Implementasi *keyed-Hash Message Authentication Code*-MD5 pada Aplikasi *Instant Messenger* Pidgin 2.2.0

Prasetyo Nugroho [1)]
1) Program Studi Teknik Informatika ITB, Bandung 40132, email: if14039@students.if.itb.ac.id

Abstract – *Makalah ini membahas bagaimana analisis implementasi salah satu varian algoritma Message Authentication Code (MAC) pada sebuah aplikasi messaging client. Varian MAC yang dipilih adalah keyed-Hash Message Authentication Code, yaitu metode untuk mengotentikasi pesan dengan menggunakan fungsi hash yang dapat dipakai, dalam hal ini menggunakan MD5. Kekuatan dari HMAC sendiri bergantung pada fungsi hash yang dipakai.*

Aplikasi messaging client yang dipilih untuk implementasi HMAC ini adalah Pidgin. Pidgin adalah salah satu messaging client open-source yang berlisensi GNU dan dikembangkan dengan Gtk+. Oleh karena itu, pengembangan aplikasi ini cenderung lebih mudah, baik secara teknis, maupun non-teknis. Ada beberapa versi Pidgin yang terdapat di internet, tetapi versi engine terbaru yang dipakai masih versi 2.2.0. Oleh karena itu, pengembangan akan dipilih menggunakan API untuk versi 2.2.0. Untuk membangun plugin Pidgin, telah tersedia libpurple, yaitu library yang digunakan aplikasi pengembang untuk berhubungan dengan core Pidgin. Jadi aplikasi plugin baru bisa memanfaatkan libpurple untuk mengakses koneksi, conversation, account list, dsb.

Kata Kunci*: Message Authentication Code (MAC), keyed-Hash Message Authentication Code (HMAC), messaging client, Pidgin, Gtk+, libpurple.*

1. PENDAHULUAN

Seiring dengan perkembangan teknologi, *messaging client* sampai saat ini juga terus berkembang pesat dan sangat bervariasi dalam penggunaannya. Sebelumnya, *messaging client* hanya digunakan sebagai sarana untuk berkomunikasi dengan orang lain saja. Namun seiring dengan perkembangan jaman, penggunaan *messaging client* menjadi semakin meluas dan tidak terbatas hanya sebagai sarana komunikasi dua orang saja, tetapi dapat digunakan untuk konferensi dan bertukar informasi yang serius, misalnya tentang ekonomi, politik, dsb. Oleh karena itu, keamanan dalam bertukar informasi tersebut sangat dibutuhkan agar informasi tersebut tidak dicuri atau diubah oleh pihak ketiga. Untuk itu, diperlukan suatu mekanisme untuk menjamin keamanan percakapan tersebut dari segi integritas data maupun ketersediaan otentikasi pesan.

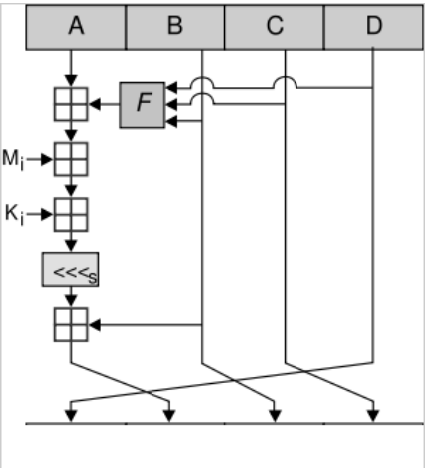
Salah satu cara yang dapat digunakan untuk menjaga integritas data sekaligus menyediakan fitur otentikasi pesan adalah teknik *Message Authentication Code* (MAC). Teknik ini adalah salah satu teknik kriptografi yang masih dipakai. MAC adalah fungsi *hash* satu-arah yang menggunakan kunci rahasia dalam pembangkitan nilai *hash*. Dengan kata lain, nilai *hash* adalah fungsi dari pesan dan kunci. MAC memiliki sifat dan properti yang sama dengan fungsi *hash* biasa, hanya saja MAC memiliki komponen kunci yang digunakan sebagai masukan. Kunci ini selanjutnya digunakan oleh penerima pesan untuk memverifikasi nilai *hash* dari pesan yang diterima. MAC digunakan untuk otentikasi pesan tanpa perlu mengenkripsi pesan. Mula-mula pengirim pesan menghitung MAC dari pesan tersebut dengan menggunakan kunci rahasia, yang telah dibagi kepada kedua belah pihak, kemudian MAC tersebut dilekatkan pada pesan yang akan dikirim. Setelah pesan tersebut diterima oleh penerima pesan yang dituju, penerima pesan tersebut selanjutnya membandingkan MAC menggunakan kunci yang sama dengan MAC yang diterimanya. Apabila MAC tersebut sama, hal itu berarti pesan tersebut masih asli, begitu pula sebaliknya.

