MIND Journal, Vol. 1, No.1, [September] 2016, hlm 9-18
ISSN : 2528-0015

Implementasi *Keyed-Hash Message Authentatication Code* Pada Sistem Keamanan Rumah

M.Ichwan[1], Milda Gustiana[1], Novan Rizky Nurjaman[1]

[1]Teknik Informatika Institut Teknologi Nasional
Email: ichwan@itenas.ac.id

ABSTRAK

Keyed-Hash Message Authentication Code (HMAC) adalah algoritma untuk menghitung nilai MAC (Message Authentication Code) yang menggunakan Fungsi Hash dikombinasikan dengan sebuah kunci rahasia, Fungsi Hash yang digunakan dalam penelitian ini adalah Secure Ha sh Algorithm 256 (SHA256). Nilai MAC digunakan sebagai otentikasi untuk menjamin integritas data dan keaslian pesan. Algoritma ini di implementasikan pada sistem keamanan rumah, dimana pertukaran pesan antara user dan sistem keamanan di otentikasi dengan menggunakan HMAC. Keamanan algoritma HMAC ini dibuktikan dengan hasil pengujian Avalanche effect yang mencapai 87.5% pada fungsi Hash yang digunakan ,dan dibutuhkan waktu sampai 84 tahun untuk serangan Brute force berhasil pada kunci dengan panjang 8 karakter.

***Kata kunci**: keyed-Hash Message Authentication Code, Hash function, Avalanche effect, Brute force attack.*

ABSTRACT

keyed-Hash Message Authentication Code (HMAC) is an algorithm to calculate a Message Authentication Code (MAC) involving a cryptographic hash function and a secret key, the hash function that was used in this research is Secure Hash Algorithm 256 (SHA256). the MAC value is used for authenticating the message to verify data integrity and authentication of a message.This algorithm is implemented in home security system, where the message exchange between user and system is authenticated with HMAC. The security of this algorithm is proved by the result of Avalanche effect testing that reach 85% on Hash function and 84 years needed time to simulate successful brute force attack on key with 8 characters length.

***Keywords**: keyed-Hash Message Authentication Code, Hash function, Avalanche effect, Brute force attack.*

1. LATAR BELAKANG

Tingginya angka kriminalitas khususnya pencurian rumah kosong yang terjadi saat ini membuat masyarakat merasa tidak tenang saat bepergian meninggalkan rumah. Tercatat tahun 2014 di Jakarta telah terjadi 83 kasus pencurian rumah kosong yang ditinggal mudik oleh pemiliknya[2]. Oleh karena itu dibutuhkan suatu penanggulangan untuk mengatasi permasalah tersebut, salah satu cara yang bisa digunakan adalah dengan membangun sistem keamanan rumah. Sistem keamanan ini menggunakan sensor gerak untuk mendeteksi pergerakan manusia. Apabila ada pergerakan manusia maka sistem akan langsung memberikan pesan kepada pemilik rumah. Sistem keamanan rumah ini juga mempunyai fitur lain yaitu dapat mendeteksi kebakaran dan kebocoran gas elpiji.

Message Authentication Code (MAC) adalah informasi yang digunakan untuk mengotentikasi pesan dan untuk menjamin integritas dan keaslian pesan. Terdapat dua cara untuk menghitung nilai MAC yaitu dengan menggunakan *Fungsi Hash* dan algoritma *block cipher*. *keyed-Hash Message*

Authentication Code (HMAC) adalah algoritma untuk menghitung nilai MAC (*Message Authentication Code*) yang menggunakan fungsi hash dikombinasikan dengan sebuah kunci rahasia. Nilai MAC digunakan sebagai otentikasi untuk menjamin integritas dan keaslian dari sebuah pesan. Algoritma ini di implementasikan pada sistem keamanan rumah, dimana pertukaran pesan antara user dan sistem keamanan di otentikasi dengan menggunakan HMAC. Pertukaran pesan antara user dengan sistem ini harus mempunyai pengamanan terhadap orang yang tidak bertanggung jawab, jika tidak maka orang yang tidak bertanggung jawab tersebut bisa mematikan notifikasi sistem dan atau memberikan notifikasi palsu kepada user.

