Aplikasi UMAC pada *Instant Messaging*

Gerard Edwin - 13507079
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
geedatz@gmail.com

***Abstract*—Aplikasi *Instant Messenger* merupakan aplikasi yang banyak digunakan sekarang ini untuk melakukan pertukaran pesan. Namun, aplikasi ini pun tidak luput dari serangan-serangan seperti penyadapan atau man-in-the-middle attack. Pada kasus ini MAC (Message Authentication Code) dapat dipergunakan. Fungsi utama MAC adalah untuk otentikasi pengirim dan menguji integritas pesan. Ada beberapa jenis MAC yang dapat digunakan, misalnya HMAC yang berbasis fungsi hash, OMAC, PMAC, atau CBC-MAC yang berbasis block cipher serta VMAC dan UMAC yang berbasis universal hash. Namun, yang paling cocok untuk digunakan dalam *instant messaging* adalah fungsi MAC yang berbasis universal hash, karena pengiriman pesan pada *Instant Messaging* relatif cepat, sehingga dibutuhkan fungsi yang dapat menghitung nilai hash dari suatu pesan dengan cepat. Dalam makalah ini, dijelaskan mengenai implementasi sederhana dari UMAC dalam aplikasi Windows Live Messenger yang dibuat dengan menggunakan aplikasi Messenger Plus!.**

***Index Terms*—*instant messaging*, UMAC, universal hash, Windows Live Messenger**

I. PENDAHULUAN

A. Message Authentication Code

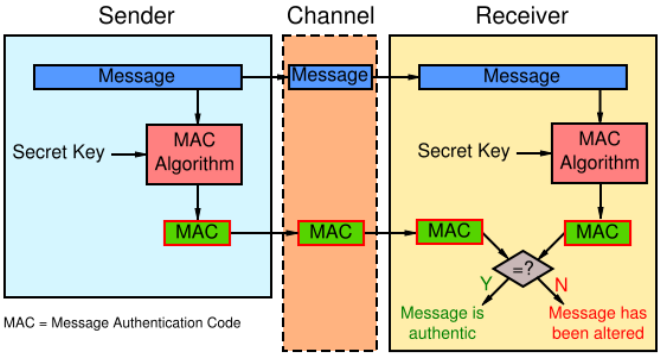
Message Authentication Code atau MAC adalah nilai hash dari suatu pesan yang dibangkitkan dengan suatu fungsi satu-arah menggunakan kunci rahasia (*secret key*). Dengan kata lain, nilai dari MAC bergantung pada pesan dan kunci rahasia yang digunakan. Nilai hash yang dihasilkan ini kemudian dilekatkan (embed) pada pesan yang akan dikirim. Nilai MAC selalu berukuran tetap (fixed) untuk ukuran pesan berapa saja. Secara umum MAC dapat dinyatakan dengan fungsi matematik :

$$MAC = C_K(M) \quad (1)$$

dengan MAC = nilai hash yang dihasilkan, C_K = fungsi hash berdasarkan kunci rahasia K, dan M = pesan yang akan dikirim.

Proses yang terjadi dalam pengiriman pesan yaitu dengan menggunakan MAC adalah sebagai berikut :

1. Pesan M di hash dengan fungsi C berdasarkan kunci K menghasilkan nilai hash MAC
   $MAC = C_K(M)$
2. MAC dilekatkan pada pesan M
   $M = M||MAC$
3. Pesan dikirimkan kepada penerima
4. Penerima memisahkan pesan M dan nilai MAC dari pesan yang diterima
5. Penerima melakukan hash terhadap pesan M yang diterima untuk menghasilkan nilai hash MAC'
6. Jika nilai MAC = MAC' maka pesan yang diterima tersebut masih otentik dan dikirim oleh lawan bicara.
7. Jika nilai MAC ≠ MAC', maka ada dua kemungkinan, yaitu :
   - Pesan berasal dari pihak ketiga yang tidak mengetahui kunci K sehingga nilai MAC yang dihasilkan akan berbeda.
   - Pesan telah diubah pada saat transmisi/ pengiriman (misalnya disadap). Maka nilai MAC yang dihasilkan akan berubah pula.
   - Nilai MAC dari pesan diubah pada saat transmisi/pengiriman. Maka nilai MAC' yang dihasilkan tidak akan sama.