Ada banyak varian MAC yang berhasil dikembangkan oleh berbagai ahli. Salah satunya adalah HMAC. HMAC atau *keyed-Hash Message Authentication Code* adalah MAC yang dihitung menggunakan fungsi hash, misalnya MD4, MD5, SHA-1, dsb. Panjang hasil dari HMAC sendiri bergantung pada fungsi *hash* yang digunakan. Dalam contoh kasus kali ini, fungsi *hash* yang digunakan adalah MD5, jadi hasil dari HMAC adalah 128-bit atau 32 karakter. Kekuatan dari HMAC sendiri bergantung pada kekuatan fungsi *hash* yang digunakan. Karena HMAC pada makalah ini menggunakan fungsi *hash* MD5, maka HMAC ini disebut juga HMAC-MD5.

2. PEMBAHASAN

2.1. MD5

MD5 (*Message Digest algorithm 5*) adalah fungsi *hash* satu-arah yang dibuat oleh Ronald Rivest pada tahun 1991. MD5 merupakan perbaikan dari MD4 setelah MD4 berhasil diserang oleh kriptanalis. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan pesan-ringkas (*message digest*) dengan panjang 128-bit.



Gambar 1: Operasi dasar MD5

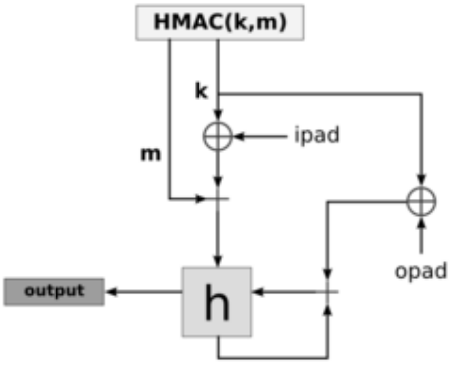
Operasi dasar MD5 yang diperlihatkan Gambar 1 dapat ditulis dengan sebuah persamaan sebagai berikut:

$$a \leftarrow b + CLS_s(a + g(b,c,d) + X[k] + T[i]) \quad (1)$$

dengan, *a*, *b*, *c*, *d* adalah 4 buah peubah penyangga 32-bit, *g* adalah suatu fungsi, *CLSs* adalah menggeser bit sebanyak *s* bit (notasi <<<*s*), *X[k]* adalah kelompok 32-bit ke-*k* dari blok 512-bit *message* ke-*q* dengan nilai *k*=0 sampai 15, *T[i]* adalah elemen tabel *T* ke-*I* (32-bit), dan + adalah operasi penjumlahan modulo 2^{32}.

2.2. HMAC

HMAC menggunakan fungsi *hash* seperti MD4, MD5, SHA-1, dsb. Dalam contoh kasus ini, fungsi *hash* yang digunakan adalah MD5. HMAC menggunakan kunci dalam melakukan *hash* pada pesan, dan kunci yang digunakan memiliki batas maksimal adalah 64 karakter.



Gambar 2: Ilustrasi HMAC

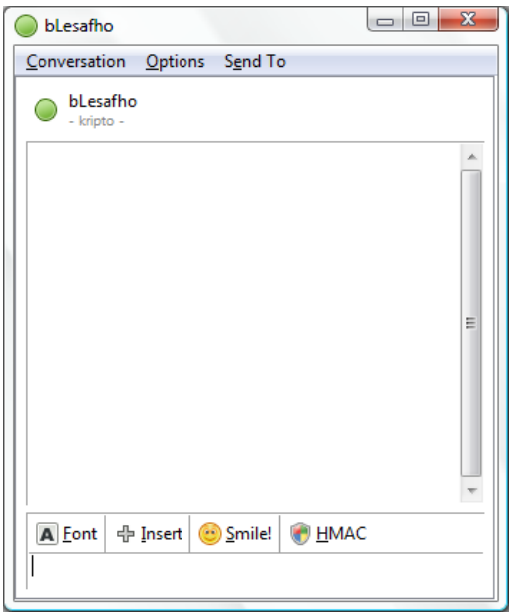
Cara kerja HMAC yang diperlihatkan Gambar 2 dapat ditulis dengan sebuah persamaan sebagai berikut:

$$HMAC_k(m) = h(k \oplus opad, h((k \oplus ipad), m)) \quad (2)$$

dengan, *k* adalah kunci yang telah ditambahkan bit 0 agar ukurannya 512-bit, *m* adalah pesan, *h* adalah fungsi *hash*, *ipad* dan *opad* adalah bit penambah yang berukuran 512-bit.