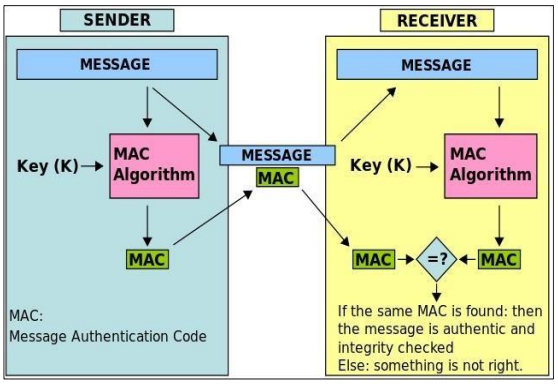
Pesan yang dikirimkan adalah berupa perintah beserta nilai MAC-nya. Pada proses penerimaan, sistem hanya akan menjalankan perintah yang mempunyai nilai MAC yang sama dengan yang dihitung oleh sistem. Adapun tujuan dari penelitian ini adalah melakukan pengujian karakteristik keamanan algoritma *keyed-Hash Messaging Authentication Code* (HMAC) pada sistem keamanan rumah dimana rumusan masalahnya mengenai bagaimana membangun sistem keamanan rumah dan mengimplementasikan Algoritma *keyed-Hash Message Authentication Code* (HMAC) untuk mengamankan komunikasi antara *user* dengan sistem. Pada penelitian ini sistem keamanan rumah yang dibangun berbasis mikrokontroler arduino, kunci privat yang digunakan pada algoritma HMAC statis. Algoritma Hash yang digunakan adalah algoritma *Secure Hash Algorithm* 256 (SHA256) dengan sensor yang digunakan adalah sensor gerak *Passive Infrared Receiver* (PIR), sensor suhu LM35, dan sensor gas MQ6. Komunikasi antara *user* (android) dengan arduino adalah menggunakan *Short Message Service* (SMS) dengan pengujian karakteristik keamanan menggunakan pengujian *Avalanche effect* dan serangan *Brute force*.

2. METODOLOGI PENELITIAN

2.1 *Message Authentication Code* (MAC)[7]

Message Authentication Code (MAC) adalah sepotong informasi yang digunakan untukmengotentikasi pesan dan untuk menjamin integritas dan keaslian pesan. Algoritma MAC mulai berkembang sekitar tahun 1995 dan banyak digunakan secara umum karena memiliki tingkat keamanan yang layak. Gambar 1 menjelaskan bahwa pengirim menghitung nilai MAC dengan menggunakan algoritma MAC dan kunci rahasia dari pesan yang akan dikirimkan. Pesan dan nilai MAC kemudian dikirimkan ke penerima. Penerima selanjutnya menghitung kembali pesan yang diterima dengan menggunakan algoritma dan kunci rahasia yang sama untuk menghasilkan nilai MAC yang kedua. Penerima kemudian membandingkan nilai MAC pertama yang diterima dari pengirim dengan nilai MAC yang dihitung sendiri. Jika nilai MAC pertama dan kedua sama maka pesan dikirim oleh orang yang sesungguhnya dan isi pesan tidak mengalami perubahan selama proses pengiriman. Namun jika nilai MAC pengirim

dan penerima berbeda maka pesan bukan berasal dari pengirim yang asli, sehingga tidak terjamin keaslian isi pesan tersebut



Gambar 1. Blok diagram MAC
(Sumber : Wikipedia.org)

2.2 Keyed-Hash Message Authentication Code (HMAC)[2]

Keyed-Hash Message Authentication Code (HMAC) adalah algoritma untuk menghitung nilai MAC (*Message Authentication Code*) yang menggunakan fungsi hash dikombinasikan dengan sebuah kunci rahasia. HMAC ini digunakan untuk memeriksa integritas data dan otentikasi dari sebuah pesan yang dikirimkan.

