mengenal beberapa macam kunci (*keys*) yang digunakan, antara lain:

a. *Region key* : Digunakan untuk memeriksa bahwa sebuah keping DVD yang diproduksi

di suatu negara (*region*) tertentu hanya dapat diputar dengan menggunakan DVD *player* yang juga diproduksi di *region* yang sama. Dalam hal ini, DVD dan DVD *player* mempunyai *region key* yang sama.

b. *Authentication key* : Digunakan untuk melakukan autentikasi terhadap DVD *player* dan DVD, yaitu keduanya harus sudah mempunyai lisensi yang legal. *Bus key* (40 bit) juga termasuk jenis kunci ini, namun hanya dipakai sewaktu proses autentikasi antara DVD *player* dan *host*.

c. *Session key* : Digunakan untuk melakukan enkripsi terhadap data dalam DVD ketika ditransfer dari DVD *player* ke *host*. Hal ini dilakukan untuk menghindari penyadapan.

d. *Player key* : Total terdapat 409 *player keys* yang telah ditetapkan oleh **DVD Copy Control Association** pada setiap pabrikan DVD *player*. Setiap DVD *player* mempunyai *player key* yang berbeda, contoh : **Sony** *player* mempunyai kuncinya sendiri, dan **Panasonic** *player* juga mempunyai *player key*-nya sendiri. Pada dasarnya, *player key* terdiri dari vektor *player key* yang berjumlah M buah.

e. *Disk key* : *Disk key* terletak di dalam DVD dan dipakai untuk mengenkripsi *title key*. Di lain pihak, *player key* melakukan dekripsi terhadap *title key. Player key* akan mencoba semua kemungkinan *disk key* dan melakukan verifikasi dengan menggunakan *disk hash*.

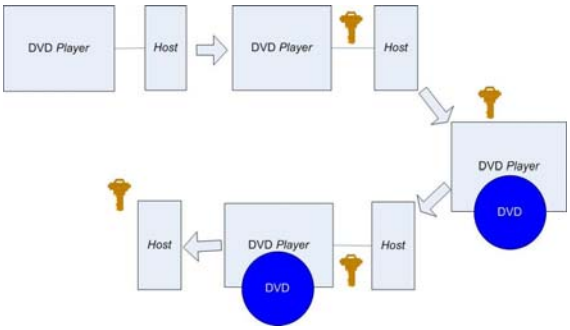
f. *Sector key* : Digunakan dalam operasi **XOR** dengan *title key* dan juga untuk melakukan enkripsi terhadap data. Di dalam tiap *sector's header*, *sector key* disimpan di dalam byte ke 80 hingga 84. Tiap *sector key*

g. *Secret key :* Terletak pada *ROM* dalam DVD, berukuran 5 byte, nantinya dipakai dalam proses autentikasi dengan *secret key* sebagai salah satu parameternya.

h. *Title key* : Digunakan dalam operasi **XOR** dengan *sector key* dan juga untuk melakukan enkripsi terhadap data. Letak *title key* dalam DVD berada dalam *Title* (*Video Title Set*).

2.4 Gambaran Umum Proteksi CSS pada DVD

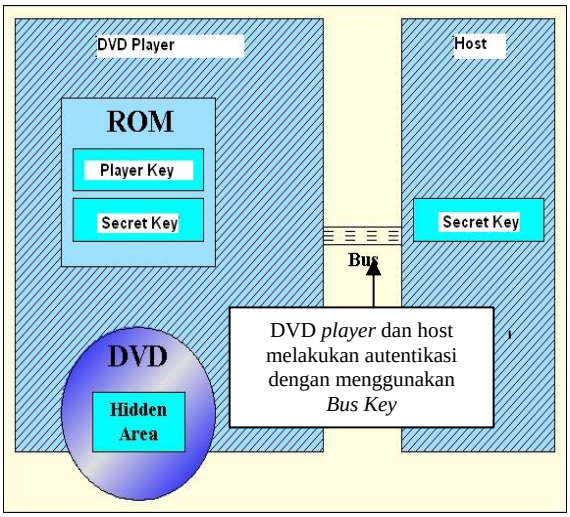
Akan lebih mudah dalam mengerti cara kerja proteksi *CSS* bila kita menjabarkan terlebih dahulu yaitu ketika sebuah DVD akan diputar. Hal ini dikarenakan inti dari proteksi *CSS* itu sendiri terletak sewaktu proses pemutaran sebuah DVD. Dalam hal ini semua elemen-elemen terkait akan terlibat. Sedangkan proses yang lebih detail, akan dibahas pada bagian dekripsi, karena proses pemutaran DVD juga merupakan proses dekripsi.



Gambar 10

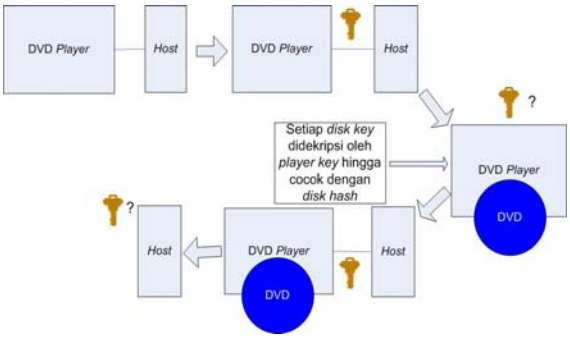
Sewaktu sebuah DVD akan diputar (telah dimasukkan ke dalam DVD *player*), maka terlebih dahulu DVD *player* dan *host* akan melakukan autentikasi satu sama lain. DVD *player* dan *host* terhubung melalui suatu protokol/transmisi yang disebut *bus*, keduanya melakukan autentikasi dengan menggunakan kunci yang namanya sesuai dengan protokol diatas, yaitu *bus* key yang berukuran 40 bit.