Gambar 1 Skema pengiriman pesan dengan MAC

Seperti pada skema di atas, dapat dilihat bahwa jika nilai MAC ≠ nilai MAC' ada kemungkinan terjadinya penyadapan atau saluran komunikasi sudah tidak aman lagi. Hal ini sesuai dengan fungsi utama dari MAC, yaitu mengotentikasi pengirim dan menjaga keaslian pesan.

MAC berbeda dengan tanda tangan digital (digital signature), karena MAC tidak menyediakan layanan kriptografi *non-repudiation*, karena semua orang yang mengetahui kunci rahasia K dapat menghitung MAC

dari suatu pesan, berbeda dengan tandatangan digital yang menggunakan kunci privat untuk perhitungannya. Kunci privat hanya dapat dimiliki oleh seorang saja, sehingga dapat dibuktikan jika orang tersebut menyangkal telah mengirimkan pesan.

B. Instant Messaging

Instant Messaging (IM) adalah salah satu bentuk 'chat' antara orang yang sudah saling mengenal satu sama lain. Untuk menggunakan IM, pengguna harus menambahkan pengguna lain terlebih dahulu ke dalam daftar kontaknya. Hal inilah yang membedakan IM dengan chat. Pada chat, pengguna bebas berbicara dengan siapa saja. Komunikasi dengan menggunakan IM berlangusng secara *real-time* sehingga untuk dapat berkomunikasi, pengguna dan lawan bicaranya harus *online* pada saat yang bersamaan.

Instant Messaging (IM) dilakukan dengan menggunakan *Instant Messenger* client. Contoh-contoh *Instant Messenger* client yang banyak dipakai antara lain Yahoo! Messenger, Windows Live Messenger, Pidgin (messenger pada sistem operasi Linux).

C. IM dan MAC

Pengiriman pesan pada IM umumnya tidak dienkripsi, sehingga IM biasanya tidak digunakan untuk pembicaraan yang sangat penting dan privat. Selain itu, pengiriman pesan melalui IM dapat disadap oleh siapa saja yang dapat mengakses jaringan yang digunakan untuk mengirim pesan. Karena itu, meskipun pengguna memulai percakapan dengan orang yang dikenalnya (orang yang berada dalam kontaknya), pengguna tidak dapat memastikan apakah benar pesan yang ia terima berasal dari orang yang seharusnya, karena tidak tertutup kemungkinan bahwa ada orang lain yang menyadap pembicaraan di tengah-tengah (*man-in-the-middle-attack*) dan mengirimkan pesan seolah-olah ia adalah orang yang diajak bicara oleh pengguna.

Salah satu cara untuk mengatasi masalah keamanan ini adalah dengan menggunakan MAC. Seperti telah dijelaskan pada subbab A, MAC dapat membantu pengguna mengotentikasi pesan yang ia terima. Namun, karena umumnya IM tidak menyediakan fitur kriptografi, hal ini dapat dicapai dengan menggunakan plugin atau add-ons pada IM yang digunakan.

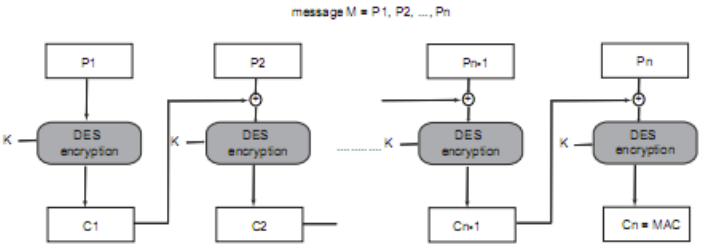
II. ALGORITMA MAC

Ada beberapa algoritma yang dapat digunakan untuk menghitung nilai MAC dari suatu pesan. Algoritma-algoritma ini didasarkan pada fungsi-fungsi kriptografi yang sudah ada, misalnya fungsi hash, fungsi block cipher, dsb. Jenis-jenis algoritma MAC yang ada yaitu algoritma MAC berbasis fungsi *hash* satu arah, algoritma MAC berbasis *block cipher*, dan algoritma MAC berbasis *universal hashing*.

A. Algoritma MAC berbasis fungsi block cipher

Salah satu algoritma MAC yang banyak digunakan adalah Data Authentication Algorithm (DAA). DAA merupakan algoritma berbasis CBC (Cipher Block Chaining) dengan menggunakan algoritma DES (Data Encryption Standard) untuk mengenkripsi pesan. MAC atau yang dalam algoritma ini disebut juga DAC (Data Authentication Code) diperoleh dari keseluruhan atau sebagian dari blok terakhir dari hasil enkripsi (C_n) dengan panjang hasil 16 <= m <= 64 bit (m adalah panjang MAC dalam bit).

