OTENTIKASI PESAN YANG PRAKTIS DAN AMAN

Mohammad Gilang Kautzar HW – NIM : 13505101

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15101@students.if.itb.ac.id

Abstrak

Otentikasi pesan sangat dibutuhkan untuk menjamin bahwa suatu pesan aman dari impersonasi, modifikasi, dan atau substitusi. Dengan otentikasi yang aman dapat dijamin bahwa hanya pihak yang berkepentingan yang dapat membuat dan mengubah pesan yang ada.

Penggunaan algoritma enkripsi sebagai cara untuk otentikasi pesan telah diganti dengan penggunaan fungsi hash yang prosesnya jauh lebih cepat. Pada bukunya, Gene Tsudik [TSU92] memberikan tiga metode otentikasi pesan berdasarkan fungi hash dan menggunakan beberapa kunci untuk keamanan. Pada makalah akan diberikan beberapa metode yang menggunakan fungi hash seperti pada MD5 dan mendukung otentikasi pesan dengan aman. Ide metode pada makalah ini mirip dengan milik Tsudik, dengan kekuatan keamanan yang sama, namun panjang kunci dapat diperkecil hingga delapan kali. Pada metode ini, kunci dibuat dengan operasi *exclusive-or* atau *assign operator* yang lebih cepat daripada proses konkatenasi. Pada makalah juga akan dibuktikan bahwa metode tersebut merupakan *secure keyed one-way hash functions*.

Kata kunci: *otentikasi pesan*, *message digest*, *fungi hash satu-arah*

1. Pendahuluan

Di masa ini, keberadaan suatu metode otentikasi pesan merupakan sesuatu yang sangat penting. Sebuah otentikasi pesan yang aman harus menjamin integritas dan keaslian pesan dari impersonasi, modifikasi, dan substitusi. Yang dimaksud dengan aman berarti bahwa hanya pihak-pihak yang berkepentingan yang dapat menulis atau mengganti isi pesan. Penggunaan *parity bits* pada komunikasi memberikan otentikasi karena setiap orang dapat memeriksa keaslian dari suatu pesan, namun ini tidak dapat disebut aman karena setiap orang dapat mengganti beberapa bit pesan lalu mencari *parity bits* yang sesuai. Dengan kata lain otentikasi tersebut tidak bergantung pada suatu kunci rahasia. Otentikasi pesan yang aman ini disebut juga *Secured Keyed One-Way Hash Functions*.

Algoritma enkripsi, misalnya DES, digunakan pada aplikasi yang berbeda untuk menjamin kerahasiaan menggunakan suatu kunci rahasia. Salah satu cara paling umum untuk melakukan otentikasi pesan adalah dengan mengirimkan pesan diikuti dengan *message digest*. Sebuah *message digest* yang aman harus memiliki panjang yang tetap dan dapat dihasilkan dengan misalnya melakukan enkripsi DES di mode CBC terhadap pesan. Contoh lainnya, penggunaan fungsi *hash* yang diikuti dengan algoritma enkripsi dapat digunakan untuk menghitung *message digest*. Kemudian pasangan pesan dan *message digest* tadi dikirimkan melalui suatu kanal. Karena algoritma enkripsi dapat di-invers, biasanya algoritma tersebut bersifat kompleks dan lambat. Sementara itu, penggunaan fungsi *hash* satu arah yang biasanya jauh lebih cepat sangat dianjurkan. Pada bukunya [1], Tsudik menawarkan tiga metode otentikasi pesan yang berdasarkan pada fungsi *hash* satu arah dan suatu kunci rahasia. Metode ini menggunakan MD4 sebagai fungi *hash* dan dinamakan *secret prefix*, *secret suffix*, dan *envelope method*.

