Penerapan Kriptografi Klasik Dalam Keamanan Penyimpanan Data Permainan

Biolardi Yoshogi / 13509035
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
vsio@students.itb.ac.id

***Abstrak*—Dalam penyimpanan data permainan, data ditulis dalam sebuah file. Jika menggunakan fungsi dasar, maka data yang ditulis pada sebuah file tersebut masih berbentuk plaintext, alias tertulis informasiinformasi data permainan yang disimpan (seperti tingkat, kekuatan serangan, pertahanan, daya tahan, dan lain-lain). Data-data yang ditulis tersebut masih tertulis dalam sederet angka atau string. Pemainyang teliti dan tertarik untuk melihat file simpanannya, tentu saja dapat membaca deretan tersebut.**

Ditambah lagi, karena informasi simpanan yang pemain simpan mudah diingat, maka tinggal perlu mencari data tersebut dalam sebuah file. Karena pemain bisa membacanya, pemain juga bisa mengubah data tersebut. Akibatnya, permainan akan memanggil data yang telah dimanipulasi oleh pemain. Karena hal tersebut, maka permainan akan menjadi tidak seimbang atau terasa kurang tantangannya. Dengan menerapkan beberapa kriptografi klasik, maka hal tersebut dapat dipersulit. Permainan juga sebaiknya dilengkapi pesan header yang telah dienkripsi agar mengubah datanya tidak mudah sehingga jika header tersebut diubah, maka data simpanan tersebut akan dianggap rusak oleh permainan. Untuk penerapan kriptografinya, bisa digunakan algoritma caesar cipher, vigenere cipher, playfair cipher, atau algoritma lainnya. Untuk keamanan yang lebih kuat, sebaiknya menggunakan lebih dari satu algoritma karena jika hanya satu, lebih mudah dipecahkan. Karakter yang digunakan pun sebaiknya terdiri dari extended karakter sehingga semakin sulit dibaca dan ditebak apa algoritma yang digunakan. Kunci yang digunakan sebaiknya juga relatif panjang agar metode crack pun semakin sulit digunakan. Setelah enkripsi dilakukan, maka algoritma dekripsinya juga harus dipasang agar pesan-pesan yang terenkripsi didekripsi kembali menjadi deretan plaintext yang bisa dibaca oleh permainan menjadi nilai informasi yang sebelumnya telah disimpan di dalam di dalam file.

***Kata Kunci*— File Simpanan, Enkripsi, Permainan, Header File**

I. PENDAHULUAN

Dalam permainan digital zaman sekarang, seringkali durasi permainan dari awal hingga akhir selalu bisa berjam-berjam untuk menyelesaikannya. Tentu saja ada saatnya pemain untuk berhenti bermain karena suatu hal. Hal ini menyebabkan permainan tentu saja harus dihentikan dan dimatikan. Pada saat itu juga, maka apa yang telah dicapai pemain akan hilang begitu saja jika data permainan tidak disimpan. Karena hal itulah permainan dikembangkan agar dapat menyimpan data-data apa yang telah dicapai dalam permainan.

Dalam aplikasinya, ketika data permainan disimpan, maka data tersebut akan ditulis dalam suatu file. Data tersebut nantinya akan dibaca oleh permainan dan digunakan untuk memanggil data-data yang telah dicapai oleh pemain yang telah disimpan sebelumnya. Dengan begini, pemain dapat melanjutkan permainan dari penyimpanan data sebelumnya.

Data yang disimpan dalam suatu file, masih disimpan dalam bentuk plaintext. Misal pemain memiliki nama "Pizza", maka data yang disimpan dalam file akan tertulis "Pizza" juga. Dalam segi keamanan, hal ini tentu saja kurang aman karena pemain dapat memanipulasi data yang telah disimpan di dalam file tersebut. Misalkan nama "Pizza" tadi yang telah disimpan di dalam suatu file, kemudian pemain membuka file tersebut, kemudian mengubah "Pizza" menjadi "Cabai", ketika permainan memanggil data dari file simpanan, maka nama pemain tadi berubah menjadi "Cabai". Di sisi pemain, hal ini bisa menjadi keuntungan atau kerugian. Beberapa keuntungannya: misalka kekuatan karakter yang digunakan pemain bisa sangat kuat dengan lebih sedikit usaha, pemain bisa mengganti nama karakter yang digunakannya jika sudah bosan dengan nama tersebut, pemain dapat melangkahi sebuah misi atau tugas tanpa perlu mengerjakannya, dan masih banyak lagi. Beberapa kerugian: jika bermain secara ganda dengan pemain lain, maka pemain yang jujur tentu saja akan dicurangi oleh pemain lain yang telah mengubah informasi di dalam file tersebut, kadang-kadang permainan bisa menjadi membingungkan karena state-nya telah diubah sehingga jalan tugas atau misinya menjadi tidak jelas. Oleh karena permasalahan tersebut, maka akan diberikan beberapa solusi yang mampu untuk minimal mempersulit hingga mencegah pemain mengubah data simpanan permainan di dalam file.