Algoritma ini pernah menjadi standar dalam standar US Government. Namun, sekarang algoritma ini sudah tidak lagi digunakan karena dianggap sudah tidak aman lagi.



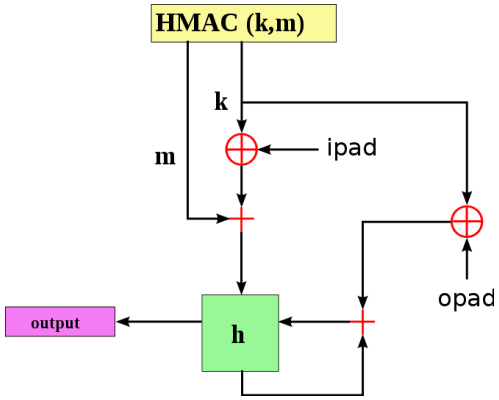
Gambar 2 DAA yang berbasis CBC

B. Algoritma MAC berbasis fungsi hash satu arah

Algoritma lain yang dikembangkan untuk menghitung MAC adalah algoritma berbasis fungsi hash satu arah (fungsi hash dalam kriptografi). Algoritma ini menggunakan fungsi hash seperti MD5 dan SHA1 untuk memperoleh nilai MAC. Namun, karena fungsi hash pada umumnya tidak menggunakan kunci, algoritma ini perlu dimodifikasi. Contoh modifikasi yang dapat dilakukan misalnya dengan menggabungkan pesan dan kunci kemudian hasilnya dihash untuk memperoleh MAC atau kunci dipecah-pecah dan dimanipulasi sedemikian rupa kemudian dioperasikan dengan pesan dan dihash. Karena kunci diikutsertakan dalam perhitungan MAC, maka orang yang tidak memiliki kunci tidak dapat menghitung nilai MAC yang benar.

Kekuatan algoritma MAC berdasarkan fungsi hash ini terletak pada kekuatan fungsi hash itu sendiri dan kunci yang digunakan. Algoritma MAC berbasis fungsi hash satu arah ini lebih cepat dalam perhitungan dibandingkan dengan algoritma yang berbasis block cipher. Algoritma ini disebut juga HMAC (Hash-based Message Authentication Code) dan biasanya disebutkan algoritma hash yang dipakai, misalnya HMAC-MD5 atau HMAC-SHA1.

\



Gambar 3 Algoritma MAC berbasis fungsi hash satu arah (HMAC)

C. Algoritma MAC berbasis universal hash

Algoritma MAC berbasis universal hash sebenarnya merupakan pengembangan dari algoritma berbasis fungsi hash satu arah biasa. Bagaimanapun cara kita memilih fungsi hash, selalu dimungkinkan bahwa ada kumpulan kunci yang akan menghasilkan nilai hash yang sama, berakibat buruknya performa dari skema hash tersebut. Untuk mengatasi permasalahan tersebut, dibuatlah skema universal hashing.

Universal hash menggunakan himpunan fungsi hash H yang terdiri dari fungsi-fungsi hash h yang dipilih secara acak yang memetakan himpunan kunci M ke N = {1,2,...,N}. H dikatakan universal jika untuk semua x, y ϵ M (x ≠ y), peluang nilai h(x) sama dengan nilai h(y) lebih kecil dari 1/|N| dengan h adalah fungsi hash yang dipilih secara acak dari H.

$$\Pr_{h \leftarrow H}[h(x) = h(y)] \leq \frac{1}{N} \quad (2)$$

Berdasarkan persamaan (2), jika h adalah fungsi hash universal, maka untuk setiap S himpunan bagian dari M, untuk setiap x anggota M dengan h acak dari H angka kolisi yang diharapkan antara x dengan elemen dari S adalah |S|/N.

