Analisis Kinerja dan Keamana dari Message Authentication Code (MAC) Berbasis Fungsi Hash SHA3

Dandu Satyanuraga
13515601
Teknik Informatika
Institut Teknologi Bandung
Jalan Ganesa 10 Bandung, Indonesia
13515601@std.stei.itb.ac.id

Abstrak – Serangan terhadap pesan yang dikirim melalui jaringan sering terjadi dan hal ini adalah isu krusial yang terus menjadi alasan sehingga pengembangan keamanan dalam jaringan terus berkembang. Metode yang terus berkembang adalah metode MAC atau Message Authentication Code terus berkembang. Dengan berkembangnya terus fungsi hash maka metode MAC terus berkembang berdasarkan funsgi hash yang digunakan. Fungsi hash pada metode MAC yang populer seperti MD5, SHA1, SHA256 sudah banyak digunakan untuk meningkatkan kinerja pada metode MAC. Fungsi hash SHA3 atau Keccak adalah metode hash terbaru yang dipercaya memiliki keamana yang baik. Karena itu dalam makalah ini dilakukan eksperimen untuk mengukur performansi dari HMAC menggunakan fungsi hash SHA3.

Keywords – MAC, HMAC, SHA3, SHA1, MD5

1. PENDAHULUAN

Saat ini berkomunikasi atau mengirim pesan jarak jau sangatlah mudah. Dengan berbagai media elektronik untuk mengirim pesan jarak sepeti tidak berarti. Pengiriman data atau informasi dapat dilakukan dimana saja dan kapan saja melalui *email,* sms, *chat* dan lain - lain. Media elektronik dapat meningkatkan efisiensi dalam pengiriman pesan atau media tertentu, namun ada hal yang perlu diperhatikan yaitu keamanan yang terjamin.

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.

Metode kode pesan autentikasi atau yang sering disebut *Message Authentication Code (MAC)* sangat populer untuk memeriksa apakah pesan yang dikirim merupakan pesan yang asli dikirim oleh pengirimi dan metode ini dapat memeriksa apakah pesan tersebut utuh sampai kepada penerima dan dapat memeriksa juga apakah pesan berubah selama pengiriman atau tidak. Metode *MAC* biasanya hanya berupa penggabungan antara pesan yang dikirim dengan hasil fungsi *MAC* yang akan diperiksa oleh penerima sebagai bukti keabsahan dari pesan. Hal ini dapat beresiko penggantian pesan namun menggunakan hasil fungsi *MAC* yang sama dengan yang dikirim oleh pengirim dan bahkan sebaliknya. Hal ini menyebabkan pesan tidak terverifikasi dan akhirnya tidak berguna mengirim pesan tersebut. Walaupun kinerja dari metode MAC sukar untuk diukur namun alangkah baiknya jika pengembangan metode untuk mengirim pesan dapat berjalan. Dengan metode *MAC* yang lebih rumit dan dengan otentikasi pesan yang lebih detail maka serangan pada pengiriman pesan dengan MAC akan lebih sukar untuk dilakukan.

Walaupun performansi dan tingkat keamanan dari Hash-based Message Authetication Code (HMAC) sedikit sulit diukur karena sangat bergantung pada fungsi hash yang digunakan. Diperlukan analisis performansi untuk menentukan apakah fungsi hash SHA3 efisien untuk berbagai kasus. Dalam makalah ini, akan diukur performasi MAC berbasis fungsi satu arah dengan menggunakan fungsi hash MD5, SHA1 dan SHA3.

Dengan menggunakan fungsi hash terbaru diharapkan metode MAC dapat meningkat kinerjanya dan dapat menjamin keamanan untuk mengirim pesan.

2. 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 m1 dan m2 yang menghasilkan nilai hash yang sama hash(m1) = hash(m2). 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:

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

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:

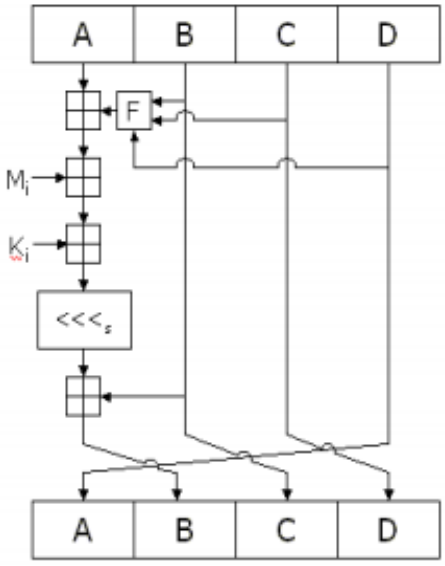
- Pesan dibagi dalam blok berukuran 512 bit dengan 64 bit ditambahkan pada blok terakhir. 64 bit tambahan terse but digunakan untuk menyimpan panjang dari masukan. Jika blok terakhir berukuran kurang dari 512 bit, sejumlah bit tambahan ditambahkan untuk menghasilkan blok berukuran 512 bit. Kemudian, 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 M1 dan konstanta pada tabel K1 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 digunakan 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:

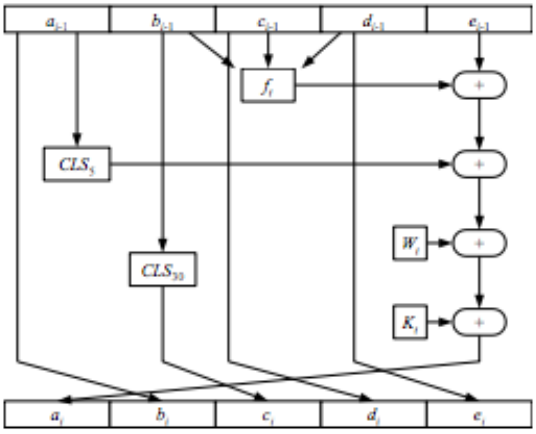
- Tambahkan bit penyangga pada pesan, dimulai dengan angka 1 dan diikuti dengan sejumlah angka 0 sehingga panjang bit kongruen dengan -64 ≡ 448 (mod 512). Pada akhir bit, tambahkan 64 bit yang menyatakan panjang pesan sehingga panjang pesan sekarang merupakan kelipatan dari 512.
- Pesan dibagi menjadi 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:

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

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



Gambar 2 operasi pada SHA1.

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

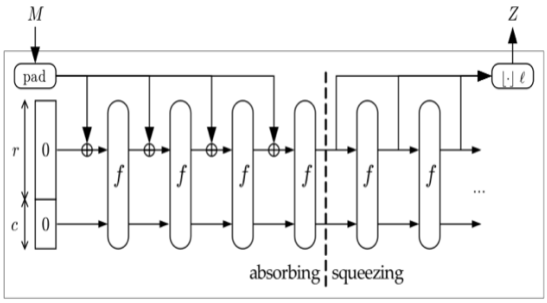
$$F(B,C,D) = (B \wedge C) \vee (\sim B \wedge D)$$
$$G(B,C,D) = B \oplus C \oplus D$$
$$H(B,C,D) = (B \wedge C) \vee (B \wedge D))$$

D. SHA3

SHA3 merupakan fungsi hash satu arah yang dihasilkan oleh kompetisi yang diadakan oleh NIST sebagai komplementer dari SHA1 dan SHA2. Tahapan dari algoritma SHA3 adalah sebagai berikut:

- Pertama, string input diisi dengan aturan padding yang dapat dibalik dan potong blok r bits. Kemudian bit b dari *state* diinisialisasi nol dan konstruksi spons berlangsung dalam dua fase:
- (a) Pada fase penyerap, blok input r-bit di-XOR ke dalam r bit kondisi pertama, disisipkan dengan aplikasi fungsi f. Ketika semua blok input diproses, konstruksi spons beralih ke fase meremas.
- (B) Dalam fase meremas, bit r kondisi awal dikembalikan sebagai blok output, disisipkan dengan aplikasi fungsi f. Jumlah blok output dipilih sesuka hati oleh pengguna.
- C bit terakhir dari kondisi tidak pernah dipengaruhi secara langsung oleh input blok dan tidak pernah output selama fase meremas.

Berikut adalah konstruksi spon dari algoritma SHA3:



Gambar 3 konstruksi spon pada SHA3.

3. IMPLEMENTASI

Untuk melakukan analisis kinerja dari fungsi hash pada Message Authentication Code (MAC), dibuat sebuah program sederhana dengan menggunakan bahasa Python dengan berbagai fungsi untuk menunjukan hasil kinerja dari HMAC.