II. TEORI DASAR

Sebelum mulai ke dalam solusi, akan dijelaskan

beberapa metode dasar dan algoritma klasik akan digunakan untuk membuat file data simpanan permainan menjadi kuat terhadap pengubahan data oleh pemain.

File Header

Semacam informasi yang menyimpan data atribut suatu file. Misalkan suatu file bmp memiliki informasi berapa lebar dan panjang layar pixelnya. Misalnya suatu file video, memiliki ada berapa frame yang dimilikinya.

Caesar Cipher

Enkripsi suatu pesan dengan mensubtitusikan suatu huruf dengan furuf lain. Caranya dengan menggeser indeksnya ke huruf indeks huruf lain, kemudian lakukan modulus dengan 26.

Rumus Enkripsi: En(P) = (P+n) mod 26

En adalah fungsi enkripsi Caesar Cipher
P adalah himpunan huruf indeks plainteks
n adalah nilai pergeseran

Rumus Dekripsi: Dn(C) = (C+n) mod 26

En adalah fungsi enkripsi Caesar Cipher
C adalah himpunan huruf indeks plainteks
n adalah nilai pergeseran

Jika nilai pergeseran melebihi 25, maka akan kembali lagi ke 0 karena hanya terdapat 26 karakter di alpabet. Jika digunakan Caesar Cipher Extended, maka jumlah karakter yang digunakan adalah 256.

Contoh 1:

Plaintext

SAYA MASIH BELUM MENDAPATKAN PIZZA

Nilai pergeseran

5

Maka hasil pergesarannya

XFDF RFXNM GJQZR RJSIFUFYPFS UNEEF

Contoh 2:

Plaintext

Pada suatu malam, diriku bermimpi sedang memakan pizza degan rasa daging.

Nilai pergeseran

10

Maka hasil pergesarannya

ZKNK CEKDE WKVKW, NSBSUE LOBWSWZS CONKXQ WOWKUKX ZSJJK NOQKX BKCK NKQSXQ.

Berikut ini algoritma enkripsi dan dekripsinya (Extended Mode) yang digunakan dalam bahasa C++:

Enkripsi

```
void CaesarCipherASCII :: encryptKey() {
        for (int i=0; i<inputText_.length(); i++) {
                outputText_ +=
(inputText_[i]+key_)%256;
                if ((int)outputText_[i] < 0) {
                        outputText_[i] +=256;
                }
        }
}
```

Dekripsi:

```
void CaesarCipherASCII :: decryptKey() {
        for (int i=0; i<inputText_.length(); i++) {
                outputText_ += (inputText_[i]-
key_)%256;
                if ((int)outputText_[i] < 0) {
                        outputText_[i] +=256;
                }
        }
}
```

Vigenere Cipher

Enkripsi dengan metode ini adalah dengan menjumlahkan indeks masing-masing huruf pada pesan dengan indeks masing-masing huruf dengan kuncinya, kemudian lakukan modulus dengan nilai 26.

Kuncinya sendiri, bisa dikembangkan dengan pengulangan kunci itu sendiri atau mengambil plaintext kemudian menggabungkannya dengan kunci hingga panjang kunci sama dengan panjang plaintext.

Contoh 1:

Plaintext

SAYA MASIH BELUM MENDAPATKAN PIZZA

Kunci

PIZZA

Berdasarkan plainteks dan kuncinya, maka kuncinya

PIZZ APIZZ APIZZ APIZZAPIZZA PIZZA

Hasil enkripsi

HIXZ BIRHH JDKUB LDNSIOZTZIM PXHYZ

Contoh 2:

Plaintext

Pada suatu malam, diriku bermimpi sedang memakan pizza degan rasa daging.

Kunci

panas

Berdasarkan plainteks dan kuncinya, maka kuncinya

PIZZ APIZZ APIZZ APIZZA PIZZAPIZ ZAPIZZ APIZZAPIZZAPI ZZAPI ZZAP IZZAPI

Hasil enkripsi

EAQA HUNTM MNLSB, DAGIXU QEEMABPV KTDNNY MRMSZAA HXZMA SETAF RNSS DNGACG.

