Analisis Performansi dan Keamanan dari *Message Authentication Code* (MAC) Berbasis Fungsi Hash Satu Arah

Zulva Fachrina
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jalan Ganesha 10 Bandung 40132, Indonesia
13513010@std.stei.itb.ac.id

***Abstrak*— Serangan terhadap pesan yang dikirimkan melalui jaringan yang tidak aman merupakan salah satu masalah dalam sistem keamanan informasi. Salah satu cara yang dapat dilakukan untuk menjaga otentikasi dan integritas dari pesan yang dikirimkan adalah dengan menggunakan *Message Authentication Code* (MAC). MAC paling umum diimplementasikan dengan menggunakan fungsi hash satu arah, seperti MD5, SHA1, dan SHA256. Performansi dan keamanan dari pesan dengan menggunakan MAC berbasis fungsi hash (HMAC) bergantung pada fungsi hash yang dipilih. Karena itu, dalam makalah ini akan dilakukan eksperimen dengan menggunakan fungsi hash SHA1, SHA256, MD5 pada HMC untuk mengukur performansi berdaarkan waktu eksekusi dan komputasi CPU. Selain itu, akan dilakukan analisis keamanan dengan menguji fungsi HMAC terhadap serangan *length extension attack*.**

Kata Kunci—Hash, MD5, Message Authentication Code, SHA1, SHA256,

I. PENDAHULUAN

Dalam berkomunikasi, jarak seringkali menjadi kendala yang mengakibatkan pertukaran informasi tidak dapat dilakukan *face to face*. Untuk menyampaikan informasi yang tidak dapat disampaikan secara langsung, diperlukan media yang dapat membantu menyampaikan pesan kepada penerima. Dengan semakin berkembangnya teknologi, media yang digunakan semakin beragam. Pesan dapat dikirimkan melalui media elektronik dalam bentuk *email*, SMS, atau *chat.* Penggunaan media elektronik tersebut sangat membantu dalam mengefisiensikan waktu dan biaya, namun masih belum menjamin keamanan pesan yang dikirimkan.

Serangan terhadap pesan yang dikirimkan melalui media elektronik merupakan isu krusial dalam keamanan jaringan dan informasi. Saluran pertukaran informasi yang tidak aman mengakibatkan pihak ketiga dapat memodifikasi dan menghapus pesan sebelum diterima oleh penerima. Penerima tidak memiliki mekanisme untuk mengetahui apakah pesan yang dikirimkan merupakan pesan yang asli atau tidak, karena pada dasarnya penerima tidak mengetahui pesan yang akan dikirimkan. Pihak ketiga juga dapat mengirimkan pesan yang berbeda dengan berpura-pura menjadi pengirim sehingga penerima tidak mengetahui apakah pesan tersebut dikirimkan oleh pengirim yang sebenarnya. Serangan-serangan tersebut menjadi berbahaya karena menyangkut otentikasi dan integritas dari pesan.

Salah satu cara yang dapat dilakukan untuk menjaga otentikasi dan integritas dari pesan adalah dengan menggunakan *Message Authentication Code* (MAC). MAC menjamin pesan yang dikirimkan berasal dari pengirim yang sah dan pesan tidak diubah selama perjalanan. MAC menggunakan kunci rahasia yang hanya diketahui oleh pengirim dan penerima. Kunci rahasia tersebut memungkinkan terjadinya pengecekan, apakah pengirim juga memiliki kunci rahasia yang dimiliki bersama. Jika pengirim tidak memiliki kunci rahasia tersebut, hasil MAC akan berbeda dan penerima dapat mengetahui bahwa pesan dikirimkan oleh pengirim yang berbeda.

Algoritma MAC dapat diimplementasikan dengan menggunakan berbagai algoritma kriptografi, seperti algoritma *block cipher* dengan mode CBC atau CFB. Namun pemanfaatan algoritma MAC yang lebih umum digunakan adalah dengan menggunakan fungsi hash satu arah. MAC berbasis fungsi satu arah dianggap lebih efeien dan cepat dibandingkan dengan menggunakan *block cipher*. Meskipun begitu, performansi dan tingkat keamanan dari *Hash-based Message Authetication Code* (HMAC) sedikit sulit diukur karena sangat bergantung pada fungsi hash yang digunakan. Oleh karena itu, diperlukan analisis performansi untuk menentukan fungsi hash yang paling efisien untuk berbagai kasus. Dalam makalah ini, akan diukur performasi MAC berbasis fungsi satu arah dengan menggunakan fungsi hash MD5, SHA1, dan SHA256.

