Algoritma MAC Berbasis Jaringan Syaraf Tiruan

Paramita [1)]

1) Program Studi Teknik Informatika STEI ITB, Bandung, email: if14040@students.if.itb.ac.id

Abstract – *MAC adalah fungsi hash satu arah yang menggunakan kunci rahasia (secret key) dalam pembangkitan nilai hash. Dengan kata lain, nilai hash adalah fungsi dari pesan dan kunci. Di dalam makalah ini, penulis mengajukan fungsi hash dengan menggunakan jaringan syaraf tiruan untuk menghasilkan nilai hash. Panjang kunci rahasia ditetapkan sebesar 128 bit, sedangkan pesan akan dipecah menjadi blok-blok berukuran 128 bit. Untuk melatih jaringan syaraf tiruan, akan digunakan beberapa rangkaian bit berukuran 128 bit.*

Kata Kunci*: Algoritma MAC, fungsi hash, kunci rahasia, jaringan syaraf tiruan.*

1. PENDAHULUAN

MAC adalah fungsi *hash* satu arah yang menggunakan kunci rahasia (*secret key*) dalam pembangkitan nilai *hash*. Dengan kata lain, nilai *hash* adalah fungsi dari pesan dan kunci.

Di dalam makalah ini, penulis mengajukan fungsi *hash C* dengan menggunakan jaringan syaraf tiruan (*Neural Network*). Jaringan syaraf tiruan merupakan salah satu teknik pembelajaran mesin. Hal ini untuk menjaga agar fungsi *hash* tidak diketahui, karena sifat dari jaringan syaraf tiruan adalah fungsi di dalamnya sulit untuk diketahui.

Panjang kunci rahasia ditetapkan sebesar 128 bit. Akan tetapi, karena panjang pesan tidak pasti, maka dibuat jaringan syaraf tiruan bertingkat, dan pesan dipecah menjadi blok-blok berukuran 128 bit. Mengenai jaringan syaraf bertingkat ini akan dijelaskan lebih lanjut pada makalah ini.

Untuk melatih jaringan syaraf tiruan, akan digunakan beberapa rangkaian bit berukuran 128 bit. Rangkaian bit akan dibangkitkan secara acak.

Untuk menguji algoritma *MAC* yang diusulkan oleh penulis, akan dibuat aplikasi jaringan syaraf tiruan sederhana yang akan digunakan untuk mengaplikasikan algoritma *MAC*. Aplikasi sederhana ini akan dibuat dengan menggunakan pustaka yang disediakan oleh NeuroBox [3].

2. DASAR TEORI

2.1. *MAC* dan Aplikasinya

MAC adalah fungsi *hash* satu-arah yang menggunakan kunci rahasia (*secret key*) dalam pembangkitan nilai *hash.* Dengan kata lain nilai *hash* adalah fungsi dari pesan dan kunci. *MAC* disebut juga *keyed hash function* atau *key-dependent one-way hash function*. *MAC* memiliki sifat dan properti yang sama seperti fungsi *hash* satu-arah yang telah didiskusikan sebelumnya, hanya saja ada komponen kunci. Kunci digunakan oleh penerima pesan untuk memverifikasi nilai *hash*.

Secara matematis, *MAC* dinyatakan sebagai:

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

yang dalam hal ini, *MAC* = nilai *hash*, *C* = fungsi *hash* (atau algoritma *MAC*), dan *K* = kunci rahasia. Fungsi *C* memampatkan pesan *M* yang berukuran sembarang dengan menggunakan kunci *K*. Fungsi *F* adalah fungsi *many-to-one*, yang berarti beberapa pesan berbeda mungkin memiliki *MAC* yang sama, tetapi menemukan pesan-pesan semacam itu sangat sulit (secara komputasi).

MAC digunakan untuk otentikasi pesan tanpa perlu merahasiakan (mengenkripsi) pesan. Mula-mula pengirim pesan menghitung *MAC* dari pesan yang hendak ia kirim dengan menggunakan kunci rahasia *K* (diasumsikan bahwa sebelum transmisi pengirim dan penerima sudah berbagi kunci rahasia). Kemudian *MAC* ini dilekatkan (*embedded*) pada pesan. Selanjutnya pesan dikirm bersama-sama dengan *MAC* ke penerima. Penerima kemudian menggunakan kunci yang sama untuk menghitung *MAC* pesan dan membandingkannya dengan *MAC* yang diterimanya. Jika kedua *MAC* ini sama maka ia menyimpulkan bahwa pesan dikirim oleh orang yang sesungguhnya dan isi pesan tidak diubah oleh pihak ketiga selama transmisi. Jika pesan tidak berasal dari pengirim yang asli, maka *MAC* yang ia hitung tidak sama dengan *MAC* yang diterima, sebab pihak ketiga tidak mengetahui kunci rahasia. Begitu juga jika pesan sudah diubah selama transmisi, maka *MAC* yang ia hitung tidak sama dengan *MAC* yang diterima.