Berikut ini algoritma enkripsi dan dekripsinya (Extended Mode) yang digunakan dalam bahasa C++:

Enkripsi

```
void VigenereCipherASCII :: encryptKey() {
        string keyRepeated = getRepeatKey();
        string tempOutput = "";
        for (int i=0; i<inputText_.length(); i++) {
                tempOutput[0] =
(char)((int)inputText_[i]+(int)keyRepeated[i])%256;
                outputText_ += tempOutput[0];
                tempOutput = "";
        }
}
```

Dekripsi

```
void VigenereCipherASCII :: decryptKey() {
        string keyRepeated =
getRepeatKey();
        string tempOutput = "";

        for (int i=0;
i<inputText_.length(); i++) {
                tempOutput[0] =
(char)((int)inputText_[i]-
(int)keyRepeated[i])%256;
                if ((int)tempOutput[0] <
0) {
                        tempOutput[0] +=
256;
                }
                outputText_ +=
tempOutput[0];
                tempOutput = "";
        }
}
```

Rail Fence Cipher

Rail Fence Cipher adalah metode enkripsi dengan menyusun huruf membentuk suatu garis zigzag, kemudian menyatukan huruf-huruf yang satu baris dengna baris lain berikutnya.

Contoh 1:

Pizza hari ini tidak ada.

Dienkripsikan:

Kunci: 3

- disusun dulu menjadi pola zig-zag

```
P   A   I   T   K
 I Z H R I  I I A A  A
  Z   A   N   D   D
```

- disatukan yang satu barus dengan baris lain berikutnya sehingga hasilnya menjadi seperti di bawah ini:

PAITKIZHRIIIAAAZANDD

Contoh 2:

Pada suatu malam, diriku bermimpi sedang memakan pizza degan rasa daging.

Dienkripsikan:

Kunci: 7

- disatukan yang satu barus dengan baris lain berikutnya sehingga hasilnya menjadi seperti di bawah ini:

PAIMGGALMMMEAEANDADRPMKDNIAMIEIGA ARGSURBSNNZAAUTIUEAPZSDAKDIA

Berikut ini algoritma enkripsi dan dekripsinya (Extended Mode) yang digunakan dalam bahasa Ruby:

Enkripsi

```
def encipher()
        # enciphers with Rail Fence Cipher Extended

                plainText = @inputText_
                totalRails = @key_
                list_Str = []
                strTemp = ""
                output = ""
                i = 0
                j = 0

                begin
                        raise unless
(totalRails!=1)
                rescue
                        if (totalRails==1)
```

```
                                        puts "\n> Deciphering Result:\n"+plainText

        @outputText_ = plainText
                                        return
                                end
                end

                for i in 0..(totalRails-1)
                        list_Str.push("")
                end

                i = 0

                begin
                        while (j != plainText.length)
                                        while (i!=totalRails-1)

        list_Str[i] += plainText[j]

        j+=1

        i+=1
                                        end
                                        while (i!=0)

        list_Str[i] += plainText[j]

        j+=1

        i-=1
                                        end
                                end
                rescue
                        # I am rescued! :D
                end

                i = 0

                for i in 0..(totalRails-1)
                        output += list_Str.push[i]
                end

                puts "\n> Enciphering Result:\n"+output

                @outputText_ = output
        end
```