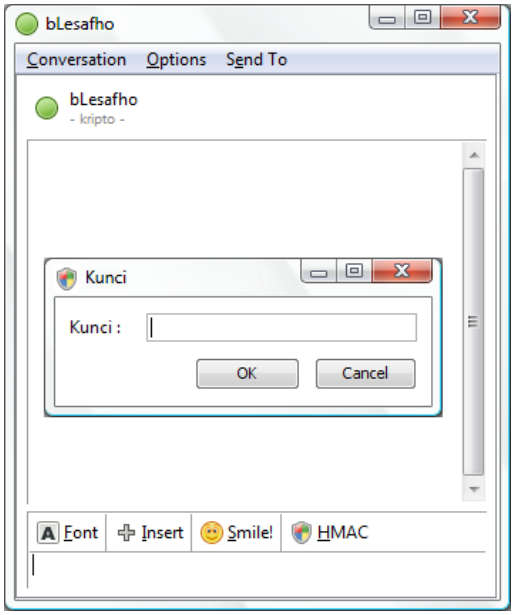
3. PENGUJIAN

Pengujian dilakukan dengan membuat plugin pada *Instant Messenger* Pidgin 2.2.0. Secara keseluruhan, tampilan tidak banyak berubah, hanya saja pada *form* percakapan plugin ini menambahkan fungsi *secure message* yang dapat diaktifkan atau di-nonaktifkan dengan menekan tombol HMAC. Gambar 3 adalah gambar tampilan *form* percakapan setelah ditambahkan plugin HMAC. Pada contoh ini, percakapan yang terjadi adalah percakapan antara "blue" (pengirim) dan "blesafho" (penerima). Gambar 3 adalah gambar *form* percakapan pada komputer "blue."



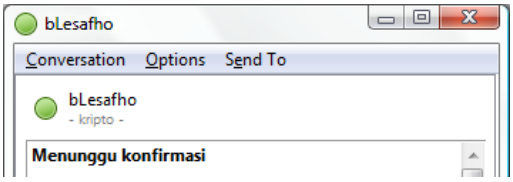
Gambar 3: Form percakapan

Untuk menggunakan fasilitas percakapan yang aman, dapat diaktifkan dengan menekan tombol "HMAC". Apabila fasilitas tersebut tidak diaktifkan, maka percakapan tersebut adalah percakapan yang biasa (tanpa menggunakan HMAC). Apabila tombol HMAC ditekan, maka akan muncul *form* "Kunci" seperti pada Gambar 4. *Form* ini adalah *form* untuk memasukkan kunci yang telah disetujui oleh kedua belah pihak. Untuk menggunakan fasilitas ini, kedua belah pihak harus memasukkan kunci yang sama, apabila kunci yang dimasukkan berbeda, atau lawan bicara tidak menyetujui mode percakapan, maka fungsi percakapan aman tidak akan berlangsung.



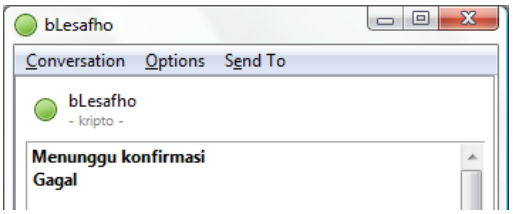
Gambar 4: Form kunci

Gambar 5 adalah gambar ketika "blue" ingin memulai percakapan yang aman, dengan metode HMAC. Setelah pengirim memasukkan kunci, maka pengirim akan menunggu konfirmasi penerima, yang digambarkan pada Gambar 5. *Form* kunci seperti Gambar 4 akan muncul pada sisi penerima.