Gambar 11



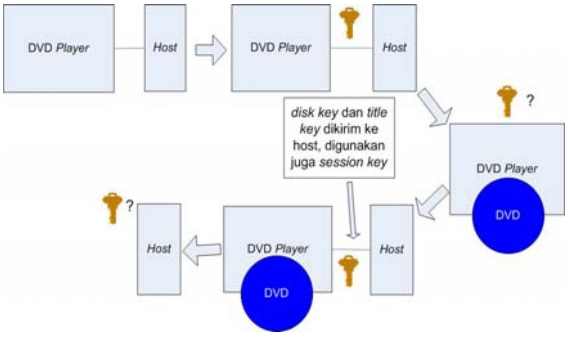
Gambar 12

Setelah proses autentikasi selesai dan telah terjamin bahwa komunikasi diantara keduanya cocok dan aman, proses selanjutnya adalah DVD *player* (dengan menggunakan *player key*) akan melakukan dekripsi terhadap setiap *disk key* yang sebelumnya terenkripsi, hingga salah satu *disk key* terverifikasi dengan *disk hash* sebagai kunci yang benar.



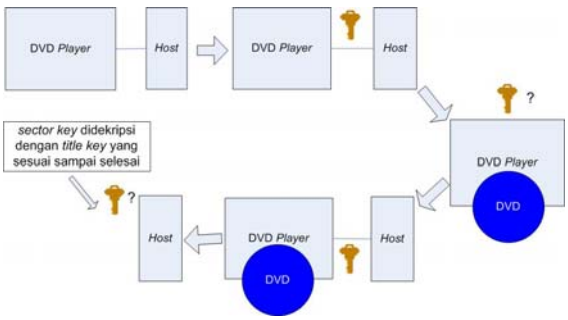
Gambar 13

Pada gambar 8, dapat dilihat bahwa bisa terdapat beberapa *title key* pembanding (*TitleKey*$_c$), karena setiap file ".VOB" mempunyai *TitleKey*$_c$-nya masing-masing. Kemudian *disk key* dan semua *TitleKey*$_c$ yang ada pada DVD akan dikirim ke *host*. Dalam proses pengiriman ini juga digunakan *session key* untuk menghindari adanya serangan ditengah-tengah proses. Selanjutnya tiap *title key* akan dideksripsi dengan menggunakan *disk key*, dan akan ditemukan *final title key* setelah di-**XOR**-kan dengan *bus key*.

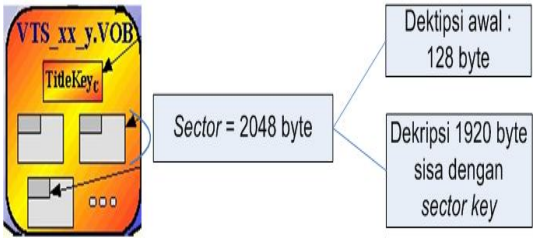


Gambar 14

Tiap file ".VOB" mempunyai sektor-sektor file tertentu dalam logikanya, atau tiap *TitleKey*$_c$ mewakili sejumlah sektor. Tiap sektor dalam tiap file ".VOB" berukuran 2048 byte. Proses berikutnya adalah tiap sektor akan didekripsikan oleh *final title key* yang sesuai dari sektor tersebut. Namun dalam kenyataannya, pada tiap sektor hanya 128 byte atau sektor ke-84 hingga sektor ke-88 saja yang dilakukan proses dekripsi. Sewaktu *title key* telah mendekripsi 128 byte dari tiap sektor, maka akan terbentuk pola berupa *sector key* dari masing-masing sektor. Kemudian proses terakhir adalah tiap *sector key* ini akan digunakan untuk mendekripsikan sisa byte dari tiap sektor, atau sebesar 1920 byte.



Gambar 15



Gambar 16

3. Enkripsi pada Data

Enkripsi merupakan komponen penting yang dimiliki oleh sistem *CSS*. Enkripsi dilakukan pada byte-byte data plainteks yang akan

disimpan dalam DVD. Dalam melakukan enkripsi ini, hanya ada dua kunci yang sangat berperan, yaitu *sector key* dan *title key*. Hal ini dikarenakan karena kedua kunci inilah yang akan menjadi perintis terbentuknya kunci baru sebesar 40 bit yang digunakan dalam melakukan enkripsi terhadap data yang akan disimpan dalam DVD.

Kunci baru tersebut merupakan jenis **aliran-bit kunci** (*keystream*). Aliran-bit-kunci dibangkitkan dari sebuah pembangkit yang dinamakan pembangkit aliran-bit-kunci (*keystream generator*). Aliran-bit-kunci nantinya (sering dinamakan *running key*) akan di-**XOR**-kan dengan aliran bit-bit plainteks, p_1, p_2, …, p_i, untuk menghasilkan aliran bit-bit cipherteks:

$$c_i = p_i \oplus k_i$$

Keamanan data yang akan disimpan dalam DVD sangar bergantung pada pembangkit aliran-bit-kunci. Jika pembangkit mengeluarkan aliran-bit-kunci yang seluruhnya nol, maka cipherteks sama dengan plainteks, dan proses enkripsi menjadi tidak artinya. Oleh karena itu pemilihan *keystream generator* yang dipilih untuk digunakan akan sangat penting dalam menjaga keamanan isi data. *Keystream generator* yang dipakai dalam *CSS* menggunakan teknik *Linear Feedback Shift Register* (*LFSR*).

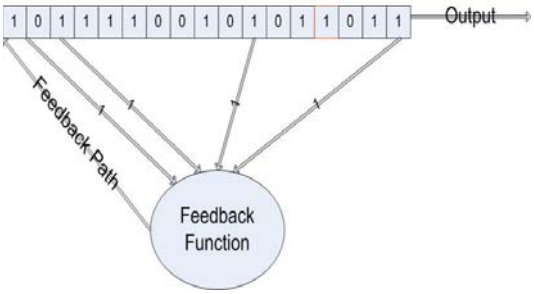
3.1. *LFSR* pada *CSS*

Linear Feedback Shift Register (LFSR) merupakan salah satu jenis *keystream generator* yang berfungsi untuk membangkitkan aliran-bit kunci yang nanti akan dipakai untuk mengenkripsi data dalam DVD. *LFSR* mempunyai dua bagian utama :

a. *Shift register*, merupakan barisan/deretan bit-bit sebanyak n buah. Sebuah bit masuk dari arah paling kiri (*leftmost*) *register*, lalu *register* ini terus bergeser ke arah kanan sebanyak 1 bit tiap pergeserannya dan bit paling kanan akan keluar dari barisan. Pada kebanykan *LFSR*, keluaran ini digunakan lebih lanjut. Asal bit masukan itu berasal dari proses pada *Tap sequence*.

b. *Tap sequence*, bagian ini mengambil beberapa bit dalam *register*, lalu mengopernya ke dalam *Feedback function* (biasanya dengan **XOR**). Lalu keluaran dari *feedback function* sebesar 1 bit menjadi masukan pada *shift register*, yaitu diletakkan di sebelah paling kiri dari *register*.

