Jurnal Teknik Elektro Vol. 10 No. 2
Juli - Desember 2018

P-ISSN 1411 - 0059
E-ISSN 2549 - 1571

Implementasi Enkripsi Data *Secure Hash Algorithm* (SHA-256) dan *Message Digest Algorithm* (MD5) pada Proses Pengamanan Kata Sandi Sistem Penjadwalan Karyawan

Santi Sulastri[1], dan Riana Defi Mahadji Putri[2]

Jurusan Teknik Elektro, Fakultas Teknik, Universitas Negeri Semarang
Kampus Sekaran, Gunungpati, Semarang, 50229, Indonesia
santisulastri52@gmail.com[1], riana.dmp@mail.unnes.ac.id[2]

Abstract— There are various types of encryption that can be used, for example MD5 and SHA-256. However, using MD5 or SHA-256 alone is not safe enough because it can be solved by using a brute force attack. The method of working is very simple is to try all possible combinations. This study aims to develop MD5 and SHA256 collaboration methods. Encryption is applied to the user's password on the web login system. Simulation results can be tested by measuring resistance to brute force attacks and the value of the avalanche effect (AE). From the results of the tests carried out by using attack software, the results of the application are quite safe from brute force attacks. From the AE test obtained the result with an AE value of 71% which means that the encoding result is good enough.

Keywords— Encryption, MD5, SHA-256, brute force, avalanche effect

Abstrak— Terdapat berbagai macam jenis enkripsi yang dapat digunakan, misalnya MD5 dan SHA-256. Namun penggunaan MD5 saja atau SHA-256 saja tidak cukup aman karena dapat dipecahkan dengan menggunakan serangan *brute force*. Cara kerja metode sangat sederhana yaitu dengan mencoba semua kombinasi yang mungkin. Penelitian ini bertujuan untuk mengembangkan metode kolaborasi dari MD5 dan SHA-256. Enkripsi diterapkan pada *password* pengguna pada sistem *login* web. Hasil simulasi dapat diuji dengan mengukur ketahanan terhadap serangan *brute force* dan besar nilai *avalanche effect* (AE). Dari hasil pengujian yang dilakukan yaitu dengan menggunakan *software attack* hasil penyandian cukup aman dari serangan *brute force*. Dari pengujian AE diperoleh hasil dengan nilai AE sebesar 71% yang artinya hasil penyandian cukup baik.

Kata kunci— Enkripsi, MD5, SHA-256, *brute force, avalanche effect*

I. PENDAHULUAN

Perkembangan teknologi dan informasi yang sangat pesat membawa dampak positif yaitu kemudahan dalam berbagi informasi atau data melalui jaringan komputer. Namun pada saat yang bersamaan juga menimbulkan dampak negatif, yaitu informasi atau data dapat diakses oleh pihak yang tidak bertanggunng jawab untuk tindak kejahatan. Maka dibutuhkan suatu mekanisme pengamanan data pada suatu sistem. Sistem perlu mengimplementasikan layanan keamanan, seperti otentikasi, enkripsi, akses kontrol, pengelolaan pengguna dan perizinannya [1].

Setya Aji *Flower Farm* merupakan agrowisata yang bertemakan kebun/taman bunga dengan jumlah karyawan yang mencapai puluhan. Khususnya untuk jenis karyawan parkir dan penjaga loket dengan jumlah personil yang berlimpah menjadikan setiap karyawan tidak dapat bekerja setiap hari. Sejauh ini pembagian waktu kerja telah dilakukan, namun masih menimbulkan kekisruhan dan kekacauan karena beberapa karyawan menginginkan jam kerja lebih dengan merubah-ubah jadwal yang menguntungkan dirinya sendiri. Karena itu diperlukan sistem pengaman yang tidak memudahkan karyawan selain admin megakses sistem penjadwalan. Keamanan sistem informasi digunakan untuk menghindari seseorang yang tidak memiliki akses untuk dapat masuk ke dalam sistem [2].