II. DASAR TEORI

A. Hash-based Message Authentication Code (HMAC)

Hash-based Message Authentication Code (HMAC) adalah salah satu metode implementasi MAC dengan mengombinasikan fungsi hash dengan kunci rahasia. Fungsi hash satu arah bertujuan untuk melakukan kompresi dari data masukan yang besar menjadi keluaran berukuran tetap dan

jauh lebih kecil. Fungsi hash memilki properti *collision resistance*, yaitu sangat sulit untuk menemukan dua buah pesan m_1 dan m_2 yang menghasilkan nilai hash yang sama hash(m_1) = hash(m_2). Properti ini digunakan pada HMAC untuk menjamin otentikasi dari pesan dengan mengombinasikannya dengan kunci rahasia. Secara garis besar, MAC berbasis fungsi hash satu arah terdiri dari 3 tahap, yaitu:

- Pengirim dan penerima menyepakati kunci rahasia bersama, atau dapat dibangkitkan secara random dengan menggunakan algoritma pembangkit kunci.
- MAC dibangkitkan berdasarkan kunci rahasia dan pesan yang dikirimkan dengan menggunakan fungsi hash yang dipilih.
- Pesan diverifikasi keasliannya berdasarkan MAC dan kunci rahasia yang dimiliki bersama.

HMAC membagi pesan menjadi sekumpulan blok dengan ukuran tertentu, kemudian menjalankan fungsi hash untuk menghasilkan MAC yang akan dilampirkan bersama pesan yang dikirimkan. HMAC didefinisikan dengan persamaan:

$$HMAC(K,m) = H\Big((K' \oplus opad) \,||\, H\big((K' \oplus ipad) \,||\, m\big)\Big)$$

dengan keterangan sebagai berikut:

- H = fungsi hash yang digunakan,
- K = kunci rahasia,
- m = pesan,
- K' = kunci rahasia lain yang diturunkan dari K dengan menambahkan angka 0 pada masukan atau dengan melakukan hashinh terhadap K jika lebih panjangan dari ukuran blok pesan,
- *opad = outer padding*, bilangan konstanta heksadesimal 0x5c5c5c…5c5c,
- *ipad = inner padding,* bilangan konstanta 0x363636…3636.

B. MD5

MD5 merupakan algoritma fungsi hash yang dirancang oleh Ronald Rivest pada tahun 1991 untuk menggantikan MD4. MD5 memproses pesan sehingga mengahasilkan keluaran dengan ukuran tetap 128 bit. Cara kerja algoritma MD5 adalah sebagai berikut:

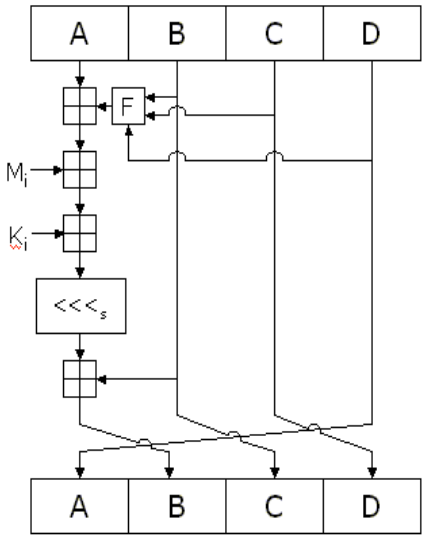
- Pesan dibagi dalam blok berukuran 512 bit dengan 64 bit ditambahkan pada blok terakhir. 64 bit tambahan tersebut digunakan untuk menyimpan panjang dari masukan. Jika blok terakhir berukuran kurang dari 512 bit, sejumlah bit tambahan ditambahkan untuk menghasilkan blok berukuran 512 bit. Kemudian, masing-masing blok dibagi menjadi 16 bagian dengan masing-masing berukuran 32 bit.
- Buffer diinisiasi dengan ukuran 32 bit. MD5 menggunakan 4 buah buffer, yaitu A, B, C, D.

$$A: 01\ 23\ 45\ 67$$
$$B: 89\ ab\ cd\ ef$$
$$D: fe\ dc\ ba\ 98$$
$$D: 76\ 54\ 32\ 10$$

- MD5 menggunakan tabel K yang terdiri dari 64 elemen. Masing-masing elemen dari tabel dikalkulasi dengan menggunakan persamaan di bawah ini.

$$K_i = abs(\sin(i+1)) * 2^{32}$$

- Blok masukan diproses dalam 4 ronde, dengan masing-masing ronde terdiri dari 16 operasi berdasarkan fungsi F, operasi modulus, dan *left rotation*. Operasi tersebut diilustrasikan seperti gambar di bawah ini.