Dekripsi

```
def decipher()

        # deciphers with Rail Fence Cipher Extended
                cipherText = @inputText_
                totalRails = @key_
                multiplier = 0
                list_Str = []
                pointer = 0
                output = ""

                begin
                        raise unless (totalRails!=1)
                rescue
                        if (totalRails==1)
                                puts "\n> Deciphering Result:\n"+cipherText

        @outputText_ = cipherText
                                return
                        end
                end

                extra = cipherText.length % (totalRails*2-2)
                base = (cipherText.length-extra) / (totalRails*2-2)
                extraPointerPeriode = extra / 2
                extraPointer = totalRails*2 - extra

                #puts extra
                #puts base
                #puts extraPointerPeriode
                #puts extraPointer

                numb = 0

                for i in 0..(totalRails-1)
                        if ( (i == 0) || (i == totalRails-1) )
                                multiplier = 1
                        else
                                multiplier = 2
                        end
                        if ((cipherText[pointer..(pointer+(base*multiplier)+numb-1)]).length != 0)
                                if (extra != 0)

        if (extraPointerPeriode == 0)

        else

        end

        if (i

        else

        end
```

```
            end


            extra -= 1
                                                else

            numb = 0
                                                end


            list_Str.push(cipherText[pointer..(pointer+(bas
e*multiplier)+numb-1)])
                                                #puts
list_Str[i]
                                                pointer =
pointer+(list_Str[i]).length
                                    else

                                    end
                        end

                        j = 0

                        begin
                                    while (output.length !=
cipherText.length)
                                                while
(j!=totalRails-1)

            output += list_Str[j][0]

            list_Str[j].slice! list_Str[j][0]

            j+=1
                                                end
                                                while (j!=0)


            output += list_Str[j][0]

            list_Str[j].slice! list_Str[j][0]
                                                            j
-=1
                                                end
                                                j = 0
                                    end
                        rescue

                        end

                        puts "\n> Deciphering
Result:\n"+output

                        @outputText_ = output
            end
```

Mode Extended

Algoritma enkripsi klasik dengan mode extended, membuat algoritma tersebut bisa menerima karakter aneh, tanda baca, dan case sensitive. Hal ini membuat algoritma menjadi lebih kuat karena karakter yang bisa digunakan pun menjadi bertambah hingga 256.

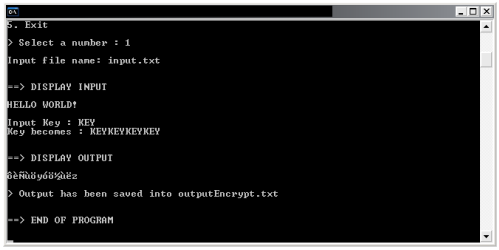
Jika suatu string memiliki panjang N, maka peluang untuk membobolnya adalaha 1 banding 256 pangkat N. Contoh: misal suatu string terenkripsi adalah 36$@)#*@, maka peluang untuk membobolnya adalah 256 pangkat 8.

Algoritma Caesar Cipher, Vigenere Cipher, dan Rail Fence Cipher dengan mode Extended, dapat mengolah karakter asing, tanda baca, dan case sensitive hingga jumlah karakter yang bisa digunakan dalam satu karakter dalah 256 karakter.

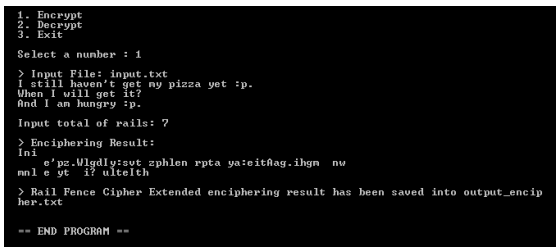
Berikut ini contoh hasil enkripsi algoritma yang disebutkan tadi dalam command prompt (gambar diambil dari blog sendiri).

Caesar Cipher Extended

Vigenere Cipher Extended



Rail Fence Cipher Extended



III. METODE PEMECAHAN MASALAH

Untuk mengatasi masalah bahwa data file simpanan bisa dibuah oleh pemain, maka digunakan

enkripsi pada data file. Pemain tentu saja menjadi lebih kesulitan untuk dapat membaca data yang tertulis pada file tersebut. Akan tetapi, hal ini tentu saja masih saja kurang karena meskipun data tidak bisa dibaca pemain, tetapi dengan mencoba untuk mengubahnya, tetap saja akan membuat permainan memanggil data dari file yang telah dibuah oleh pemain tersebut.

Untuk itulah, selain dilakukan enkripsi, file simpanan juga dilengkapi oleh file header. File header juga akan dienkripsi sehingga pemain akan kesulitan menebak mana yang header dan mana yang bukan. Jika header diubah, permainan harus bisa menolak header yang tidak valid (biasanya muncul pesan bahwa file simpanan telah rusak). Dengan ditambahkan file header, maka file simpanan permainan terdiri dari dua bagian, yaitu Header dan Body. Body yang akan menyimpan informasi-informasi dari game yang disimpan.

Algoritma enkripsi yang akan digunakan adalah Caesar Cipher, Rail Fence Cipher, dan Vigenere Cipher. Ketiga algoritma tersebut akan digunakan bersama untuk lebih mempersulit kriptanalisis ciphertext. Mode yang digunakan pun adalah yang Extended yang mana kemungkinan karakter yang digunakan dibandingkan total karakter adalah 256 pangkat n, yang mana n adalah panjang teks.

