BAB 2

LANDASAN TEORI

2.1 Deskripsi Teori

2.1.1 Plagiarisme

Plagiarisme adalah suatu tindakan untuk mengambil ide, tulisan, gambar, kata, atau kepemilikan intelektual yang serupa dari orang lain dan mempublikasikan selayaknya adalah hak milik sendiri tanpa seijin dari yang empunya. Plagiarisme berbeda dengan *copyright* yang berbasiskan hukum, plagiarisme merupakan pelanggaran etika. Sebagai contoh, kita dapat menggunakan ulang suatu karya tulisan kepada khalayak umum tanpa merisaukan tentang masalah pelanggaran legalitas karya tersebut. Tetapi jika mempublikasikan karya tersebut selayaknya adalah hak milik sendiri tanpa mencantumkan nama dari penulis / pembuat karya yang bersangkutan maka tindakan tersebut merupakan plagiarisme, dan plagiarisme selalu melanggar etika.

Self-plagiarism terjadi ketika kita menggunakan ulang karya kita sendiri dari suatu publikasi sebelumnya tanpa mencantumkan referensi karya lama ke suatu publikasi yang baru. Hal ini memang tidak terlihat layaknya plagiarisme pada umumnya karena kedua karya yang baru dan yang lama merupakan hasil karya kita sendiri. Namun jika kita menggunakan ulang sejumlah tulisan yang sangat penting, sebaiknya kita mencantumkan referensi dari publikasi yang sebelumnya.

Terdapat pula jenis plagiarisme lainnya, yakni *accidental plagiarism*. Sebagai contoh, seorang ibu membacakan sebuah cerita kepada anaknya yang masih kecil secara berkala. Saat dewasa, ia membuat tugas atau laporan berdasarkan cerita ibunya tersebut.

Jika beberapa orang secara independen menuangkan ide dan analisis yang sama maka hal ini bukan dikatakan sebagai plagiat.

2.1.2 Sekuriti

Sekuriti merupakan faktor yang sangat penting dalam menyembunyikan *source code* suatu program dan mengirimkan data pada jaringan internet. Serangan yang dilakukan untuk membongkar dan melihat isi dari *source code* mempunyai banyak bentuk misalnya meng-decompile file, memodifikasi *source code*, dan lainnya. Vulnerabilitas yaitu kelemahan di dalam sistem sekuriti yang dapat dieksploitasi sehingga menyebabkan kehilangan atau kerugian. Ancaman terhadap jaringan internet dapat disebabkan oleh manusia, bencana alam, kesalahan manusia, dan kerusakan perangkat keras atau perangkat lunak.

Ada empat macam ancaman dari sistem komputasi menurut **Charles P. Pfleeger (1997, p22)** yaitu :

- Interupsi yaitu suatu cara yang dapat menyebabkan suatu asset penting dapat menjadi hilang, tidak dapat digunakan, atau tidak berguna sama sekali.
- Intersepsi yaitu suatu cara dimana beberapa pihak yang tidak berhak dapat memperoleh hak untuk mempunyai akses ke sebuah informasi.
- Modifikasi yaitu suatu cara dimana pihak luar itu tidak hanya dapat memperoleh hak akses, tetapi dapat merusak dengan cara mengubah pesan atau informasi tersebut.
- Fabrikasi atau produktif pemalsuan yaitu salah satu faktor ancaman yang cukup membahayakan dan pihak luar yang tidak berhak dapat memalsukan sebuah informasi atau program yang akan digunakan.

Menurut **S. Garfinkel (2002,p1)**, karakteristik sekuriti komputer dikatakan aman berkaitan dengan tiga tujuan berikut ini yaitu :

- Kerahasiaan, berarti sebuah asset dari sistem komputer yang dapat diakses hanya oleh pihak yang berhak atau legal.
- Integritas, berarti sebuah asset hanya bisa dimodifikasi oleh pihak-pihak yang berhak atau dengan cara yang legal.
- Ketersediaan, berarti sebuah asset dapat diakses oleh orang yang berhak saja dimana tidak dibatasi hanya mengakses suatu bagian saja.