Aplikasi *MAC* lainnya adalah untuk menjaga otentikasi arsip yang digunakan oleh dua atau lebih pengguna. Selain itu *MAC* juga digunakan untuk menjamin integritas (keaslian) isi arsip terhadap perubahan, misalnya karena serangan virus. Caranya: hitung *MAC* dari arsip, kemudian simpan *MAC* di dalam sebuah tabel basis data. Jika pengguna, menggunakan fungsi *hash* satu-arah biasa (seperti MD5), maka virus dapat menghitung nilai *hash* yang baru dari arsip yang sudah diubah, lalu mengganti

nilai *hash* yang lama di dalam tabel. Tetapi, jika digunakan *MAC*, virus tidak dapat melakukan hal ini karena ia tidak mengetahui kunci.

2.2. Algoritma *MAC*

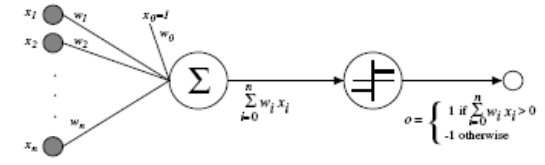
Algoritma *MAC* dapat dirancang dengan dua pendekatan. Pendekatan pertama menggunakan algoritma kriptografi kunci-simetri berbasis blok (*cipher block*), sedangkan pendekatan kedua menggunakan fungsi *hash* satu-arah.

Untuk pendekatan kedua, fungsi *hash* satu-arah seperti MD5 dapat digunakan sebagai *MAC*. Caranya adalah pesan *M* yang akan dihitung nilai *hash*-nya disambung (*concatenate*) dengan kunci rahasia *K*. Kemudian hasil penyambungan tersebut dihitung nilai *hash*-nya dengan menggunakan fungsi *hash*. Nilai *hash* ini adalah *MAC* dari pesan tersebut. Pengirim pesan akan mengirimkan pesan *M* dan *MAC.* Kemudian penerima pesan akan menyambung *M* dengan kunci *K* yang ia miliki untuk memperoleh *MAC*. *MAC* yang ia peroleh akan dibandingkan dengan *MAC* yang dikirimkan oleh pengirim untuk otentikasi pesan.

Di dalam makalah ini akan dibahas mengenai algoritma *MAC* yang dirancang dengan menggunakan pendekatan kedua, yaitu dengan menggunakan fungsi *hash* satu-arah. Akan tetapi fungsi *hash* yang digunakan adalah fungsi jaringan syaraf tiruan bertingkat. Penjelasan lebih lanjut mengenai aplikasi jaringan syaraf tiruan dalam algoritma *MAC* dapat ditemukan pada bagian selanjutnya dari makalah ini.

2.3. Jaringan Syaraf Tiruan

Salah satu tipe dari sistem jaringan syaraf tiruan didasarkan pada sebuah unit yang disebut sebagai *perceptron*, seperti yang dapat dilihat pada Gambar 1 [2].



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Gambar 1 *Perceptron* pada jaringan syaraf tiruan

Sebuah *perceptron* menerima sebuah vektor dari masukan bernilai real, menghitung kombinasi linear dari masukan-masukan tersebut, dan menghasilkan nilai 1 jika hasil lebih besar dari ambang batas tertentu, dan -1 jika sebaliknya, seperti yang dapat dilihat pada Gambar 1.

Proses pembelajaran pada sebuah *perceptron* dilakukan dengan mengkoreksi nilai bobot untuk setiap masukan ($w_0, \ldots, w_n$).

Dalam implementasinya, jaringan syaraf tiruan dapat lebih kompleks daripada hanya terdiri dari sebuah *perceptron*. Keluaran tidak hanya terdiri dari sebuah simpul, melainkan beberapa simpul. Selain itu, di antara layer masukan dan keluaran mungkin terdapat beberapa layer antara yang disebut *hidden layer*.

