Implementasi Algoritma MAC Berbasis *Cipher* Blok Sebagai Program *Add-in* di *Microsoft Word* untuk Otentikasi Dokumen

Yudha Adiprabowo – 13506050
Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha No. 10 Bandung
if16050@students.if.itb.ac.id

Dr. Ir. Rinaldi Munir, M.T. - 132084796
Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha No. 10 Bandung
rinaldi-m@stei.itb.ac.id

Abstrak - Salah satu metode untuk mengamankan dokumen secara digital adalah Message Authentication Code (MAC). MAC digunakan untuk tujuan otentikasi dokumen tanpa harus merahasiakan atau mengenkripsi data dokumen tersebut. MAC juga digunakan untuk menjamin integritas (keaslian) isi arsip terhadap perubahan dari pihak ketiga. Pada tugas akhir ini dilakukan implementasi salah satu algoritma MAC berbasis *cipher* blok yaitu *Cipher-based Message Authentication Code* (CMAC). CMAC menggunakan algoritma *cipher* blok berbentuk *Advanced Encryption Standard* (AES). Implementasi dilakukan dalam bentuk program *add-in* pada aplikasi pengolah kata *Microsoft Word* supaya bisa melakukan otentikasi pada dokumen *Word* yang sedang dibuka.

***Add-in* memiliki kemampuan untuk membangkitkan nilai MAC dari dokumen yang sedang dibuka lalu menyimpannya di akhir dokumen ataupun di *file* eksternal. Setelah itu *add-in* juga memiliki kemampuan untuk membaca nilai MAC yang sebelumnya disimpan baik di akhir dokumen walaupun di *file* eksternal. Pada akhirnya, *add-in* mampu melakukan validasi nilai MAC dengan membandingkan nilai MAC yang dibangkitkan dari dokumen dan nilai MAC yang sudah tersimpan sebelumnya.**

Berdasarkan hasil pengujian, dapat disimpulkan bahwa nilai MAC yang dibangkitkan oleh *add-in* dapat digunakan untuk otentikasi dokumen Word.

Kata kunci: MAC, *cipher* blok, otentikasi dokumen, *add-in Microsoft Word*, CMAC, AES.

I. PENDAHULUAN

Penyampaian informasi dari satu pihak ke pihak lain merupakan hal yang vital dalam dunia informatika. Teknologi informasi memungkinkan pengiriman pesan tanpa harus bertatap muka dari dua tempat yang jauh berbeda. Ada kelemahan dari penyampaian pesan melalui teknologi informasi yaitu keamanan yang kurang terjamin. Informasi dalam pesan yang terkirim haruslah aman dari orang-orang yang hendak membaca, mengubah, atau menghapus tanpa izin. Walau begitu, dalam penyampaian pesan ini ada kemungkinan pihak ketiga mampu menangkap, membuka, dan mengubah pesannya. Untuk melindungi pesan, diperlukanlah metode-metode untuk mengamankan data pesan secara digital. Salah satu metode untuk mengamankan data pesan secara digital adalah *Message Authentication Code* (MAC).

MAC digunakan untuk tujuan otentikasi data tanpa harus merahasiakan atau mengenkripsi arsip data tersebut. MAC juga digunakan untuk menjamin integritas (keaslian) isi arsip terhadap perubahan dari pihak ketiga. Prinsip kerja MAC adalah menambahkan kata kunci ke suatu hasil penghitungan dari keseluruhan pesan, lalu menambahkannya ke akhir pesan atau bisa juga secara terpisah. MAC ini kemudian bisa diperiksa oleh pengirim dan penerima pesan selama kedua pihak mengetahui metode MAC dan kata kunci yang mereka gunakan. Jika hasil MAC yang dihitung penerima berbeda dengan MAC asli, berarti ada perubahan yang dilakukan terhadap data pesan tersebut.