Secara sistematis, langkah-langkah yang dilakukan algoritma HMAC dalam menghitung nilai tag adalah:

1. Menentukan kunci rahasia yang akan digunakan.
2. Melakukan XOR antara kunci rahasia dengan *inner Pad* (ipad).
3. Menambahkan pesan pada hasil dari langkah 2.
4. Melakukan hash pada hasil langkah 3.
5. Melakukan XOR antara kunci rahasia dengan *outer Pad* (opad).
6. Menambahkan hasil langkah 5 ke hasil langkah 4.
7. Melakukan hash untuk hasil langkah 6.

Berikut adalah persamaan dari algoritma HMAC.

$$HMAC(K,m)=h((K\oplus opad)||h((K\oplus ipad)||m)) \quad [1]$$

Dimana:

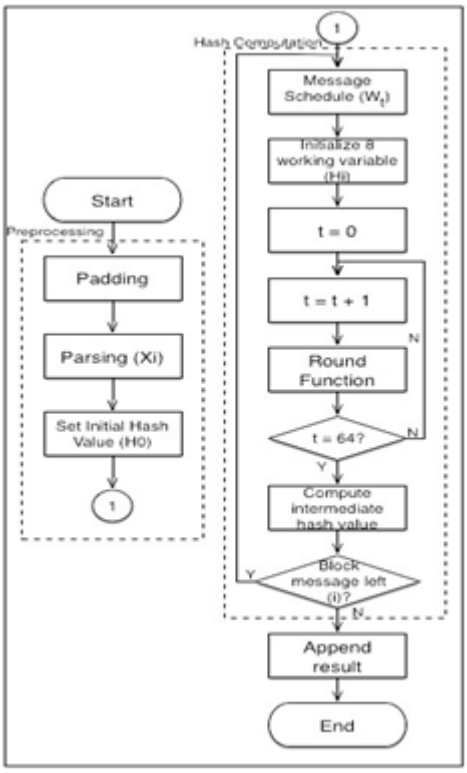
K : kunci rahasia
h : fungsi hash
m : pesan
opad : outer Pad
ipad : inner Pad

2.3 Cryptographic Hash Function[9]

Fungsi hash atau algoritma hash adalah suatu metoda yang menghasilkan suatu kode atau fingerprint dari sebuah data. Fungsi ini memecah dan mengolah data untuk menghasilkan kode atau nilai hashnya. Nilai hash dari suatu fungsi hash memiliki panjang yang tetap untuk masukan dengan panjang sembarang.

2.4 Secure Hash Algorithm 256 (SHA256) [6]

Secure Hash Algorithm (SHA) adalah fungsi hash satu arah yang dibuat oleh NIST (National Institute of Standards and Technology) sebagai standar pengolah informasi bagi pemerintah Amerika Serikat. SHA di dasarkan pada MD4 yang dibuat oleh Ronald L. Rivest.



Gambar 2. Flowchart SHA256
(Sumber : Nist.gov)