Gambar 1 Operasi pada MD5

Buffer A,B,C dan D dikombinasikan dengan blok pesan M_1 dan konstanta pada tabel K_1 dengan menggunakan persamaan F. Persamaan F merupakan persamaan pada MD5 yang menerima masukan 32 bit dan memiliki 4 fungsi yang berbeda dan masing-masing digunakan pada ronde yang berbeda:

$$F(B,C,D) = (B \wedge C) \vee (\sim B \wedge D)$$
$$G(B,C,D) = (B \wedge D) \vee (C \wedge \sim D)$$
$$H(B,C,D) = B \oplus C \oplus D$$
$$I(B,C,D) = C \oplus (B \vee \sim D)$$

C. SHA1

SHA1 merupakan fungsi hash satu arah yang dikembangkan oleh NIST dan deigunakan bersama dengan DSS (*Digital Signature Standard*). Algortima SHA1 menghasilkan *message digest* yang panjangnya 160 bit, lebih panjang dari *message digest* yang dihasilkan oleh MD5. Tahapan dari algoritma SHA1 adalah sebagai berikut:

- Tambahkan bit penyangga pada pesan, dimulai dengan angka 1 dan diikuti dengan sejumlah angka 0 sehingga panjang bit kongruen dengan -64 ≡ 448 (mod 512). Pada akhir bit, tambahkan 64 bit yang

menyatakan panjang pesan sehingga panjang pesan sekarang merupakan kelipatan dari 512.

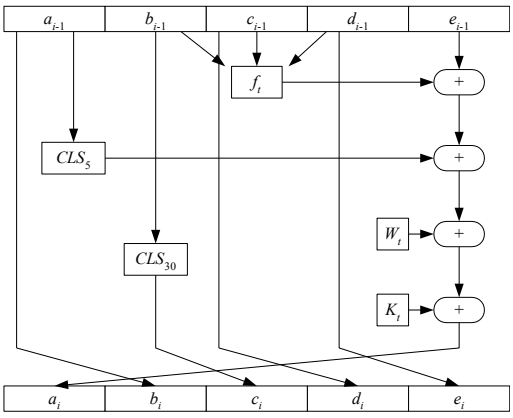
- Pesan dibagi menjadi blok dengan panjang 512 bit dan pemrosesan dilakukan pada masing-masing blok. Untuk masing-masing blok dibagi menjadi 16 bagian dengan masing-masing memiliki panjang 32 bit. 16 bagian tersebut diperbesar sehingga menjadi 80 bagian dengan panjang masing-masing 32 bit.
- Inisisasi penyangga yang digunakan pada SHA1. Terdapat 5 penyangga yang digunakan

$$A = 0x67452301$$
$$B = 0xEFCDAB89$$
$$C = 0x98BADCFE$$
$$D = 0x10325476$$
$$E = 0xC3D2E1F0$$

- Proses hash pada SHA terdiri dari 80 putaran, dengan masing masing putaran menggunakan bilangan penambah K yaitu:

$$Putaran\ 0 \leq t\ \leq 19, K = 0x5A827999$$
$$Putaran\ 20 \leq t \leq 39, K = 0x6ED9EBA1$$
$$Putaran\ 40 \leq t \leq 59, K = 0x8F1BBCDC$$
$$Putaran\ 60 \leq t \leq 79, K = 0xCA62C1D6$$

- Operasi dasar pada setiap putaran digambarkan dalam ilustrasi berikut ini.



Gambar 2 Operasi Dasar SHA1

Masing-masing putaran memilliki aturan fungsi yang berbeda. Terdapat 3 fungsi yang digunakan pada algoritma SHA1:

$$F(B,C,D) = (B \wedge C) \vee (\sim B\ \wedge D)$$
$$G(B,C,D) = B \oplus C \oplus D$$
$$H(B,C,D) = (B\ \wedge C) \vee (B \wedge D))$$

D. SHA256

SHA256 merupakan fungsi hash pengembangan dari SHA1 yang menghasilkan *message digest* sepanjang 256 bit. Berikut adalah langkah-langkah dari algoritma SHA256:

- Menginisiasi 8 buah buffer dengan nilai kuadrat dari 8 bilangan prima pertama antara 2 dan 19.
- Menginisiasi tabel konstanta yang merupakan 32 bit pertama dari bagian pecahan akar kuadrat dari 64 bilangan pertama antara 2 dan 31.
- Menambahkan angka 1 pada akhir pesan dan diikuti dengan sejumlah angka 0 dehingga panjang pesan menjadi kongruen dengan $64 \equiv 448 \pmod{512}$. Tambahkan 64 bit yang menyatakan panjang pesan sehingga panjang pesan sekarang menjadi kelipatan 512.
- Pesan dibagi menjadi blok-blok dengan panjang 512 bit. Operasi dilakukan pada masing-masing blok. Sebelumnya, blok yang memiliki panjang 512 bit dibagi menjadi 16 bagian dengan panjang masing-masing 32 bit dan diperbesar hingga menjadi 64 bagian dengan panjang masing-masing 64 bit.
- Pemrosesan dilakukan sebanyak 64 putaran dengan masing-msaing putaran menggunakan fungsi yang sama

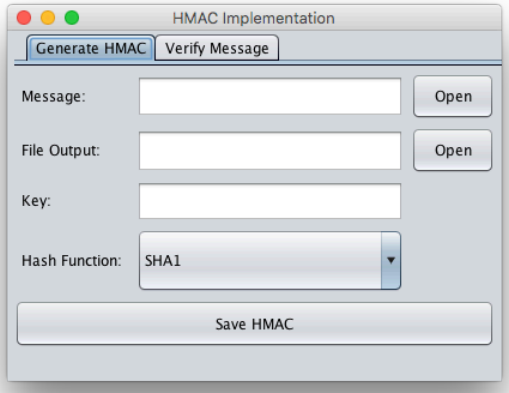
$$Ch(E,F,G) = (E \wedge F) \oplus (\sim E\ \wedge G)$$
$$Ma(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$F(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$G(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

III. IMPLEMENTASI

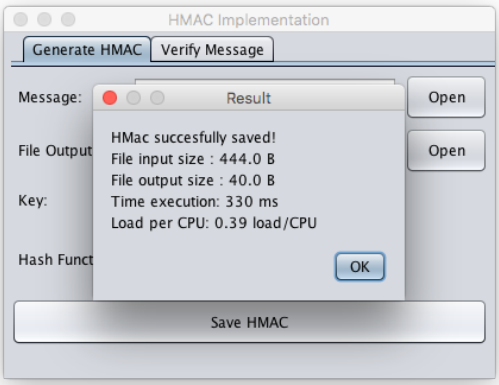
Untuk melakukan analisis performansi fungsi hash pada *Message Authentication Code* (MAC), sebuah program diikembangkan untuk mempermudah proses analisis. Program dikembangkan dalam bahasa pemrograman Java dan menggunakan Java Swing sebagai Graphical User Interface (GUI). Program memanfaatkan kelas bawaan Java *crypto.Mac* untuk melakukan perhitungan MAC dengan menggunakan fungsi hash SHA1, SHA256, dan MD5.

Program yang diimplementasikan memiliki 2 fungsionalitas utama, yaitu untuk membangkitkan nilai MAC berdasarkan fungsi hash yang dipilih dan melakukan verifikasi terhadap nilai HMAC dan pesan yang dimasukkan. Program juga dapat menerima masukan kunci dengan panjang sesuai keinginan pengguna. Analisis performansi dilakukan dengan membandingkan waktu eksekusi, ukuran file output, dan jumlah rata-rata load CPU yang ditampilkan oleh program ketika membangkitkan nilai HMAC. Analisis keamanan dilakukan dengan menerapkan *length extension attack* ketika melakukan verifikasi pesan dengan nilai MAC-nya.

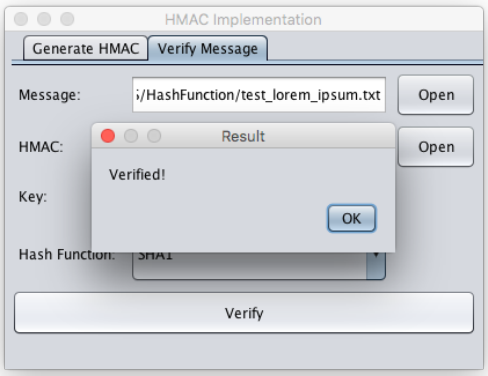
Berikut adalah tampilan dari program yang diimplementasikan.



Gambar 3 Tampilan awal Program



Gambar 4 Tampilan Program untuk Membangkitkan HMAC



Gambar 5 Tampilan Program untuk Melakukan Verifikasi HMAC

Berikut adalah contoh masukan dan keluaran yang dihasilkan oleh program.