Modifikasi terhadap metode Tsudik ini dilakukan dengan mengurangi panjang kunci dari 512 menjadi 128 bit. Pengurangan kunci ini tidak hanya mempercepat proses operasi, namun juga mempermudah penanganan kunci. Kunci 128 bit berarti hanya berukuran 16 *byte*, sehingga mudah untuk dihapalkan. Metode ini menggunakan MD5 ini, sementara kunci rahasia digunakan untuk menambah keamanan. Konkatenasi kunci terhadap pesan digantikan dengan operasi XOR (*exclusive or*) terhadap pesan dan menjadikannya vektor inisialisasi (*vector initialization*). Hal ini menyebabkan penghematan sebanyak satu blok pada pemrosesan fungsi *hash*. Fungsi *hash* selain MD5 juga dapat digunakan, tergantung keperluan keamanaan dan kecepatan.

2. MD5

MD5 adalah algoritma *hash* yang memetakan suatu pesan ke 128 bit *message digest* (nilai *hash*). MD5 didesain cepat pada mesin dengan *registers* 32 bit.

MD5 selalu menambahkan *padding* beberapa bit yang terdiri dari sebuah '1' dan diikuti dengan '0', sehingga panjang bit pesan merupakan kelipatan dari 512. Setiap 512 bit blok pesan diproses dalam satu *loop* MD5 yang terdiri dari empat putaran. 128-bit keluaran dari satu putaran disebut dengan *intermediate value* dari algoritma MD5. Pada tahap awal, MD5 menggunakan sebuah *initialization vector* untuk 512 bit blok pertama. Untuk blok-blok setelahnya, pemrosesan menggunakan keluaran blok sebelumnya. Dan hasil keluaran blok terakhirlah yang menjadi nilai *message digest*.

Pada bukunya [2], Ronald L Rivest menyatakan bahwa:

1. Tingkat kesulitan mencari suatu pesan dari suatu nilai *message digest* MD5 berada pada orde 2^{128} operasi.
2. Tingkat kesulitan mencari sepasang pesan dengan nilai *message digest* MD5 yang sama berada pada orde 2^{64} operasi.

Melalui pernyataan dua buah di atas, maka dapat diasumsikan bahwa MD5 aman dari serangan.

3. Metode Tsudik

Tsudik memberikan tiga buah metode otentikasi pesan, yakni *secret prefix*, *secret suffix*, *envelope method*.

Penjelasannya masing-masing metode adalah sebagai berikut:

1. Metode *secret prefix*:
   Misalkan Alice ingin mengirimkan pesan ke Bob, maka tahapannya adalah sebagai berikut:

   - Gunakan kunci 512-bit S^p_{AB},
   - Hitung MD = MD4 ($S^p_{AB} \| M$),
   - Kirim pasangan [M, MD] ke Bob.

   Karena Bob mengetahui kunci S^p_{AB}, maka ia dapat memverifikasi MD dengan menghitung ulang MD4 ($S^p_{AB} \| M$). Perlu dicatat bahwa panjang kunci adalah 512 bit, maka MD4 memerlukan 64 langkah (satu *loop*) untuk memroses tambahan 512 bit tersebut.

2. Metode *secret suffix*:
   Adapun Alice melakukan langkah-langkah berikut:

   - Gunakan 512 bit kunci S^s_{AB},
   - Hitung MD = MD4 ($M \| S^s_{AB}$),
   - Kirimkan pasangan [M, MD] ke Bob.

   Bob dapat memverifikasi MD karena ia mengetahui S^s_{AB}. Seperti metode sebelumnya, 512 bit ditambahkan ke pesan (satu blok extra).

3. Metode *envelope*:
   Metode ini merupakan kombinasi dari dua metode sebelumnya dimana tahapan yang dilakukan Alice adalah sebagai berikut:

   - Gunakan kunci S^p_{AB} dan S^s_{AB} (1024 bit),
   - Hitung MD = MD4 ($S^p_{AB} \| M \| S^s_{AB}$),
   - Kirim pasangan [M, MD] ke Bob.

