IMPLEMENTASI SANDI AFFINE UNTUK PENGAMANAN FILE MICROSOFT OFFICE

Indra Oloan Nainggolan

Widyaiswara Muda
Kementerian Perindustrian R.I
Balai Diklat Industri Medan

olo.nainggol123@gmail.com

ABSTRAK

Aplikasi perkantoran yang paling sering digunakan adalah Microsoft Office. Permasalahan keamanan muncul apabila file Office yang dibuat bersifat rahasia. Tanpa adanya pengamanan, maka file Office dapat dicuri atau disadap oleh orang lain. Metode kriptografi yang dapat digunakan adalah sandi Affine. Sandi Affine merupakan algoritma kriptografi stream cipher, yang memproses data per karakter. Kunci yang digunakan pada sandi Affine adalah 2 (dua) bilangan integer, yaitu a dan b. Proses enkripsi dilakukan pada setiap nilai byte di dalam file. Dengan menggunakan kunci yang sama, maka proses dekripsi akan mengembalikan file Office ke keadaan semula. Aplikasi dapat melakukan pengamanan terhadap file Microsoft Office, dengan melakukan proses enkripsi menggunakan sandi Affine. Aplikasi ini akan memberi tanda ekstension *.aff kepada file yang telah terenkripsi, sehingga hasil enkripsi tidak akan dapat dibuka dan dikenali lagi sebagai format file Office.

Kata Kunci: Sandi affine, Pengamanan, File Microsoft Office

ABSTRAK

*Aplikasi perkantoran yang paling sering digunakan adalah Microsoft Office. Permasalahan keamanan muncul apabila file Office yang dibuat bersifat rahasia. Tanpa adanya pengamanan, maka file Office dapat dicuri atau disadap oleh orang lain. Metode kriptografi yang dapat digunakan adalah sandi Affine. Sandi Affine merupakan algoritma kriptografi stream cipher, yang memproses data per karakter. Kunci yang digunakan pada sandi Affine adalah 2 (dua) bilangan integer, yaitu a dan b. Proses enkripsi dilakukan pada setiap nilai byte di dalam file. Dengan menggunakan kunci yang sama, maka proses dekripsi akan mengembalikan file Office ke keadaan semula. Aplikasi dapat melakukan pengamanan terhadap file Microsoft Office, dengan melakukan proses enkripsi menggunakan sandi Affine. Aplikasi ini akan memberi tanda ekstension *.aff kepada file yang telah terenkripsi, sehingga hasil enkripsi tidak akan dapat dibuka dan dikenali lagi sebagai format file Office.*

Kata Kunci: *Sandi affine, Pengamanan, File Microsoft Office*

I. PENDAHULUAN`

Microsoft office merupakan aplikasi perkantoran yang paling sering digunakan untuk memudahkan tugas perkantoran. Dengan Microsoft Office kita bisa mengetik proposal, surat, laporan keuangan, presentasi, database karyawan, membuat logo, dan lain – lain. Permasalahan keamanan muncul apabila data yang dibuat dengan menggunakan aplikasi tersebut bersifat rahasia. Pemecahan dari permasalahan ini adalah dengan menggunakan ilmu kriptografi. Dengan menggunakan kriptografi, maka file Office dapat dienkripsi menjadi file teracak, sehingga dapat disimpan atau dikirim dengan aman. Apabila data terenkrip disadap atau dicuri oleh orang lain, maka data tersebut tidak berguna karena file telah diacak sebelumnya, dan Microsoft Office atau bahkan Windows tidak akan mengenal format file yang telah dienkripsi tersebut.

Salah satu metode kriptografi yang dapat digunakan adalah sandi Affine. Sandi Affine termasuk pada algoritma kriptografi stream cipher, yang memproses data per karakter. Sistem sandi Affine merupakan sandi monoalfabetik yang menggunakan teknik substitusi yang menggunakan fungsi linier ap + b untuk enkripsi teks asli p dan fungsi a-1c – b untuk dekripsi teks sandi c.

Fungsi dekripsi menggunakan fungsi invers yang akan diselesaikan dengan algoritma Extended Euclidean. Implementasi nyata dari fungsi enkripsi menggunakan metode kali dan tambah dari kelas aritmatika modular untuk melakukan perhitungan c[i] = a x p[i] + b, sedangkan fungsi dekripsi menggunakan metode invers, kali, tambah dan

negatif dari kelas aritmatika modular untuk melakukan perhitungan, p[i] = a-1 x c[i] – b. Kunci yang digunakan pada sandi Affine adalah 2 bilangan integer, yaitu a dan b.

II. TINJAUAN PUSTAKA

2.1. Sistem Kriptografi

Kata kriptografi (cryptography) berasal dari bahasa Yunani, yaitu kriptos yang artinya secret (rahasia), dan graphein, yang artinya writing (tulisan). Jadi, kriptografi berarti secret writing (tulisan rahasia). Ada beberapa definisi kriptografi yang telah dikemukakan di dalam berbagai literatur. Pada pengertian modern, kriptografi adalah ilmu yang bersandarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi entitas.

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan (cryptography is the art and science of keeping messages secure). Defenisi lain dari kriptografi adalah ilmu yang mempelajari teknik-teknik matematka yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data serta otentikasi.

Kriptografi bertujuan untuk memberi layanan keamanan (yang juga dinamakan sebagai aspek-aspek keamanan) sebagai berikut:

a. Kerahasiaan (confidentiality), adalah layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak. Di dalam kriptografi, layanan ini direalisasikan dengan menyandikan pesan menjadi ciphertext. Misalnya pesan "Harap datang pukul 8" disandikan menjadi "TrxC#45motyptre!%". Istilah lain yang senada dengan confidentiality adalah secrecy dan privacy.
b. Integritas data (data integrity) , adalah layanan yang menjamin bahwa pesan masih asli / utuh atau belum pernah dimanipulasi selama pengiriman. Dengan kata lain, aspek keamanan ini dapat diungkapkan sebagai pertanyaan: "Apakah pesan yang diterima masih asli atau tidak mengalami perubahan (modifikasi)?". Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubstitusian data lain ke dalam pesan yang sebenarnya. Di dalam kriptografi, layanan ini direalisasikan dengan menggunakan tanda-tangan digital (digital signature). Pesan yang telah ditandatangani menyiratkan pesan yang dikirim adalah asli.
c. Otentikasi (authentication), adalah layanan yang berhubungan dengan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (user authentication atau origin authentication). Dua pihak yang saling berkomunikasi harus dapat mengotentikasi satu sama lain sehingga ia dapat memastikan sumber pesan.

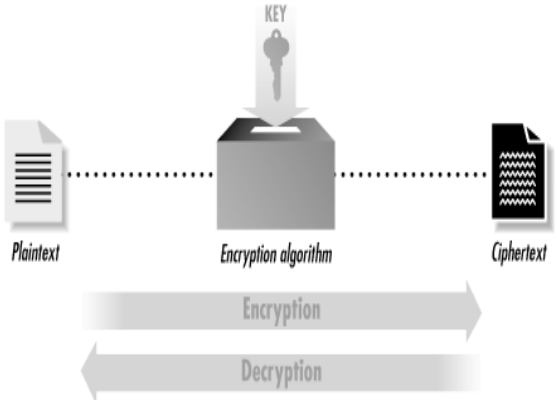
2.2. Jenis – Jenis Cryptosystem

Kriptografi membentuk sebuah sistem yang dinamakan sistem kriptografi. Sistem kriptografi (cryptographic system atau cryptosystem) adalah kumpulan yang terdiri dari algoritma kriptografi, semua plaintext dan ciphertext yang mungkin dan kunci. Di dalam sistem kriptografi, cipher hanyalah salah satu komponen saja.