Terdapat dua pendekatan untuk merancang algoritma MAC. Yang pertama adalah MAC berbasis *cipher* blok, yaitu metode enkripsi teks yang kunci

kriptografis dan algoritmanya diterapkan langsung dalam suatu blok data dan tidak dilakukan per bit. Yang kedua adalah MAC berbasis fungsi *hash* satu-arah. Fungsi *hash* satu arah merupakan sebuah prosedur deterministik yang mengambil suatu blok data dan mengembalikan suatu *string* dengan panjang bit tetap yang disebut sebagai nilai *hash*.

Di dalam Tugas Akhir ini dilakukan pendekatan implementasi dengan *Cipher-based* MAC (CMAC) yang merupakan salah satu algoritma MAC berbasis *cipher* blok. Implementasi dilakukan dalam bentuk program *add-in* dalam aplikasi pengolah kata *Microsoft Word* karena *Microsoft Word* merupakan aplikasi pengolah kata yang paling populer dan *add-in* untuk keperluan otentikasi dokumen masih belum ada dibuat. Sementara CMAC dipilih karena merupakan algoritma yang masih cukup aman dan cenderung baru, sehingga penerapannya belum banyak dilakukan. Selain itu, CMAC dirancang untuk mendeteksi modifikasi yang disengaja dan tidak sah dari data, serta modifikasi yang tidak sengaja (Dworkin, 2005).

II. DASAR TEORI

Pada makalah ini akan dibahas mengenai CMAC, AES, dan Program *Plug-in*.

1. *Cipher-based Message Authentication Code* (CMAC)

Berikut ini dipaparkan penjelasan rekomendasi untuk salah satu mode operasi *cipher* blok, yaitu mode CMAC untuk otentikasi yang diusulkan oleh Dworkin (2005). Inti dari algoritma *Cipher-based* MAC (CMAC) adalah sebuah variasi dari CBC-MAC yang diajukan dan dianalisis Black dan Rogaway dengan nama XCBC dan diajukan ke NIST (Black & Rogaway, dikutip oleh Dworkin, 2005). Algoritma XCBC secara efisien menunjukkan kurangnya keamanan CBC-MAC. Iwata dan Kurosawa kemudian mengajukan perbaikan dari XCBC dan menamakan hasilnya *One-key* CBC-MAC (OMAC) saat mengajukan ke NIST pertama kali. Belakangan, Iwata dan Kurosawa mengajukan OMAC1 yang merupakan perbaikan OMAC lebih lanjut dan tambahan analisis keamanan (Iwata & Kurosawa, dikutip oleh Dworkin, 2005). Variasi OMAC1 mengurangi ukuran kunci dari XCBC secara efisien. CMAC ekuivalen dengan OMAC1.

Algoritma CMAC tergantung dengan pilihan *cipher* blok kunci simetris yang digunakan. Algoritma CMAC dengan demikian merupakan salah satu mode operasi dari *cipher* blok. Kunci CMAC adalah kuncinya *cipher* blok.

Untuk setiap kunci yang diberikan, *cipher* blok yang mendasari mode ini terdiri dari dua fungsi yang merupakan *invers* satu sama lain. Pada *cipher* blok yang terpilih di dalamnya telah didesain dua fungsi. Satu fungsi sebagai fungsi maju / transformasi dan satu fungsi lagi sebagai fungsi *invers* seperti dalam spesifikasi algoritma AES (Advanced Encryption Standard) dan TDEA (Triple Data Encryption Algorithm). Mode CMAC tidak menggunakan fungsi *invers*.

Fungsi *cipher* maju adalah permutasi pada *string-string* bit dengan panjang tetap, dengan *string-string* tersebut disebut sebagai blok. Panjang bit blok dinotasikan dengan b, dan panjang blok itu disebut dengan ukuran blok. Untuk algoritma AES, b = 128; untuk TDEA, b = 64. Kunci diberi notasi K, dan hasil dari fungsi *cipher* maju dari *cipher* blok diberi notasi $CIPH_K$.