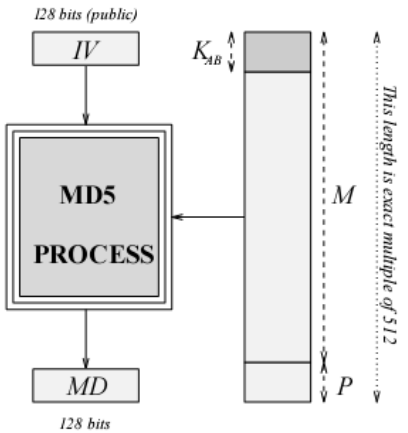
   Pada metode ini, pesan M berada di antara S^p_{AB} dan S^s_{AB}. Dua blok (1024 bit) yang ditambahkan pada pesan menyebabkan MD4 melakukan $2 \times 64 = 128$ langkah.

Dapat dilihat bahwa penambahan panjang kunci pada metode-metode di atas tidak akan menambah keamanan dan tetap ekivalen dengan kriptanalisis terhadap MD4. Dengan kata lain pemendekan panjang kunci ke 128 bit akan tetap menjamin keamanan sistem. Terlebih lagi, kunci tidak perlu ditambahkan ke pesan.

4. Metode Baru

Di sini dilakukan modifikasi terhadap metode Tsudik dan diberikan enam buah metode yang masing-masing berbeda tergantung kebutuhan pengguna. Kunci 128 bit dipercaya cukup kuat terhadap serangan *exhaustive search*. Kunci yang digunakan 8 kali lebih kecil daripada kunci pada metode Tsudik. Kunci di operasi XOR-kan dengan message dan terkadang digunakan sebagai *initialization vector*.

4.1 Metode 1, K_{AB} sebagai prefix M



Gambar 1: K_{AB} (128 bit) sebagai *prefix M*.

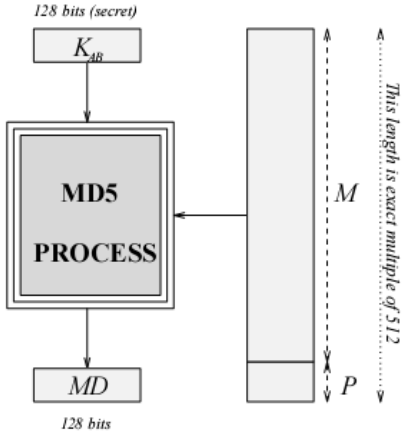
Pada metoden ini (gambar 1), sebuah kunci 128 bit K_{AB} digunakan sebagai *prefix* dari pesan M untuk

menghitung *message digest*. Alice melakukan langkah-langkah berikut untuk mengirim *M* ke Bob:

1. Gunakan kunci 128 bit $K_{AB,}$
2. Hitung MD = MD5 ($K_{AB} \oplus M$),
3. Kirimkan pasangan [M, MD] ke Bob.

Metode ini adalah hasil modifikasi dari *secret prefix method*, di mana dilakukan pemendekan kunci dari 512 ke 128 bit (4 kali) dan operasi XOR (bukan *prepend*) kunci dengan awal dari pesan agar mempercepat proses, penghematan 1 satu blok.

4.2 Metode 2, K_{AB} sebagai *initialization vector* MD5



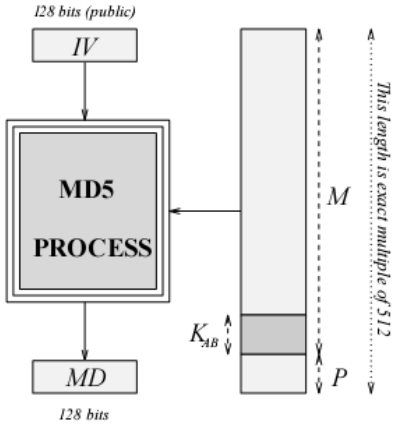
Gambar 2: K_{AB} (128 bit) sebagai *initialization vector MD5*.

Sebagai ganti penambahan kunci pada pesan, *initialization vector* dari MD5 dapat digunakan sebagai 128 bit kunci rahasia. Lalu, fungsi hash memulai pemrosesan pesan. Metode ini diilustriasikan pada gambar 2 di atas. Langkah-langkahnya adalah sebagai berikut:

1. Gunakan kunci rahasia 128 bit sebagai K_{AB}
2. Set $IV = K_{AB}$
3. Hitung MD = MD5(M),
4. Kirimkan pasangan [M, MD].