Tabel 1 Contoh Isi File Masukan dan Keluaran Program dengan Menggunakan SHA1

Pesan	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident,
Kunci Rahasia	THEQUICKBROWNFOXJUMPSOVERTHELAZYDOGS
HMAC	f96741c116552bf779dc0f9653f5014b8d0e0ffb

IV. EKSPERIMEN DAN ANALISIS

A. *Spesifikasi Komputer*

Untuk melakukan eksperimen dengan menjalankan program yang telah diimplementasikan, digunakan *Personal Computer* MacBook Pro 13-inch dengan spesifikasi sebagai berikut:

- Operating System: OS X El Capitan Version 10.11.16
- Processor: 2.5 GHz Intel Core i5
- Memory: 4 GB 1600 MHz DDR3
- Startup Disk: Macintosh HD
- Graphics: Intel HD Graphics 4000 1536 MB

B. *Analisis Performansi*

Eksperimen dilakukan dengan menggunakan 4 buah data dengan ukuran yang berbeda, yaitu 5 KB, 25 KB, 50 KB, dan 100 KB. Untuk kunci rahasia digunakan 3 buah kunci yang berbeda, masing-masing dengan panjang 8 karakter (64 bit), 16 karakter (128 bit), dan 32 karakter (256 bit). Masing-masing algoritm SHA1, SHA256, dan MD5 akan dijalankan dengan kombinasi file dan kunci yang berbeda. Ukuran file HMAC, waktu eksekusi, dan eksekusi CPU dicatat untuk melakukan analisis performansi.

Berikut adalah hasil yang didapat dengan mengunakan fungsi SHA1, SHA256, dan MD5

Tabel 2 Hasil Eksperimen dengan SHA1

Ukuran File Masukan (KB)	Ukuran Kunci Rahasia (bit)	Ukuran File Keluaran (B)	Waktu Eksekusi (ms)	Load Per CPU
5	64	40	230	0.46
5	128	40	231	0.46
5	256	40	241	0.45
25	64	40	238	0.48
25	128	40	268	0.43
25	256	40	291	0.44
50	64	40	268	0.47
50	128	40	300	0.43

50	256	40	280	0.46
100	64	40	275	0.44
100	128	40	310	0.44
100	256	40	322	0.41

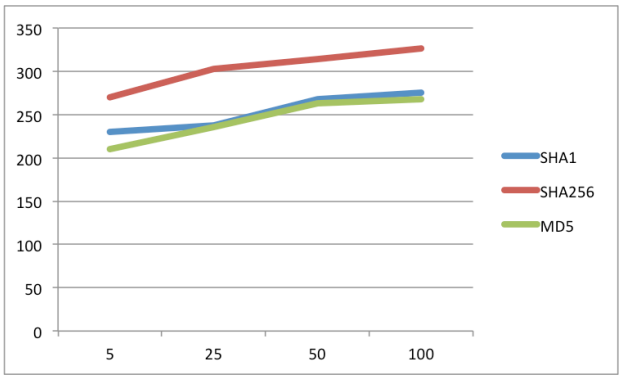
Tabel 3 Hasil Eksperimen dengan SHA256

Ukuran File Masukan (KB)	Ukuran Kunci Rahasia (bit)	Ukuran File Keluaran (B)	Waktu Eksekusi (ms)	Load Per CPU
5	64	64	270	0.48
5	128	64	290	0.49
5	256	64	294	0.59
25	64	64	303	0.56
25	128	64	310	0.59
25	256	64	321	0.50
50	64	64	314	0.55
50	128	64	312	0.56
50	256	64	322	0.55
100	64	64	326	0.45
100	128	64	336	0.48
100	256	64	340	0.46