Gambar dibawah ini menggambarkan bagaimana teknik LFSR pada umumnya bekerja. *Indexing* dilakukan dari kiri ke kanan. Jadi bit ke 1 terletak di sebelah paling kiri *register*, sedangkan bit ke n, terletak di sebelah paling kanan *register*.



Gambar 17

Dengan menggunakan mekanisme *tap sequence* di atas, *LFSR* berputar dengan memiliki sebanyak 2^{n-1} kemungkinan kombinasi semua bit di dalam *register*, yang disebut juga dengan *maximal length LFSR*. Mengapa bukan 2^n? hal ini dikarenakan ada sebuah kondisi dimana semua bit dalam *register* bernilai 0, dan jika hal ini terjadi maka, *feedback function* hanya akan terus menghasilkan bit 0 *keystream*, kondisi ini disebut dengan *null-cycling*. Itulah mengapa kondisi ini tidak dimasukkan ke dalam kemungkinan kombinasi bit.

Dalam *CSS* digunakan dua jenis *LFSR*, yaitu *LFSR*-17 dan *LFSR*-25.

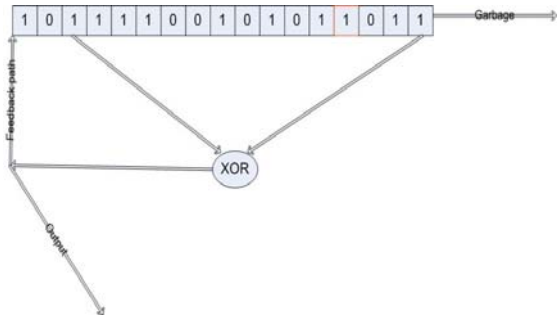
3.1.1. *LFSR*-17

Sesuai dengan namanya, *register* dalam *LFSR*-17 berjumlah 17 bit (2 byte + 1 bit tambahan), Pada *LFSR*-17 dan *LFSR*-25, 1 bit tambahan digunakan untuk menghindari terjadinya *null cycling* seperti yang telah dijelaskan di atas dan selalu bernilai 1. Bit tambahan ini diletakkan pada posisi bit ke 4.

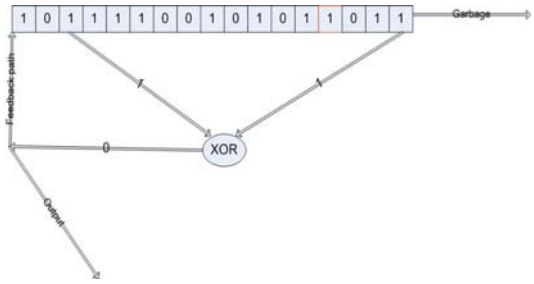
Asal bit di dalam *register LFSR*-17 tersebut berasal dari *title key* dan *sector key* di dalam DVD. Seperti diketahui bahwa *title key* dan *sector key* masing-masing mempunyai panjang 40 bit (5 byte). Asal bit *register* pada *LFSR*-17 didapat dari operasi **XOR** pada 16 bit (2 byte) awal *title key* dan 16 bit (2 byte) awal *sector key* ditambah 1 bit tambahan pada posisi ke 4.

Register LFSR-17 = (16 bit awal *title key* $\oplus$ 16 bit awal *sector key*) + 1 bit tambahan

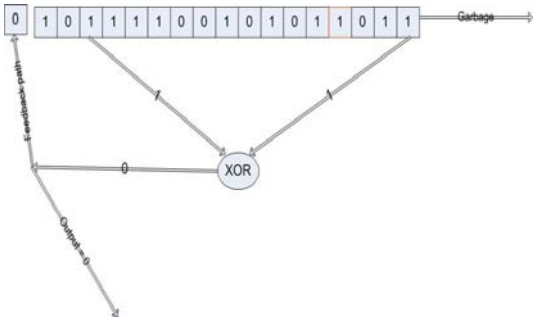
Pada *LFSR*-17, *tap sequence* mengambil bit *register* ke 1 dan 15 sebagai masukan pada *feedback function* pada setiap pergeseran. Di *feedback function* dilakukan operasi **XOR** pada kedua masukan bit ini. Dari operasi ini dihasilkan bit keluaran *LFSR*-17 yang juga dipakai sebagai keluaran yang selanjutnya digunakan sebagai masukan pada *register* dan diletakkan di posisi paling kiri pada *register*, sekaligus menggeser *LFSR* kekanan sebanyak 1 bit. Bit paling kanan yang tergeser akan dibuang ke *garbage*. Gambar-gambar dibawah ini menjelaskan bagaimana *LFSR*-17 bekerja dalam menghasilkan *output*.



Gambar 18



Gambar 19



Gambar 20

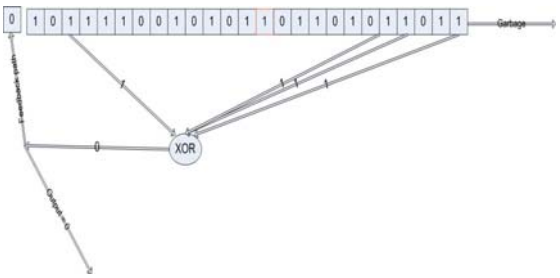
3.1.2. *LFSR*-25

Jumlah bit pada *register* dalam *LFSR*-25 berjumlah 25 bit (3 byte + 1 bit tambahan). 1 bit tambahan juga diletakkan pada posisi bit ke 4 dan juga selalu bernilai 1.