Karena tidak ada proses konkatenasi pada metode ini, maka kecepatannya sama dengan algoritma MD5.

4.3 Metode 3, K_{AB} sebagai *suffix M*



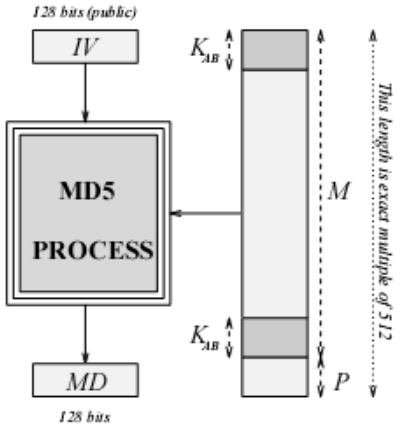
Gambar 3: K_{AB} (128 bit) sebagai *suffix M*

Seperti terlihat pada gambar 3, kunci 128 bit K_{AB} di-XOR kan dengan akhir dari pesan M sebelum dicari *message digest*-nya. Ketika Alice ingin mengirim M ke Bob, maka yang dilakukan adalah:

1. Gunakan kunci 128-bit K_{AB},
2. Hitung MD = MD5 (M $\oplus$ K_{AB}),
3. Kirimkan pasangan [M, MD] ke Bob.

Metode ini merupakan modifikasi dari *secret suffix method*, di mana panjang kunci dipendekkan dari 512 ke 128 bit (4 kali) dan kunci di-XOR kan dengan pesan sebagai ganti dari *append*.

4.4 Metode 4, M di tengah-tengah K_{AB}



Gambar 4: pesan M berada di tengah dua kunci identik K_{AB}

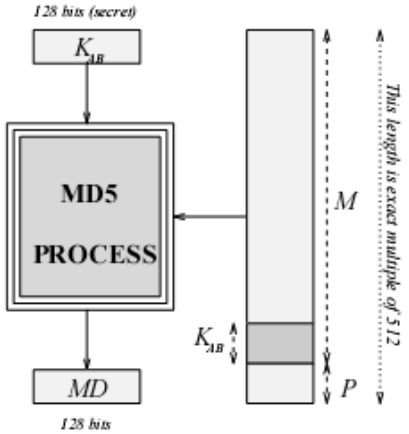
Pada metode ini, kunci K_{AB} dipergunakan sebanyak dua kali untuk menambah keamanan (kombinasi dari metode pertama dan ketiga). Langkah-langkahnya adalah sebagai berikut:

1. Gunakan kunci 128-bit K_{AB},
2. Hitung MD = MD5 ($K_{AB} \oplus$ M $\oplus$ K_{AB}),
3. Kirimkan pasangan [M, MD].

Metode ini hanya menggunakan kunci 128 bit yang berarti 8 kali lebih kecil dari yang digunakan pada

envelope method Tsudik. Pemendekan kunci ini juga mempercepat proses sebanyak dua blok. Perlu dicatat bahwa, jika panjang *M* kurang dari 256 bit, kedua kunci akan *overlap*, oleh karena itu harus digunakan *padding*.

4.5 Metode 5, K_{AB} sebagai *initialization vector* dan *suffix* M



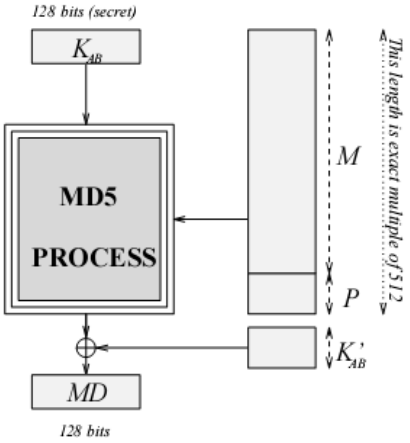
Gambar 5, kunci K_{AB} sebagai *initialization vector* dan *suffix*

Ini merupakan kombinasi dari metode 2 dan 3, seperti ilustrasi pada gambar 5. Langkah-langkahnya adalah sebagai berikut:

1. Gunakan kunci K_{AB},
2. Set $IV = K_{AB}$,
3. Hitung MD = **MD5** (M $\oplus$ K_{AB}),
4. Kirimkan pasangan [M, MD].

Metode ini mirip dengan metode ke-4, yakni kunci hanya berukuran 128 bit, dan kecepatannya hampir sama dengan MD5 standard.