Fungsi ini sering dikatakan sebagai fungsi hash yang lemah dan seringkali disebut sebagai fungsi |S|-universal. Fungsi |S|-universal dikatakan kuat jika untuk x, x' dalam M dan y, y' dalam N dipenuhi :

$$\Pr_{h \leftarrow H}[h(x) = h(y) \; and \; h(x') = h(y')] \leq \frac{1}{N^2} \quad (3)$$

Namun, biasanya fungsi universal hash yang lemah pun sudah cukup untuk melakukan universal hashing. Universal hashing unggul dari fungsi hash biasa karena perhitungan nilai hash lebih cepat dan lebih tahan terhadap collision-attack, karena seorang attacker akan kesulitan untuk mencari kolisi disebabkan oleh fungsi h yang dipilih secara acak.

III. UMAC

UMAC merupakan algoritma MAC yang cepat dan telah dibuktikan keamanannya. UMAC merupakan algoritma MAC tercepat berdasarkan laporan salah satu literatur kriptografi. UMAC pertama kali didefinsikan pada tahun 1999 dan kemudian direvisi cukup banyak pada tahun 2000 untuk mempercepat proses UMAC pada pesan berukuran kecil. Kemudian pada tahun 2004, UMAC direvisi lagi untuk menghilangkan beberapa option dengan tujuan membuat UMAC lebih sederhana.

A. Overview UMAC

UMAC merupakan salah satu algoritma yang berbasis universal hashing. Tidak seperti algoritma MAC lainnya, UMAC bersifat stateful. Ketika menghitung nilai MAC dari suatu pesan, selain membutuhkan Pesan M dan Kunci K, UMAC juga membutuhkan nonce (number used once) berukuran 64-bit. Nonce ini sesuai dengan namanya hanya digunakan sekali saja untuk satu pengiriman pesan. Biasanya nilai nonce di-*generate* oleh program secara acak atau dengan menggunakan counter (yang ditambah setiap kali pengguna mengirimkan pesan).

Karena perhitungan MAC dalam algoritma ini menggunakan noncemaka pada pesan yang dikirimkan, selain pesan dan kunci, informasi nonce juga harus disisipkan, berbeda dengan algoritma MAC lainnya. Setelah dikirim, penerima kemudian menghitung nilai MAC berdasarkan pesan dan nilai nonce yang diterima serta kunci yang digunakan untuk mengecek keaslian pesan.

UMAC menggunakan *subkey generation process* untuk membangkitkan kunci internal berdasarkan kunci rahasia k yang dipilih dalam pengiriman pesan. Karena biasanya kunci k digunakan selama pertukaran pesan, proses ini hanya dilakukan sekali saja pada awal sesi pembicaraan. Contoh proses perhitungan MAC dengan UMAC yaitu :

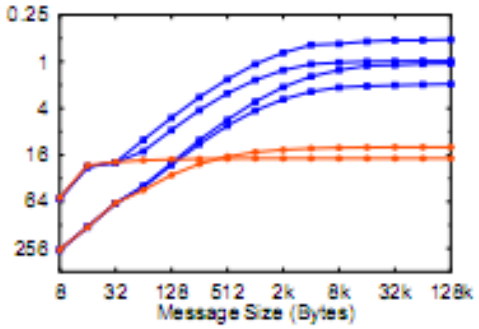
1. Subkey generation : menggunakan pseudo-random generator (PRG) untuk memetakan kunci K = $K_1K_2...K_{1024}$, (dengan K_i adalah word berukuran 32-bit) ke A (dengan |A| = 512-bit). Contoh perhitungan
2. Melakukan hash dari *Msg* ke *HM* (Hashed Message) = $NHX_{Key}(Msg)$:
   Len = |*Msg*| mod 4096, diencode dalam string 2-byte
   Msg dipadding dengan bit 0 agar |Msg| kelipatan 8.
   $Msg = Msg_1 \; || \; Msg_2 \; || \; ... \; || \; Msg_t$, dengan Msg_i berukuran 1024 word, kecuali Msg_t yang berukuran 2-1024 word.
   $HM = NH_K(Msg_1) \; || \; NH_K(Msg_2) \; || \; ... \; || \; NH_K(Msg_t) \; ||$ Len
3. Menghitung MAC : MAC = $HMAC\text{-}SHA1_A(HM||Nonce)$

B. Parameter UMAC

Parameter yang dipakai dalam perhitungan MAC adalah ukuran blok dan ukuran word, misalnya 1024 dan 32 pada kasus di atas. Secara alami, fungsi pseudorandom (PRF) yang diterapkan kepada HM || Nonce (fungsi HMAC-SHA1 pada contoh di atas) merupakan parameter. Fungsi ini merupakan fungsi yang dipilih secara acak dari sekumpulan fungsi hash, jadi tidak harus selalu menggunakan fungsi hash HMAC-SHA1.