Contoh 1 :

Dalam contoh ini, Caesar Cipher Extended menenkripsi plaintext, hasil Caesar Cipher Extended dienkripsi Vigenere Cipher Extended, dan hasil Vigenere Cipher Extended dienkripsi Rail Fence Cipher Extended.

Urutan:
1. Caesar Cipher Extended (Key:7)
2. Vigenere Cipher Extended (Key:pizza)
3. Rail Fence Cipher Extended (Key:3)

Plaintext:

Saya belum makan pizza hari ini dan kemarin.

Ciphertext:
Caesar Cipher Extended:

Zh€h'ils|t'thrhu'wp□□h'ohyp'pup'khu'rlthypu5

Caesar Cipher Extended+Vigenere Cipher Extended:

ÊÑúâˆÙÕíöÕ—Ýâì Éå□ñêâñÑ¡éÉéÙ¡êÖà□åâÖ—Ûæî ÉéÙï¯

Caesar Cipher Extended+Vigenere Cipher Extended +Ruby Fence Cipher:

Êˆöâ□ñÉêåÛéÑâÙíÕÝìåñâÑéé¡Ö□â—æÉÙ¯úÕ—Éê¡ÙàÖîï

Contoh 2 :

Dalam contoh ini, Rail Cipher Extended meng-enkripsi plaintext, hasil Cipher Extended dienkripsi Caesar Cipher Extended, dan hasil Caesar Cipher Extended dienkripsi Vigerene Cipher Extended.

Urutan:
1. Rail Fence Cipher Extended (Key:3)
2. Caesar Cipher Extended (Key:7)
3. Vigenere Cipher Extended (Key:pizza)

Plaintext:

Saya belum makan pizza hari ini dan kemarin.

Ciphertext:
Fence Cipher Extended:

S ua zaidkraablmmknpzahr n a eai.ye ai iinmn

Fence Cipher Extended + Caesar Cipher Extended:

Z'|h'□hpkryhhisttruw□hoy'u'h'lhp5€l'hp'pputu

Fence Cipher Extended + Caesar Cipher Extended + Vigenere Cipher:

Ê□öâˆñÑêåÓéÑâãÔäÝìïØñÑéóˆå□â¡ÍØÙ¯úÍ—Ñê¡ÑàÞîï

Contoh 3 :

Dalam contoh ini, Vigenere Cipher Extended menenkripsi plaintext, hasil Vigenere Cipher Extended dienkripsi Caesar Cipher Extended, dan hasil Caesar Cipher Extended dienkripsi Rail Fence Cipher Extended.

Urutan:
1. Vigenere Cipher Extended (Key:pizza)
2. Caesar Cipher Extended (Key:7)
3. Rail Fence Cipher Extended (Key:3)

Plaintext:

Saya belum makan pizza hari ini dan kemarin.

Ciphertext:
Vigenere Cipher Extended:

ÃÊóÛ□ÒÎæïÎ□ÖÛåÂÞ‰êãÛêÊšâÂâÒšãÏÙ‰ÞÛÏ□Ô ßçÂâÒè¨

Vigenere Cipher Extended + Caesar Cipher Extended:

ÊÑúâˆÙÕíöÕ—Ýâì Éå□ñêâñÑ¡éÉéÙ¡êÖà□åâÖ—Ûæî ÉéÙï¯

Vigenere Cipher Extended + Caesar Cipher Extended + Ruby Fence Cipher:

Êˆöâ□ñÉêåÛéÑâÙíÕÝìåñâÑéé¡Ö□â—æÉÙ¯úÕ—Éê¡ÙàÖîï

Dari ketiga contoh tadi, didapat bahwa meskipun urutan algoritmanya tidak sama, namun hasil yang diperoleh ketika menggunakan algoritma dan kunci yang sama, maka hasilnya pun sama.

IV. EKSPERIMEN DAN HASILNYA

Dalam eksperimen ini, permainan yang digunakan adalah game yang pernah dibuat oleh tim saya. Permainannya adalah "Destined Way" (bisa diunduh di http://sandbox.yoyogames.com/games/121597-destined-way). Dibuat dengan game engine "Game Maker 7.0 dan 8.0 dengan bahasa pemrogramannya adalah GML (Game Maker Language).