2.1.3 SWF (*Small Web Format*)

Format *Small Web Format* (SWF) yang dirancang khusus oleh perusahaan Macromedia. Inc ini ditujukan untuk mengantarkan animasi interaktif dan gambar berbentuk vector melalui jaringan internet dengan menggunakan Macromedia *Flash Player*-nya. Format SWF dirancang sebagai format masa depan animasi web dan aplikasi multimedia dengan menghadirkan keunggulannya dalam membuat animasi kompleks dan interaktifitas yang tinggi terhadap pengguna jaringan internet dengan ukuran file yang kecil. Selain itu, keunggulan lain yang ditonjolkan adalah sebagai berikut:

a. On-Screen Display

Tujuan format SWF adalah untuk *on-screen display* dan mendukung *anti-aliasing*, kecepatan *render* yang tinggi terhadap file gambar dari berbagai bentuk format, animasi, dan tombol-tombol interaktif.

b. Extensibility

Dapat membaca format SWF pada versi terdahulu tanpa mengurangi kualitas yang ada.

c. *Network Delivery*

Dapat dikirimkan melalui jaringan internet di seluruh dunia. Format SWF merupakan hasil kompresi dan mendukung *incremental rendering* melalui *streaming*. Format SWF merupakan sebuah format biner dan tidak dapat dibaca oleh manusia layaknya file HTML (*Hyper Text Markup Language*). Format SWF menggunakan teknik *bit-packing* untuk mereduksi ukuran file.

d. *Simplicity*

Format SWF sangat sederhana sehingga mudah digunakan secara luas.

e. *File Independence*

Menampilkan isi file tanpa ketergantungan terhadap sumber luar seperti *font*.

f. *Scalability*

Dapat digunakan pada berbagai resolusi monitor dengan ketajaman file yang sama. Namun hal ini juga tergantung pada *hardware* pengguna.

g. *Speed*

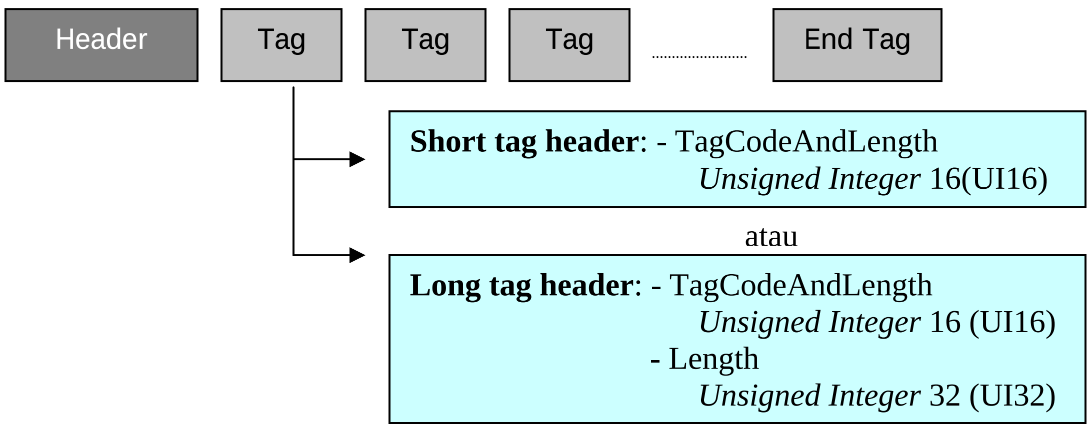
File di-render pada kualitas yang tinggi dengan kecepatan yang tinggi.

h. *Scriptability*

Format ini mengandung *tag-tag* yang menjalankan kode *byte* secara berurutan yang dapat diinterpretasikan oleh mesin komputer. Kode *byte* mendukung bahasa pemrograman *ActionScript*. Bahasa pemograman *ActionScript* merupakan kontrol utama bagi file SWF jika terdapat interaktifitas di dalamnya. Pengaturan dan kontrol bagaimana suatu objek dijalankan atau berinteraksi dengan objek lain dilakukan oleh bahasa ini. Untuk itulah peranan bahasa ini sangat penting dan vital bagi file SWF.