Fungsi universal hash pada contoh di atas memiliki peluang untuk terjadinya kolisi sebesar 2^{-32}. Untuk mengecilkan nilai ini, dibuat suatu ketentuan. Untuk menguadratkan nilai kolisi tersebut misalnya, dapat dilakukan hashing terhadap pesan 2 kali dengan kunci yang berbeda dan independen kemudian menggabungkan hasilnya. Tetapi, optimisasi dari UMAC adalah dua kunci yang digunakan saling terkait satu sama lain. Salah satu kunci merupakan hasil shift dari kunci lainnya dengan tambahan beberapa word. Hal ini dikenal dengan "Toeplitz construction".

C. Performa UMAC



Gambar 4 Performa UMAC dibandingkan dengan algoritma MAC lainnya seperti CBC-MAC dan HMAC

Seperti terlihat pada gambar di atas, UMAC (ditunjukkan oleh garis biru, 4 buah garis biru adalah beberapa varian dari UMAC, yaitu dari atas-ke-bawah UMAC-MMX-30, UMAC-MMX-60, UMAC-STD-30, UMAC-STD-60) jauh lebih cepat dibandingkan fungsi HMAC-SHA1 dan CBC-MAC-RC6 (ditunjukkan oleh dua buah garis merah).

Namun, pada pesan berukuran kecil algoritma MAC berjalan dengan lambat, hal ini dikarenakan banyak proses yang harus dilakukan untuk pesan yang berukuran kecil. Hal ini sudah diperbaiki pada revisi UMAC terbaru, sehingga waktu pemrosesan pesan yang berukuran kecil juga relatif cepat.

D. Kekuatan UMAC

Kekuatan UMAC bergantung pada kekuatan fungsi-fungsi kriptografi yang mendasarinya, meliputi KDF (Key-Derivation Function) yang digunakan untuk menghasilkan bit-bit pseudorandom untuk kunci pada fungsi hash yang digunakan dan PDF (Pad-Derivation Function) yang digunakan untuk menghitung pseudorandom pad berdasarkan key dan nonce. Pad ini digunakan dalam perhitungan MAC. Panjang pad bervariasi, antara 4, 8, 12, atau 16 byte.

Dalam spesifikasi ini, kedua fungsi tersebut diimplementasikan menggunakan block cipher, biasa digunakan Advanced Encryption Standard (AES). Namun, desain UMAC memungkinkan bahwa fungsi ini diganti dengan fungsi block cipher lain maupun dengan fungsi hash untuk perhitungan KDF dan PDF.

Inti dari desain UMAC, yaitu fungsi UHASH (Universal Hash) tidak tergantung pada asumsi kriptografik. Kekuatannya ditentukan secara murni oleh properti matematis dari fungsi universal, yaitu peluang terjadinya kolisi. Hal ini berarti kekuatan UHASH dijamin terhadap serangan kriptanalis.

Analisis UMAC menunjukkan bahwa skema ini memiliki keamana yang telah dibuktikan dalam kriptografi modern, melalui pengurangan yang ketat. Artinya, serangan pada UMAC dengan probabilitas yang lebih besar dari probabilitas kolisi dari UHASH akan menimbulkan tingkat kesulitan serangan yang sebanding.

Algoritma MAC umumnya digunakan untuk otentikasi pesan antara dua pihak yang berbagi kunci rahasia K. Nilai MAC dari pesan dihitung berdasarkan kunci K dan nilai Nonce (untuk UMAC). Serangan terhadap MAC berarti penyerang menghasilkan sendiri, tanpa mengetahui kunci K, sebuah pesan M dan menghitung nilai MAC berdasarkan kunci K yang dipakai.

Jika panjang MAC ditetapkan menjadi t panjang, maka penyerang secara trivia dapat memecahkan MAC dengan probabilitas 1/2^t. Untuk kasus ini, penyerang dapat menghitung MAC dari pesan acak dan kemungkinan untuk menemukan bahwa MAC tersebut benar adalah 1/2^t. Dalam kasus UMAC-64, seorang penyerang dapat menebak dengan benar nilai MAC 8-byte dengan probabilitas 1/2^64 dengan hanya menebak nilai secara acak. Hasil analisis menunjukkan bahwa tidak ada strategi serangan yang dapat menghasilkan nilai MAC yang bernar dengan probabilitas lebih baik dari 1/2^60 jika UMAC menggunakan fungsi hash secara acak dalam implementasinya, bukan dengan AES saja.