Ada dua jenis cryptosystem, yakni : public-key cryptography dan secret-key cryptography. Dalam secret-key cryptography, kunci yang sama digunakan pada saat enkripsi maupun dekripsi. Sedangkan dalam public-key cryptography, masing-masing user memiliki satu public-key (kunci umum) dan satu private-key (kunci rahasia). Kunci publik dapat diumumkan (dibuat publik) tetapi kunci rahasia tetap dirahasiakan. Enkripsi dilakukan dengan menggunakan kunci publik sedangkan dekripsi dilakukan dengan mengunakan kunci rahasia.

2.2.1 Secret-Key Cryptography (Symmetric Cryptosystem)

Ini adalah jenis kriptografi yang paling umum dipergunakan. Kunci untuk membuat pesan yang disandikan sama dengan kunci untuk membuka pesan yang disandikan itu. Jadi pembuat pesan dan penerimanya harus memiliki kunci yang sama persis. Siapapun yang memiliki kunci tersebut, termasuk pihak-pihak yang tidak diinginkan, dapat membuat dan membongkar rahasia ciphertext. Problem yang paling jelas disini terkadang bukanlah masalah pengiriman ciphertext-nya, melainkan masalah bagaimana menyampaikan kunci simetris tersebut kepada pihak yang diinginkan. Ilustrasi dari proses enkripsi dan dekripsi dari secret-key cryptography dapat dilihat pada gambar 2.1 berikut:



Gambar 2.1 Enkripsi dan Dekripsi dengan Secret-key Cryptography

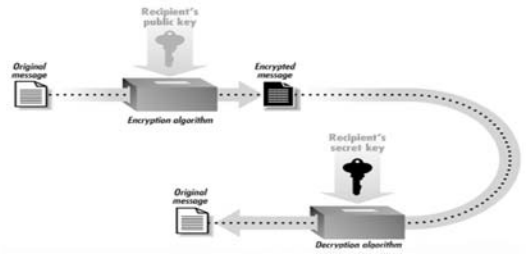
Algoritma kunci-simetri mengacu pada metode enkripsi yang dalam hal ini baik pengirim maupun penerima memiliki kunci yang sama. Algoritma kunci-simetri modern beroperasi dalam mode bit dan dapat dikelompokkan menjadi dua kategori:

a. Cipher aliran (stream cipher)
Algoritma kriptografi beroperasi pada plaintext / ciphertext dalam bentuk bit tunggal, yang dalam hal ini rangkaian bit dienkripsikan / didekripsikan bit per bit. Cipher aliran mengenkripsi satu bit setiap kali.

b. Cipher blok (block cipher)
Algoritma kriptografi beroperasi pada plaintext / ciphertext dalam bentuk blok bit, yang dalam hal ini rangkaian bit dibagi menjadi blok-blok bit yang panjangnya sudah ditentukan sebelumnya. Misalnya panjang blok adalah 64 bit, maka itu berarti algoritma enkripsi memperlakukan 8 karakter setiap kali enkripsi (1 karakter = 8 bit dalam pengkodean ASCII). Cipher blok mengenkripsi satu blok bit setiap kali.

2.2.2 Public Key Cryptography (Assymmetric Cryptosystem)

Dalam kriptografi tradisional, pengirim dan penerima pesan mengetahui dan menggunakan kunci rahasia yang sama, pengirim menggunakan kunci tertentu untuk mengenkripsi pesan, dan penerima menggunakan kunci yang sama untuk melakukan dekripsi terhadap pesan yang dia terima. Metode seperti ini dikenal dengan nama secret-key atau symmetric cryptography. Persoalan utama dalam hal ini adalah menyepakati kunci yang digunakan antara pengirim dan penerima pesan tanpa diketahui atau dicuri oleh seseorang yang kebetulan menguping. Jika keduanya berada pada dua lokasi yang berbeda secara fisik, mereka harus mempercayai pembawa pesan (courier), sistem telepon, media transmisi lainnya untuk tidak memberitahu atau meletakkan kunci sembarangan yang memungkinkan orang lain dapat menggunakannya. Seseorang yang secara kebetulan mendengar atau memotong (intercept) kunci dalam perjalanan (transit) dapat membaca, mengubah, dan memalsukan sebagian atau keseluruhan pesan tersebut untuk kepentingan pribadi dengan menggunakan kunci yang dia peroleh. Proses untuk menghasilkan, transmisi dan menyimpan kunci disebut dengan manajemen kunci (key management); semua kriptosistem (cryptosystems) harus dan selalu berhubungan dengan masalah-masalah manajemen kunci. Karena semua kunci dalam secret-key cryptosystem harus tetap rahasia (secret), maka secret-key cryptography sering mengalami kesulitan dalam menyediakan atau menjamin manajemen kunci yang aman, terutama dalam sebuah sistem yang terbuka dengan pengguna yang sangat banyak dan sangat bervariasi. Ilustrasi dari proses enkripsi dan dekripsi dari public-key cryptography dapat dilihat pada gambar 2.2 berikut:



Gambar 2.2 Enkripsi dan Dekripsi dengan Public-key Cryptography

2.3 Aplikasi Kriptografi

Kriptografi telah banyak digunakan dalam aplikasi-aplikasi khususnya aplikasi pengamanan data pada saat sekarang ini. Aplikasi khas dari kriptografi adalah sistem yang dikembangkan dengan teknik dasar. Sistem seperti ini dapat memiliki tingkat kompleksitas yang beraneka ragam. Beberapa aplikasi dari kriptografi dapat dilihat pada perincian berikut ini:

2.3.1 Komunikasi yang aman (Secure Communication)

Komunikasi yang aman merupakan penggunaan kriptografi yang paling sederhana. Dua pihak dapat berkomunikasi secara aman dengan cara mengenkripsi pesan-pesan yang mereka kirimkan di antara mereka. Hal ini dapat dicapai sedemikian rupa sehingga pihak ketiga yang mendapat bocoran (menyadap) pembicaraan antar kedua pihak tadi mungkin tidak pernah mengembalikan pesan dalam bentuk acak ke dalam bentuk yang berarti.

2.3.2 Identifikasi dan Otentikasi (Identification and Authentication)

Identifikasi (Identification) adalah proses verifikasi identitas seseorang atau sesuatu. Sebagai contoh, ketika menarik uang dari bank, petugas (teller) meminta tanda pengenal (misalnya SIM, KTP, dan lain-lain) untuk memverifikasi identitas pemilik rekening dari mana uang ditarik. Proses yang sama dapat dilakukan secara elektronik dengan menggunakan kriptografi. Setiap kartu Automatic Teller Machine (ATM) dilengkapi dengan satu Personal Identification Number (PIN) yang rahasia yang memadukan pemilik kartu dengan kartu tersebut demikian juga dengan rekening yang bersangkutan. Ketika kartu dimasukkan ke dalam mesin ATM, mesin tersebut akan meminta pengguna kartu untuk memasukkan PIN. Jika PIN yang dimasukkan benar, mesin akan mengidentifikasikan orang tersebut benar sebagai pemilik kartu sehingga kepadanya diberikan akses.

Otentikasi (Authentication) merupakan sebuah istilah yang digunakan dalam pengertian yang luas. Secara tersirat kata tersebut mempunyai arti lebih dari sekedar menyampaikan ide yaitu bahwa alat tersebut

telah menyediakan jaminan bahwa informasi tidak dimanipulasi oleh pihak yang tidak mempunyai wewenang. Otentikasi bersifat spesifik dalam topik keamanan yang berusaha dicapai. Contohnya meliputi pengendalian akses, otentikasi entity, otentikasi pesan, integritas data, non-repudiation, dan otentikasi kunci.