2.1.4 Struktur File SWF

Struktur file SWF terdiri dari rangkaian *tag-tag* blok data. Banyak program dapat memodifikasi terhadap *tag-tag* blok data tersebut dengan menghilangkan, menyisipkan, ataupun mengubah struktur file SWF.



Gambar 2.1 Struktur File SWF

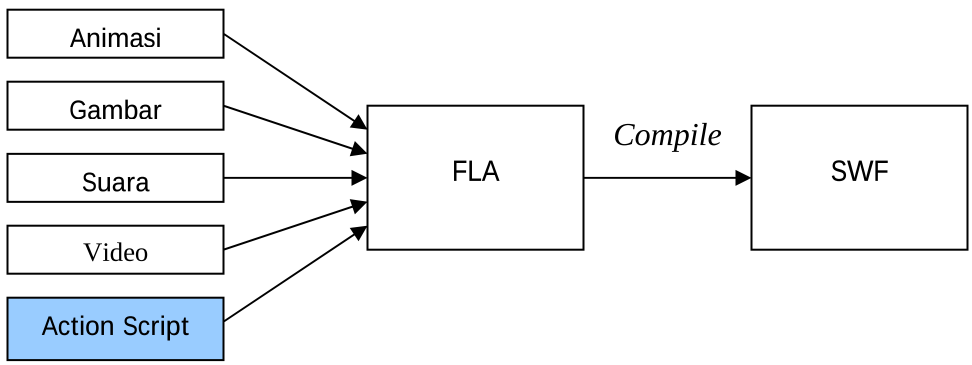
Setiap *tag* dimulai dengan *tag type* dan *tag length*. Terdapat dua format *tag header*, *short tag header* dan *long tag header*. *Short tag header* digunakan untuk data yang berukuran 62 bytes atau kurang. *Long tag header* dapat digunakan oleh berbagai tipe *tag* dengan ukuran 63 bytes hingga 4GB.

Terdapat dua kategori *tag* yang tersedia di dalam struktur file SWF, yakni *definition tags* dan *control tags*. *Definition tags* merupakan *tag* yang mendefinisikan objek dari file SWF seperti bentuk, teks, gambar, suara, dll. Setiap *definition tag* memiliki ID unik, disebut sebagai *character ID*, yang menunjuk ke objek yang didefinisikan. Flash Player kemudian menyimpan *character* di tempat penampung yang disebut *dictionary*. *Definition tags* tidak memberikan perintah untuk me-*render* objek.

Control tags berfungsi sebagai inisialisasi dan memanipulasi *rendered instance* dari *character* yang ada di dalam *dictionary*, serta mengatur alur berjalannya file SWF.

2.1.5 Proses Pembentukan File SWF

File SWF merupakan hasil kompilasi dari file yang berekstensi .FLA. Format FLA adalah file asli yang menyimpan keseluruhan informasi mengenai segala sesuatu yang kita buat pada Macromedia Flash.



Gambar 2.2 Proses Pembentukan File SWF

Berbagai format *multimedia* yang telah terintegrasi di dalam format FLA akan di-*compile* menjadi file berformat SWF. Format SWF inilah yang kemudian di-*publish* ke dalam jaringan internet untuk berbagai kegunaan tersendiri. File SWF mempunyai ukuran file yang kecil dikarenakan di-*compile* dengan menggunakan metode kompresi *Zlib*.

2.1.6 Algoritma Enkripsi RSA

Algoritma enkripsi RSA adalah sebuah *public-key cryptosystem* yang dikembangkan oleh para profesor MIT: **Ronald L. Rivest, Adi Shamir**, dan **Leonard M. Adleman** pada tahun 1977 dalam rangka menjaga sekuritas pada jaringan internet. **Steve Burrnett** dari RSA Data Security, Inc. menjelaskan, *cryptosystem* adalah algoritma sederhana yang dapat mengubah data inputan menjadi sesuatu yang tidak dapat dikenal lagi (enkripsi) dan dapat mengembalikan data tersebut ke bentuk aslinya (dekripsi).