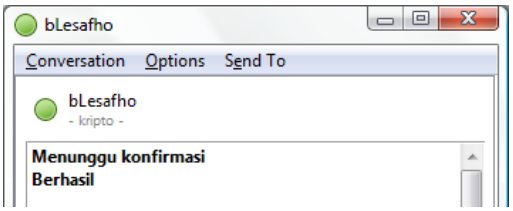


Gambar 5: Konfirmasi

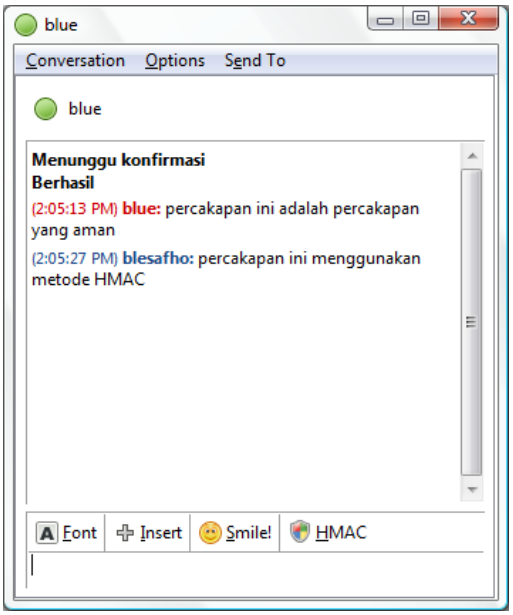
Apabila penerima memasukkan kunci yang sama, maka mode percakapan aman dengan HMAC akan diaktifkan, apabila penerima menekan tombol "Cancel" atau memasukkan kunci yang berbeda, maka penerima menolak mode tersebut, dengan ditandai dengan tulisan "Gagal" seperti Gambar 6. Sedangkan apabila penerima memasukkan kunci yang sama, maka akan muncul tulisan "Berhasil" dan mode percakapan aman akan aktif seperti pada Gambar 7.
Setelah itu, setiap percakapan yang dilakukan akan dikonfirmasi terlebih dahulu dan percakapan berlangsung seperti biasa, hanya saja sebenarnya pesan yang dikirim telah ditambahkan dengan HMAC yang telah dihitung. Dan pada sisi penerima, pesan yang dikirim akan dicek kebenarannya dengan cara membandingkan HMAC pada pesan tersebut dengan kunci yang sama. Apabila HMAC tersebut sama, maka pesan tersebut akan tampil seutuhnya seperti percakapan biasa, seperti pada Gambar 8, dan apabila HMAC tersebut berbeda, maka pesan tersebut tidak akan muncul dan akan ada tulisan "Pesan telah berubah" pada kedua belah pihak, namun pesan yang dikirim oleh pengirim tetap muncul pada *form* percakapan pengirim, namun pesan tersebut tidak muncul pada sisi penerima, seperti pada Gambar 9. Untuk contoh kasus ini, karena penulis tidak mengetahui cara pengubahan pesan oleh pihak ketiga, atau cara untuk mengubah pesan yang dikirim pengirim agar pesan yang diterima penerima berbeda, maka ada bagian dari *source code* program yang diubah agar pengecekan HMAC selalu salah, sehingga pesan yang dikirim dianggap telah berubah. Apabila pengirim atau penerima menekan tombol "HMAC" kembali, maka mode percakapan aman akan selesai dan menampilkan tulisan "Mode HMAC selesai." Setelah mode tersebut selesai, maka percakapan akan berlangsung seperti biasa tanpa ada penambahan HMAC pada pesan dan pengecekan HMAC.



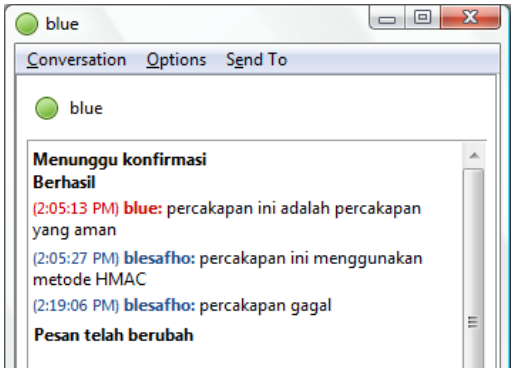
Gambar 6: Gagal konfirmasi



Gambar 7: Berhasil konfirmasi



Gambar 8: Percakapan yang aman



Gambar 9: Pesan yang telah berubah

4. KESIMPULAN

HMAC adalah salah satu dari varian dari MAC yang menggunakan fungsi *hash* seperti MD4, MD5, SHA-1, dsb. Kekuatan dari HMAC sendiri bergantung pada fungsi *hash* yang dipakai dan panjang hasil dari HMAC juga bergantung pada fungsi *hash* yang digunakan. Karena fungsi *hash* yang digunakan adalah MD5, maka hasil dari HMAC adalah karakter dengan panjang 32-byte atau 128-bit. Penggunaan otentikasi pesan dan penjagaan integritas data pada aplikasi *client messanging* termasuk bagus untuk diterapkan, misalnya pada Pidgin. Pidgin adalah salah satu *Instant Messenger* yang mampu ditambahkan plugin, sehingga implementasi HMAC ini dapat dicoba pada Pidgin. Pidgin menyediakan sebuah *library* yaitu *libpurple* yang didesain sebagai *front end* dari Gtk+, lingkungan pengembangan Pidgin. Implementasi pada *Instant Messenger* ini dapat dikatakan berhasil, karena program dapat berjalan dengan baik.

Dari segi keamanan, memang HMAC hanya bergantung pada fungsi *hash* yang digunakan, hal ini menjadi salah satu kekurangan HMAC, padahal banyak kriptanalis yang selalu mencoba untuk menyerang fungsi *hash* yang ada, dalam hal ini MD5 telah berhasil diserang walaupun hal itu sangat susah untuk dilakukan.

5. DAFTAR REFERENSI

[[1] Munir, Rinaldi, Diktat Kuliah IF5054 Kriptografi, Institut Teknologi Bandung, 2007.
[2] http://developer.pidgin.im/wiki/CHowTo.
[3] http://tools.ietf.org/html/rfc2202