Gambar 2 menjelaskan bahwa pesan yang akan di hash harus melalui tahap *preprocessing*. Di dalam tahap preprocessing ini pesan harus melalui tahap *padding* (penambahan nilai pengganjal) sehingga panjang pesan menjadi 512 bit atau kelipatannya. Kemudian masuk pada tahap *parsing* dimana pesan tersebut dibagi-bagi menjadi i buah blok X_0^i, X_1^i, X_2^i, ..., X_n^i. masing-masing sepanjang 32-bit. Initial hash value adalah nilai pembanding pertama (H0) yang digunakan pada *round function*.Satu blok pesan masuk ke dalam perhitungan *message schedule function*. Kemudian tahap selanjutnya adalah memecah nilai pembanding menjadi 8 blok a,b,c,d,e,f,g,h dengan panjang masing-masing blok32-bit. Lakukan round function sebanyak 64 kali terhadap nilai pembanding dengan pesan yang sebelumnya telah melalui tahap message schedule. Setelah itu lakukan modulo 2^{32} terhadap nilai keluaran terakhir round function dengan nilai pembanding (Hi). Apabila masih terdapat blok pesan yang tersisa, lakukan kembali perhitungan hash dengan mengganti nilai pembanding (H_i) dengan keluaran terakhir dari round function yang telah di modulo 2^{32}. Apabila sudah tidak ada blok pesan, selanjutnya adalah gabungkan (*append*) dari blok a-h yang mempunyai panjang masing-masing 32- bit menjadi 256-bit.

2.5 Avalanche Effect[5]

Avalanche secara harfiah berarti longsoran salju yang bergerak cepat menuruni lereng. *Avalanche* biasanya terjadi karena longsoran kecil yang terjadi di puncak atau bagian atas gunung, dan kemudian dengan cepat berakselerasi menuruni lereng yang dalam prosesnya longsoran salju tersebut menjadi semakin membesar. Istilah *Avalanche effect* muncul karena prosesnya mirip dengan *avalanche* atau longsoran salju dimana dengan longsoran kecil dapat menghasilkan longsor yang jauh lebih besar. *Avalanche effect* adalah sebuah teknik untuk mengukur level keamanan dari sebuah metode kriptografi. Dimana perubahan kecil yang terjadi pada *plain text* (*input*) dapat menghasilkan perubahan yang signifikan pada *cipher text* (*output*). Jika sebuah algoritma *block cipher* atau fungsi Hash tidak memiliki *Avalanche effect* minimal 50% maka algoritma tersebut memiliki tingkat pengacakan yang

buruk[5], sehingga bisa membuat seseorang dapat memprediksi input hanya dari output algoritma tersebut.

2.6 Hamming Distance[3]

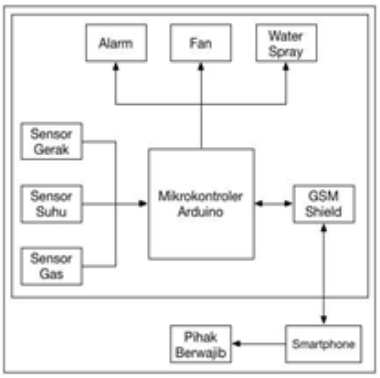
Hamming distance mengukur jarak antara dua *string* yang ukurannya sama dengan membandingkan simbol-simbol yang terdapat pada kedua string pada posisi yang sama. *Hamming distance* antara 2 buah *string* dengan panjang yang sama adalah jumlah posisi simbol yang berbeda. Contohnya adalah hamming distance antara "karolin" dengan "kathrin" adalah 3.

2.7 Analisis Sistem

Sistem keamanan rumah ini memerlukan perangkat keras dalam pelaksanaannya. Berikut adalah spesifikasi kebutuhan perangkat keras dalam sistem keamanan rumah:

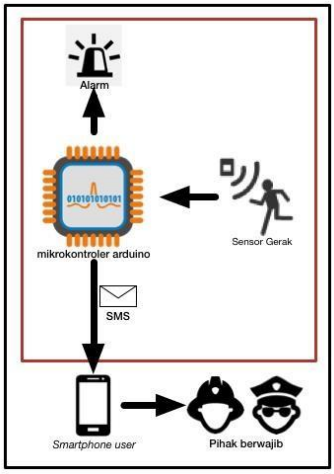
- Arduino Mega 2560
- Sensor gerak *Passive Infrared Receiver* (PIR)
- Sensor suhu LM35
- Sensor gas MQ-6
- Icomsat SIM900 GSM *Shield*
- *Smartphone* Android dengan sistem operasi minimal 4.0 (*Ice Cream Sandwich*).