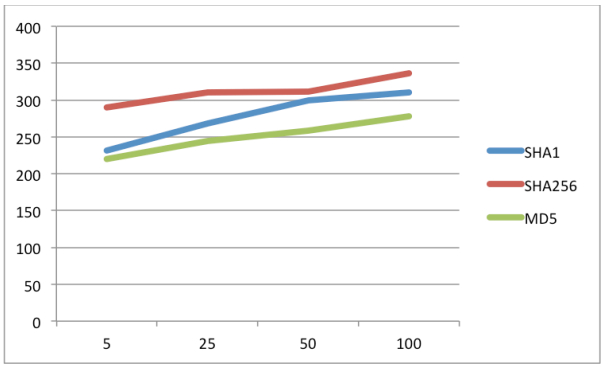
Tabel 4 Hasil Eksperimen dengan MD5

Ukuran File Masukan (KB)	Ukuran Kunci Rahasia (bit)	Ukuran File Keluaran (B)	Waktu Eksekusi (ms)	Load Per CPU
5	64	32	210	0.40
5	128	32	220	0.47
5	256	32	236	0.49
25	64	32	236	0.50
25	128	32	244	0.48
25	256	32	272	0.48
50	64	32	263	0.45
50	128	32	258	0.49
50	256	32	248	0.40
100	64	32	268	0.44
100	128	32	278	0.44
100	256	32	289	0.46

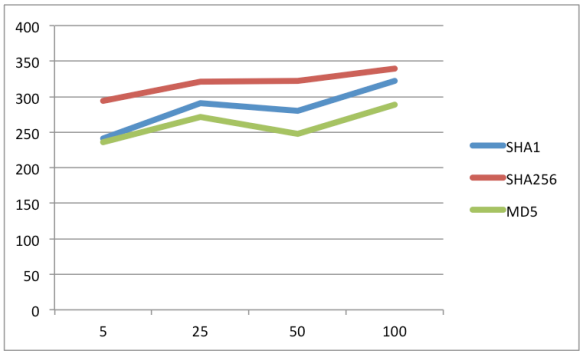
Ketiga hasil eksperimen dengan menggunakan tiga fungsi yang berbeda tersebut dapat dirangkaum dalam grafik untuk mempermudah proses analisis. Berikut adalah grafik perbandingan performansi ketiga fungsi hash dengan parameter waktu eksekusi dengan panjang kunci masing-masing 64 bit, 128 bit, dan 256 bit.



Gambar 6 Grafik Perbandingan Fungsi Hash SHA1, SHA256, MD5 dengan Parameter Ukuran File dalam KB (Sumbu X) dan Waktu dalam ms (Sumbu Y) dengan Panjang Kunci 64 bit

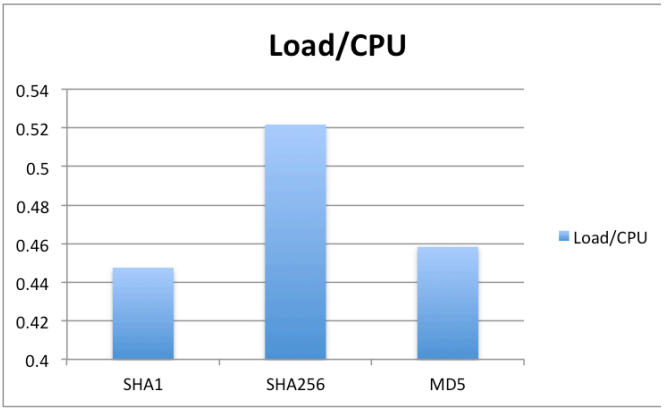


Gambar 7 Grafik Perbandingan Fungsi Hash SHA1, SHA256, MD5 dengan Parameter Ukuran File dalam KB (Sumbu X) dan Waktu dalam ms (Sumbu Y) dengan Panjang Kunci 128 bit



Gambar 8 Grafik Perbandingan Fungsi Hash SHA1, SHA256, MD5 dengan Parameter Ukuran File dalam KB (Sumbu X) dan Waktu dalam ms (Sumbu Y) dengan Panjang Kunci 256 bit

Berikut adalah grafik yang menggambarkan rata-rata pemrosesan CPU masing-masing fungsi hash untuk semua kunci.



Berdasarkan hasil eksperimen dan grafik di atas, dapat dilihat bahwa untuk masing-masing fungsi hash, waktu ekseskusi berbanding lurus dengan ukuran file pesan, sedangkan ukuran kunci tidak terlalu berpengaruh terhadap waktu eksekusi. Semakin besar ukuran pesan maka waktu eksekusi semakin lama dan semakin kecil ukuran pesan maka waktu eksekusi semakin kecil. Hal ini disebabkan karena ketika ukuran file semakin besar, jumlah blok yang terbentuk juga semakin banyak. Pemrosesan dilakukan untuk msing-masing blok sehingga semakin banyak blok yang terbentuk maka waktu eksekusi semakin lama, begitu pula sebaliknya.