Salah satu mekanisme pengamanan yang dapat dilakukan yaitu dengan menggunakan *access control. Access control* seringkali dilakukan dengan kombinasi *user id* dan *password.* Penggunaan *password* yang statis dapat memungkinkan terjadinya pencurian *password* oleh *hacker*, maka dibutuhkan suatu mekanisme pengamanan *password* yang dapat dilakukan dengan menerapkan ilmu kriptografi didalamnya.Kriptografi terdiri dari dua proses yaitu proses mentransformasikan *plaintext* menjadi *chipertext* atau yang disebut dengan enkripsi dan proses mentranformasikan kembali *chipertext* menjadi *plaintext* yang disebut dengan dekripsi [3].

Dalam enkripsi data dikenal suatu fungsi yang disebut dengan *hashing*. Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan dikonversikan menjadi *string* dengan keluaran yang panjangnya tetap (*fixed*) [4].

Terdapat dua macam fungsi *hash* yaitu fungsi *hash* satu arah dan dua arah. Dimana dalam fungsi *hash* satu arah hasil *hash* (*hash value*) sangat sukar dikembalikan ke nilai *hash* awal. Terdapat berbagai macam fungsi *hash* satu arah yang dapat digunakan untuk mengkodekan teks, diantaranya fungsi enkripsi *Message Digest* 5 (MD5) dan SHA-256. MD5 adalah fungsi matematika yang merubah *variable* dari suatu data yang berukuran besar menjadi lebih sederhana [5]. Sedangkan SHA-256 merupakan fungsi *hash* satu arah yang dirancang oleh *The National Instutute of Standars and Technology* (NIST) pada tahun 2002 dan versi dari SHA-2.

Pada penelitian [6] penggunaan MD5 hanya untuk menghindari pengiriman *password* secara apa adanya tanpa adanya perlindungan atau pengamanan ke *webserver*. Karena pada masa sekarang banyak *tools* yang dapat digunakan untuk mendekripsi hasil MD5 sehingga penggunaan MD5 saja tidak cukup aman. Penggunaan MD5 dapat dikolaborasikan dengan model enkripsi lain. Pada penelitian ini model enkripsi yang akan digunakan yaitu SHA-256. Penggunaan MD5 saja atau SHA-256 saja tidak cukup aman, maka penggunaannya dapat dimodifikasi [7].

Keamanan SHA-256 pernah diuji pada penelitian yag dilakukan oleh peneliti [8]. Penggunaan SHA-256 yang digabungkan dengan algoritma *Message Authentication Code* (MAC) dari hasil pengujian 64 *round* menghasilkan nilai rata-rata *avalanche effect* (AE) sebesar 85,9%. Ini menunjukan bahwa keluaran SHA-256 memiliki tingkat pengacakan yang bagus.

Pada penelitian ini penggabungan atau kolaborasi model enkripsi MD5 dan SHA-256 akan dilakukan dengan mengambil *string* dari hasil *generate* MD5 yang kemudian *string* tersebut akan dijadikan *padding* pada *password* untuk selanjutnya dienkripsi menggunakan model enkripsi SHA-256.

II. Model Matematis

A. Fungsi Hash

Fungsi *hash* adalah fungsi yang melakukan pemetan pesan dengan panjang sembarang ke sebuah teks khusus yang disebut *message digest* dengan panjang tetap [9]. Fungsi *hash* dapat menerima masukan *string* apa saja, jika *string* menyatakan pesan (*message*), maka sembarang pesan *M* berukuran bebas dikompresi oleh fungsi *H* melalui persamaan algoritma berikut:

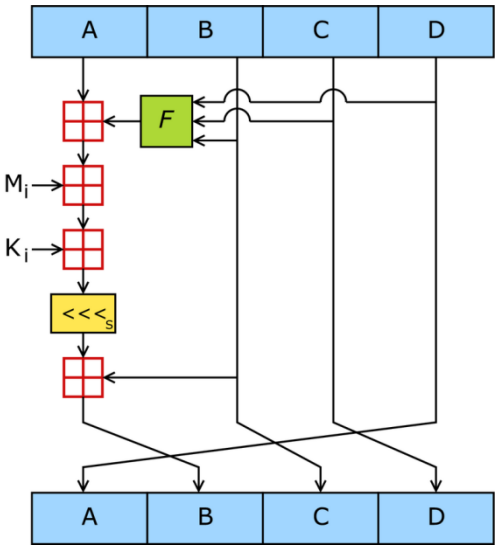
$$h = H(M) \quad (1)$$

Pada persamaan (1), *h* adalah nilai *hash* atas *message digest* dari *H* untuk masukan *M*. Dengan kata lain fungsi *hash* mengonversikan sembarang pesan yang ukurannya selalu tetap [10].

B. MD5

Algoritma MD5 dirancang Ron Rivest dan penggunaanya sangat populer dikalangan komunitas *open source* sebagai *checksum* untuk *file* yang dapat di *download*. Besarnya blok untuk MD5 adalah 512 bit sedangkan *digest size* adalah 128 bit.

MD5 mengolah blok 512 bit, dibagi ke dalam 16 blok berukuran 32 bit. Keluaran algoritma diset menjadi 4 blok yang masing-masing berukuran 32 bit yang setelah digabungkan akan membentuk nilai *hash* 128 bit. MD5 terdiri atas 64 operasi, dikelompokan dalam empat putaran dari 16 operasi proses tersebut dapat dilihat pada Gambar 1.



Gambar 1. Satu operasi MD5

Keterangan :
F : adalah fungsi *nonlinear*, atau fungsi digunakan pada tiap-tiap putaran,
M : menunjukan blok 32 bit dari masukan pesan,
K_i : menunjukan konstanta 32 bit, berbeda untuk tiap-tiap operasi,
$<<<_i$: menunjukan perputaran bit kiri oleh s;s bervariasi untuk tiap-tiap operasi
: menunjukan tambahan modulo 2^{32}

C. SHA-256

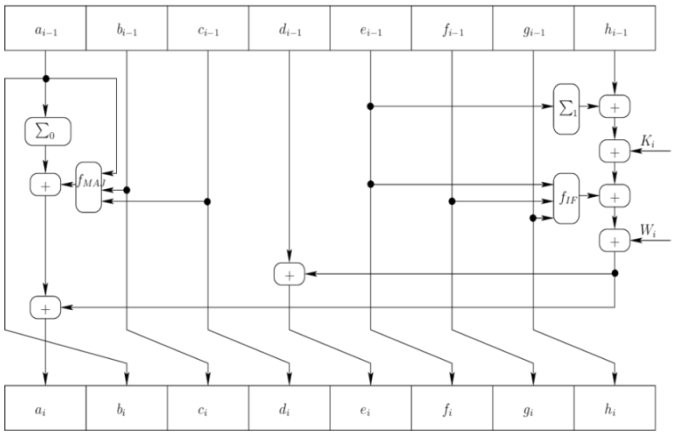
Fungsi *hash* SHA-256 merupakan versi SHA dengan ukuran *digest* 256 pada versi SHA-2. SHA didasarkan pada MD4 yang dibuat oleh Ronald L, Rivest dari MIT.

SHA-256 menggunakan enam logika yang merupakan kombinasi dasar seperti AND, OR, XOR, pergeseran bit kekanan (*shift right*), dan rotasi bit kekanan (*rotate right*). Algoritma ini mengubah sebuah *message schedule* yang terdiri dari 64 element 32-bit *word*, delapan buah variabel 32-bit, dan variabel penyimpan nilai *hash* 8 buah *word* 32-bit.

Algoritma ini menggunakan sebuah *message schedule* yang terdiri dari 64 element 32-bit *word*, delapan buah variable 32-bit, dan *variable* penyimpan nilai *hash* 8 buah *word* 32-bit. Hasil akhir dari algoritma ini adalah sebuah *message digest* sepanjang 256-bit.