Cipher blok yang mendasari CMAC ini haruslah sudah menjadi standard atau sudah disetujui. Kunci yang digunakan juga haruslah dibangkitkan secara acak atau sangat mendekati acak sehingga setiap kunci yang sudah dibangkitkan hampir tidak mungkin untuk dibangkitkan persis sama lagi. Kuncinya harus dirahasiakan dan digunakan secara eksklusif pada mode CMAC dari cipher blok yang terpilih untuk dipakai. Agar kunci memenuhi syarat kerahasiaan, kuncinya harus ditetapkan antara pihak-pihak pengguna dengan struktur manajemen kunci yang berada di luar lingkup rekomendasi Dworkin (2005) maupun tugas akhir ini.

Kunci *cipher* blok digunakan untuk menurunkan 2 nilai rahasia tambahan yang dikenal sebagai subkunci dan dinotasikan sebagai K1 dan K2. Panjang setiap subkunci adalah sama dengan ukuran blok. Subkunci selalu tetap untuk pembangkitan CMAC dengan kunci yang diberikan. Sehingga, subkunci bisa

dihitung sebelumnya dan disimpan bersama kunci untuk pemakaian berulang-ulang.

Setiap nilai yang dihasilkan di antara perhitungan subkunci juga harus dijadikan rahasia, khususnya $CIPH_K(0^b)$. Persyaratan ini melarang sistem tempat CMAC diimplementasikan untuk menggunakan nilai perantara ini secara publik, baik untuk nilai tak terduga atau nilai pemeriksaan integritas pada kunci.

Salah satu elemen penting dalam proses pembangkitan subkunci adalah sebuah bit *string*, dinotasikan Rb, yang ditentukan dari jumlah bit dalam suatu blok. Secara khusus, untuk dua ukuran blok yang sudah disetujui untuk cipher blok adalah R_{128} = 012010000111 dan R_{64} = 05911011.

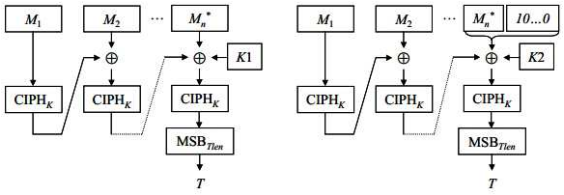
Proses pembangkitan kunci di CMAC adalah sebagai berikut. Langkah pertama, *cipher* blok diaplikasikan ke sebuah blok yang terdiri dari bit '0'. Langkah kedua, subkunci pertama diturunkan dari *string* hasilnya dengan left shift 1 bit ke kiri, dan tergantung kondisi, dilakukan XOR dengan konstanta yang ukurannya bergantung dengan ukuran blok. Di langkah ketiga, subkunci kedua diturunkan dengan cara yang sama dengan subkunci pertama.

Proses pembangkitan MAC dari CMAC adalah sebagai berikut. Pada langkah pertama, subkunci-subkunci dibangkitkan dari kunci. Di langkah 2, 3, dan 4, pesan masukan diformat menjadi urutan blok komplit yang blok finalnya sudah ditutupi (*masked*) oleh subkunci. Ada dua kasus:

1. Kalau panjang pesan adalah kelipatan yang bulat dari ukuran blok, maka pesan tersebut dipartisi menjadi blok-blok lengkap. Blok final ditutupi dengan subkunci pertama; dengan kata lain blok final di partisi diganti dengan hasil XOR antara blok final dengan subkunci pertama. Hasil urutan blok yang dihasilkan merupakan pesan yang telah diformat.
2. Kalau panjang pesan bukan merupakan kelipatan bulat dari ukuran blok, maka pesan dipartisi ke blok-blok lengkap sebisa mungkin dan diikuti oleh bit string final yang panjangnya kurang dari ukuran blok. Sebuah *string padding* ditambahkan ke bit *string* final, dimulai dengan sebuah bit '1' diikuti oleh sejumlah bit '0' seminimal mungkin, kalau bisa tidak ada, yang dibutuhkan untuk melengkapi sbuah blok. Blok final komplit ditutupi (*masked*) dengan subkunci kedua. Hasil urutan blok yang dihasilkan merupakan pesan yang telah diformat.