IV. IMPLEMENTASI UMAC

Pada kesempatan kali ini, penulis mencoba membuat program sederhana yang mengimplementasikan algoritma UMAC dalam salah satu aplikasi *Instant Messenger* yang banyak digunakan, yaitu Windows Live Messenger. (Pada kasus ini, diasumsikan pengirim dan penerima pesan sudah menginstal aplikasi Messenger Plus! dan script UMAC).

A. Windows Live Messenger

Windows Live Messenger (WLM) merupakan aplikasi *Instant Messenger* yang dibuat oleh Microsoft. Pada awalnya aplikasi ini bernama MSN Messenger. Windows Live Messenger sendiri tidak menyediakan fitur untuk penambahan plugin atau add-on. Namun, hal itu diatasi dengan dibuatnya extension untuk WLM bernama Messenger Plus!.

B. Messenger Plus!

Messenger Plus!, seperti telah disebutkan pada subbab sebelumnya, merupakan ekstensi dari aplikasi Windows Live Messenger yang dikembangkan oleh Yuna Software. Ekstensi ini memungkinkan penambahan berbagai macam script pada aplikasi WLM. Script ini akan dijalankan pada WLM dengan Messenger Plus! sebagai perantaranya. Contoh script yang ada misalnya auto-reply, duplicate contact, dan sebagainya. Pengguna dapat memperoleh script-script ini dari laman Messenger Plus!. Selain itu pengguna juga dapat membuat sendiri script sesuai dengan kebutuhan mereka.

Script untuk Messenger Plus! dibuat atau ditulis dalam bahasa Javascript dengan menggunakan tambahan library dari Messenger Plus!. Messenger Plus! juga menyediakan fitur untuk menjalankan dan debugging dari script yang telah dibuat oleh pengguna.

C. Script UMAC

Ada tiga fungsi utama dari script yang ditulis untuk implementasi UMAC, yaitu sebagi berikut :

1. **Fungsi OnEvent_ChatWndCreated**

```
13 function OnEvent_ChatWndCreated(ChatWnd)
14 {
15      // Meminta masukan key
16      key = PromptKey();
17      // Generate angka acak sebagai Nonce
18      nonce = GenerateRandomNonce();
19 }
```

Gambar 5 Source code untuk fungsi OnEvent_ChatWndCreated

Fungsi ini dipanggil ketika sebuah chat window baru dibuat. Dalam fungsi ini, program akan meminta kunci yang akan digunakan pada chat tersebut. Kunci ini akan digunakan dalam pengiriman pesan selama window chat tidak ditutup.

2. **Fungsi OnEvent_ChatWndSendMessage**

```
21 function OnEvent_ChatWndSendMessage(ChatWnd, Message)
22 {
23      // Append Message dengan MAC dan nonce
24      var msg = Message +
25                "[MAC]"+hash(Message, key, nonce)+
26                "[\MAC]"+ nonce + "\0";
27      // Increment nonce
28      nonce++;
29      // Kirim pesan
30      return msg;
31 }
```

Gambar 6 Source code untuk fungsi OnEvent_ChatWndSendMessage

Fungsi ini dipanggil ketika pengguna mengirimkan pesan melalui suatu window chat. Yang dilakukan adalah menghitung nilai MAC dari pesan yang dikirim berdasarkan kunci untuk window chat tersebut dan nilai nonce (yang pertama kali dibangkitkan secara acak kemudian diincrement), kemudian kode MAC ditambahkan di belakang pesan (di antara tag [MAC] dan [\MAC]) beserta kode nonce (setelah tag [\MAC]). Pesan tersebut kemudian dikirim kepada penerima.