4.6 Metode 5, K_{AB} sebagai *initialization vector* dan K'_{AB} sebagai *message digest*



Gambar 6, Sebuah kunci 128 bit untuk *initialization vector* dan kunci 128 bit lain sebagai *masking* pesan *M*

Metode ini, menggunakan 128 bit kunci K'_{AB} untuk meng-XOR-kan hasil MD5. Berikut adalah langkah-langkahnya:

1. Gunakan kunci K_{AB} dan K'_{AB} (256 bit),
2. Set $IV = K_{AB}$,
3. Hitung MD = **MD5**(M) $\oplus$ K'_{AB},
4. Kirimkan pasangan [M, MD].

Pada metode ini, *message digest* diproteksi dengan menambah lapisan keamanan pada awal dan akhir proses. Kecepatan metode ini hampir sama dengan algoritma MD5 biasa, namun kunci 256 bit diperlukan. Metode ini adalah metode satu-satunya yang menggunakan kunci 256 bit, namun tetap 4 kali lebih kecil dari pada *envelope method* Tsudik.

5. *Secured Key One-Way Hash Function*

Menurut Berson [3], definisi dari sebuah *Secured Key One-Way Hash Function* adalah:

Definisi 1 *Sebuah fungsi f() yang memetakan suatu kunci K dengan panjang tetap (fixed) dan sebuah pesan dengan panjang berapapun (arbitrary) , jika memenuhi sifat- sifat berikut:*

1. *Diberikan K dan M, mudah untuk mencari MD = f(M,K),*
2. *Diberikan K dan MD, sulit untuk mencari M dengan MD = f(M,K),*
3. *Diberikan K, sulit untuk mencari dua nilai M dan M' ($\neq$M) di mana f(M,K) =f (M',K),*
4. *Diberikan sepasang [M, MD], sulit untuk mencari kunci rahasia K,*
5. *Tanpa K, sulit menghitung f(M,K) untuk setiap M.*

Secara umum, sifat- sifat di atas sangat penting untuk menjamin suatu fungsi *hash* tahan terhadap serangan. Sifat pertama menjamin bahwa algoritma mudah dihitung. Sifat kedua berarti kesearahan fungsi jika diberikan suatu kunci. Sifat ketiga menjamin *collision freeness* fungsi. Sifat keempat menjaga kunci hanya diketahui pihak-pihak yang terlibat. Sementara sifat terakhir mencegah pemunculan suatu *digest* palsu.

Berdasarkan definisi di atas, ketika K ditentukan, SKOWHF menjadi suatu fungsi *hash* normal biasa (*One-Way Hash Function*). Keberadaan kunci rahasia menyebabkan desainer dapat menentukan algoritma sesuai dengan kebutuhan keamanan dan efisiensi.

Preposisi 1 *Keenam metode memenuhi seluruh sifat SKOWHF, di mana f() adalah MD5, kunci adalah K_{AB} (dan K'_{AB} pada metode 6).*

Pembuktian:
MD5 adalah suatu fungsi *hash* yang tergolong cepat dan $MD = f(M, K_{AB})$ dapat dihitung dengan mudah. Hal ini membuktikan bahwa metode tersebut memiliki **sifat pertama**.