Untuk dapat mengenkrip suatu data, masukkan data tersebut (*plaintext*) beserta kata kunci enkripsi ke dalam algoritma enkripsi. Untuk mendekrip kembali, dibutuhkan sebuah kunci dekripsi yang sesuai. Kunci enkripsi disebut juga sebagai *public key*, sedangkan kunci dekripsi disebut juga sebagai *secret key*. Nilai *public key* maupun *secret key* didapat dengan perhitungan *key generation* sebelum kita mulai mengenkrip data.

Berikut adalah langkah-langkah untuk mengenkripsi pesan:

1. *Key Generation*

a. Inisialisasikan dua angka bilangan prima yang besar, bilangan pertama sebagai p dan bilangan kedua sebagai q.

b. Hitung nilai $n = pq$.

c. Hitung nilai $m = (p-1)(q-1)$.

d. Pilih angka terkecil e, *coprime* berdasarkan nilai m.

e. Cari nilai $d = (1 + nm)/e$ dengan nilai n dimulai dari nol hingga didapat nilai $(1 + nm)/e = 1$.

Dari perhitungan diatas,nilai *e* dan *n* adalah sebagai *public key*. Sedangkan nilai *d* dan *n* sebagai *secret key*.

2. *Encryption*

$$C = P^{e} \% n$$

3. *Decryption*

$$P = C^{d} \% n$$

2.1.7 Reverse Engineering

Reverse engineering merupakan implementasi teknik untuk memilah objek dengan tujuan untuk mengetahui dan mempelajari bagaimana objek tersebut bekerja sehingga seseorang dapat belajar dan mengembangkan pengetahuan untuk menciptakan sesuatu yang lebih baik. *Reverse engineering* secara umum tidak dipandang sebagai sesuatu yang illegal atau tak beretika selama tujuan penggunaannya adalah seperti yang telah disebutkan diatas. Yang tergolong penggunaan *reverse engineering* secara ilegal dan tak beretika ini adalah dengan membuat tiruan secara langsung dari produk pesaing dengan hanya mengganti sedikit, bahkan tanpa adanya perubahan atau peningkatan yang dilakukan tanpa mengetahui bagaimana cara produk tersebut dibuat dan bekerja.