Otentikasi merupakan salah satu hal yang paling penting dalam keamanan informasi. Hingga pertengahan tahun 1970-an, dipercaya bahwa kerahasiaan dan otentikasi terhubung secara erat. Dengan penemuan dari fungsi-fungsi hash dan digital signatures, disadari bahwa kerahasiaan dan otentikasi sebenarnya adalah masalah yang terpisah dan independen. Awalnya tidak kelihatan penting untuk memisahkan keduanya tetapi terdapat situasi dimana hal tersebut tidak hanya berguna tetapi juga penting. Contohnya, jika terdapat komunikasi dua pihak antara Lauren dan John yang berlangsung, dimana Lauren berada di suatu negara dan John di negara lainnya, Negara tuan rumah mungkin tidak memperbolehkan kerahasiaan dalam saluran komunikasi karena satu atau kedua negara mungkin ingin memonitor semua komunikasi. Namun, Lauren dan John ingin meyakinkan identitas masing-masing, dan juga integritas serta keaslian dari informasi yang mereka kirim dan mereka terima.

Skenario diatas menggambarkan beberapa aspek otentikasi yang independen. Jika Lauren dan John ingin meyakinkan identitas masing-masing, terdapat dua kemungkinan yang dapat dipertimbangkan yaitu:

a. Lauren dan John dapat berkomunikasi tanpa penundaan waktu. Berarti mereka berkomunikasi secara real time.
b. Lauren atau John dapat saling menukar pesan dengan penundaan waktu. Berarti pesan mungkin dirouting melalui jaringan yang berbeda, disimpan, dan disampaikan pada beberapa saat kemudian.

Dalam kemungkinan pertama, Lauren dan John akan memverifikasi identitas masing-masing secara real time. Hal ini dapat dicapai oleh Lauren dengan mengirimkan beberapa challenge kepada John dimana hanya John yang dapat meresponnya secara tepat. John dapat melakukan tindakan yang sama untuk mengidentifikasi Lauren. Jenis otentikasi ini secara umum dikenal sebagai otentikasi entity atau secara sederhana disebut identifikasi.

Dalam kemungkinan kedua, tidaklah tepat untuk mengirimkan challenge dan menunggu respon, dan selain itu jalur komunikasi mungkin hanya tersedia pada satu arah. Teknik yang berbeda sekarang diperlukan untuk meng-otentikasi keaslian pesan. Bentuk otentikasi ini disebut otentikasi keaslian data (data origin authentication).

2.3.3 Tanda Tangan Digital (Digital Signature)

Otentikasi pesan dengan fungsi hash memang berhasil melindungi kedua belah pihak yang saling bertukar pesan dari pihak ketiga. Tetapi, otentikasi pesan tidak bisa mencegah kemungkinan kedua belah pihak saling menyerang satu sama lain.

Sebagai contoh, A mengirim sebuah pesan yang dilengkapi dengan nilai hash yang dienkripsi dengan algoritma kunci simetris kepada B. Kemungkinan di bawah ini bisa terjadi :

a. B bisa memalsukan pesan lain dan mengaku bahwa pesan itu dikirim oleh A. B hanya perlu untuk membuat pesan lain dan menambahkan nilai hash yang dienkripsi dengan kunci yang diketahui kedua belah pihak.
b. A bisa menyangkal bahwa ia telah mengirimkan pesan kepada B karena mungkin saja B memalsukan pesan tersebut. Sama sekali tidak ada cara untuk membuktikan bahwa A memang mengirimkan pesan tersebut.

Pada situasi dimana tidak ada kepercayaan penuh antara pengirim dan penerima pesan, diperlukan suatu mekanisme yang lebih daripada sekedar otentikasi. Solusi yang paling menarik dari masalah ini adalah tanda tangan digital (digital signature). Tanda tangan digital adalah suatu mekanisme otentikasi yang memungkinkan pembuat pesan menambahkan sebuah kode yang bertindak sebagai tanda tangannya. Tanda tangan tersebut menjamin integritas dan sumber dari sebuah pesan.

Tanda tangan digital dalam banyak hal mirip dengan tanda tangan biasa. Ada beberapa fungsi dari tandatangan digital yaitu :

a. Untuk memverifikasi pembuat pesan serta waktu ditandatanganinya pesan tersebut.
b. Untuk mengotentikasi isi pesan pada waktu ditandatangani pesan tersebut.
c. Harus bisa diverifikasi oleh pihak ketiga untuk menyelesaikan perselisihan mengenai integritas dan sumber pesan.

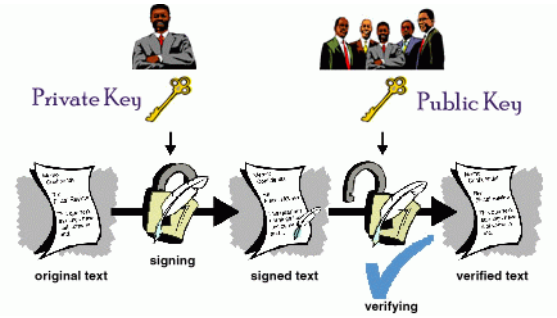
Dari fungsi-fungsi di atas maka dapat disusun persyaratan-persyaratan yang harus dipenuhi oleh sebuah tanda tangan digital yaitu :

a. Tanda tangan tersebut haruslah berupa pola bit yang tergantung pada pesan yang ditandatangani.
b. Tanda tangan tersebut harus menggunakan beberapa informasi yang menunjukkan pengirimnya, baik untuk mencegah pemalsuan pesan atau pengingkaran.
c. Relatif mudah untuk mengenali dan memverifikasi tanda tangan digital.
d. Tidak bisa secara komputasi untuk memalsukan sebuah tanda tangan digital, baik dengan mengkonstruksi sebuah pesan baru baru dari tanda tangan digital yang ada ataupun dengan mengkonstruksi

sebuah tanda tangan digital dengan menggunakan pesan yang diberikan.

e. Duplikat tanda tangan digital tersebut mudah disimpan.

Gambar proses kerja dari Digital Signature dapat dilihat pada gambar 2.3.



Gambar 2.3 Pembentukan tanda tangan digital

2.4 Protokol Pertukaran Kunci (Key Exchange Protocol)

Key exchange protocol, juga dikenal dengan key agreement protocol, adalah sebarisan langkah yang dilakukan bila dua atau lebih pihak perlu sepakat atas suatu kunci yang digunakan untuk suatu secret-key cryptosystem. Protokol ini memungkinkan orang menggunakan kunci secara bersama dengan bebas dan aman melalui suatu medium yang tidak aman, tanpa perlu terlebih dahulu ada pembentukan kunci rahasia bersama.

Misalkan Rio dan John ingin menggunakan satu secret-key cryptosystem untuk berkomunikasi secara aman. Mereka terlebih dahulu harus memutuskan apa kunci yang mereka gunakan. John bukannya menelepon Rio dan mendiskusikan apa kunci yang mereka gunakan, yang bisa saja didengar oleh pihak ketiga, mereka memutuskan menggunakan satu protokol kesepakatan kunci (key agreement protocol). Dengan menggunakan key agreement protocol, Rio dan John dapat saling mempertukarkan kunci dalam lingkungan yang tidak aman. Salah satu contoh protokol ini adalah Diffie-Hellman key agreement. Dalam beberapa kasus, public-key cryptography digunakan dalam suatu key agreement protocol. Contoh lainnya adalah penggunaan amplop dijital (digital envelopes) untuk key agreement.