Program yang diimplementasi memiliki fungsionalitas utama yaitu untuk menerima masukan pesan dan kunci, membangkitkan nilai MAC berdasarkan fungsi hash dari masing – masing HMAC dan melakukan verifikasi terhadap nilai HMAC dari pesan yang dimasukan. Program ini dapat menerima panjang pesan dan kunci sesuai dengan 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:

```
masukan file: messagefile
masukan key: okey
CPU usage:  [24.8, 5.0, 21.8, 7.0]
hasil hmac sh1 : vSABkGdOCWb9J_Z2y7vyIdptVNU=
1005.8588981628418 ms
size input:  5480 B
size ouutput:  224 B
```

Gambar 3 Tampilan hasil program menggunakan SH1

```
verifikasi sh1
masukan key: okey
CPU usage:  [19.0, 5.9, 15.8, 7.0]
verified? :  True
```

Gambar 4 Tampilan Verifikasi HMAC menggunakan SH1

Berikut ini adalah contoh pesan dan kunci yang digunakan pada penelitian ini:

Tabel 1 Contoh Pesan, Kunci dan Hasil dari HMAC

Pesan	Kota kembang merupakan sebutan lain untuk kota ini, karena pada zaman dulu kota ini dinilai sangat cantik dengan banyaknya pohon-pohon dan bunga-bunga yang tumbuh di sana. Selain itu Bandung dahulunya disebut juga dengan Parijs van Java karena keindahannya. Selain itu kota Bandung juga dikenal sebagai kota belanja, dengan mall dan factory outlet yang banyak tersebar di kota ini, dan saat ini berangsur-angsur kota Bandung juga menjadi kota wisata kuliner. Dan pada tahun 2007, konsorsium beberapa LSM internasional menjadikan kota Bandung sebagai pilot project kota terkreatif se-Asia Timur. Saat ini kota Bandung merupakan salah satu kota tujuan utama pariwisata dan pendidikan.
Kunci	okey
Keluaran SH1	gY0hjO4NkmLQbvI0CSdWnr8DOrc=
Keluaran MD5	IQuy51Fi_GTJhpmkMWjrQA==
Keluaran SHA3	Hkf5oONy_Sv-Ikld7Halz6gh3ZZ-N8PsNyUHRwf68QA=

4. Analisis

A. Komputer

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

- Operating System: OS X Sierra
- Processor: 2.5 GHz Intel Core i5
- Memory: 8 GB 1600 MHz DDR3
- Startup Disk: Macintosh HD
- Graphics: Intel HD Graphics 3000 512 MB

B. Analisis Kinerja

Tabel 2 Hasil Eksperimen HMAC SHA1

Ukuran File Masukan (KB)	Ukuran Kuci (B)	Ukuran File Output (B)	Waktu Eksekusi (MS)	CPU Load
5	64	224	1004	21.8
5	128	224	1005	24

5	256	224	1004	24.8
25	64	224	1004	23
25	128	224	1005	20
25	256	224	1006	25.7
50	64	224	1006	23.2
50	128	224	1002	21
50	256	224	1004	27
100	64	224	1003	14
100	128	224	1005	21
100	256	224	1004	17

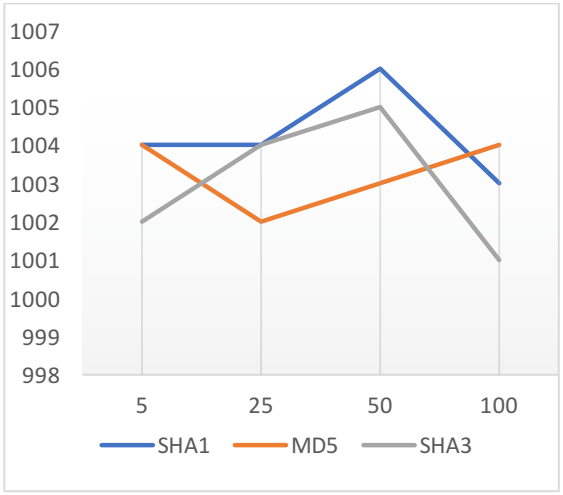
Tabel 3 Hasil Eksperimen HMAC MD5

Ukuran File Masukan (KB)	Ukuran Kuci (B)	Ukuran File Output (B)	Waktu Eksekusi (MS)	CPU Load
5	64	192	1004	23.8
5	128	192	1004	22
5	256	192	1003	31
25	64	192	1002	14
25	128	192	1005	23.8
25	256	192	1004	22
50	64	192	1003	15.7
50	128	192	1005	23
50	256	192	1004	25
100	64	192	1004	21.2
100	128	192	1005	15.8
100	256	192	1003	24