2.1.8 Konsep Dasar Rekayasa Piranti Lunak

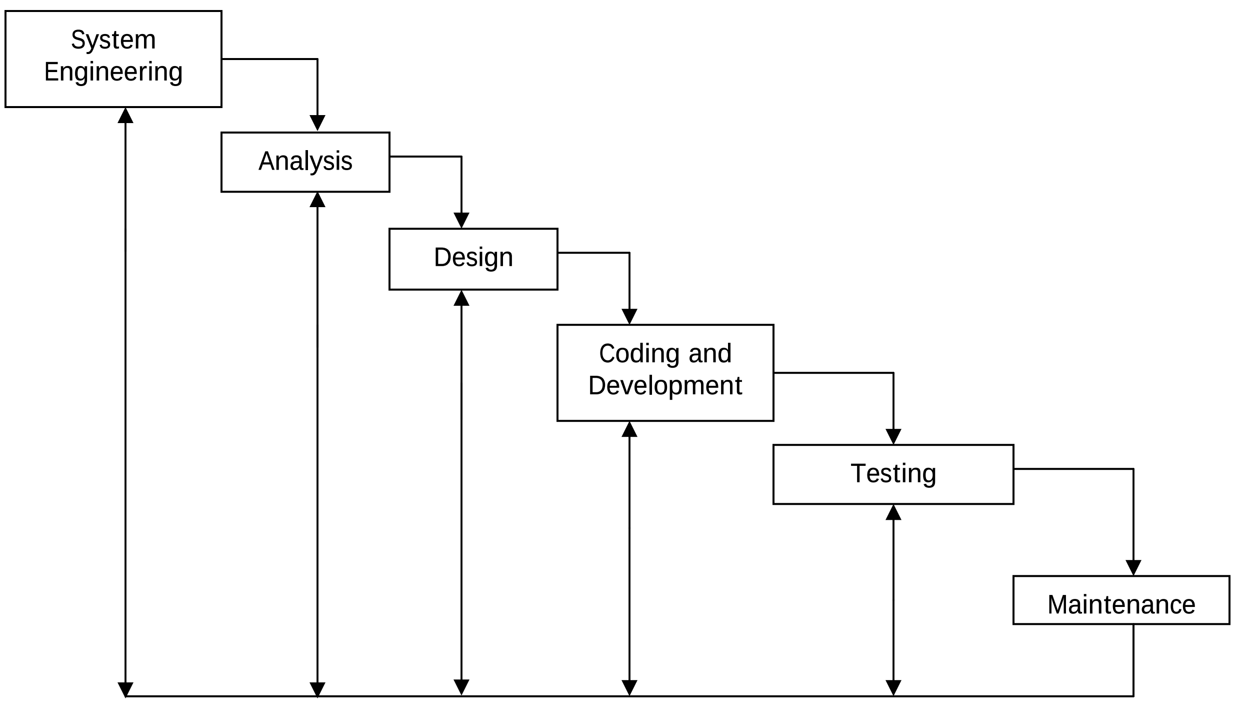
Pengertian Rekayasa Piranti Lunak

Pengertian rekayasa piranti lunak pertama kali diperkenalkan oleh **Fritz Bauer** sebagai penetapan dan penggunaan prinsip-prinsip rekayasa dalam usaha mendapatkan piranti lunak yang ekonomis, yaitu piranti lunak yang terpercaya dan bekerja secara efisien pada mesin atau komputer (Pressman, 1992, p19).

Paradigma Rekayasa Piranti Lunak

Terdapat lima paradigma (model proses) dalam merekayasa suatu piranti lunak, yaitu *The Classic Life Cycle* atau sering juga disebut *Waterfall Model*, *Prototyping Model*, *Fourth Generation Techniques (4 GT)*, *Spiral Model*, dan *Combine Model*. Pada penulisan skripsi ini dipakai model *Waterfall Model*.

Menurut **Pressman** (1992, p20-21), ada enam tahap dalam *Waterfall Model*, seperti pada gambar berikut:



Gambar 2.3 Model *Waterfall*

a. Rekayasa sistem (*System Engineering*)

Aktivitas ini dimulai dengan penetapan kebutuhan dari semua elemen sistem. Gambaran sistem ini penting jika perangkat lunak harus berinteraksi dengan elemen-elemen lain, seperti *hardware*, manusia dan *database*.

b. Analisis kebutuhan perangkat lunak (*Software Requirement Analysis*)

Yang dilakukan pada tahap ini adalah untuk mengetahui kebutuhan piranti lunak, sumber informasi piranti lunak, fungsi-fungsi yang dibutuhkan, kemampuan piranti lunak dan antarmuka piranti lunak tersebut.

c. Perancangan (*Design*)

Tahap ini menitikberatkan pada empat atribut program, yaitu struktur data, arsitektur piranti lunak, rincian prosedur dan karakter antarmuka. Tahap ini pula menerjemahkan kebutuhan ke dalam sebuah representasi perangkat lunak yang dapat dinilai kualitasnya sebelum dilakukan pengkodean.

d. Pengkodean (*Coding*)

Tahap pengkodean yang dilakukan adalah memindahkan hasil perancangan menjadi suatu bentuk yang dapat dimengerti oleh mesin, yaitu dengan membuat program.

e. Pengujian (*Testing*)

Tujuan dari tahap pengujian adalah agar output yang dihasilkan oleh program sesuai dengan yang diharapkan. Pengujian dilakukan secara menyeluruh hingga semua elemen, perintah dan fungsi dapat berjalan sebagaimana mestinya.

f. Pemeliharaan (*Maintenance*)

Tahap pemeliharaan dilakukan dengan tujuan mengantisipasi kebutuhan pemakai terhadap fungsi-fungsi baru yang dapat timbul sebagai akibat munculnya sistem operasi baru, teknologi baru dan *hardware* baru.