Pada langkah 5 dan 6, teknik *Cipher Block Chaining* (CBC) dengan blok nol sebagai vektor inisialisasi diterapkan pada pesan yang telah diformat. Pada langkah 7 dan 8, blok keluaran CBC final dipotong berdasarkan parameter panjang MAC yang berhubungan dengan kunci dan hasilnya dikembalikan sebagai MAC.

Kedua kasus dari pembangkitan MAC diilustrasikan pada Gambar 1, pada gambar di kiri menunjukkan kasus jika panjang pesan adalah kelipatan yang bulat dari ukuran blok. Gambar di kanan menunjukkan kasus jika panjang pesan bukan kelipatan yang bulat dari ukuran blok.



Gambar 1: Ilustrasi kedua kasus pembangkitan MAC pada CMAC (Dworkin, 2005)

2. *Advanced Encryption Standard* (AES)

Dalam pemakaian CMAC dapat dipilih antara dua algoritma *cipher* blok, yaitu antara AES atau *Triple Data Encryption Algorithm* (TDEA). Dalam tugas akhir ini, yang digunakan adalah AES, sehingga bisa dikatakan yang diimplementasikan dalam tugas akhir ini adalah algoritma AES-CMAC.

AES itu sendiri menggunakan substitusi dan permutasi dan sejumlah putaran (*cipher* berulang) – setiap putaran menggunakan kunci internal yang berbeda (kunci setiap putaran disebut *round key*). AES tidak berorientasi bit melainkan *byte*, dan tidak menggunakan jaringan Feistel (Munir, 2006).

Garis besar algoritma AES / Rijndael yang beroperasi pada blok 128-bit dengan kunci 128-bit dapat dijabarkan sebagai berikut:

1. *AddRoundKey*: melakukan XOR antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
2. Putaran sebanyak 9 kali untuk AES 128-bit. Proses yang dilakukan pada tiap putaran adalah:
   a. *SubBytes*: substitusi *byte* dengan menggunakan tabel substitusi (S-box).
   b. *ShiftRows*: pergeseran baris-baris *array state* secara *wrapping*.
   c. *MixColumns*: mengacak data di masing-masing kolom *array state*.
   d. *AddRoundKey*: melakukan XOR antara *state* sekarang dengan *round key*.
3. *Final round*: proses untuk putaran terakhir:
   a. *SubBytes*.
   b. *ShiftRows*.
   c. *AddRoundKey*.

3. Program *Plug-In*

Perangkat lunak *add-in* merupakan salah satu contoh dari *plug-in*. Program *plug-in* adalah sebuah tambahan untuk sebuah aplikasi yang menambahkan fungsionalitas aplikasi tersebut. Sebagai contoh, Photoshop *plug-in* (seperti *Eye Candy*) dapat menambahkan filter tambahan yang dapat digunakan untuk memanipulasi gambar. Sebuah *browser plug-in* (seperti *Macromedia Flash* atau *Apple QuickTime*) memungkinkan pengguna *browser* untuk memutar *file* multimedia tertentu. VST *plug-in* menambahkan efek untuk merekam audio dan aplikasi *sequencing* seperti Cubase dan Logic Audio.

Kebanyakan aplikasi grafis dan audio hari ini mendukung *plug-in* karena mereka adalah cara mudah untuk memperluas kemampuan aplikasi. Meskipun beberapa *plug-in* dapat dibundel dengan aplikasi, sebagian besar dikembangkan oleh pihak ketiga dan dijual secara terpisah. Karena perusahaan yang membuat browser *plug-in* sering bersaing untuk standar (seperti *Flash* dan *QuickTime*), *plug-in* ini biasanya tersedia sebagai *download* gratis dari internet. (TechTerms, 2012).

Alasan aplikasi-aplikasi untuk mendukung keberadaan *plug-in* bermacam-macam. Beberapa alasannya adalah:
a. Untuk memungkinkan pengembang pihak ketiga menciptakan kemampuan yang memperkaya aplikasi.
b. Untuk memudahkan ditambahnya fitur baru.
c. Untuk mengurangi ukuran aplikasi.
d. Untuk memisahkan *source code* dari aplikasi karena lisensi perangkat lunaknya tidak kompatibel.