Pertukaran pesan yang terjadi adalah dari user ke sistem keamanan rumah dan sebaliknya seperti ditunjukan pada Gambar 3



Gambar 3. Blok diagram sistem keamanan rumah

Gambar 3 menjelaskan bahwa mikrokontroler arduino terhubung dengan sensor gerak, sensor suhu, sensor gas, alarm, fan, water spray, dan GSM *shield*. GSM *shield* berperan sebagai masukan dan keluaran bagi arduino, dimana GSM *shield* ini berfungsi sebagai media komunikasi dengan user menggunakan SMS (*Short Message Service*). Isi pesan SMS yang dikirimkan oleh arduino dan smartphone user terlebih dahulu dihitung dengan menggunakan algoritma HMAC-SHA256. Nilai MAC yang dihasilkan kemudian dikirimkan bersama pesan. Pesan SMS yang diterima oleh arduino dan smartphone user melalui tahap verifikasi, dimana pesan yang masuk dihitung kembali nilai MAC- nya menggunakan algoritma HMAC-SHA256. kemudian membandingkan nilai MAC dari nilai yang dihitung sendiri dengan nilai yang terdapat di dalam pesan, apabila nilai MAC sama maka sistem akan menjalankan pesan yang diterima Berikut adalah workflow dari sistem pada saat terjadinya pencurian, seperti ditunjukan pada Gambar 4



Gambar 4. Workflow pencurian

Gambar 4 menjelaskan bahwa sensor gerak PIR (Passive Infrared Receiver) mendeteksi adanya pergerakan. sensor ini bekerja dengan cara mengukur pancaran sinar infra merah yang dikeluarkan oleh benda-benda di sekitarnya. Kemudian sensor gerak memberikan masukan kepada arduino berupa pulse bernilai high yang menunjukan bahwa telah terjadi pergerakan. Arduino kemudian menghidupkan alarm dan mengirimkan pesan yang berisikan perintah " "maling"dan "1bae299aa4d9c4a96cf89ef4ed90b178a020dda96757ccb592ea88de1b5e1541" untuk nilai MAC-nya (Message Authentication Code) yang dihitung dengan menggunakan algoritma HMAC-SHA256.

Hal pertama yang harus dilakukan dalam penghitungan nilai MAC dari kata "maling" adalah sebagai berikut:

1. Merubah perintah "maling" ke dalam bilangan hexa"6d616c696e67".
2. Merubah kunci rahasia "secret1*" yang digunakan ke dalam bilangan hexa.
   K
   73656372 6574312a
3. Melakukan *padding* (penambahan nilai pengganjal) pada kunci rahasia dengan angka 0 sehingga panjang kunci menjadi sama dengan panjang masukan fungsi hash yang digunakan (512 bit).
   K_0 73656372 6574312a 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
4. Melakukan XOR antara K0 dengan ipad (inner pad) yang mempunyai nilai 0x363636…36 sepanjang 512 bit.
   K0 ⊕ ipad
   45535544 5342071c 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636
5. Menggabungkan (*append*) hasil langkah 4 dengan pesan.
   $K_0 \oplus$ ipad || pesan
   45535544 5342071c 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 36363636 73656372 6574312a
6. Menghitung nilai hash dari langkah 5.
   H($K_0 \oplus$ ipad || pesan)
   c73792f8 219814a0 87b0142e 5704aa86 9a868a85 c9e61a7d eb261b11 9f495401
7. Melakukan XOR antara K0 dengan opad (*outer pad*) yang mempunyai nilai 0x5c5c5c…5c sepanjang 512 bit.
   K0 ⊕ opad