SHA-256 mengubah pesan masukan ke dalam *message digest* 256 bit. Berdasarkan *Secure Hash Signature Standard*, pesan masukan yang penjangnya lebih pendek dari 2^{64} bit,

harus dioperasikan oleh 512 bit dalam kelompok dan menjadi sebuah *message digest* 256-bit. Transformasi pembaruan status pada SHA-256 dapat dilihat pada Gambar 2.



Gambar 2. Satu langkah tranformasi pembaruan status SHA-256

III. METODE PENELITIAN

Pada penelitian ini menggunakan metode kolaborasi dari enkripsi SHA-256 dikombinasikan dengan fungsi lain yang dihasilkan dari proses *generate* MD5. Kunci dibentuk melalui beberapa tahap dengan memanfaatkan model enkripsi dari MD5 dan juga SHA-256. Tahap pertama pembentukkan kunci menggunakan model enkripsi MD5 dengan kunci masukan pengguna sebagai bahan pembentukan kunci enkripsi. Dimana dalam tahap pertama ketika suatu pengguna menginputkan *password* pada halaman *login* inputan akan di *generate* menggunakan MD5 dan disimpan dalam sebuah variabel yang diberi nama ”atr”. Hasil dari tahap pertama digunakan untuk membentuk kunci pada tahap kedua. Pada tahap kedua, *string* yang didapatkan dari tahap pertama pada variabel ”atr” akan diambil karakter pertama dan juga terakhir. Setelah didapatkan dua karakter dari variabel ”átr” kemudian karakter akan ditambahkan sebagai *padding* pada *password* yang diinputkan oleh pengguna. Sehingga *password* mendapatkan dua karakter tambahan yaitu pada awal dan juga akhir. Menjadi seperti pada Gambar 3.

Padding “atr”	Password	Padding “atr”

Gambar 3. Model pembentukan kunci

Setelah mendapatkan *padding* kemudian pada tahap terakhir kunci akan dienkripsi menggunakan SHA-256. Hasil *chipertext* dari tahap terakhir akan disimpan ke dalam *database*.

Contoh dari simulasi penyandian pada penelitian ini adalah sebagai berikut:

1. Data yang digunakan adalah *password* yang diinputkan oleh pengguna, misalkan ”123456”
2. *Plaintext* akan di*generate* menggunakan MD5. MD5 digunakan untuk *generate string* untuk mendapatkan hasil *string* yang bervariasi sehingga mendapatkan kombinasi karakter yang beragam. Hasil dari *generate* MD5 yaitu e10adc3949ba59abbe56e057f20f883e
3. Hasil dari *generate* MD5 akan diambil dua karakter, yaitu karakter pertama (e) dan karakter terakhir (e)
4. Dua karakter tersebut akan ditambahkan dalam *password*, dengan skema / model seperti yang ditunjukan pada Gambar 3. Yaitu akan ditambahkan 2 karakter / *string* pada awal dan akhir dari *password*
5. Sehingga data menjadi = e123456e
6. Kemudian data dienkripsi menggunakan SHA-256. Dimana dalam proses pembentukan kunci SHA-256 melalui beberapa tahapan yang mana data yang akan dienkripsi dirubah ke dalam bentuk biner untuk kemudaian mendaptkan padding.
7. Padding dilakukan dengan menambahkan bit ’1’ dan sisanya bit ’0’ hingga panjang pesan tersebut kongruen dengan 448 modulo 512.
8. Selanjutnya pesan yang sudsah di padding dibagi menjadi N buah blok 512 bit.
9. Masing-masing blok 512-bit tadi lalu dipecah menjadi 16 buat word 32-bit : M0(i), M1(i), …, M15(i) yang manantinya diperluas menjadi 64 word yang diberi label W0, W1, …, W63 dengan aturan tertentu yang sudah ditentukan sebelumnya oleh standar SHA-2.