Asal bit *register* pada *LFSR*-25 juga berasal dari *title key* dan *sector key*. Asal bit *register* pada *LFSR*-25 didapat dari operasi **XOR** pada 24 bit sisa *title key* dan 24 bit sisa *sector key* 1 bit tambahan pada posisi ke 4.

Register LFSR-17 = (24 bit terakhir *title key* ⊕ 24 bit terakhir *sector key*) + 1 bit tambahan

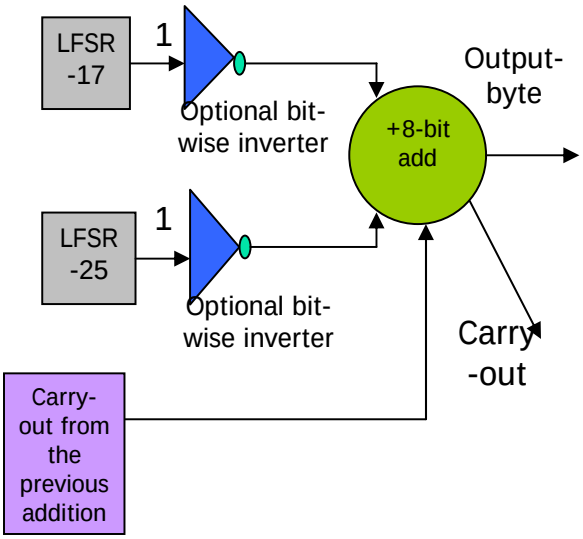
Pada *LFSR*-25, *tap sequence* mengambil bit *register* ke 1, 4, 5, dan 15 sebagai masukan pada *feedback function* pada setiap pergeseran. Di *feedback function* juga dilakukan operasi **XOR** pada keempat masukan bit ini. Dari operasi ini dihasilkan bit keluaran *LFSR*-25 yang juga dipakai sebagai keluaran yang selanjutnya digunakan sebagai masukan pada *register* dan diletakkan di posisi paling kiri pada *register*, sekaligus menggeser *register* kekanan sebanyak 1 bit. Bit paling kanan yang tergeser akan dibuang ke *garbage*. Gambar dibawah ini menjelaskan bagaimana *LFSR*-25 bekerja dalam menghasilkan *output*.



Gambar 21

3.1.3. Pembangkitan *Keystream*

Setelah kedua *LFSR* masing-masing telah menghasilkan 1 byte *output*, pada kedua 1 byte *output* tersebut dikenakan operasi penjumlahan untuk membentuk 1 byte *keystream*. Operasi penambahan ini dilakukan sebanyak 5 kali putaran atau hingga terbentuk *keystream* berukuran 40 bit (5 byte). *Carry out* dari setiap penjumlahan 1 byte/1 putaran digunakan dalam penjumlahan 1 byte selanjutnya hingga terbentuk 5 byte *keystream*.



Gambar 22

```
for (i=0;i<5;i++) {
 for (j=0;j<8;j++) {
  keystream[i*j+1] = !O1[j] + !O2[j] + C0[j];
  C0[j+1] = (!O1[j] + !O2[j] + C0[j]) >> 8
 Endoffor
endofFor
```

Gambar 23

Seperti bisa dilihat pada gambar di atas, juga digunakan *bit wise inverter*, yang mengubah bit 1 ke 0 atau bit 0 ke 1. Namun penggunaan *inverter* ini bersifat opsional. Pada kebanyakan

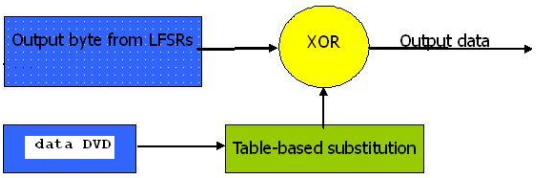
Proses autentikasi antara DVD dan DVD *player*, digunakan 1 *inverter* pada *LFSR*-17 namun tidak pada *LFSR*-25. Pada enkripsi dan dekripsi terhadap kunci-kunci (*session key*, *title key*, dan lainnya), penggunaan *inverter* juga bersifat opsional (lihat tabel).

Mode	LFSR-17	LFSR-25
Autentikasi DVD dan DVD *Player* (CSS_{Auth})	*Yes*	*No*
Encryption/Decryption pada *Session Key* / *disk key* (CSS_A/CSS^{-1}_A)	*No*	*No*
Encryption/Decryption pada *Title key* (CSS_B/CSS^{-1}_B)	*No*	*Yes*
Encryption/Decryption Data / sector di DVD	*Yes*	*No*

Tabel 1. Penggunaan inverter

3.1.4. XOR keystream dan data DVD

Setelah didapatkan *keystream*, langkah selanjutnya adalah mengkombinasikannya dengan bit-bit data dalam DVD dengan menggunakan operasi **XOR**. Sebelum itu dilakukan metode substitusi (menggunakan tabel substitusi) pada data-data dalam DVD, yaitu mengganti bit-bit data dengan bit lain tanpa mengubah urutannya (mengganti 1 karakter dengan karakter yang lain). Metode ini berfungsi untuk menambah rumit algoritma enkripsi yang digunakan. Setelah itu, dilakukan operasi **XOR** terhadap *keystream* dan data-data DVD yang telah disubstitusi tersebut. Hasil dari operasi **XOR** ini adalah *output* data.



Gambar 24

Output data pada gambar diatas inilah yang merupakan hasil enkripsi akhir pada sistem *CSS* dan juga yang akan disimpan secara permanen dalam kepingan DVD.

4. Dekripsi dengan *CSS*

Seperti telah dijelaskan diatas bahwa dekripsi pada DVD dengan *CSS*, tidak lain adalah proses yang berjalan ketika sebuah DVD akan diputar dengan DVD player. Pada bagian sebelumnya telah dijelaskan gambaran secara umum bagaimana proteksi *CSS* bekerja pada pemutaran DVD. Pada bagian ini akan dijelaskan bagaimana dekripsi dengan *CSS* itu berlangsung. Juga akan dijelaskan lebih detail hubungan antara ketiga komponen *CSS*, yaitu dengan memakai fungsi dekripsi dan autentikasi dengan *CSS*.