Berdasarkan grafik waktu ekseskusi ketiga fungsi hash tersebut dapat dilihat juga bahwa fungsi hash MD5 memiliki waktu eksekusi yang paling kecil dan SHA256 memiliki waktu eksekusi yang paling besar untuk semua panjang kunci. Hal ini disebakan karena fungsi hash MD5 tidak terlalu kompleks, hanya terdiri dari 4 putaran dengan masing-masing terdiri dari 16 operasi. Fungsi hash SHA1 dan SHA256 lebih kompleks karena putaran yang dilakukan mencapai 64 – 80 putaran sehingga menghasilkan waktu eksekusi lebih besar.

Ukuran file keluaran tidak bergantung pada ukuran masukan maupun panjang kunci, karena masing-masing fungsi hash memiliki ukuran file keluaran yang tetap, yaitu SHA1 40 B (160 bit), SHA256 64 B (256 bit), dan MD5 32 B (128 bit). Sedangkan dari segi pemrosesan CPU, dapat dilihat bahwa SHA256 mengkonsumsi lebih banyak proses pada CPU dibandingkan MD5 dan SHA1 yang selisihnya tidak terlalu jauh. Meskipun pemrosesan CPU tidak bisa menjadi tolok ukur yang baik dalam mengukur performansi karena sangat bergantung pada sistem operasi dan faktor eksternal lainnya, namun dapat digunakan untuk mengambil perbandingan kasar antara ketiga fungsi hash dan didapatkan bahwa SHA256 mengonsumsi lebih banyak proses dibandingkan kedua algoritma lainnya.

Berdasarkan hasil eksperimen dan analisis tersebut dapat dilihat bahwa dari segi waktu eksekusi dan pemrosesan CPU, fungsi hash MD5 memiliki performansi paling baik, diikuti oleh SHA1 kemudian SHA256.

C. Analisis Keamanan

Untuk menganalisis keamanan dalam melakukan verifikasi HMAC, fungsi hash diujicoba dengan menjalankan *length extension attack*. *Length extension attack* merupakan serangan ketika penyerang mengetahui Hash(m1) dan panjang kunci untuk melakukan kalkulasi Hash(m2). *Length extension attack* memungkinkan penyerang untuk mendapatkan hasil fungsi hash yang sama dengan menambahkan padding dan kata baru pada pesan tanpa menegetahui kunci rahasia yang disepakati sebelumnya. Dengan begitu, penyerang akan terverifiksi meskipun tidak memiliki kunci rahasia. *Length extension attack* merupakan jenis serangan yang cukup umum pada fungsi hashing, dan sangat rentan dikenakan terutama untuk ketiga fungsi SHA1, SHA256, dan MD5. Pada eksperimen ini, akan diujicoba apakah HMAC dengan menggunakan ketiga fungsi hash tersebut juga rentan terhadap serangan *length extension attack*.

Untuk melakukan analisis keamanan, dipilih pesan dan kunci yang sama yang memiliki panjang 128 bit. Tools *hash_extender*[5] digunakan untuk mendapatkan nilai hash dalam melakukan serangan. Berikut adalah hasil HMAC dengan menggunakan pesan asli dan kunci rahasia dengan menggunakan algoritma SHA1.

Tabel 5 Masukan, Kunci dan HMAC untuk Analisis Keamanan dengan menggunakan SHA1

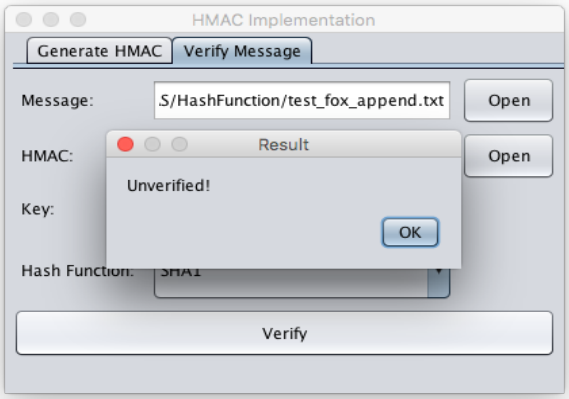
Pesan	The Quick Brown Fox Jumps Over The Lazy Dogs
Kunci	InformatikaITB13
HMAC	5818167b2167581ce5d2650fb4baf15daa6dd1b1

Untuk melakukan penyerangan dengan *length extension attack*, penyerang menempelkan kata tambahan di pesan untuk mendapatkan HMAC yang baru. Kata yang digunakan pada percobaan adalah kata “And Cats”. Dengan mengetahui pesan dan HMAC asli, HMAC untuk (Pesan + “And Cats”) dapat dicari dengan menggunakan tools hash_extender. Berikut hasil yang didapatkan