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases} \quad (2)$$

Dengan fungsi σ_0 dan $\sigma_0$dirumuskan sebagai berikut:

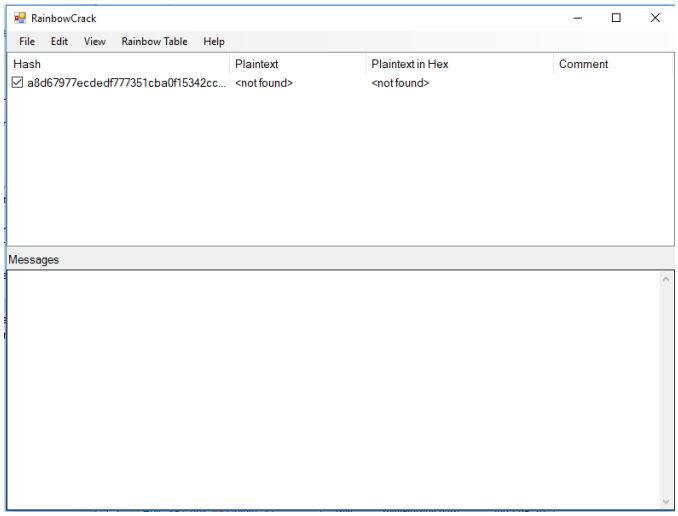
$$\sigma_0(x) = ROTR^7(X) \oplus ROTR^{18}(X) \theta SHR^3(X) \quad (3)$$

$$\sigma_1(x) = ROTR^{17}(X) \oplus ROTR^{19}(X) \theta SHR^{10}(X) \quad (4)$$

10. Masing-masing dari 64 word yang diberi label W0, W1, …, W63 tadi kemudian diproses dengan algoritma fungsi hash SHA-256.
11. Dalam proses tersebut, inti utama dari algoritma SHA-256 adalah membuat 8 variabel yang diberikan nilai untuk nilai awal dari H0(0)-H7(0) di awal masing-masing fungsi hash. Algoritma ini melakukan perhitungan sebanyak 64 kali putaran untuk setiap perhitungan blok.
12. Sehingga mendapatkan hasil akhir sebagai berikut: 3a95dfe70908959bd571ad2c8e2ea86d965bd8462a28ae08 16f2c9597c9a512f
13. Hasil akhir / *Chipertext* akan disimpan dalam *database* dan akan diuji keamanannya menggunakan *software attack* dan juga AE.

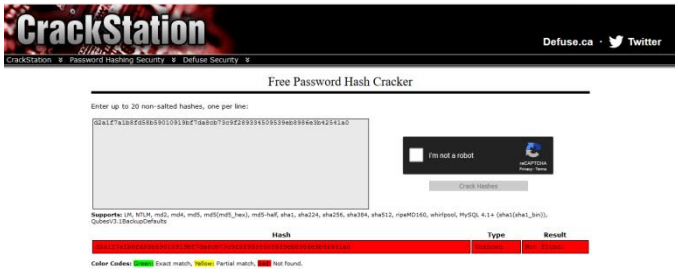
IV. HASIL DAN PEMBAHASAN

Pada penelitian ini hasil penyandian akan diuji ketahanannya dari serangan *brute force* dengan menggunakan *software* penyerang. *Software* penyerang yang digunakan dalam pengujian yaitu Rainbow Table dan CrackStation. Pengujian Rainbow Table dapat dilihat pada Gambar 4. Rainbow Table digunakan untuk mengembalikan fungsi kriptografi hash dan menemukan plaintext kata sandi dalam database. Pada pengujian Rainbow Table, *chipertext* yang dihasilkan oleh metode kolaborasi MD5 dan SHA-256 yang telah dimodifikasi tidak dapat dipecahkan atau dikembalikan ke dalam bentuk *plaintext*. Hasil dapat dilihat pada Gambar 4.



Gambar 4. Tampilan pengujian Rainbow Table

Pengujian ketahanan terhadap serangan *brute force* juga dilakukan dengan menggunakan *software* CrackStation. Pada pengujian menggunakan CrackStation dimaksudkan untuk mendapatkan *plaintext* serta mengetahui model enkripsi apa yang digunakan pada *chipertext* yang diuji. Data uji yang digunakan pada pengujian ini yaitu "iwan09", hasil dari pengujian dapat dilihat pada Gambar 5.