Screen Shot Permainan "Destined Way"

Untuk eksperimen ini, permainan ini juga dimodifikasi sehingga menampilkan sebuah nilai string ("Pizza" di screenshot) dan menyimpan informasi string "This is a header" sebagai File Header serta validasi apakah headernya telah dimodifikasi atau tidak. Jika header mengalami modifikasi (tidak sama dengan nilai string "This is a header", maka akan muncul pesan kesalahan seperti di bawah ini:

Pesan Kesalahan jika Heaeder Tidak Valid

Dalam eksperimen ini, permainan ini belum dilengkapi oleh algoritma enkripsi dan dekripsi sehingga akan digunakan program luar (yang juga telah dibuat sendiri) yang mana sebagai berikut:

Caesar Cipher (ASCII) (bisa diunduh di http://virucodesoup.blogspot.com/2012/01/cpp-caesar-cipher-alphabet.html)

Vigenère Cipher (ASCII) (bisa diunduh di http://virucodesoup.blogspot.com/2012/02/cpp-vigenere-cipher-ascii.html)

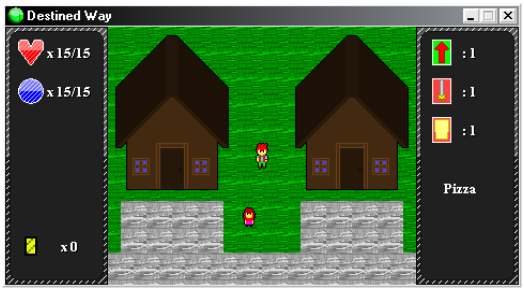
Rail Fence Cipher Extended (bisa diunduh di http://virucodesoup.blogspot.com/2012/03/ruby-rail-fence-cipher-extended.html)

Berikut ini hasil eksperimen yang dilakukan:

Asumsi

- Algoritma pemrosesan file simpanan tidak diketahui
- Pemain tidak bisa membedakan mana yang header dan mana yang bukan

Screenshot



Kondisi Awal

Input File Simpanan

Untuk yang tidak terenkripsi:

This is a header
Pizza
6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 16 16 16 16 1 2 2 2 2 2 26 1 1 217 181 76 76 2 2 2 2 1 1 1 1 1 1 1 1 1 1

Untuk yang Terenkripsi (nilai dari yang tidak terenkripsi):

Di sini, hasil enkripsi saling menenkripsi yaitu Caesar Cipher Extended mengekripsi plaintext, Vigenere Cipher Extended mengenkripsi hasil enkripsi dari Caesar Cipher Extended, dan Rail Fence Cipher Extended mengenkripsi hasil enkripsi dari Vigenere Cipher Extended.

Urutan Enkripsi:

1. Caesar Cipher Extended

Key: 10

^rs}*s}*k*rokno|Zs„„k*@*;@*;@*;@*;@*;*<*<*<*<*<*<*<@*;*;*<;A*;B;*A@*A@*<*<*<*<*;*;*;*;*;*;*;*;*;*;*;*;*;

2. Vigenere Cipher Extended

Key: pizza

ÎÛí÷‹ãæ¤å‹âØåèÐì}ÔíåôÔŽ¤¡š¤¤µ‹«“µ¤œš¤¤µ‹«“µ¤œš¤¤µ‹«“µ¤œš¤¤µ‹«“µ¤œ°“µº‹«©¤µ¡š¤¤¶‹¬“¶¤š¥¤¶¡š¤¤µ‹

¬¤»¤œ²¤¤»¡š^{a0}¤□š¥¤¶‹¬“µ¤œš¤¤µ‹‹“µ¤œš¤¤µ‹«

3. Rail Fence Cipher Extended

Key: 5

Îå}¡µ«µ¤œ¤¬¶¤¡¤š¤Û¤‹ìÔ¤š“¤‹“¤‹š¤¤°©µ‹“¤¡¬»»š¥¶ œ¤µœ«íæâÐíŽ¤«œµµ¤«œµµ“«¡¶¶¥š‹¤¤ªš‹¤¤“š‹÷ãØèåÔ¤‹ š¤¤š“¤‹“µ‹š¤¤š¤µœ¤º□¬µµ«¤µ‹åôµ¤œµ«º□¤²¤“‹¤

Hasil yang digunakan adalah dari Rail Fence Cipher Extended yang mana menrupakan hasil enkripsi algoritma enkripsi lainnya.