2.5 Sandi Affine

Sandi Affine adalah substitution cipher yang dikhususkan untuk melakukan enkripsi atau dekripsi per karakter. Oleh karena itu, sandi Affine termasuk algoritma kriptografi stream cipher.

Di dalam sandi Affine, banyaknya alfabet / karakter yang digunakan disimbolkan dengan nilai dari 0 ... m-1. Dalam hal ini yang dienkripsi adalah byte, dengan nilai 0 … 255, sehingga nilai m yang digunakan adalah 256. Kemudian, metode ini menggunakan fungsi modulo aritmatika untuk mengubah nilai dari setiap karakter menjadi kode rahasia.

Fungsi enkripsi untuk setiap karakter adalah sebagai berikut:

$$E(p) = (a.p + b) \bmod m$$

dimana: p = plaintext

a dan b = kunci numerik yang digunakan.

m = 256 (maksimum nilai byte +1)

Syarat pemilihan nilai kunci adalah nilai a dan m harus relatif prima atau GCD(a, m) = 1, sedangkan nilai b dapat dipilih secara acak. Sedangkan, fungsi dekripsi untuk setiap karakter adalah:

$$D(c) = a^{-1}(c - b) \bmod m$$

Dimana :c = ciphertext, a dan b = kunci numerik yang digunakan.

m = 256 (maksimum nilai byte +1)

a^{-1} = invers dari perkalian modulo dengan m.

Nilai a^{-1} harus memenuhi ketentuan berikut:

$$1 = a \,.\, a^{-1} \bmod m$$

Contoh dari pembuktian persamaan di atas dapat dilihat pada perhitungan berikut:

a = 9663 m = 256

$a^{-1} \bmod 256 = 9663^{-1} \bmod 256 = 63$

(gunakan algoritma Extended Euclidean untuk menyelesaikan perhitungan $a^{-1} \bmod 256$)

Bukti: $1 = a \,.\, a^{-1} \bmod m$

$1 = ((a \bmod m) \,.\, (a^{-1} \bmod m)) \bmod m$

$1 = ((9663 \bmod 256) \,.\, (9663^{-1} \bmod 256)) \bmod 256$

$1 = (191 \,.\, 63) \bmod 256$

$1 = 12033 \bmod 256$

1 = 1 (Benar)

2.6 Operasi - Operasi Yang Digunakan Pada Sistem Kriptografi

2.6.1. Relatif Prima

Dua buah bilangan dikatakan relatif prima jika mereka tidak memiliki faktor prima yang sama kecuali 1. Dengan perkataan lain, jika Greatest Common Divisor (GCD) dari a dan n adalah sama dengan satu, maka a dan n adalah relatif prima. Bentuk ini dapat ditulis seperti berikut : GCD (a,n) = 1

Salah satu metode untuk menghitung GCD dari dua buah bilangan adalah dengan menggunakan algoritma Euclid yang ditulis dalam bukunya 'Elements' sekitar 300 tahun sebelum masehi. Algoritma ini bukan hasil rancangannya. Para ahli sejarah menyatakan bahwa algoritma mungkin 200 tahun lebih tua. Algoritma ini merupakan algoritma nontrivial tertua yang masih eksis di dunia. Knuth mendeskripsikan algoritma ini dengan beberapa modifikasi modern untuk menghitung GCD(a, b) seperti berikut:

A ← a
B ← b
While B > 0 do
  Q ← A / B
  R ← A – (Q * B)
  A ← B
  B ← R
End while
return A

II.6.2 Inverse Modular

Inverse merupakan operasi kebalikan. Inverse perkalian dari 4 adalah ¼ karena 4 * ¼ = 1. Dalam modulo, persoalan ini agak rumit. Misalkan

$4 * x \equiv 1 \pmod 7$

Atau bentuk penulisan ekivalen lainnya dapat dinyatakan dengan :

$4x = 7k + 1$

Dengan x dan k adalah bilangan integer.

Persoalannya adalah untuk mencari x sehingga,

$1 = (a * x) \bmod n$

Atau dapat ditulis seperti berikut :

$a^{-1} \equiv x \pmod n$

Persoalan inverse modular agak sulit untuk diselesaikan. Kadang-kadang dapat memiliki penyelesaian, dan kadang-kadang tidak. Sebagai contoh, inverse modular dari 5 modulo 14 adalah 3, sedangkan 2 mod 14 tidak memiliki inverse modular.

Secara umum, $a^{-1} \equiv x \pmod n$ memiliki sebuah solusi unik jika a dan n adalah relatif prima. Jika a dan n bukan relatif prima, maka $a^{-1} \equiv x \pmod n$ tidak memiliki solusi. Jika n adalah bilangan prima maka setiap bilangan dari 1 sampai (n – 1) adalah relatif prima dengan n dan memiliki tepat satu inverse modulo n dalam range tersebut.

Untuk menghitung inverse modular dari $a^{-1} \equiv x \pmod n$ dapat digunakan algoritma extended Euclidean yang dapat dijabarkan seperti berikut :

a. Bentuk Array A[3x2] dimana A[1,1] = n dan A[1,2] = a
b. Isikan A[2,1] .. A[3,2] dengan matriks Identitas.
c. Hitung bilangan bulat m dengan kondisi m * A[1,2] ≤ A[1,1]; usahakan m maksimum.
d. Hitung nX = A[1,1] - m * A[1,2].
e. Ubah nilai A[1,1] = A[1,2] dan A[1,2] = nX.
f. Lakukan langkah 4 dan 5 untuk baris kedua dan ketiga dari array A.
g. Ulang langkah 3 sampai 5 hingga elemen terakhir dari baris 1 = 0.
h. Jika A[3,1] ≥ 0 maka x = A[3,1] sebaliknya x = A[3,1] + n.

III. PEMBAHASAN

Untuk melakukan enkripsi dan dekripsi terhadap file Microsoft Office dengan menggunakan sandi Affine langkah – langkah yang dilakukan adalah sebagai berikut:

a. Proses pemilihan nilai kunci.
b. Proses untuk melakukan enkripsi file Microsoft Office dengan menggunakan sandi Affine.
c. Proses untuk melakukan dekripsi terhadap file Microsoft Office dengan menggunakan sandi Affine.

3.1. Proses Pemilihan Nilai Kunci

Terdapat tiga buah parameter di dalam sandi Affine, yaitu nilai a dan b sebagai kunci, dan nilai m sebagai jumlah karakter. Oleh karena, 1 byte di dalam sebuah file bernilai antara 0 sampai 255, maka nilai m ditetapkan sebesar 256. Nilai m = 256 akan dipakai pada proses enkripsi dan dekripsi, sehingga hasil operasi matematika modulo m tidak akan pernah melewati 255 atau batas maksimum nilai di dalam suatu byte.

Pada proses pemilihan kunci, nilai kunci a harus relatif prima terhadap m, atau dengan kata lain a dan m tidak boleh memiliki faktor yang sama, kecuali angka satu. Nilai GCD antara a dan m harus sama dengan satu.