Aplikasi *host* menyediakan layanan yang dapat digunakan *plug-in*, termasuk cara untuk *plug-in* untuk mendaftarkan diri dengan aplikasi *host* dan protokol untuk pertukaran data dengan *plug-in*. *Plug-in* tergantung pada layanan yang disediakan oleh aplikasi *host* dan biasanya tidak bekerja sendiri. Sebaliknya, aplikasi *host* beroperasi secara independen dari *plug-in*, sehingga memungkinkan bagi pengguna akhir untuk menambah dan meng-update *plug-in* secara dinamis tanpa perlu melakukan perubahan pada aplikasi *host*.

Dalam *Microsoft Office*, *add-in* bekerja dalam kerangka .NET *Framework* bawaan *Microsoft*. Salah satu aplikasi untuk mempermudah pembuatan *add-in* untuk *Microsoft Office* adalah *Microsoft Visual Studio*, dan *tools* khusus di aplikasi itu untuk membuat *add-in* disebut *Visual Studio Tools for Office* (VSTO).

III. IMPLEMENTASI

Masalah yang dibahas dalam tugas akhir ini adalah membangun sebuah program *add-in* untuk aplikasi pengolah kata *Microsoft Word* yang mampu melakukan otentikasi dokumen dengan mengimplementasikan algoritma CMAC. Dalam implementasi algoritma CMAC, ada tiga hal penting yang perlu diperhatikan yaitu pembangkitan nilai MAC, penyimpanan nilai MAC, dan validasi nilai MAC.

1. Pembangkitan Nilai MAC

Pembangkitan nilai MAC membutuhkan sebuah nilai kunci. Dalam rekomendasi Dworkin (2005) disarankan untuk menggunakan semacam fungsi *random* untuk membangkitkan kunci agar keamanan dari nilai MAC terjamin. Walau begitu, manajemen kunci dan keamanan kunci tidak dibahas dalam makalah ini. Oleh karena itulah diputuskan bahwa masalah nilai kunci cukup diselesaikan dengan mengambil nilai kunci dari masukan pengguna.

Nilai kunci masukan pengguna kemudian digunakan sebagai kunci bagi AES 128 bit yang digunakan untuk mengenkripsi blok '0', blok yang berisi bit 0 sebanyak 128 buah / blok kosong. Hasilnya disebut sebagai kunci. Setelah kunci dibangkitkan, dilakukan pembangkitan subkunci-subkunci seperti pada. Begitu kunci dan subkunci didapatkan, nilainya disimpan dan dilakukan langkah berikutnya.

Langkah berikutnya yang menjadi masalah adalah pengambilan pesan dari dokumen *Word* yang sedang dibuka. Ada beberapa metode untuk mengambil pesan dari dokumen *Word* ini, dan diputuskan untuk menggunakan metode paling sederhana, yaitu hanya mengambil nilai teks. Metode satu lagi adalah mengambil nilai *xml*. Jika hanya mengambil nilai teks maka banyak informasi dokumen tidak ikut diproses. Contohnya, bahasa di dokumen, *font* yang digunakan, *page margin* dan ukuran kertas di dokumen, *encoding charset* dari dokumen (standardnya adalah UTF-8), dan lain-lain.

Alasan tidak menggunakan metode pengambilan pesan berbentuk *xml* yang berisi informasi-informasi tambahan dari dokumen adalah adanya informasi yang cukup mengganggu dalam *xml* tersebut, yaitu *revision id*. *Revision id* adalah suatu nilai *random* unik yang berubah setiap kali pengguna menambahkan sesuatu pada dokumen *Word* yang sedang dibuka. Guna *revision id* adalah untuk menandai posisi-posisi di dokumen *Word* yang berubah dan nanti akan digunakan oleh *Microsoft Word* itu sendiri untuk fungsionalitas *Undo* dan *Redo*.