2f393f2e 39286d76 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c

8. Menggabungkan (*append*) hasil langkah 7 dengan langkah 6.
   $K_0 \oplus$ opad || H($K_0 \oplus$ ipad || pesan)

   2f393f2e 39286d76 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c 5c5c5c5c
   c73792f8 219814a0 87b0142e 5704aa86 9a868a85 c9e61a7d eb261b11 9f495401
9. Menghitung nilai hash dari langkah 8.
   H($K_0 \oplus$ opad || H($K_0 \oplus$ ipad || pesan))

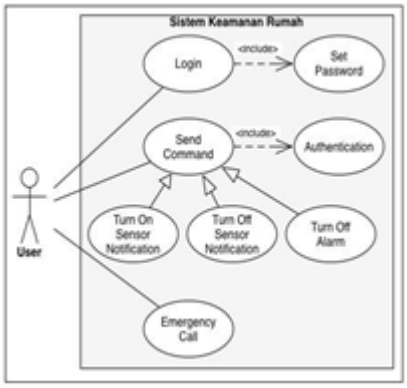
   1bae299a a4d9c4a9 6cf89ef4 ed90b178 a020dda9 6757ccb5 92ea88de 1b5e1541

Pesan yang dikirimkan kepada *user* berupa SMS (*Short Message Service*) menggunakan GSM *shield*. GSM shield adalah perangkat tambahan pada arduino yang memiliki kemampuan untuk mengirimkan pesan melalui jaringan GSM (*Global System for Mobile communications*). Pesan yang diterima pada *smartphone user* kemudian melalui tahap verifikasi. Pada tahap ini sistem mencari perintah pada setiap pesan SMS yang masuk kepada *smartphone user*. Apabila terdapat perintah di dalamnya maka sistem kemudian akan menghitung nilai MAC dari perintah tersebut dengan menggunakan algoritma HMAC-SHA256. Kemudian sistem akan membandingkan nilai MAC yang ada pada pesan dengan nilai MAC yang dihitung sendiri. Apabila nilai dari kedua MAC tersebut sama maka sistem kemudian akan memberikan notifikasi kepada *user*. User kemudian dapat melakukan panggilan kepada pihak berwajib melalui aplikasi sistem keamanan rumah.

3. HASIL DAN PEMBAHASAN

3.1 Perancangan

Dalam perancangan sistem keamanan rumah ini digunakan UML (*Unified Model Language*) sebagai *tools* untuk memodelkan secara visual fungsi. Gambaran interaksi pengguna dengan aplikasi digambarkan pada diagram use case seperti yang ditunjukan pada Gambar 5.

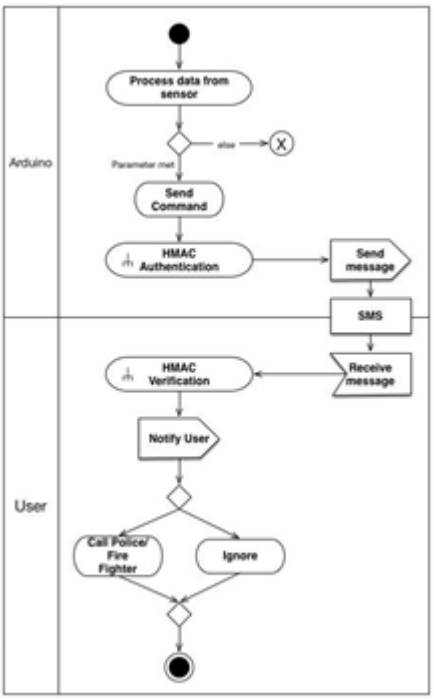


Gambar 5. Diagram Use Case

Penjelasan dari **Gambar 5** adalah sebagai berikut :