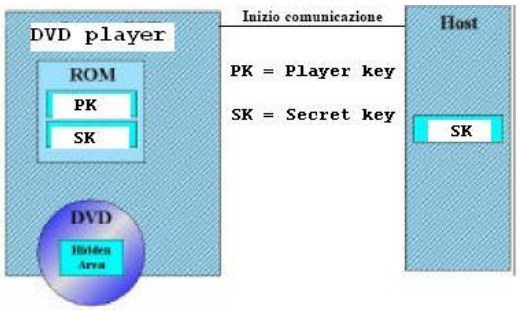
Dekripsi di sini terdiri dari empat proses yang berjalan secara bertahap, yaitu :

a. *Mutual autentication*

Autentikasi yang dimaksud disini adalah autentikasi antara DVD *player* dan *host* dengan menggunakan sebuah protokol *bus*. Pada tahap ini akan dihasilkan *bus key*, namun bila *bus key* tidak dihasilkan, itu

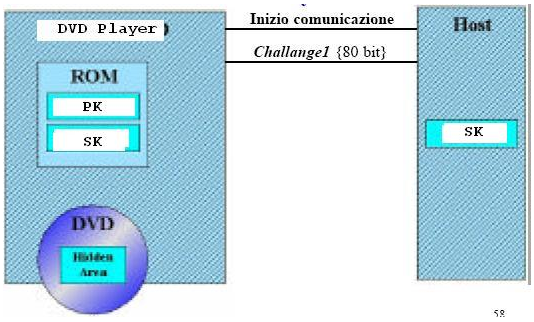
berarti autentikasi antara DVD *player* dan *host* gagal. Tahap-tahapnya :

- DVD *player* me-*request AGID* (Autentication Grant ID) dari *host*, yang berfungsi untuk mengidentifikasi *session*.



Gambar 25

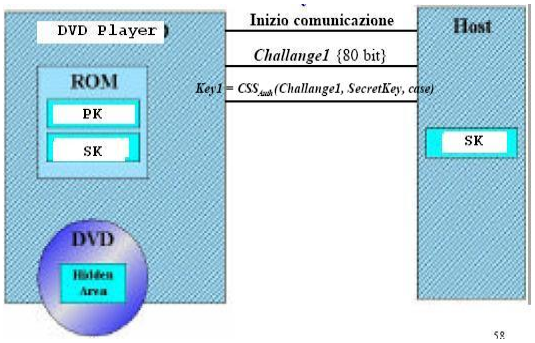
- DVD *player* menerima konstanta *challenge1* sebesar 80 bit dari *host.*



Gambar 26

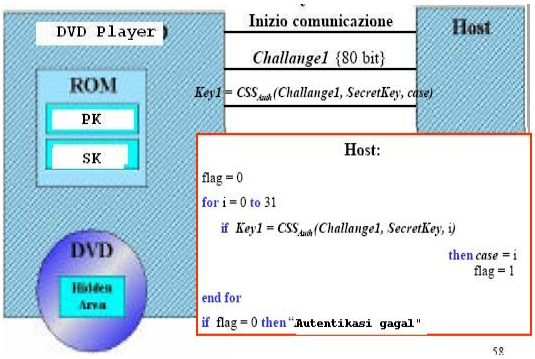
- Dengan konstanta *case* (0 hingga 31) dan *secret key* yang dimiliki DVD *player* dan *chellenge1* dilakukan fungsi autentikasi *CSS* untuk menghasilkan *Key1* sebesar 40 bit.

  $Key_1 = CSS_{Auth}(Challenge1, Case, Key)$



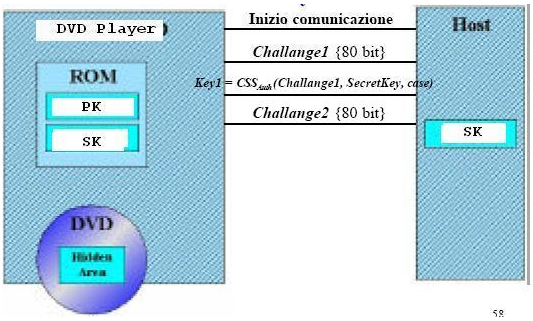
Gambar 27

- Selanjutnya Key_1 dikirim ke *host.* Dengan Key_1 ini, *host* akan melakukan pengecekan apakah Key_1 tersebut cocok dengan Key_1 ‘ yang dihasilkan di *host* dengan melakukan fungsi autentikasi *CSS* dengan menggunakan *chellenge1*, *secret key* dan *case* awal (*case* = 0) yang dimiliki *host.* Bila Key_1 dan Key_1 ‘ cocok, maka proses akan berlanjut dan nilai konstanta *case* pada *host* akan bertambah 1, namun bila gagal berarti autentikasi gagal dan proses berhenti.



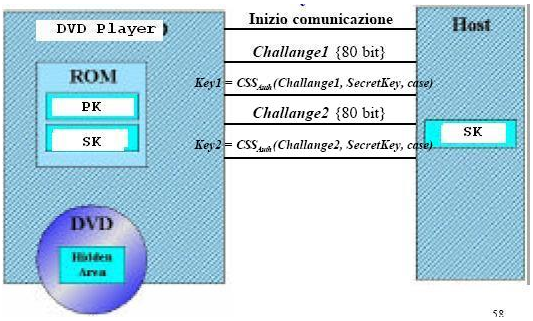
Gambar 28

- Diasumsikan proses berlanjut. Selanjutnya giliran DVD *player* yang mengirimkan konstanta *challenge2*-nya sebesar 80 bit kepada *host.*



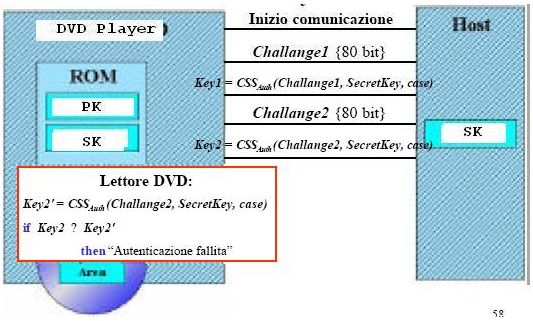
Gambar 29

- Lalu pada *host* akan dilakukan fungsi autentikasi *CSS*, dengan menggunakan *challenge2*, *secret key*, *dan case* yang telah bertambah 1 (*secret key* dan *case* disini milik *host*) sebagai parameter fungsi dan akan dihasilkan Key_2 (40 bit). Selanjutnya Key_2 ini dikirim ke DVD *player*.