Algoritma yang umum digunakan untuk proses pembelajaran jaringan syaraf tiruan adalah algoritma *Backpropagation*. Berikut ini adalah gambaran umum dari algoritma *Backpropagation* [2].

BACPROPAGATION(*training_examples*,η,n_{in},n_{out},n_{hidden})

- Bangun jaringan dengan jumlah unit masukan n_{in}, jumlah unit antara n_{hidden}, dan jumlah unit keluaran n_{out}.
- Inisialisasi nilai semua bobot dalam jaringan dengan bilangan acak kecil (misalnya antara -0.05 sampai 0.05).
- Sampai kondisi terminasi terpenuhi, lakukan:
  - Untuk setiap $\langle \vec{x}, \vec{t} \rangle$ pada *training_examples*, lakukan:

    Propagasi maju nilai masukan di dalam jaringan

    1. Masukkan $\vec{x}$ ke dalam jaringan dan hitung keluaran o_u untuk setiap unit *u* di dalam jaringan.

    Propagasi balik nilai error di dalam jaringan

    2. Untuk setiap unit keluaran *k*, hitung nilai error δ_k

    $$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

    3. Untuk setiap unit antara *h*, hitung nilai error δ_h

    $$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

    4. Update nilai setiap bobot w_{ji} dalam jaringan

    $$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

    dimana

    $$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

Algoritma jaringan syaraf tiruan memiliki karakteristik-karakteristik sebagai berikut [2]:

1. Masukan dapat berupa nilai diskrit atau real yang memiliki banyak dimensi.
2. Keluaran berupa vektor yang terdiri dari beberapa nilai diskrit atau real.
3. Dapat mempelajari permasalahan secara *black box*, dengan hanya mengetahui nilai masukan serta keluarannya saja.
4. Mampu menangani pembelajaran terhadap data yang memiliki derau (*noise*).
5. Bentuk dari fungsi target pembelajaran tidak diketahui, karena hanya berupa bobot-bobot nilai masukan pada setiap neuron.
6. Karena harus mengubah banyak nilai bobot pada proses pembelajaran, maka waktu pembelajaran menjadi lama, sehingga tidak cocok untuk masalah-masalah yang memerlukan waktu cepat dalam pembelajaran.
7. Jaringan saraf tiruan hasil pembelajaran tiruan dapat dijalankan dengan cepat.

3. RANCANGAN *MAC* DENGAN JARINGAN SYARAF TIRUAN

3.1. Langkah-langkah Pembuatan *MAC*

Berikut ini adalah langkah-langkah pembuatan *MAC* secara garis besar:

1. Penambahan bit-bit pengganjal (*padding bits*) sehingga pesan menjadi berukuran kelipatan 128 bit, dikurangi 8 bit untuk menyimpan panjang bit-bit pengganjal.
2. Penambahan jumlah bit-bit pengganjal berukuran 8 bit.
3. Pengolahan pesan dalam blok berukuran 128 bit dengan menggunakan jaringan syaraf tiruan.
4. Peng-*XOR*-an hasil pengolahan pesan berukuran 128 bit dengan kunci berukuran 128 bit.

3.2. Penambahan Bit-bit Pengganjal

Agar pesan dapat diubah menjadi blok-blok berukuran 128 bit, maka pesan harus ditambahkan dengan bit-bit pengganjal, sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 120 modulo 128. Ini berarti panjang pesan setelah ditambah bit-bit pengganjal adalah 8 bit kurang dari kelipatan 128.

Angka 128 bit muncul karena fungsi *hash* yang digunakan akan memproses pesan dalam blok-blok yang berukuran 128. Sedangkan 8 bit ini nantinya akan digunakan untuk menyimpan jumlah bit pengganjal yang digunakan, karena panjang bit-bit pengganjal maksimum yang mungkin digunakan adalah sejumlah 128 bit.

Pesan dengan panjang 120 bit pun akan tetap ditambah dengan 128 bit pengganjal sehingga menjadi 248 bit. Kemudian ditambah 8 bit lagi untuk menyimpan jumlah bit-bit pengganjal sehingga total panjang pesan menjadi sejumlah 256 bit (kelipatan 128 bit). Bit-bit pengganjal yang digunakan akan terdiri dari sebuah bit 1 diiku
ti dengan sisanya bit 0.

3.3. Penambahan Nilai Panjang Bit-bit Pengganjal

Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 8 bit yang menyatakan panjang pesan semula. Dengan penambahan jumlah bit-bit pengganjal ini, panjang pesan akan menjadi kelipatan 128 bit.