1. Manipulasi File Simpanan Permainan Tanpa Header dan Enkripsi

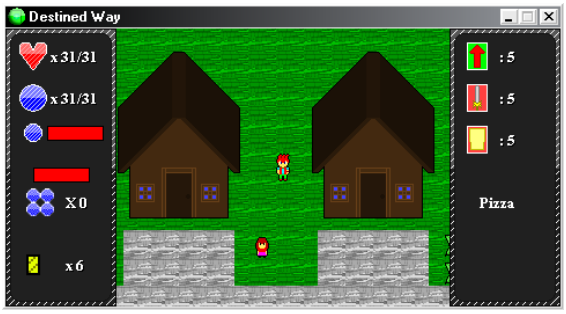
Asumsi

- Dalam eksperimen ini, asumsi file header tidak diperiksa oleh file.

Manipulasi Input

Pizza
6 8 8 8 8 7 7 6 7 7 7 7 8 8 7 7 6 10 3 3 32 32 32 32 5 6 6 6 6 6 26 7 7 217 181 76 76 2 5 8 8 5 5 5 5 5 5 3 3 3 3

Screenshot Hasil Manipulasi



Penjelasan Hasil Manipulasi

Terjadi perubahan pada nilai string dan beberapa nilai integer.

2. Manipulasi File Simpanan Permainan dengan Header, tetapi Tanpa Enkripsi

Asumsi

Pemain mengetahui mana yang header dan mana yang bukan

Manipulasi Input

This is a burger
Pizza
6 8 8 8 8 7 7 6 7 7 7 7 8 8 7 7 6 20 22 8 32 32 38 32 9 9 9 9 9 6 26 9 9 9 17 181 76 76 2 5 10 8 10 5 5 10 5 5 3 3 3 10

Screenshot Hasil Manipulasi

Penjelasan Hasil Manipulasi

Muncul pesan kesalahan bahwa file simpanan rusak

3. Manipulasi File Simpanan Permainan tanpa Header, tetapi Body Dienkripsi

Asumsi

Dalam eksperimen ini, asumsi file header tidak diperiksa oleh file.

Manipulasi Input

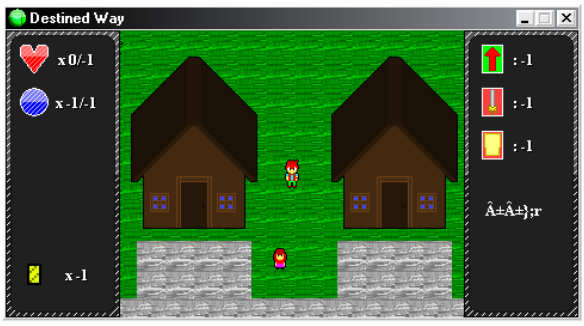
Dienkripsi:

Îå}¡µ«µ¤œ¤¬¶¤¡¤š¤Û¤‹ìÔ5¤š“¤‹“¤‹š¤¤°©µ‹“¤5¡¬»»š¥ ¶œ¤µœ«íæ5í5Ž¤«œµµ¤IKUHµµ“«¡¶¶¥š‹¤¤ªš‹¤¤“š‹÷ãØèå Ô¤‹š¤¤š10()š¤¤š50¤º□¬µµ100µ‹åôµ¤œµ«º□¤10¤“‹¤

Didekripsi:

T1iz}sa□»ppT,Z
±±};r
6 1P 91 1 1)1 1 1 J*1 1 Ï"-à 1 ÑÅ@µÄ 16B¥ /;6 1(** ' 2 9*C 2Ê<Â-¬:'1(9¶G76A1A*' !?; 1K)- ¶1 J*½ 1("h'

Screenshot Hasil Manipulasi



Penjelasan Hasil Manipulasi

Terjadi perubahan pada nilai string dan beberapa nilai integer yang mana tidak sesuai dengan yang dimasukkan

di file yang terenkripsi tadi. Di tambah lagi, nilai string juga menjadi terkesan acak.

4. Manipulasi File Simpanan Permainan dengan Header dan Enkripsi

Asumsi

- Pemain mengetahui mana yang header dan mana yang bukan

Manipulasi Input

Dienkripsi:

```
Î$%^µ«µ¤œ¤¬$%^š¤Û¤‹ìÔ¤š“¤‹“¤‹šaa)(&¤¤$#°©µ‹“¤
¡¬»»š¥¶œ$#%#$¤***æâÐí#$%Ž¤«œµµ¤«œµ%%¡¶¶¥š‹¤
¤ªš‹¤¤“š‹÷ãØèåÔFGHFH¤¤š¤µœ¤º□¬µµ«¤µ‹åô%$^µ«º¤
□¤²¤“‹¤
```

Didekripsi:

```
T±¡ÚJ1B 92±¡Ú/*h □Z19 /çî¥¤»*1 ŸE/B9'977/+C ¸«°
¿°·b^es° ¡#*81J*81º«.22: ? ( s_mnrPÂÜÎÓÄ 9
119@*(1J111zz² ÚJ1G 9819
```

Screenshot Hasil Manipulasi

Penjelasan Hasil Manipulasi

Muncul pesan kesalahan bahwa file simpanan rusak

V. ANALISIS

Dari hasil yang didapat, pemasangan enkripsi dan header, memiliki efek yang berbeda satu sama lain. Berikut ini beberapa analisis yang bisa diambil dari hasil eksperimen yang didapat.