Gambar 30

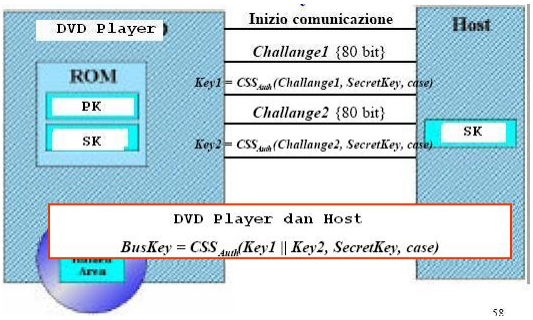
- Sebelumnya pada DVD *player* juga dilakukan fungsi autentikasi *CSS* yang menghasilkan Key_2' (40 bit) dengan *challenge2*, *secret key*, dan *case* sebagai parameternya. Lalu DVD *player*, Key_2 akan dibandingkan dengan Key_2' tadi. Bila tidak sama maka autentikasi gagal dan proses berhenti. Namun bila sama, maka tahap selanjutnya akan berjalan.



Gambar 31

- Tahap terakhir pada proses ini adalah dihasilkannya *bus key* yang sama pada DVD *player* dan *host*. *Bus key* dihasilkan dengan operasi :

  $BusKey = CSS_{Auth}(Key_1 \parallel Key_2, SecretKey, case)$



Gambar 32

b. Menemukan *final Disk Key* yang sesuai

Pada proses ini akan di cari *final disk key* yang sesuai. Seperti yang telah disebutkan diatas, bahwa *disk key* pada DVD terdiri dari kumpulan berupa vektor *disk key* yang berjumlah N buah (dk_n) dan sebuah *disk key* pembanding ($DiskKey_c$). Begitu juga *player key* berupa vektor *player key* yang berjumlah M buah.

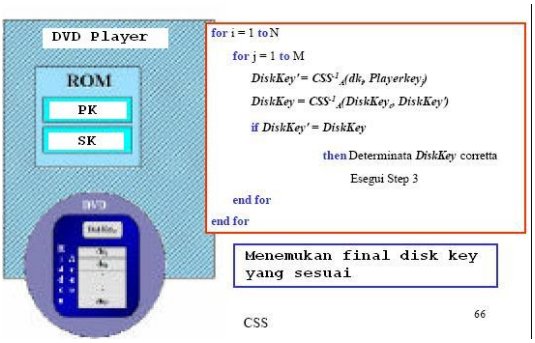
Pada proses ini terdapat *nested looping*. *Looping* sebelah luar sebesar N buah (*disk key*) dan *looping* bagian dalam sebesar M buah (*player key*). Pada *looping* awal dilakukan fungsi dekripsi *CSS* yang akan menghasilkan *DiskKey'* (sementara). Pada dekripsi ini dilibatkan anggota vektor *player key* ke 1 dan anggota vektor *disk key* ke 1 (dk_1) sebagai parameter fungsinya.

$$DiskKey' = CSS^{-1}_A(dk_i, Playerkey_j)$$

Dengan menggunakan *disk key* pembanding ($DiskKey_c$) dan *DiskKey'* sebagai parameter, dilakukan fungsi dekripsi lagi untuk menghasilkan *DiskKey*.

$$DiskKey = CSS^{-1}_A(DiskKey_c, DiskKey')$$

Selanjutnya *DiskKey'* dan *DiskKey* dibandingkan, bila sama maka *final disk key* adalah *DiskKey*. Namun bila tidak sama, maka lakukan *looping* selanjutnya dengan menggunakan *player key* ke 2 dan selanjutnya (*looping* dalam) hingga yang ke m. Jika masih belum menemukan, maka akan dilakukan *looping* selanjutnya dengan menggunakan dk_2 dan selanjutnya hingga yang ke n (*looping* luat) dan ditemukan *final DiskKey* yang sesuai.



Gambar 33

c. Menentukan *TitleKey'*

TitleKey' adalah *title key* sementara yang dihasilkan di DVD *player* dan nanti akan dikirim ke *host*. Untuk setiap file ".VOB", memiliki $TitleKey_c$-nya masing-masing.

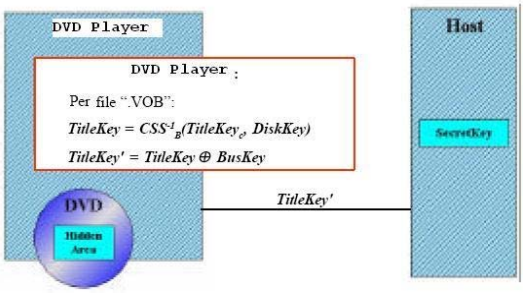
Pada tiap file ini akan dilakukan fungsi dekripsi untuk mencari *TitleKey*. Fungsi ini menggunakan *final DiskKey* yang sudah ditemukan pada proses sebelumnya dan $TitleKey_c$ sebagai parameter fungsi.

$TitleKey = CSS^{-1}{}_B(TitleKey_C, DiskKey)$

Selanjutnya *TitleKey* akan di-**XOR**-kan dengan *bus key* milik DVD *player* yang sudah ditemukan pada proses a, hasil dari ini adalah *TitleKey'*.

$TitleKey' = TitleKey \oplus BusKey$

Selanjutnya *TitleKey'* ini akan dikirim ke *host*.



Gambar 34

d. Menentukan *sector key*

Proses ini terjadi pada *host*. Dengan menggunakan *TitleKey'* yang sebelumnya dikirim oleh DVD *player*, akan di-**XOR**-kan dengan *bus key* milik *host* yang sudah ditemukan pada proses a, hasil dari **XOR** ini adalah *TitleKey*.