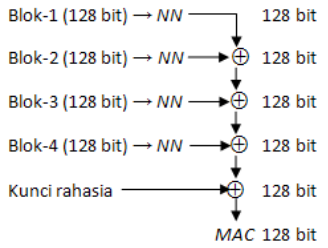
3.4. Pengolahan Pesan untuk Menghasilkan *MAC*

Alasan pemecahan pesan ke dalam blok-blok pesan adalah karena jumlah masukan dan keluaran dari jaringan syaraf tiruan harus tetap dan pasti. Solusinya adalah dengan membuat jaringan syaraf tiruan bertingkat, dan pesan dipecah menjadi blok-blok berukuran 128 bit. Agar pesan menjadi berukuran kelipatan 128 bit sudah dijelaskan pada bagian sebelumnya.

Misalkan, panjang pesan setelah ditambahkan dengan bit-bit pengganjal dan juga ditambahkan dengan panjang bit-bit pengganjal, sejumlah 512 bit. Pesan akan dipecah menjadi 4 blok berukuran 128 bit. Masing-masing blok pesan akan dimasukkan ke dalam jaringan syaraf tiruan. Hasil yang diperoleh dari blok pertama akan di-*XOR*-kan dengan hasil yang diperoleh dari blok kedua. Kemudian hasil *XOR* tersebut akan di-*XOR*-an lagi dengan hasil yang diperoleh dari blok ketiga, dan seterusnya.

Hasil akhir dari blok terakhir akan di-*XOR*-kan dengan kunci rahasia yang juga berukuran 128 bit sehingga menghasilkan *MAC* berukuran 128 bit.

Gambar 2 mengilustrasikan pemrosesan blok-blok pesan tersebut hingga menghasilkan *MAC* berukuran 128 bit.

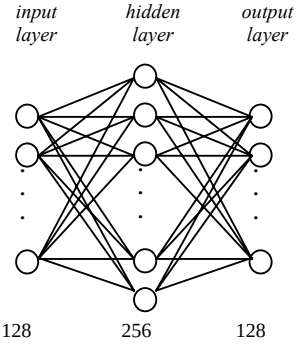


Gambar 2 Pengolahan pesan dengan jaringan syaraf tiruan bertingkat

3.5. Rancangan Jaringan Syaraf Tiruan

Jaringan syaraf tiruan yang digunakan dalam penghitungan nilai *MAC* ini merupakan jaringan syaraf tiruan sederhana dengan masukan 128 bit dan keluaran juga 128 bit.

Jumlah simpul masukan sebanyak 128 buah, dan simpul keluaran juga sebanyak 128 buah. Di antara layer masukan dan keluaran dibangun sebuah layer antara dengan jumlah simpul sebanyak 256 buah. Pemilihan jumlah simpul untuk masukan dan keluaran didasarkan pada rancangan algoritma *MAC* yang memiliki masukan 128 bit dan keluaran 128 bit. Sedangkan pemilihan jumlah simpul antara tidak didasarkan pada apapun.



Gambar 3 Arsitektur jaringan syaraf tiruan

4. IMPLEMENTASI

4.1. Lingkungan Implementasi

Implementasi dilakukan dengan menggunakan perangkat lunak Microsoft Visual Studio 2005 C#, dengan spesifikasi lingkungan implementasi:

- Microsoft Windows XP Professional SP2
- Intel(R) Core(TM)2 Duo processor T5600 @ 1.83GHz
- RAM 512 MB
- Microsoft .NET Framework 2.0

4.2. Pelatihan Jaringan Syaraf Tiruan

Pelatihan jaringan syaraf tiruan menggunakan 5 buah data pelatihan dengan format 128 bit masukan dan 128 bit keluaran. Data pelatihan tersebut dibangkitkan secara acak, kemudian disimpan ke dalam file untuk kemudian digunakan oleh proses pembelajaran.

4.3. Proses Pembelajaran Jaringan Syaraf Tiruan

Implementasi jaringan syaraf tiruan menggunakan pustaka yang disediakan oleh NeuroBox [3]. Algoritma pembelajaran yang digunakan adalah algoritma *Backpropagation*.

Dari hasil implementasi ditemukan bahwa proses yang membutuhkan waktu lama adalah proses pembangunan jaringan syaraf tiruan dengan arsitektur jaringan seperti pada Gambar 3, yaitu ± 3 detik. Sedangkan proses pembelajarannya sendiri, dengan 5 buah data pelatihan, hanya membutuhkan waktu kurang dari 1 detik.