Sifat kedua SKOWHF menekankan bahwa fungsi tidak boleh bersifat *reversible*. Pada keenam metode di atas, karena pesan M selalu didampingi oleh suatu *message digest MD*, kesearahan tidak diperlukan lagi (secara umum, metode tersebut sudah searah). Ditambah lagi, pencarian M' untuk menghasilkan suatu *MD* yang sama harus sulit. Dengan kata lain, ketika [M, *MD*] dengan $MD = f(M,K)$ didapat, pencarian M tidak perlu sulit bila diketahui K dan *MD*, namun pencarian M' harus sulit di mana $f(M',K) = f(M,K)$.

Pandang metode 1, di mana $f(M,K) = \mathbf{MD5}(K_{AB} \oplus M)$. Jika mencari $M \neq M'$ di mana $f(M,K_{AB}) = f(M',K_{AB})$ adalah mudah, kita set $X = (K_{AB} \oplus M)$ dan $X' = (K_{AB} \oplus M')$. Ini berarti mudah untuk mencari $X = X'$ di mana:

$\mathbf{MD5}(X) = \mathbf{MD5}(K_{AB} \oplus M) = f(M,K_{AB}) = f(M',K_{AB}) = \mathbf{MD5}(K_{AB} \oplus M') = \mathrm{MD5}(X')$.

Dengan kata lain, mencari *collision* pada MD5 adalah mudah, sebuah kontradiksi. Hal ini membuktikan **sifat ketiga**.

Untuk sifat keempat, kita harus membuktikan bahwa, menghitung suatu kunci ketika diberikan pasangan pesan dan *message digest* adalah sulit. Pada metode-metode di atas, kunci di-XOR-kan dengan pesan (misalnya metode 1). Hal ini terbukti karena pada pemrosesean setiap blok, nilai awal selalu ditambahkan pada hasil dari blok tersebut (*loop*), hal ini menyebabkan operasi mundur (*backwards*) tidak mungkin (*one-wayness*).

Pada metode 6, hasil dari algoritma, sebelum di-XOR-kan dengan kunci K_{AB}, tidak diketahui (karena *vector initialization* adalah rahasia). *Intermediate value* ini disebut dengan X. Pertanyan dapat diganti menjadi "Apakah sulit untuk mencari K'_{AB} dari $X \oplus K'_{AB} = MD$ ketika hanya *MD* yang diketahui?" Jawabannya adalah ya, sehingga **sifat keempat** telah terbukti.

Untuk **sifat kelima**, pembuktian yang mirip dengan sebelumnya digunakan. Perlu diketahui bahwa pada MD5 *message digest* sangat bergantung terhadap *initialization vector* dan pesannya. Sehingga, kesalahan beberapa bit masukan saja dapat mempengaruhi setiap bit pada *message digest*. Hal ini menjamin bahwa penghitungan $f(M,K)$ tanpa diketahui K adalah tidak mungkin.

6. Kesimpulan

Pada keenam metode yang diberikan, tidak hanya ada pemendekan kunci sampai 87.5% saja, namun juga digunakannya operasi XOR untuk menggantikan konkatenasi kunci pada pesan. Keberdaan kunci rahasia tidak menambah panjang masukan, sehingga tidak mengurangi kecepatan proses (waktu untuk XOR dapat diabaikan). Perlu dicatat bahwa kecepatan adalah alasan utama digantinya algoritma *hash-then-encrypt* dengan fungsi *hash* berkunci.

DAFTAR PUSTAKA

[1] Tsudik, Gene, "*Message Authentication with One Way Hash Functions*", IEEE INFOCOM, 1992.

[2] Rivest, Ronald L., "*The MD Message Digest Algorithm*", RFC 1321, Network Working Group MIT Laboratory for Computer Science and RSA Data Security Inc., 1992.

[3] T. A. Berson, L. Gong, dan T. M. A. Lomas, "*Secure Keyed and Collisionful Hash Functions*", Technical Report SRI-CSL-94-08, SRI International Laboratory Menlo Park, California, 1993.

[4] Munir, Rinaldi, "*Bahan Kuliah IF5054 Kriptografi*", Departemen Teknik Informatika, Institut Teknologi Bandung, 2004.

[5] J Pieprzyk dan B Sadeghiyan, "*Design of Hashing Algorithms*", Springer Verlag, 1993.