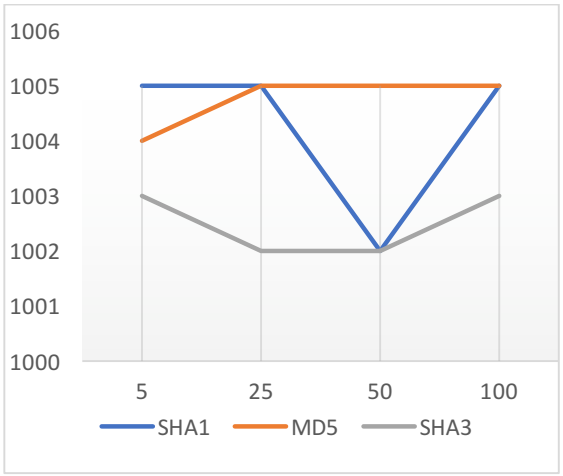
Tabel 4 Hasil Eksperimen HMAC SHA3

Ukuran File Masukan (KB)	Ukuran Kuci (B)	Ukuran File Output (B)	Waktu Eksekusi (MS)	CPU Load
5	64	352	1002	25.3
5	128	352	1003	14.9
5	256	352	1003	28.4
25	64	352	1004	22
25	128	352	1002	29.7
25	256	352	1002	16.8
50	64	352	1005	21.2
50	128	352	1002	25.7
50	256	352	1001	23
100	64	352	1001	25
100	128	352	1003	23
100	256	352	1003	24

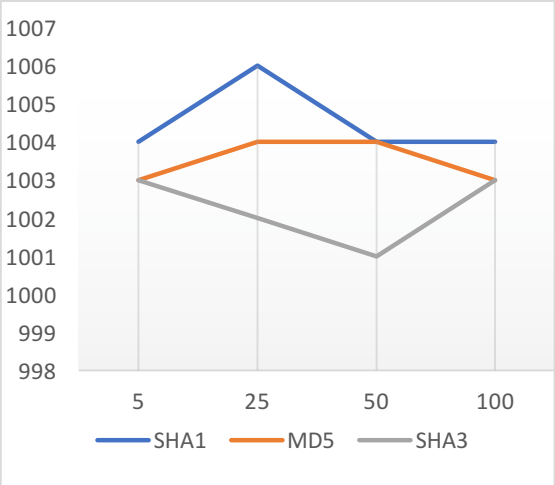
Dari hasil eksperimen menggunakan fungsi HMAC dengan fungsi hash yang berbeda yaitu SHA1, MD5 dan SHA3. Dalam Memudahkan analsis dibuat grafik perbandingan antara ketiga hasil eksperimen sebagai berikut:



Gambar 5 Grafik perbandingan hasil HMAC SH1, MD5 dan SHA3 dengan parameter Ukuran file dalam KB dan waktu eksekusi dengan panjang kunci 64 KB.

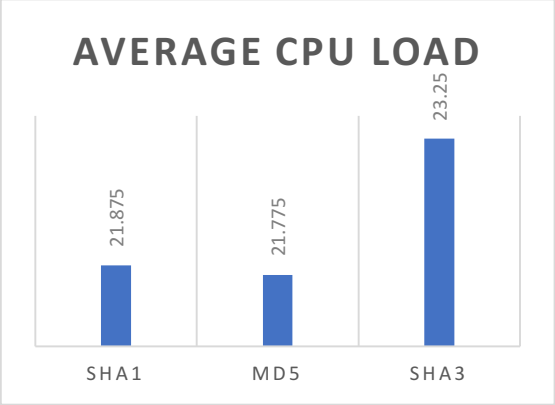


Gambar 6 Grafik perbandingan hasil HMAC SH1, MD5 dan SHA3 dengan parameter Ukuran file dalam KB dan waktu eksekusi dengan panjang kunci 128 KB.



Gambar 7 Grafik perbandingan hasil HMAC SH1, MD5 dan SHA3 dengan parameter Ukuran file dalam KB dan waktu eksekusi dengan panjang kunci 256 KB.

Berikut adalah grafik rata – rata CPU Load yang diperlukan untuk menggunakan fungsi masing – masing untuk semua kunci dan pesan.



Berdasarkan hasil eksperimen dan grafik diatas, didapat fariasi menarik tergantung panjang kungci yang diberikan. Karena tidak ada keseragaman antara ketiga eksperimen yang dilakukan. Seharunsnya semakin besar file yang dipakai maka waktu yang diperlukan untuk menghasilkan *message digest* hasrusnya lebih besar. Namun dalam eksperimen ini ukuran file ataupun ukuran kunci tidak berpengaruh dengan kecepatan pemrosesan dalam menggunakan fungsi hash.