Misalkan dipilih a = 4324 dan b = 1256, maka pemeriksaan faktor antara nilai a dan m adalah sebagai berikut:

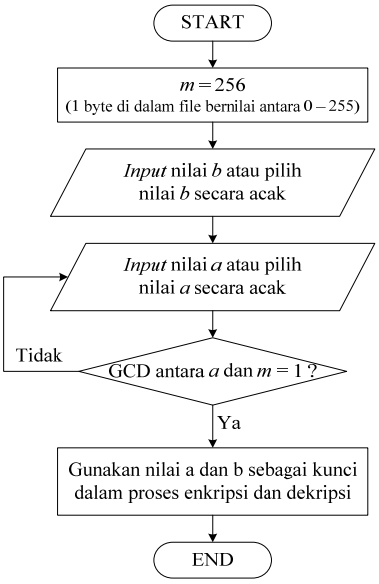
$a = 4324 = 2 \times 2 \times 23 \times 47 \times 1$
$= 2^2 \times 23 \times 47 \times 1$
$m = 256 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 1$
$= 2^8 \times 1$
GCD (a, m) $= 2^2 \times 1 = 4$ (Tidak relatif prima)

Oleh karena nilai a dan m memiliki nilai GCD tidak sama dengan satu, maka pemilihan nilai a tidak valid. Misalkan dipilih a = 9663 dan b = 1256, maka pemeriksaan faktor antara nilai a dan m adalah sebagai berikut:

$a = 9663 = 3 \times 3221 \times 1$
$m = 256 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 1 = 2^8 \times 1$
GCD (a, m) = 1 (Relatif Prima)

Oleh karena GCD antara nilai a dan m sama dengan satu, dan keduanya tidak mempunyai faktor yang sama kecuali angka satu, maka nilai a dan b

valid dan dapat dipilih sebagai kunci. Proses pemilihan nilai kunci dapat digambarkan dalam bentuk flowchart, seperti terlihat pada gambar 3.1 berikut :.



Gambar 3.1 Flowchart Proses Pemilihan Nilai Kunci

3.2 Proses Enkripsi

Proses ini digunakan untuk mengenkripsi file Microsoft Office, sehingga file Office tidak dapat dibuka dan diakses oleh pihak yang tidak berkepentingan. Proses enkripsi dilakukan pada setiap byte di dalam file dengan menggunakan fungsi enkripsi sandi Affine sebagai berikut:

$$E(p) = (a.p + b) \bmod m$$

dimana: m = maksimum nilai byte + 1 = 255 + 1 = 256

a dan b = kunci numerik yang digunakan

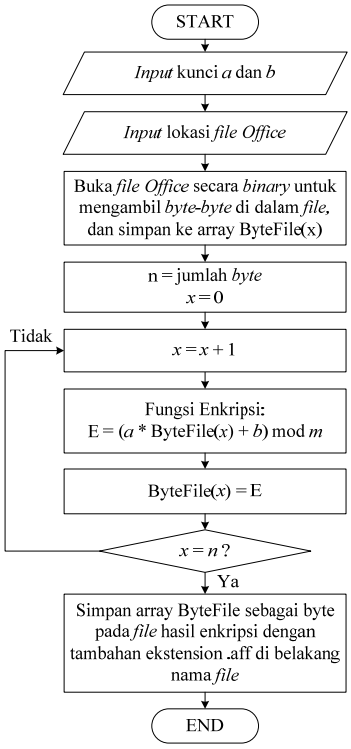
p = nilai byte

Sebagai contoh, digunakan kunci numerik a = 9663 dan b = 1256 sebagai kunci enkripsi, maka proses enkripsi terhadap file berukuran 5 byte adalah sebagai berikut:

Input file berukuran 5 byte sebagai berikut:
byte ke-1 = 80
byte ke-2 = 156
byte ke-3 = 115
byte ke-4 = 221
byte ke-5 = 40
Proses enkripsi terhadap masing-masing byte:
Byte ke-1 (p) = 80
E(p) = (a.p + b) mod m
E(p) = (9663 . 80 + 1256) mod 256
E(p) = (773040 + 1256) mod 256
E(p) = 774296 mod 256
E(p) = 152

Byte ke-2 (p) = 156
E(p) = (a.p + b) mod m
E(p) = (9663 . 156 + 1256) mod 256
E(p) = (1507428 + 1256) mod 256
E(p) = 1508684 mod 256
E(p) = 76

Byte ke-3 (p) = 115
E(p) = (a.p + b) mod m
E(p) = (9663 . 115 + 1256) mod 256
E(p) = (1111245 + 1256) mod 256
E(p) = 1112501 mod 256
E(p) = 181
Byte ke-4 (p) = 221
E(p) = (a.p + b) mod m
E(p) = (9663 . 221 + 1256) mod 256
E(p) = (2135523 + 1256) mod 256
E(p) = 2136779 mod 256
E(p) = 203
Byte ke-5 (p) = 40
E(p) = (a.p + b) mod m
E(p) = (9663 . 40 + 1256) mod 256
E(p) = (386520 + 1256) mod 256
E(p) = 387776 mod 256
E(p) = 192
Byte-byte hasil enkripsi secara berurutan adalah: 152 76 181 203 192
Proses enkripsi file Microsoft Office dengan menggunakan sandi Affine dapat dilihat pada gambar 3.2 berikut.



Gambar 3.2 Flowchart Proses Enkripsi

3.3 Proses Dekripsi

Proses dekripsi digunakan untuk mengembalikan file hasil enkripsi ke file Office semula. Proses dekripsi harus menggunakan kunci yang sama dengan proses enkripsi, agar dapat diperoleh file Office semula. Proses dekripsi menggunakan fungsi sebagai berikut:

$$D(c) = a^{-1}(c - b) \bmod m$$

dimana: a^{-1} = invers dari perkalian modulo dengan m.

m = maksimum nilai byte + 1 = 255 + 1 = 256

a dan b = kunci numerik yang digunakan.

c = nilai byte dari file hasil enkripsi

Sebagai contoh, digunakan kunci numerik a = 9663 dan b = 1256 sebagai kunci dekripsi, maka proses dekripsi akan menggunakan fungsi dekripsi sebagai berikut:

$D(c) = a^{-1} (c - b) \bmod m$

$D(c) = 9663^{-1} (c - 1256) \bmod 256$

$D(c) = ((9663^{-1} \bmod 256) . (c - 1256) \bmod 256) \bmod 256$

Pada tahapan ini, proses penyelesaian operasi invers $(9663^{-1} \bmod 256)$ harus menggunakan algoritma Extended Euclidean sebagai berikut:

Susun tabel awal

Tempatkan nilai 256 pada sel A(1,1) dan 9663 pada A(1,2). Lalu tempatkan matriks identitas [1, 0, 0, 1] di bawahnya, sehingga susunan tabel awal adalah sebagai berikut:

A[1,*]	256	9663
A[2,*]	1	0
A[3,*]	0	1

Bentuk kolom baru dan lakukan perhitungan untuk kolom tersebut sebagai berikut:

256 div 9663 = 0

			0	→ 256 div 9663 = 0
A[1,*]	256	9663		
A[2,*]	1	0		
A[3,*]	0	1		

256 – (9663 x 0) = 256

			0	
A[1,*]	256	9663	256	→ 256 – (9663 x 0) = 256
A[2,*]	1	0		
A[3,*]	0	1		

1 – (0 x 0) = 1

			0	
A[1,*]	256	9663	256	
A[2,*]	1	0	1	→ 1 – (0 x 0) = 1
A[3,*]	0	1		

0 – (1 x 0) = 0

			0	
A[1,*]	256	9663	256	
A[2,*]	1	0	1	
A[3,*]	0	1	0	→ 0 – (1 x 0) = 0

Bentuk kolom baru dan lakukan perhitungan untuk kolom tersebut sebagai berikut:

256 div 9663 = 0

			0	37	→ 9663 div 256 = 37
A[1,*]	256	9663	256		
A[2,*]	1	0	1		
A[3,*]	0	1	0		

9663 – (256 x 37) = 191

			0	37
A[1,*]	256	9663	256	191
A[2,*]	1	0	1	
A[3,*]	0	1	0	

→ 9663 – (256 x 37) = 191

0 – (1 x 37) = -37

			0	37
A[1,*]	256	9663	256	191
A[2,*]	1	0	1	-37
A[3,*]	0	1	0	

→ 0 – (1 x 37) = -37

1 – (0 x 37) = 1

			0	37
A[1,*]	256	9663	256	191
A[2,*]	1	0	1	-37
A[3,*]	0	1	0	1

→ 1 – (0 x 37) = 1

Bentuk kolom baru dan lakukan perhitungan untuk kolom tersebut sebagai berikut:

256 div 191 = 1

			0	37	1
A[1,*]	256	9663	256	191	
A[2,*]	1	0	1	-37	
A[3,*]	0	1	0	1	

→ 256 div 191 = 1

9663 – (256 x 37) = 191

			0	37	1
A[1,*]	256	9663	256	191	65
A[2,*]	1	0	1	-37	
A[3,*]	0	1	0	1	

→ 256 – (191 x 1) = 65

1 – (-37 x 1) = 38

			0	37	1
A[1,*]	256	9663	256	191	65
A[2,*]	1	0	1	-37	38
A[3,*]	0	1	0	1	

→ 1 – (-37 x 1) = 38

0 – (1 x 1) = -1

			0	37	1
A[1,*]	256	9663	256	191	65
A[2,*]	1	0	1	-37	38
A[3,*]	0	1	0	1	-1

→ 0 – (1 x 1) = -1

Bentuk kolom baru dan lakukan perhitungan untuk kolom tersebut sebagai berikut:

191 div 65 = 1

			0	37	1	2
A[1,*]	256	9663	256	191	65	
A[2,*]	1	0	1	-37	38	
A[3,*]	0	1	0	1	-1	

191 div 65 = 2

191 – (65 x 2) = 61

			0	37	1	2
A[1,*]	256	9663	256	191	65	61
A[2,*]	1	0	1	-37	38	
A[3,*]	0	1	0	1	-1	

191 – (65 x 2) = 61

-37 – (38 x 2) = -113

			0	37	1	2
A[1,*]	256	9663	256	191	65	61
A[2,*]	1	0	1	-37	38	-113
A[3,*]	0	1	0	1	-1	

-37 – (38 x 2) = -113

1 – (-1 x 2) = 3

			0	37	1	2
A[1,*]	256	9663	256	191	65	61
A[2,*]	1	0	1	-37	38	-113
A[3,*]	0	1	0	1	-1	3

1 - (-1 x 2) = 3

Bentuk kolom baru dan lakukan perhitungan seperti sebelumnya, sehingga tabel menjadi:

			0	37	1	2	1
A[1,*]	256	9663	256	191	65	61	4
A[2,*]	1	0	1	-37	38	-113	151
A[3,*]	0	1	0	1	-1	3	-4

Bentuk kolom baru dan lakukan perhitungan seperti sebelumnya, sehingga tabel menjadi:

			0	37	1	2	1	15
A[1,*]	256	9663	256	191	65	61	4	1
A[2,*]	1	0	1	-37	38	-113	151	-2378
A[3,*]	0	1	0	1	-1	3	-4	63

Pada perhitungan berikutnya, baris pertama mendapatkan nilai 0, sehingga perhitungan berhenti dan hasil dari perhitungan adalah nilai pada kolom terakhir dan baris terakhir yaitu 63, seperti terlihat pada tabel 3.1.

Tabel 3.1 Extended Euclidean

			0	37	1	2	1	15	4
A[1,*]	256	9663	256	191	65	61	4	1	0
A[2,*]	1	0	1	-37	38	-113	151	-2378	
A[3,*]	0	1	0	1	-1	3	-4	63	

Dengan demikian, perhitungan 9663-1 mod 256 = 63.
Sebagai contoh, digunakan kunci numerik a = 9663 dan b = 1256 sebagai kunci dekripsi, maka proses dekripsi terhadap file berukuran 5 byte adalah sebagai berikut:
Input file berukuran 5 byte setelah terenkripsi sebagai berikut: 152 76 181 203 192
Proses dekripsi terhadap masing-masing byte:

Byte ke-1 (c) = 152

$D(c) = a^{-1} (c – b) \bmod m$
$D(c) = 9663^{-1} (152 – 1256) \bmod 256$
$D(c) = ((9663^{-1} \bmod 256) . (152 – 1256) \bmod 256) \bmod 256$
$D(c) = (63 . 176) \bmod 256$
$D(c) = 11088 \bmod 256$
$D(c) = 80$

Byte ke-2 (c) = 76

$D(c) = a^{-1} (c – b) \bmod m$
$D(c) = 9663^{-1} (76 – 1256) \bmod 256$
$D(c) = ((9663^{-1} \bmod 256) . (76 – 1256) \bmod 256) \bmod 256$
$D(c) = (63 . 100) \bmod 256$
$D(c) = 6300 \bmod 256$
$D(c) = 156$

Byte ke-3 (c) = 181

$D(c) = a^{-1} (c – b) \bmod m$
$D(c) = 9663^{-1} (181 – 1256) \bmod 256$
$D(c) = ((9663^{-1} \bmod 256) . (115 – 1256) \bmod 256) \bmod 256$
$D(c) = (63 . 205) \bmod 256$
$D(c) = 12915 \bmod 256$
$D(c) = 115$

Byte ke-4 (c) = 203

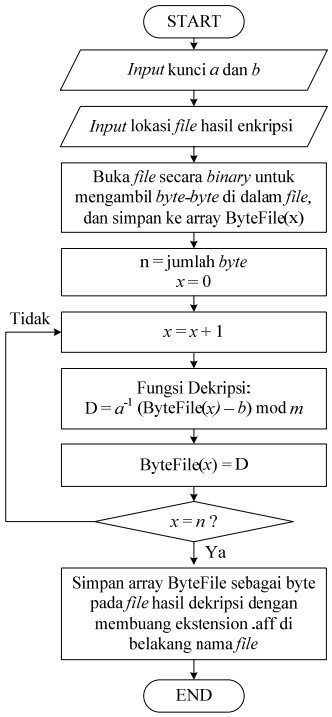
$D(c) = a^{-1} (c – b) \bmod m$
$D(c) = 9663^{-1} (203 – 1256) \bmod 256$
$D(c) = ((9663^{-1} \bmod 256) . (203 – 1256) \bmod 256) \bmod 256$
$D(c) = (63 . 227) \bmod 256$
$D(c) = 14301 \bmod 256$
$D(c) = 221$

Byte ke-5 (c) = 192

$D(c) = a^{-1} (c – b) \bmod m$
$D(c) = 9663^{-1} (192 – 1256) \bmod 256$
$D(c) = ((9663^{-1} \bmod 256) . (203 – 1256) \bmod 256) \bmod 256$
$D(c) = (63 . 216) \bmod 256$
$D(c) = 13608 \bmod 256$
$D(c) = 40$

File hasil dekripsi adalah sebagai berikut: 80 156 115 221 40

Pada contoh di atas, 5 byte hasil dekripsi (80 156 115 221 40) sama dengan byte dari file asli. Proses dekripsi berhasil mengembalikan nilai byte seperti semula dengan menggunakan kunci yang sama. Proses dekripsi dapat digambarkan dalam bentuk flowchart, seperti terlihat pada gambar 3.3 berikut.