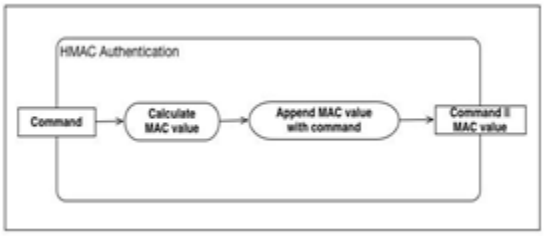
1. Fitur *Login*. Pada fitur ini user harus memasukan *password* untuk dapat mengakses aplikasi sistem keamanan rumah.
2. Fitur *Send Command*. Pada fitur ini user dapat mengirimkan perintah dari perangkat android ke arduino dimana fungsionalitas ini memanggil fungsi otentikasi dan fungsionalitas ini mempunyai *generalization relationship*. *Generalization relationship* ini menunjukan fungsionalitas kirim perintah yang lebih spesifik seperti menghidupkan / menyalakan notifikasi sensor, dan mematikan alarm.

3. Fitur Emergency Call. Pada fitur ini user dapat melakukan panggilan kepada pihak berwajib. Untuk mengilustrasikan aliran fungsional dalam sebuah sistem dan menggambarkan aliran bisnis atau bussiness workflow, maka dibuat activity diagram dari fitur Emergency Call seperti digambarkan pada Gambar 5.



Gambar 6. Activity diagram emergency call

Di dalam *Activity diagram* pada **Gambar 6** terdapat *sub-activity diagram* HMAC *authentication* dan HMAC *verification* yang di jelaskan pada **Gambar 7** dan **Gambar 8**.



Gambar 7. Sub-Activity diagram HMAC authentication

3.2 Pengujian

Keamanan algoritma HMAC-SHA256 di tentukan oleh algoritma Hash dan kunci rahasia yang digunakan. Keamanan algoritma hash dibuktikan dengan *Avalanche Effect*, dan keamanan kunci dengan pengujian *Brute force attack*. Pengujian *Avalanche Effect* ini dilakukan di bagian dari algoritma SHA256 pada *round function*, *round function* ini di eksekusi sebanyak 64 putaran (*round*). Pengujian ini dilakukan dengan membandingkan 2 pesan pada tiap putaran (round) sebanyak 16 putaran pertama. Pesan yang digunakan pada pengujian ini adalah "on" dan "om". Perbedaan antara kedua pesan yang digunakan pada pengujian terletak pada huruf "n" (0110 1110) dan "m" (0110 1101), yang hanya memiliki perbedaan 1-bit. **Tabel 1** menunjukan hasil perhitungan pengujian *Avalanche effect* antara kedua pesan

. **Tabel 1. Pengujian *Avalanche effect***

Round	*Hamming distance*	*Avalanche effect (%)*
1	3	4.6
2	14	21.8
3	29	45.3
4	43	67.1
5	53	85.9
6	55	85.9
7	55	85.9
8	55	85.9
9	51	79.6
10	54	84.3
11	55	85.9
12	57	89
13	56	87.5
14	54	84.3
15	52	81.2
16	54	84.3
64	56	87.5

Pada kolom perhitungan Avalanche effect menggunakan persamaan sebagai berikut.

$$\text{Avalanche effect} = \frac{\text{Hamming distance}}{\text{Block size}} \times 100\% \quad \text{.....[2]}$$

Dimana:

Hamming distance : jumlah perbedaan antara antara dua buah deretan bit (words) yang mempunyai ukuran sama.

Block size : ukuran blok pesan pada perhitungan algoritma SHA256.

Dari hasil pengujian terlihat bahwa hanya dengan mengambil 16 *round* pertama dari *round function* yang terdapat di dalam algoritma SHA256 menghasilkan nilai rata-rata *Avalanche effect* sebesar 72%, dan setelah 64 *round* menghasilkan nilai *Avalanche effect* sebesar 87,5%.

Pada pengujian Brute force attack algoritma HMAC-SHA256 akan di uji dengan melakukan Brute force attack pada kunci rahasia dimana di asumsikan panjang kunci rahasia diketahui. Pengujian Brute force attack ini dilakukan dengan cara melakukan penghitungan ulang algoritma HMAC- SHA256 satu per satu guna untuk memperoleh kunci rahasia yang di maksud. Perhitungan jumlah kombinasi serangan pada Tabel 2 menggunakan persamaan berikut ini

$$S = C^n \quad \text{.....[3]}$$

Dimana :

S : jumlah kombinasi serangan.