2.1.9 Representasi Data

- *Binary Numbers*

  Komputer menyimpan intruksi dan data ke dalam memori sebagai kumpulan sinyal listrik dengan konsep *on* dan *off* atau *true* dan *false*. Angka biner adalah angka berbasis 2 dimana setiap digitnya bernilai 0 atau 1.

- *Hexadecimal Integer*

  Bilangan biner dengan angka yang banyak dapat direpresentasikan sebagai bilangan *hexadecimal*. Bilangan *hexadecimal* adalah angka berbasis 16. Setiap digit pada angka *hexadecimal* mewakili empat *binary bits* dan dua digit *hexadecimal* mewakili delapan binary bits (*byte*). Sebuah digit *hexadecimal* memiliki nilai dari 0 sampai 15. Untuk digit yang lebih besar dari 9 diwakili dengan huruf abjad (A = 10, B = 11, C = 12, D = 13, E = 14, F = 15).

- *Decimal Number*

  Bilangan decimal merupakan bilangan berbasis 10 yang selalu kita gunakan dalam keseharian kita untuk mempermudah perhitungan dan representasi data. Bilangan ini memiliki nilai dari 0 sampai 9.

Binary	Decimal	Hexadecimal	Binary	Decimal	Hexadecimal
0000	0	0	1000	8	8
0001	1	1	1001	9	9
0010	2	2	1010	10	A
0011	3	3	1011	11	B
0100	4	4	1100	12	C
0101	5	5	1101	13	D
0110	6	6	1110	14	E
0111	7	7	1111	15	F

Gambar 2.4 Tabel Nilai Ekuivalen Bilangan *Binary*, *Decimal* dan *Hexadecimal*

2.2 Program Pendukung yang Dipakai

Program aplikasi yang dibuat membutuhkan pendukung program lainnya sebagai pelengkap perancangan. Program-program pendukung tersebut adalah sebagai berikut:

1. Macromedia Flash MX 2004

   Macromedia Flash MX 2004 digunakan untuk merancang dan membuat animasi interaktif menggunakan bahasa pemrograman *ActionScript*. Hasil akhir pembuatan animasi tersebut akan dikompilasi menjadi file SWF yang siap untuk di-*publish* ke jaringan internet. File SWF ini nantinya yang akan menjadi data input untuk diobfuskasi.

2. Macromedia Flash Player 7.0

   Program ini merupakan sebuah media player yang dirancang khusus untuk menjalankan file SWF.

3. FileSee 3.1

   Program ini dapat menampilkan struktur sebuah file dalam bentuk *ASCII* dan *hexadecimal*. Dalam penggunaannya, program ini membantu penulis dalam mempelajari struktur file SWF.

4. Sothink SWF Decompiler MX 2005

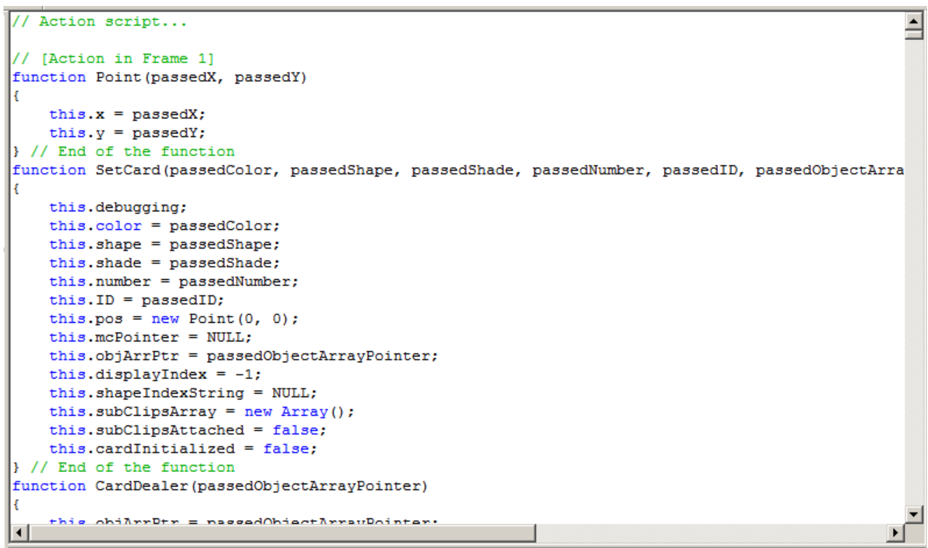
   Program ini sebenarnya merupakan aplikasi *decompiler* bagi file SWF yang digunakan banyak orang untuk mengambil data objek animasi 2D, video, suara, , gambar, serta untuk melihat *source code ActionScript* sehingga memungkinkan terjadinya tindakan plagiarisme. Pada perancangan program aplikasi ini, penulis menggunakannya untuk membandingkan file SWF sebelum dan sesudah di-obfuskasi.