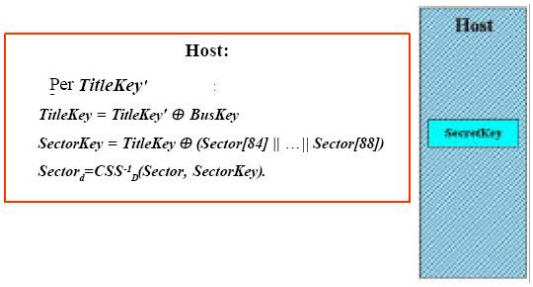
$TitleKey = TitleKey' \oplus BusKey$

Selanjutnya *TitleKey* ini akan di-**XOR**-kan lagi dengan sejumlah sektor, yaitu sektor ke-84 hingga 88, dan dihasilkan *SectorKey*.

$SectorKey = TitleKey \oplus (Sector[84] \,||\, Sector[85] \,||\, Sector[86] \,||\, Sector[87] \,||\, Sector[88])$

Selanjutnya akan dilakukan fungsi dekripsi *CSS* untuk menentukan byte tiap sektor yang besisa pada tiap file ".VOB". Dekripsi ini menggunakan sektor-sektor sisa (belum terdekripsi) dan *SectorKey* sebagai parameter fungsi. Fungsi ini akan menghasilkan tiap byte sektor dari tiap file ".VOB" akan terdekripsi semua.

$Sector_d = CSS^{-1}{}_D(Sector, SectorKey)$



Gambar 35

Gambaran keseluruhan proses Dekripsi dapat dilihat pada gambar 36

Setelah semua sektor pada tiap file".VOB" terdekripsi, maka film siap untuk dimainkan.

4. Kriptanalis *CSS*

Terdapat beberapa pendekatan kriptanalis yang dapat digunakan pada CSS.

a. *Brute force attack* pada *disk key*

Karena *CSS* hanya menggunakan kunci yang berjumlah 40 bit, maka dimungkinkan melakukan pencarian *disk key* berjumlah sebanyak 2^{40} kemungkinan *disk key* dengan menggunakan serangan *brute force*. Kompleksitas dengan memakai serangan ini adalah 2^{25}, yaitu dengan menyerang fungsi *hash* pencari *final disk* key ($CSS^{-1}{}_A$).
$DiskKey = DeCSS_A(DiskKey_c, CombKey)$

Jika kemungkinan kunci *CombKey* sama dengan *DiskKey* yang ada, maka *disk key* yang dimaksud adalah *DiskKey*.

b. Serangan dengan 6 byte *LFSR output*

Jika kriptanalis memiki 6 byte dari keluaran *LFSR*, maka dimungkinkan dicari *input* (dalam hal ini kunci yang digunakan) dari *LFSR*. Namun cara ini sangat jarang berhasil, karena sangat jarang pula dapat diketahui dengan mudah 6 byte *LFSR* tersebut. Total serangan adalah 2^{16} serangan untuk menemukan *input* dari *LFSR* yang juga berarti kunci yang digunakan. Berikut adalah proses serangan:

- Untuk setiap *conten/input* pada *LFSR*
  - Ambil 4 dari 6 byte tersebut
  - Dapatkan *output LFSR*-25 dengan mengurangi 4 byte tersebut dari kombinasi *output*

- Coba semua kemungkinan *content LFSR*-25 dengan menggunakan *output LFSR*-25 tersebut
- Ambil 2 byte sisa
- Jika benar, maka didapat *initial state LFSR* dan juga berarti kunci yang digunakan
- Jika salah, coba lagi dengan kemungkinan *output LFSR* yang lain

c. Serangan dengan 5 byte *LFSR output*

Serangan ini lebih mungkin berhasil karena juga digunakan metode *CSS mangling* dalam menemukan 5 byte *output LFSR*.

- Untuk setiap *content*/*input* pada *LFSR*
  - Ambil 3 dari 5 byte tersebut
  - Temukan byte-byte yang berkorespondensi pada *LFSR*-25 dengan mengurangi *output LFSR*-17 dari banyaknya kemungkinan kombinasi *output*. Didapatkan semua byte-byte tersebut kecuali urutan bit yang paling tinggi pada *LFSR*-25.

d. *CSS Mangling*

Dengan menggunakan metode ini, dapat dicari 5 byte *output LFSR* untuk selanjutnya digunakan pada serangan diatas.

- Untuk setiap *byte*
  - Untuk setiap tebakan
  - Lakukan *backward* terlebih dahulu dengan melakukan enkripsi
  - Lakukan verifikasi *input* yang ingin dicari dengan *input* yang sudah ada

4.5. DeCSS

Pada September 1999, **Jon Johanson**, warga negara Norwegia berusia 16 tahun, dan seorang *hacker* asal Jerman dari kelompok ilmiah **MoRE** (Master of Reverse Engineering) membuat sebuah program yang memungkinkan seseorang untuk menyaksikan DVD dengan menggunakan **Linux** sebagai sistem operasinya. Program ini kemudian diberi nama *DeCSS* (*Descramble system*). Program kecil ini sudah tersebar luas di internet terutama oleh kalangan *open source* (Linux).

Cara kerja dari *DeCSS* ini sebenarnya sangat mirip dengan cara kerja sebuah *software* DVD *player* (*host*). *DeCSS* menggunakan sebuah *player key* untuk menormalkan kembali konten dari DVD yang sudah teracak-acak. Dengan kunci tersebut *DeCSS* membuat file-file MPEG-2 yang dapat dimainkan. Kunci itu dikenal dengan *Xing player key*. Seperti telah diketahui bahwa setiap produsen DVD *player* harus membeli lisensi *CSS* berupa sebuah *player key* untuk memproduksi DVD *player*. *Xing player key* ditemukan oleh *hacker* Jerman tersebut dengan memanfaatkan kecerobohan produsen DVD *player* tersebut. Disamping itu sebenarnya *CSS* sangat mudah dipecahkan, karena kunci yang digunakan hanya 40 bit. Hal ini berarti semua kemungkinan kuncinya hanya 2^{40}. Sebuah serangan *brute force* akan sangat mudah menmukan kunci walaupun *CSS* dibuat dengan menggunakan algoritma *scrambling* yang baik.