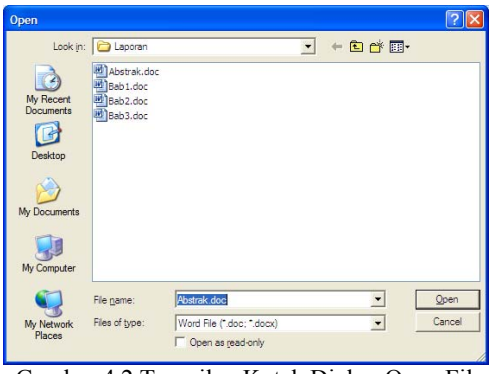
Gambar 3.3 Flowchart Proses Dekripsi

IV. Implementasi

1. Tampilan form Utama seperti terlihat pada gambar 4.1 berikut :

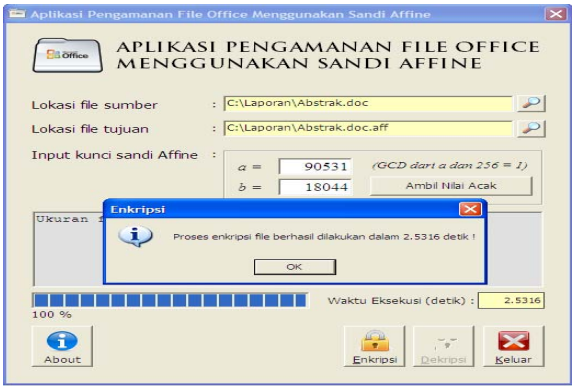
Gambar 4.1 Tampilan Form Utama

2. Untuk memilih *file* sumber untuk dienkripsi, maka tekan tombol ‘ ’ di samping *textbox* lokasi file sumber, dan kotak dialog *Open File* akan tampil seperti terlihat pada gambar 4.2 berikut :



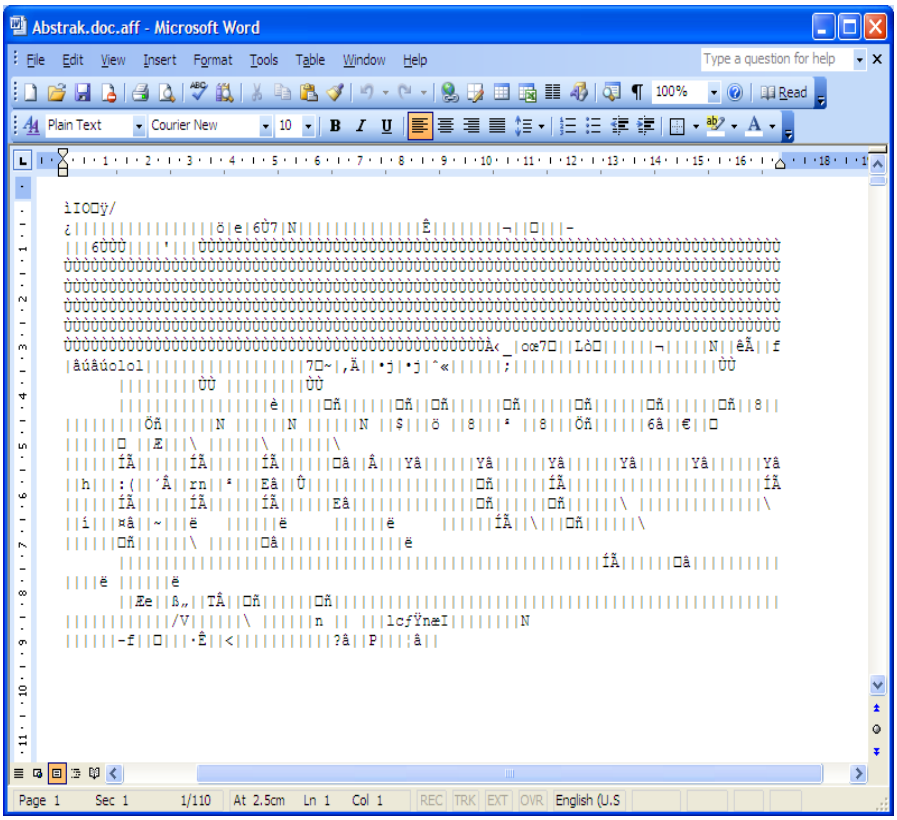
Gambar 4.2 Tampilan Kotak Dialog Open File

3. Misalkan, dipilih *file* ‘Abstrak.doc’ untuk dienkripsi, tekan tombol ‘Ok’, maka kotak dialog akan tertutup dan kembali ke *form* Utama. Pada *textbox* lokasi *file* tujuan, akan terisi secara otomatis nama *file* baru, berupa nama *file* sumber ditambah dengan ekstension .aff. Pilih nilai kunci *a* dan *b* yang memenuhi syarat GCD(a, 256) = 1, dan tekan tombol ‘Enkripsi’ untuk melakukan proses enkripsi terhadap *file* ‘Abstrak.doc’. Proses enkripsi *file* dapat dilihat pada gambar 4.3 berikut :



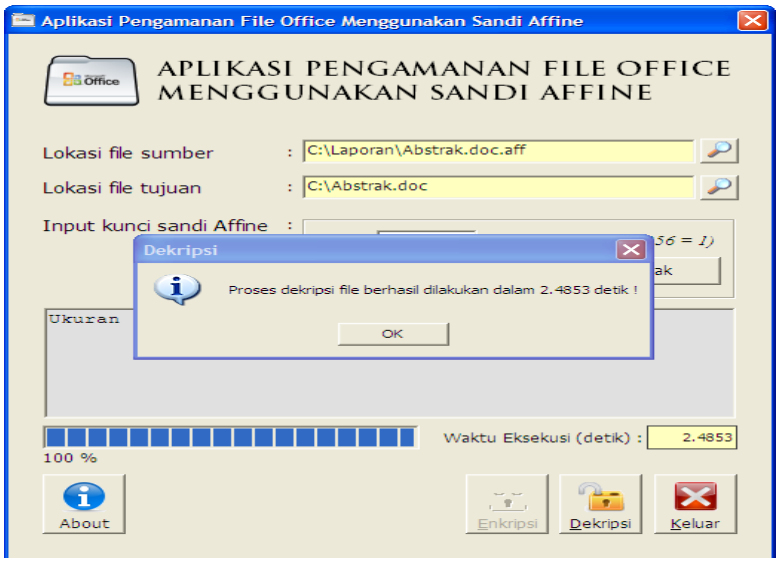
Gambar 4.3 Tampilan Proses Enkripsi

4. Hasil enkripsi dari *file* ‘Abstrak.doc’, yang berukuran 48640 *bytes* adalah ‘Abstrak.doc.aff’ dengan ukuran yang sama, yaitu 48640 *bytes* dengan waktu proses 2.5316 detik. Pada proses enkripsi, ukuran *file* hasil enkripsi tidak berubah, karena sandi *Affine* hanya melakukan penggantian nilai *byte*, tanpa menambah *byte* yang baru. Apabila *file* hasil enkripsi ‘Abstrak.doc.aff’ dibuka dengan Microsoft Word, maka kode-kode teracak akan ditampilkan, seperti terlihat pada gambar 4.4. Ini dikarenakan file ‘Abstrak.doc’ telah terenkripsi.