Tabel 6 Keluaran dari hash_extension dengan Masukan Pesan, HMAC, dan Panjang Kunci

```
Type: sha1
Secret length: 16
New signature:
734045c34c81799f435a5753c5fe14c8185ec5e
2
New string:
54686520517569636b2042726f776e20466f782
04a756d7073204f76657220546865204c617a79
20446f677380000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000000000000000000000
0000000000000000000000001e020416e6420
43617473
```

Dengan memanfaatkan *signature* baru yang didapatkan dan menggunakan pesan yang sudah ditambahkan pada kalimat terakhir, verifikasi dilakukan dengan menggunakan fungsionalitas verifikasi pada program yang diimplementasikan.



Gambar 9 Tampilan Program ketika Verifikasi HMAC

Dapat dilihat bahwa meskipun sudah menambahkan kata taambahan di akhir pesan dan membangkitkan hash yang baru dengan tools khusus, verifikasi tetap gagal. Hal ini terjadi karena berbeda dengan fungsi hash biasa, pada HMAC kunci tidak ditempelkan langsung pada pesan seperti perhitungan hash pada umumnya. Pada fungsi hash biasa, hash dihitung dengan menggunakan H(key|message), sedangkan pada HMAC kunci tidak langsung digabungkan pada pesan, namun di-XOR terlebih dahulu dengan *inner* dan *outer padding*. *Inner* dan *outer* padding inilah yang menyebabkan HMAC menjadi *collision-resistance* dan terjaga otentikasinya. Pada eksperimen di atas, meskipun fungsi SHA1 rentan terhadap serangan *length extension attack*, namun HMAC dengan menggunakan SHA1 terbukti aman dari serangan tersebut. Kesimpulan yang sama berlaku untuk fungsi hash SHA256 dan MD5.

V. KESIMPULAN DAN SARAN

HMAC merupakan salah satu metode untuk menjamin otentikasi dan integritas dari isi pesan dengan menggunakan fungsi hash berbasis satu arah. Fungsi hash yang cukup umum digunakan untuk HMAC adalah SHA1, SHA256, dan MD5. Untuk mengukur performansi dari ketiga fungsi hash tersebut, diimplementasikan program dalam bahasa pemrograman Java. Berdasarkan eksperimen dan analisis yang dilakukan, dapat disimpulkan bahwa MD5 memiliki performansi yang lebih baik dari segi waktu eksekusi dibandingkan SHA1 dan SHA256. Dari segi penggunaan CPU, baik MD5 dan SHA1 memiliki performansi lebih baik dibandingkan SHA256 karena membutuhkan lebih sedikit proses. Ketiga fungsi hash tersebut dalam pemanfaatan HMAC terbukti aman dari serangan *length extension attack*. Meskipun begitu, analisis dan penelitian yang lebih lanjut masih diperlukan untuk menguji ketahanan ketiga fungsi tersebut dari jenis serangan lainnya.

UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada Allah SWT, karena atas rahmat dan karunia-Nya lah makalah ini dapat selesai pada waktunya. Penulis juga ingin mengucapkan terima kasih kepada kedua orang tua yang tidak pernah letih mendukung dan mendoakan anaknya, serta Bapak Dr. Ir. Rinaldi Munir selaku dosen mata kuliah Kriptografi. Tidak lupa penulis juga ingin mengucapkan terima kasih kepada pihak-pihak lain yang telah membantu dalam penyelesaian makalah ini.

REFERENSI

[1] Bellare, M., Canetti. R., Krawczyk H. "Keying Hash Function for Message Authentication". 1996. Proceedings of Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag

[2] Gupta, S., Yadav. S.K. 2013. "Performance Analysis of Cryptographic Hash Function", International Journal of Science and Research.

[3] Munir, R. "MAC". 2015. Slide Kuliah IF4020 Kriptografi.

[4] Munir, R. "SHA". 2015. Slide Kuliah IF4020 Kriptografi.

[5] "Hash Extender by Ron Bowes" https://github.com/iagox86/hash_extender diakses tanggal 18 Desember 2016

[6] "Hash Length Extension Attack" https://www.whitehatsec.com/blog/hash-length-extension-attacks/ diakses tanggal 18 Desember 2016

[7] "The MD5 Cryptographic Hash Function" http://www.iusmentis.com/technology/hashfunctions/md5/ diakses tanggal 17 Desember 2016

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, 19 Desember 2016

Zulva Fachrina/13513010