Untuk file simpanan yang tidak dilengkapi header dan enkripsi, lebih mudah dimanipulasi oleh pemain. Nilai yang dimanipulasi tersebut, berikutnya dipanggil oleh permainan sehingga nilainya pun sesuai yang dimasukkan pengguna.

Untuk file simpanan yang dilengkapi header tetapi tidak dienkripsi, menjadi lebih sulit dimanipulasi namun masih kurang signifikan. Terutama jika pemain dapat mengenali mana yang header dan mana yang body dan dapat memanipulasi body filenya tanpa memanipulasi headernya. Permainan hanya menolak file simpanan jika header file-nya diubah, tidak dengan manipulasi bod file-nya.

Untuk file simpanan yang tidak dilengkapi header, namun body-nya dienkripsi, masih mudah dimanipulasi, namun keakuratan untuk memanipulasinya sangat rendah karena nilai-nilainya hampir tidak mungkin bisa dibaca dan dimengerti oleh pemain. Ditambah lagi spasinya tidak jelas yang mana karena enkripsi, jadi berganti dengan karakter lain. Akan tetapi, meskipun menjadi sulit dibaca dan sangat kecil akurasinya, permainan masih bisa memanggil isi simpanannya dan memasukkannya ke dalam nilai permainan. Akan tetapi, nilainya relatif tidak akurat dan tidak diharapkan.

Untuk file simpanan yang dilengkapi header dan dienkripsi, karena sulit dibaca oleh pemain, maka memanipulasi datanya lebih cenderung dapat merusak header file dan permainan akan menolak isi file sehingga permainan menampilkan pesan kesalahan. Akibatnya permainan tidak memanggil isi file ke dalam nilai permainannya.

VI. KESIMPULAN

Dari hasil analisis tadi, dapat disimpulkan bahwwa agar penyimpanan data permainan dalam file aman dan tidak dapat dimanipulasi oleh pemain, maka sebaiknya file tersebut sebaiknya menggunakan header. Hal ini dilakukan agar permainan menolak file jika headernya telah sedikit saja dimanipulasi sehingga header-nya rusak atau tidak sesuai dengan spesifikasi pemeriksaan header oleh permainan.

Bukan itu saja, sebaiknya file simpanan juga dienkripsi dengan algoritma enkripsi sehingga pemain sulit membedakan mana yang merupakan header file dan manapula yang merupakan body. Ditambah lagi, jila header-nya pun tidak rusak, nilai yang dibaca dari permainan, tidak akan sesuai dengan harapan pemain.

Algoritma yang digunakan pun sebaiknya bukan satu saja dan bisa menerima karakter extended. Dengan karakter extended, maka satu karater yang benar adalah 1 banding 256. Ditambah lagi, jika panjangnya sama dengan n, maka 1 banding 256 pangkat n. Untuk algoritmanya, semakin banyak algoritma yang digunakan, maka semakin aman karena pemain harus menebak lagi apa algoritma dan kunci yang digunakan berikutnya untuk mengenkripsi dan mendekripsikan file simpanan yang telah dienkripsi.

Tentu saja masih ada cara lain yang lebih baik untuk melindungi file yang berisi simpanan data. Misalkan, algoritma enkripsi yang digunakan bisa yang block cipher atau stream cipher yang meruapakan algoritma modern yang mengolah data hingga perbit. Salah cara proteksi file simpanan yang dibahas di sini, masih mampu untuk melindungi file simpanan dari pemain.

REFERENSI

[1] http://www.virucodesoup.blogspot.com/ (Source code, gambar, dan simulasi enkripsi dan dekripsi). Waktu Akses: 20 Maret 2012, Pukul 10:00 WIB.

[2] Munir, Rinaldi. Ir. M.T. *Diktat Kuliah IF5054 Kriptografi*

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Maret 2012

Biolardi Yoshogi, 13509035