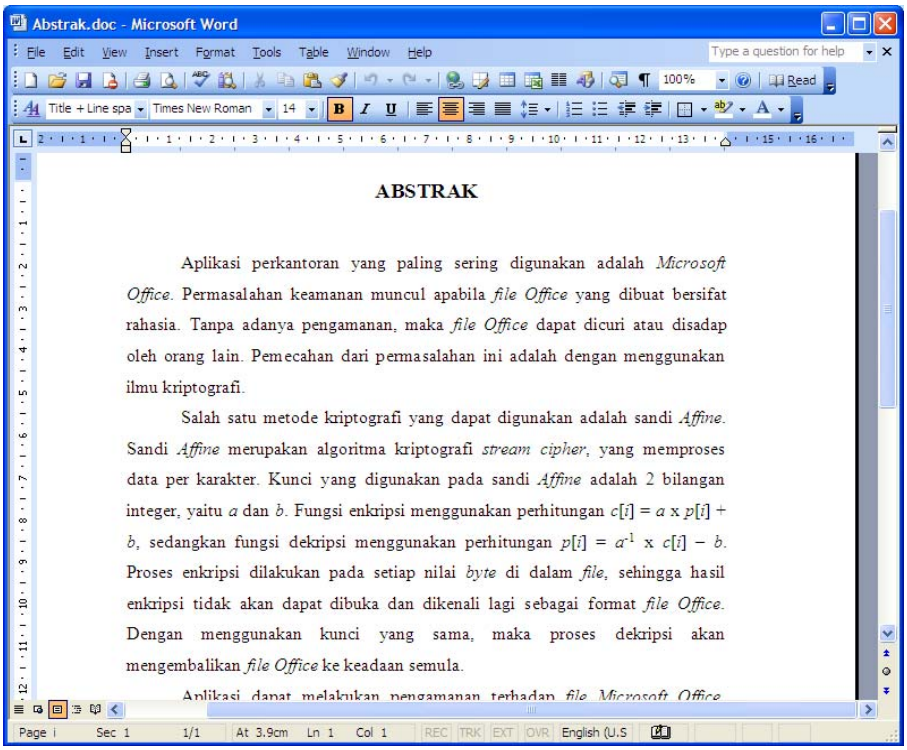
Gambar 4.4 File Terenkripsi 'Abstrak.doc.aff' dibuka dengan Ms. Word

5. Untuk mengembalikan *file* terenkripsi ke bentuk semula, maka lakukan proses dekripsi dengan menggunakan kunci yang sama. Pada gambar 4.5, aplikasi berhasil mendekripsi *file* 'Abstrak.doc.aff' menjadi *file* 'Abstrak.doc', dengan ukuran yang sama yaitu 48640 *bytes* dan membutuhkan waktu proses selama 2.4853 detik.



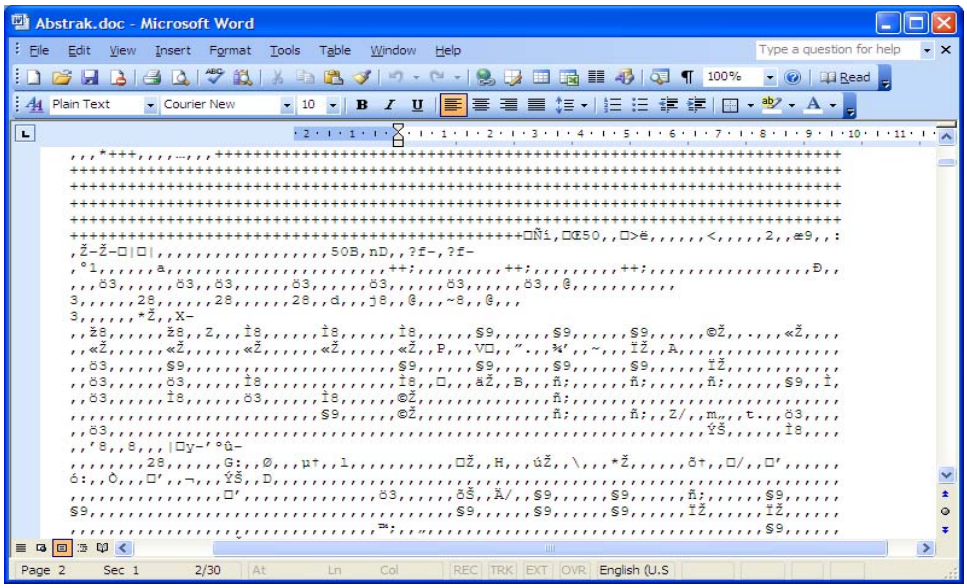
Gambar 4.5 Proses Dekripsi

6. Apabila *file* hasil dekripsi, yaitu 'Abstrak.doc' dibuka dengan Microsoft Word, maka hasilnya dapat dilihat pada gambar 4.6



Gambar 4.6 File Hasil Dekripsi 'Abstrak.doc' dibuka dengan Ms. Word

7. Hasil dekripsi pada gambar 4.7 menggunakan kunci a = 90531 dan b = 18044, atau kunci yang sama dengan proses dekripsi. Sebagai pengujian, apabila proses dekripsi menggunakan kunci yang salah atau tidak sama dengan kunci pada proses enkripsi, sebagai contoh a = 90531 dan b = 18040, maka *file* hasil dekripsi masih tetap teracak, seperti terlihat pada gambar 4.8.



Gambar 4.7 File Hasil Dekripsi dengan Kunci yang Salah

8. Dari implementasi aplikasi, dapat disimpulkan bahwa aplikasi dapat mengamankan *file Office* dengan proses enkripsi, dan berhasil mengembalikan *file* terenkripsi melalui proses dekripsi dengan kunci yang sama, tanpa mengubah ukuran *file*.

V. Kesimpulan dan Saran

Kesimpulan :

1. Proses enkripsi berhasil mengacak *file Office* sehingga *file* tidak dapat dibuka dengan benar, dan proses dekripsi berhasil mengembalikan *file Office* seperti semula.
2. Proses enkripsi dan dekripsi tidak mengubah ukuran *file*, karena sandi *Affine* hanya mengganti nilai *byte* dari *file*, tanpa menambah *byte* yang baru.
3. Penggunaan kunci yang salah akan menyebabkan hasil dekripsi tetap berada dalam keadaan teracak.

Saran :

1. Aplikasi dapat ditambahkan proses enkripsi dan dekripsi terhadap teks, disertai dengan langkah-langkah perhitungan, sehingga dapat membantu pembelajaran sandi *Affine* dalam proses enkripsi dan dekripsi.
2. Aplikasi dapat dikembangkan untuk dapat mengenkripsi semua jenis *file*.

DAFTAR PUSTAKA

Ariyus, D., Pengantar Ilmu Kriptografi (Teori,

Analisis dan Implementasi), Andi, Yogyakarta, 2008.

Kurniawan, J., Kriptografi, Keamanan Internet dan Jaringan Komunikasi, Informatika, Bandung, 2004.

Munir, R., Kriptografi, Informatika, Bandung, 2006.

Sadikin, R., Kriptografi untuk Keamanan Jaringan, Penerbit Andi, Yogyakarta, 2012.

Schneier, B., Applied Crytography, Second Edition, John Willey and Sons Inc, 1996.

Bender. 1996. Techniques For Data Hiding. IBM Systems Journal.

Hariyanto, B. 2009. Sistem Operasi. Bandung : Informatika.

Lusiana, V. 2011. Implementasi Kriptografi Pada File Dokumen Menggunakan Algoritma AES-128. Jurnal Dinamika Informatika.

Munir, R. 2006. Kriptografi. Bandung : Penerbit Informatika.

Stallings, W. 2006. Cryptography and Network Security Principles and Practice. Fifth Editon. USA : Prentice Hall.