4.4. Hasil Pengolahan Pesan Berupa *MAC*

Berikut ini adalah hasil pembelajaran dan pengolahan pesan sehingga menghasilkan *MAC*.

Gambar 4 Hasil pengolahan pesan berupa *MAC*

Proses pengolahan pesan menggunakan jaringan syaraf tiruan untuk menghasilkan *MAC* membutuhkan waktu yang relatif cepat. Berikut ini adalah hasil pengujian untuk beberapa file dengan ukuran yang berbeda (untuk kemudahan proses pengujian, hanya digunakan file teks). Kunci yang digunakan untuk setiap pengujian adalah sama, yaitu '1234567890987654' (16 byte).

Tabel 1 Hasil pengujian

Kategori ukuran	Ukuran file (byte)	Waktu
< 1 KB	31	< 1 detik
< 50 KB	44.923	43,89 detik
< 100 KB	74.867	53,10 detik

Akan tetapi, untuk file-file berukuran besar (> 1 MB), proses pengujian tidak dapat dilakukan dengan baik, karena terdapat keterbatasan memori. Untuk selanjutnya, mungkin dapat dilakukan pengubahan pada arsitektur jaringan syaraf tiruan yang digunakan dan algoritma yang digunakan supaya dapat diperoleh hasil maksimal dengan waktu dan penggunaan memori yang seminimal mungkin.

5. ANALISIS

5.1. Kelebihan

Salah satu kelebihan dari penggunaan jaringan syaraf tiruan untuk menghasilkan *MAC* adalah waktu yang dibutuhkan relatif cepat. Selain itu, nilai *MAC* dihasilkan dari fungsi yang tidak dapat diketahui, karena bergantung pada pembelajaran jaringan syaraf tiruan yang dihasilkan. Hal ini dapat meningkatkan keamanan, karena walaupun pihak ketiga (penyerang) memiliki kunci, dia tetap tidak dapat menghasilkan *MAC* jika tidak memiliki jaringan syaraf tiruan yang sama dengan pihak pengirim dan penerima.

5.2. Kekurangan

Kekurangan dari penggunaan jaringan syaraf tiruan untuk menghasilkan *MAC* adalah pihak pengirim dan pihak penerima harus memiliki jaringan syaraf tiruan yang sama persis agar dapat menghasilkan nilai *MAC* yang sama.

Yang dimaksud dengan sama di sini bukan hanya sama dari segi arsitektur jaringan saja, melainkan juga dari hasil pembelajaran (nilai bobot-bobot di dalam jaringan). Dari hasil implementasi diperoleh bahwa dengan menggunakan data pelatihan yang sama persis, tidak dapat menjamin jaringan syaraf tiruan yang dihasilkan akan sama, sehingga dapat menghasilkan nilai *MAC* yang berbeda.

Untuk itu harus dibuat suatu mekanisme yang dapat menjamin bahwa pengirim dan penerima menggunakan jaringan syaraf tiruan yang sama.

6. KESIMPULAN

Dengan menggunakan jaringan syaraf tiruan, akan sulit untuk megetahui fungsi yang digunakan untuk menghasilkan *MAC*. Hal ini akan mempersulit penyerang, karena walaupun penyerang memiliki kunci, penyerang tetap tidak dapat menghasilkan *MAC* yang sama jika tidak memiliki jaringan syaraf tiruan yang sama dengan pengirim dan penerima.

Akan tetapi, salah satu kesulitan dari penggunaan jaringan syaraf tiruan sebagai fungsi *hash* untuk menghitung *MAC* adalah pengirim dan pengguna harus memiliki jaringan syaraf tiruan yang sama, sebab jika tidak maka *MAC* yang dihasilkan akan berbeda.

Keuntungan lain dari penggunaan jaringan syaraf tiruan untuk menghitung *MAC* adalah waktu pemrosesan yang relatif cepat. Hal ini disebabkan karena salah satu karakterisitk dari jaringan syaraf tiruan yaitu hasil pembelajaran dapat dijalankan dengan cepat, walaupun untuk proses pembelajarannya itu sendiri membutuhkan waktu yang lebih lama.

DAFTAR REFERENSI

[1] Munir, Rinaldi, *Diktat Kuliah IF5054: Kriptografi*, Institut Teknologi Bandung, 2006.

[2] Mitchell, Tom M., *Machine Learning*, McGraw-Hill, 1997.

[3] http://www.cdrnet.net/projects/neuro/