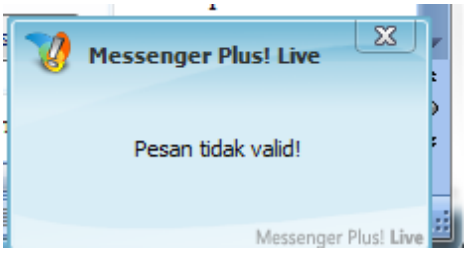
3. **Fungsi OnEvent_ChatReceiveMessage**

```
33 function OnEvent_ChatWndReceiveMessage(ChatWnd, Origin, Message, MsgKind)
34 {
35     // Pecah Message menjadi pesan asli, MAC, dan nonce
36     var idx = Message.indexOf("[MAC]");
37     var idx2 = Message.indexOf("[\MAC]");
38     var msg = Message.substr(0,idx);
39     var MAC = Message.substr(idx).replace("[MAC]","").replace("[\MAC]","");
40     var _nonce = Message.substr(idx2).replace("[\MAC]","");
41     // Periksa pengirim
42     if(Origin!=Messenger.MyName) {
43         // Jika MAC = MAC' maka valid
44         if(MAC == hash(msg,key,_nonce))
45         {
46             Debug.Trace("VALID");
47         }
48         // Jika MAC != MAC'
49         else
50         {
51             // Tampilkan popup dan mainkan suara peringatan
52             MsgPlus.DisplayToast("","Pesan tidak valid!","error.mp3");
53             Debug.Trace("INVALID");
54         }
55     }
56     // Tampilkan pesan
57     return msg;
58 }
```

Gambar 7 Source code untuk fungsi OnEvent_ChatReceiveMessage

Fungsi ini dipanggil ketika penerima menerima pesan dari seseorang (termasuk pesan yang dikirimkan dari pengguna ke orang lain). Pertama-tama, pesan akan dipecah menjadi tiga bagian, yaitu pesan asli, MAC, dan nilai nonce. Jika pesan yang diterima adalah pesan yang dikirimkan oleh pengguna maka pesan langsung ditampilkan (tanpa MAC dan nonce). Jika pesan berasal dari orang lain, maka program akan menghitung nilai MAC' berdasarkan pesan asli, nilai nonce dan kunci rahasia (kunci dimasukkan user pada saat window chat dibuat. Window chat

dibuat secara otomatis ketika pengguna menerima pesan baru dan belum ada window chat dengan pengirim). Jika nilai MAC sama dengan MAC', maka pesan yang diterima masih valid dan program tidak menampilkan apa-apa (akan mengganggu jika pesan valid ditampilkan terus menerus, karena kemungkinan besar pesan yang diterima masih valid). Jika ternyata MAC tidak sama dengan MAC', program akan mengeluarkan popup "Pesan tidak valid!" dan memberikan suara peringatan.



Gambar 8 Popup yang menunjukkan bahwa pesan yang diterima tidak valid

V. KESIMPULAN

Dari penjelasan dalam makalah ini, penulis menyimpulkan beberapa hal, yaitu :

1. Aplikasi *Instant Messaging* merupakan aplikasi yang banyak dipakai, tetapi masih rentan terhadap serangan, sehingga perlu tindakan pengamanan.
2. Salah satu bentuk pengamanan pesan untuk *Instant Messaging* adalah MAC (Message Authentication Code).
3. Algoritma UMAC merupakan algoritma MAC yang cepat dalam menghitung nilai MAC.
4. Algoritma UMAC cocok untuk dipakai dalam aplikasi *Instant Messaging* untuk menjaga integritas pesan yang dikirim dan mengecek kebenaran pengirim.
5. Implementasi plugin pada Windows Live Messenger dapat menggunakan aplikasi Messenger Plus!.

VI. UCAPAN TERIMA KASIH

Pertama-tama penulis ingin mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas selesainya Makalah Kriptografi ini. Kedua, penulis ingin berterima kepada Bapak Rinaldi Munir selaku dosen pengajar kuliah IF3058 Kriptografi atas kesempatan yang telah diberikan untuk membuat makalah ini dan atas materi-materi yang telah diberikan selama satu semester ini.

DAFTAR REFERENSI

[1] http://fastcrypto.org/umac, tanggal akses : 15 Mei 2010
[2] http://netforbeginners.about.com/od/blogchatinstantmessaging/f/email_vs_im.htm, tanggal akses : 15 Mei 2010
[3] http://mpscripts.net/docs/, tanggal akses : 16 Mei 2010
[4] T. Krovetz, UMAC : Message Authentication Code using Universal Hashing. CSU Sacramento, March 2006.
[5] Rinaldi, Munir. Diktat Kuliah IF5054 Kriptografi, Departemen Teknik Informatika Institut Teknologi Bandung. 2005, hal. 178-179

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, 16 Mei 2010

Gerard Edwin
13507079