Pada akhirnya, pesan yang diproses hanya berisi teks. Pesan ini dihitung panjangnya, dan karena menggunakan AES, panjang pesan kita bagi dengan nilai b = 128 bit lalu dibulatkan. Pesan setelah itu dibagi-bagi menjadi blok terpisah dan diproses per blok dalam sebuah *loop*. Di dalam *loop* ini kita memproses suatu blok pesan dengan melakukan operasi XOR terhadap blok sebelumnya. Jika blok pesan ternyata adalah blok pertama, maka blok sebelumnya adalah blok nilai kunci.

Setelah satu selesai diproses, hasil proses ditetapkan sebagai blok sebelumnya sementara algoritma mengakses blok yang baru. Hal ini diulang terus menerus sampai akhirnya ke blok terakhir. Operasi untuk blok terakhir dibedakan karena pada blok terakhir dilakukan *padding* sejumlah bit yang tersisa jika blok tidak penuh. Pemakaian K1 atau K2 juga tergantung dari bloknya penuh atau tidak. Hasil dari pemrosesan blok terakhir inilah yang disebut sebagai nilai MAC. Pengguna bisa memilih untuk menyimpan nilai MAC di akhir dokumen atau di *file* terpisah.

2. Penyimpanan Nilai MAC

Ada dua cara penyimpanan MAC yang dapat diimplementasikan, yaitu penambahan nilai MAC pada akhir dokumen *Microsoft Word* atau penyimpanan MAC pada *file* terpisah di luar dokumen tersebut. Berhubung operasi yang dilakukan dalam algoritma CMAC merupakan operasi *byte*, MAC yang dihasilkan kemungkinan besar tidak bisa dibaca. Oleh karena itu, nilai MAC akan disimpan setelah dikonversi menjadi berbentuk *hexadecimal*. Tujuan konversi ini supaya MAC yang dihasilkan bisa dibaca, baik oleh pengguna ataupun oleh program *add-in*. Selain itu juga supaya tidak mengganggu dokumen dengan karakter-karakter aneh di akhir dokumen.

Penyimpanan nilai MAC baik di dalam maupun di luar dokumen akan diapit oleh *tag xml*, yaitu <AESCMAC128>. Alasan pengapitan nilai MAC dengan *tag xml* ini supaya nilai MAC mudah dibaca kembali oleh program *add-in* karena di algoritma bawaan program sudah tersedia algoritma untuk membaca *xml*. Selain itu, dengan diapit *tag xml* pengguna bisa langsung tahu bahwa *xml* tersebut berisi nilai MAC.

Untuk penyimpanan MAC di luar *file* dokumen, *file* terpisah ini disimpan dalam format *file* teks biasa dengan nama terserah masukan pengguna. Di dalam *file* ini hanya ada nilai MAC diapit *tag xml* sehingga bisa dibaca oleh program *add-in* untuk proses verifikasi MAC.

3. Validasi Nilai MAC

Untuk validasi nilai MAC di akhir dokumen, program dapat memisahkan sementara MAC yang sudah ditambahkan di akhir dokumen dan melakukan

penghitungan MAC pada dokumen yang MAC-nya telah dipisahkan. Untuk validasi nilai MAC dari luar dokumen, program juga dapat membuka MAC yang disimpan dalam *file* terpisah untuk dibandingkan dengan MAC yang baru dihitung dari dokumen. Berhubung dalam proses validasi algoritma pembangkitan MAC adalah sama, tidak perlu ada pembuatan algoritma baru.

Nilai MAC yang perlu divalidasi disimpan dan dibandingkan dengan nilai MAC yang dibangkitkan dari dokumen. Dalam proses validasi yang dibutuhkan hanyalah suatu fungsi pembanding antara dua nilai MAC yang mengembalikan nilai boolean VALID atau INVALID.

IV. PENGUJIAN

Pengujian yang dilakukan terhadap program *add-in* yang dibuat mencakup dua jenis pengujian, yaitu pengujian fungsional perangkat lunak dan pengujian kehandalan otentikasi. Hasil pengujian fungsional perangkat lunak dapat dilihat pada Tabel 1.