Ukuran file yang menyimpan fungsi hash yang dihasilkan oleh fungsi HMAC tidak berpengaruh pada panjang pesan ataupun panjang kunci yang diberikan dengan ukuran file 224 B untuk HMAC SHA1, 192 B untuk HMAC MD5, dan 352 B untuk HMAC SHA3. Hal ini disebabkan karena *message digest* yang dihasilkan oleh fungsi MD5 paling kecil yaitu 64 B, sedangkan SHA1 menghasilkan 160 B dan yang terbesr adalah SHA3 yaitu 256 B. Sehingga fungsi HMAC dengan fungsi hash SHA3 memiliki panjang karakter terpanjang.

Jika dibandingkan semua eksperimen pada fungsi HMAC SHA1, MD5 dan SHA3 didapat fungsi HMAC dengan fungsi hash SHA3 memiliki waktu eksekusi rata – rata terbaik namun dengan CPU Load yang paling tinggi. Hal ini dapat disimpulkan bahwa SHA3 memerlukan kinerja dari CPU yang lebih banyak dibandingkan dengan HMAC dengan fungsi hash lainnya. Walaupun dengan pengukuran performa CPU yang kurang baik karena dapat dipengaruhi oleh faktor – faktor lain seperti sistem operasi yang sedang sibuk atau dengan memory yang kurang baik, namun dari data yang didapat, dapat disimpulkan bahwa fungsi MD5 memiliki beban CPU terbaik diikuti dengan SHA1 dan terakhir adalah SHA3.

Berdasarkan eksperimen didapatkan bahwa kinerja terbaik tergantung penggunaan. Jika ingin menggunakan HMAC untuk kinerja pada kecepatan maka pemenangnya adalah HMAC SHA3 namun jika memilih kinerja beban CPU yang kecil maka HMAC MD5 adalah pilihan terbaik.

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 analisi keamanan, dipilih pesan kunci yang sama yaitu 128 bit. Dengan menggunakan tools *hash-extender* digunakan dengan untuk melakukan eksperimen.

Tabel 5 Hasil HMAC yang akan dieksperimen dengan hash –extension

Pesan	The Quick Brown Fox Jumps Over The Lazy Dogs
Kunci	berdasarkanhasil
HMAC SHA1	gJTgLmtlNygo0-AO3JRVS9Odgas=
HMAC MD5	n30Fea-oyK68p2Mg6b9o8A==
HMAC SHA3	Ey2eCuSQ_Q-qniZ7FiUuXeWKEFHVazfeFlZ7cdBo0o0=

Untuk melakukan penyerangan dengan lenght extension attack, penyerangan pada pesan pada HMAC akan ditambah dengan kata tertentu. Hasil dari penelitian sebagai berikut:

Tabel 6 Hasil dari hash extension

Pesan	The Quick Brown Fox Jumps Over The Lazy Dogs
Kunci	berdasarkanhasil
HMAC SHA1	NnlrZneJEEy9c0000000000000000000000000mq2H1vN3m1BHLE
HMAC MD5	n30Fea-oyK0000000000000000000000000000000068p2Mg6b9o8A==
HMAC SHA3	Ey2eCuSQ_Q-qniZ7FiUuXeWKEFH0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000VazfeFlZ7cdBo0o0=

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.

```
masukan key: berdasarkanhasil
verifikasi sha1
CPU usage:  [25.7, 8.1, 23.8, 11.0]
verified? :  False

verifikasi md5
CPU usage:  [19.0, 5.0, 17.0, 6.9]
verified? :  False

verifikasi sha3
CPU usage:  [17.8, 7.9, 12.9, 7.8]
verified? :  False
```

Gambar 8 Hasil dari verifikasi menggunakan fungsi hash yang telah dibuat.

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.

5. 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, SHA3, dan MD5. Untuk mengukur performansi dari ketiga fungsi hash tersebut, diimplementasikan program dalam bahasa pemrograman Python. Berdasarkan eksperimen dan analisis yang dilakukan, dapat disimpulkan bahwa SHA3 memiliki performansi yang lebih baik dari segi waktu eksekusi dibandingkan SHA1 dan MD4. Dari segi penggunaan CPU, baik MD5 dan SHA1 memiliki performansi lebih baik dibandingkan SH3 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.

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”. 2018. Slide Kuliah IF4020 Kriptografi.

[4] Munir, R. “SHA”. 2018. Slide Kuliah IF4020 Kriptografi.

[5] Munir, R. “SHA3”. 2018. Slide Kuliah IF4020 Kriptografi

[6]“ Pure Python hash length extension module ” https://github.com/stephenbradshaw/hlextend diakses tanggal 18 Mei 2016

[6]“Hash Length Extension Attack” https://www.whitehatsec.com/blog/hash-length-extension-attacks/ diakses tanggal 18 Mei 2018

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.

Semarang, 18 Mei 2018

Dandu Satyanuraga - 13515601