2.3 Penelitian yang Relevan

Penelitian ini pernah dilakukan oleh Jon Bott pada tahun 2003. Program yang bernama SWOB (SWF Obfuscator) ini merubah nama variabel menjadi kode biner. Kelemahan program ini adalah mengenkripsi seluruh variabel beserta nilainya tanpa memberi kesempatan kepada *user* untuk memilih variabel atau nilai mana yang tidak perlu diacak. Pemilihan ini sangat penting dilakukan dikarenakan tidak semua variabel atau nilai boleh diacak, terutama bila kita menggunakan fasilitas komponen (*scrollbar, combo box, progress bar,* dll) yang telah disediakan. Apabila terdapat integrasi dengan bahasa pemrograman web

(ASP, PHP, XML, .NET, dll) untuk aplikasi database dengan cara me-*load* file eksternal, SWF tersebut tidak akan pasti berjalan sesuai harapan. Kelemahan ini dapat menyebabkan file SWF yang telah diacak tidak dapat berjalan dengan semestinya, bahkan merusak file SWF tersebut. Selain itu, ukuran file SWF tersebut menjadi bertambah besar akibat penambahan karakter-karakter bilangan biner.

Berikut adalah contoh yang dihasilkan oleh program SWOB sebelum dan sesudah dienkrip:



```
// Action script...

// [Action in Frame 1]
function Point(passedX, passedY)
{
    this.x = passedX;
    this.y = passedY;
} // End of the function
function SetCard(passedColor, passedShape, passedShade, passedNumber, passedID, passedObjectArra
{
    this.debugging;
    this.color = passedColor;
    this.shape = passedShape;
    this.shade = passedShade;
    this.number = passedNumber;
    this.ID = passedID;
    this.pos = new Point(0, 0);
    this.mcPointer = NULL;
    this.objArrPtr = passedObjectArrayPointer;
    this.displayIndex = -1;
    this.shapeIndexString = NULL;
    this.subClipsArray = new Array();
    this.subClipsAttached = false;
    this.cardInitialized = false;
} // End of the function
function CardDealer(passedObjectArrayPointer)
{
    this.objArrPtr = passedObjectArrayPointer;
```

Gambar 2.5 File SWF sebelum dienkripsi dengan program SWOB

```
// Action script...

// [Action in Frame 1]
function a__111000000101011(a__1000110000010111, a__100001111100110)
{
    this.a__101010111001011 = a__1000110000010111;
    this.a__10101101010101 = a__100001111100110;
} // End of the function
function a__110110000(a__111001000010110, a__11111010111110, a__111111011111110, a__110110100101
{
    this.a__10111011011;
    this.color = a__111001000010110;
    this.a__1001011100100 = a__11111010111110;
    this.a__110001100111000 = a__111111011111110;
    this.number = a__1101101001010;
    this.a__101011100010011 = a__110101011100010;
    this.a__101111100110111 = new a__111000000101011(0, 0);
    this.a__10000110101 = NULL;
    this.a__100101111100111 = a__110100010010;
    this.a__1000010010110000 = -1;
    this.a__111110010110011 = NULL;
    this.a__11110010001001 = new Array();
    this.a__110111100000 = false;
    this.a__10000011000100 = false;
} // End of the function
function a__111011100100111(a__110100010010)
{
```

Gambar 2.6 File SWF sesudah dienkripsi dengan program SWOB