Tabel 1. Pengujian Fungsional Perangkat Lunak

Aksi	Hasil
Menambahkan MAC pada dokumen.	Berhasil
Menyimpan MAC pada file eksternal.	Berhasil
Menyimpan dokumen yang telah diberi MAC.	Berhasil
Membuka dokumen yang telah diberi MAC.	Berhasil
Membaca MAC yang sudah terdapat pada dokumen.	Berhasil
Membaca MAC yang terdapat pada file eksternal.	Berhasil
Melakukan otentikasi dokumen.	Berhasil

Sementara pengujian kehandalan otentikasi dapat dilihat pada Tabel 2.

Tabel 2. Pengujian Kehandalan Otentikasi

Aksi	Hasil
Menghapus satu huruf	MAC berubah
Menghapus satu kalimat.	MAC berubah
Menghapus satu paragraf.	MAC berubah
Menggunakan kunci MAC yang salah.	Validasi gagal
Mengubah MAC yang sudah ada secara manual.	Validasi gagal
Menghapus MAC yang sudah ada secara manual	*Add-in* tidak mendeteksi MAC
Merusak *tag xml* yang mengapit MAC.	*Add-in* tidak mendeteksi MAC
Menambahkan gambar.	MAC berubah
Menambahkan objek..	MAC berubah
Menambahkan fungsi.	MAC berubah

V. ANALISIS

Berdasarkan hasil pengujian dapat dilihat bahwa CMAC yang terdapat pada *add-in* dapat melakukan otentikasi terhadap dokumen yang sedang dibuka dengan baik. Setiap ada perubahan pada dokumen atau ada kesalahan dalam memasukkan nilai kunci, nilai CMAC otomatis berubah drastis. Jika pihak ketiga mengubah MAC di akhir dokumen sementara tidak mengetahui nilai kunci, maka nilai MAC pasti berbeda saat penerima yang sesungguhnya melakukan validasi.

Yang perlu diperhatikan adalah MAC yang dibangkitkan oleh *add-in* ini dibalut dalam *tag xml* <AESCMAC128>. Perusakan terhadap *tag* dapat menyebabkan MAC tidak dapat ditangkap oleh *add-in* dan dianggap tidak ada. Dalam hal ini penerima mungkin bisa menambahkan atau memperbaiki *tag* tersebut secara manual, walau itu berarti ada kemungkinan juga nilai MAC tersebut ikut rusak / sudah diubah oleh pihak ketiga. Jika kemungkinan ini terjadi, pihak penerima dokumen sebaiknya meminta pengirim untuk mengirimkan ulang dokumennya lengkap dengan nilai MAC yang seharusnya.

VI. KESIMPULAN

Kesimpulan yang diambil dari studi yang telah dilakukan adalah:

1. Program *add-in* untuk *Microsoft Word* yang dibangun bekerja dengan baik untuk otentikasi dokumen. Perubahan pada dokumen seperti penambahan / pengurangan huruf, kalimat, paragraf, bahkan objek seperti gambar dan

function mengakibatkan perubahan pada nilai MAC. Selain itu, kesalahan memasukkan kunci juga akan mengakibatkan validasi nilai MAC gagal.

2. Perubahan terhadap nilai MAC atau pada dokumen tersebut pasti menggagalkan validasi karena nilai MAC yang dibangkitkan saat proses validasi berbeda dengan yang berada di dokumen. Sementara kerusakan pada cara penyimpanan nilai MAC yang dibungkus terhadap *tag xml* tertentu dapat menyebabkan nilai MAC tidak terbaca oleh program.

REFERENSI

[1] Dworkin, M. (2005). Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. National Institute of Standard and Technology Special Publication 800-3B. U.S. Department of Commerce.

[2] Munir, R. (2006). Diktat Kuliah IF5054 Kriptografi. Bandung: Penerbit Institut Teknologi Bandung.

[3] TechTerms. (2012). Plug-in Definition. http://www.techterms.com/definition/plugin, diakses 27 Juli 2012.