C : jumlah karakter dalam charset, yang terdiri simbol, angka, huruf uppercase dan lowercase.

n : panjang kunci.

Perhitungan waktu yang dibutuhkan untuk mendapatkan kunci rahasia dengan melalui Brute force attack menggunakan persamaan berikut ini:

$$T = S / K \quad \text{.....[4]}$$

Dimana :

T : waktu.

S : jumlah kemungkinan serangan yang terjadi.

K: kecepatan dalam melakukan serangan (*key per second*).

Berikut adalah tabel pengujian dari *Brute force attack* seperti yang ditunjukan pada **Tabel 2**.

Tabel 2. Pengujian Brute force attack

Panjang kunci	Kunci Rahasia (*sample*)	Jumlah serangan	Kecepatan	Waktu
1	a	95	2.480.000	1 detik
2	ab	9025	2.480.000	1 detik
3	abc	857375	2.480.000	1 detik
4	q2+M	81.450.625	2.480.000	32 detik
5	hf8s!	7.737.809.375	2.480.000	52 menit
6	Ds6*=t	735.091.890.625	2.480.000	82 jam
7	fD12@#e	69.833.729.609.375	2.480.000	325 hari
8	qw34%^1M	6.634.204.312.890.625	2.480.000	84,8 tahun

Dengan pengujian serangan *brute force* dapat disimpulkan bahwa panjang kunci rahasia berbanding lurus dengan waktu yang dibutuhkan untuk mendapatkan kunci rahasia yang digunakan algoritma HMAC-SHA256.

4. KESIMPULAN

Berdasarkan hasil penelitian yang diperoleh dari pengujian maka dapat disimpulakan sebagai berikut :

1. Hasil pengujian *Avalanche effect* yang di dapat dari 16 *round* pertama pada *round function* yang terdapat di dalam algoritma SHA256 dapat menghasilkan nilai rata-rata *Avalanche effect* sebesar 62%, dan setelah 64 *round* menghasilkan nilai *Avalanche effect* sebesar 85,9%. Ini menunjukan bahwa keluaran dari algoritma SHA256 memiliki tingkat pengacakan yang bagus sehingga membuat seseorang tidak dapat memprediksi pesan masukan hanya dari keluaran dari algoritma SHA256.
2. Keamanan algoritma HMAC-SHA256 ditentukan dengan kunci rahasia yang digunakan. Semakin panjang dan bervariasi kunci yang digunakan maka akan semakin panjang pula waktu yang dibutuhkan *Brute force attack* untuk berhasil mendapatkan kunci rahasia

DAFTAR RUJUKAN

[1] Stallings, W.: Cryptography and Network Security: Principles and Practice Second Edition.
Springer, Heidelberg (1998).
[2] Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. ln: Koblitz, N. Springe
[3] r, Heidelberg (1996)
[4] http://news.detik.com/read/2014/08/04/102528/2652267/10/waspada-pencurian-di-rumah-kosong-selama-mudik-lebaran-2014-meningkat. Diakses pada tanggal 09 oktober 2014.
[5] https://en.wikipedia.org/wiki/Avalanche_effect. Diakses pada tanggal 04 agustus 2015.
[6] https://en.wikipedia.org/wiki/SHA-2. Diakses pada 08 oktober 2014.
[7] http://en.wikipedia.org/wiki/Message_authentication_code. Diaksess pada 08 oktober 2014. [8] http://en.wikipedia.org/wiki/Cryptographic_hash_function. Diaksess pada 08 oktober 2014.
[9] http://en.wikipedia.org/wiki/Hamming_distance. Diaksess pada 04 agustus 2015