Namun umur dari *DeCSS* ini tidak bertahan lama, karena *Xing player key* sudah ditarik kembali dari pasaran. Hal ini mengakibatkan tidak akan ada lagi DVD film terbaru yang dapat di-*descramble* oleh kunci ini, karena *DeCSS* tidak akan bekerja pada DVD film terbaru ini.

5. Teknologi Proteksi Lain

Disamping *CSS* terdapat beberapa teknologi proteksi DVD yang lain, antara lain:

a. *CGMS*

CGMS (*Copy Generation Management System*) merupakan sebuah sistem pengaturan yang bekerja pada level proses pengkopian sebuah DVD. Informasi pada CGMS terintegrasi dengan sinyal video dari DVD *player*, dan bagian *recording* pada *CGMS* harus menanggapi sinyal ini.

Sebuah standar digital seperti **IEEE 1394/Firewire** akan diikut sertakan dalam koneksi digital. Tujuan dari *CGMS* adalah untuk mencegah proses pengkopian, dengan menggunakan bantuan indikator (*flags*) dan menghentikan pengkopian *master support* secara massal.

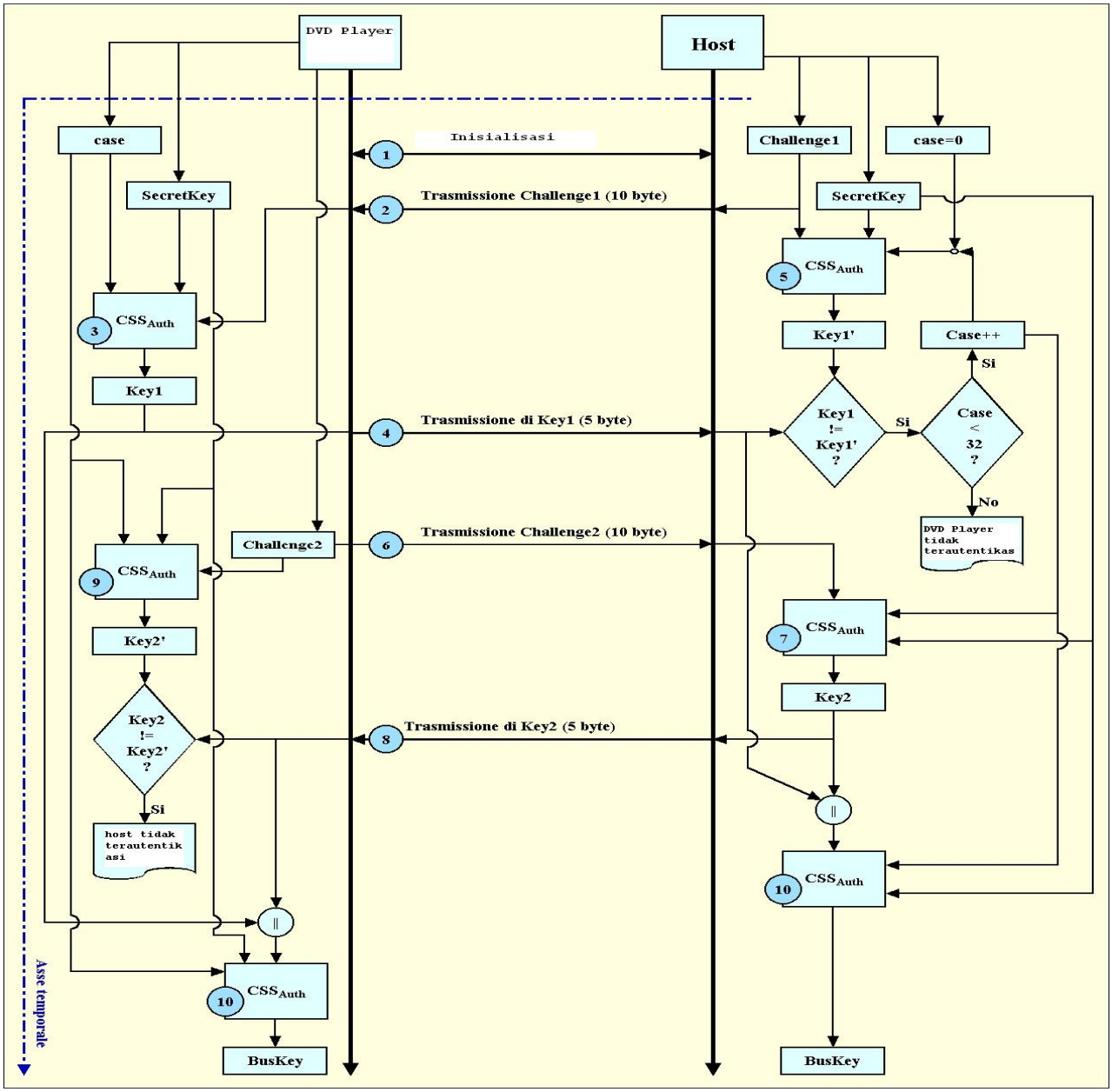
b. APS

APS (*Analog Protection System*) dikembangkan oleh Macrovision. *APS* telah terintegrasi dengan setiap DVD *player* untuk mencegah pengkopian VHS secara analog. Kartu video komputer juga menggunakan sistem ini.

c. Pendekatan lain

Sekarang ini, *Consumer-Electronics and Movie Industrie* telah membentuk beberapa *working group* yang berfungsi untuk menemukan metode-metode lain untuk melindungi *copyright*.

Contohnya, *Copy Protection Technical Working Group (CPTWG)*, yang dibuat oleh Hitachi, Intel, Matsushita (MEI), Sony and Toshiba. Contoh lain adalah 4C yang dibuat oleh *Advanced Access Content System Listening Administrator (AACS LA)* dengan Microsoft, Sony, Walt Disney and Warner Bros.



Gambar 36

DAFTAR PUSTAKA

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[2] De Santis, Alfredo (2002). *http://www.dia.unisa.it/~ads/*. Tanggal akses : 5 September 2006.