Gambar 5. Pengujian CractStation

Pada pengujian seperti yang ditunjukan oleh Gambar 5, CrackStation tidak dapat menemukan *plaintext* dan juga metode enkripsi apa yang digunakan pada data uji, ini menunjukan bahwa *chipertext* yang dihasilkan dari simulasi penyandian menggunakan SHA-256 yang dikombinasikan dengan fungsi lain memiliki tingkat pengacakan dan kombinasi yang bagus sehingga sulit untuk dipecahkan.

Dalam fungsi *hash* tidak hanya diukur ketahananya dari serangan *brute force*, juga diukur level keamanannya. AE adalah sebuah teknik untuk mengukur level keamanan dari sebuah metode kriptografi. Dimana perubahan kecil yang terjadi pada *plaintext* dapat menghasilkan perubahan yang signifikan pada *chipertext*. Pada pengujian ini, peneliti juga melakukan pengujian terhadap simulasi penyandian sebelum dimodifikasi dan mendapatkan nilai *AE* sebesar 63%. Sebuah algoritma yang baik memiliki AE yang tinggi [11]. Hal ini dapat dilihat dari nilai yang dihasilkan setidaknya setengah dari jumlah bit mengalami perubahan [12]. Pegujian AE dihitung dengan menggunakan rumus *AE* pada persamaan (5).

$$AE = \frac{hamming\ distance}{block\ size}\ x\ 100\% \quad (5)$$

Keterangan:

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

Block size : ukuran blok pesan

Pada pengujian *avalanche effect password* yang akan digunakan yaitu "skripsi" dan skrips1". Berikut tahapan dalam pengujian AE :

1. Data dienkripsi menggunakan kolaborasi dari MD5 dan SHA-256 yang telah dimodifikasi.
   Plaintext : skripsi
   Chipertext : bee87aebdb9a93bd193d5f3881b6e0ad87d6a55147fff109f47969a914dbfebd
   Plaintext : skrips1
   Chipertext : 69d59eabc8f9e009ec59b2edfbca65aae57785d2865f6402ad1f2825fc650d40
2. *Chipertext* hasil penyandian dirubah ke dalam bentuk biner dan dihitung *hamming distance* atau jumlah bit yang berubah. Pada penelitian ini *hamming distance* atau jumlah bit yang berubah yaitu sebanyak 181 bit.
3. Setelah mendapatkan hasil dari nilai *hamming distance*, selanjutnya yaitu dihitung sesuai rumus AE, sehingga:

$$AE = \frac{181}{256}\ x\ 100\%$$
$$AE = 71\%$$

Hasil dari enkripsi atau *hash* dikatakan baik apabila satu perubahan bit pada *input* menghasilkan perubahan besar pada *output* [12]. Dari pengujian yang telah dilakukan, *chipertext* yang dihasilkan dari SHA-256 yang dikombinasikan dengan fungsi lain mengalami perubahan lebih dari 50% dari 256 bit sebanyak 181 bit mengalami perubahan. Semakin banyak perubahan yang terjadi akan semakin sulit bagi kriptanalis untuk melakukan *attack* (serangan) [13]. Modifikasi dapat dilakukan, karena hanya dengan sedikit modifikasi tingkat keamanan sebuah algoritma bisa berubah [14]. Peningkatan nilai AE juga terjadi pada penelitian [15], nilai AE meningkat sebesar 5% dengan mengkombinasikan algoritma GOST dan MD5.

V. PENUTUP

Hasil dari simulasi menunjukan metode kolaborasi dari modifikasi MD5 dan SHA-256 memiliki tingkat pengacakan yang bagus sehingga tahan terhadap serangan *brute force* yang diuji dengan menggunakan *software* penyerang Rainbow Table dan CrackStation. *Chipertext yang* dihasilkan oleh kolaborasi dan modifikasi MD5 dan SHA-256 mendapatkan nilai AE sebesar 71%, terdapat peningkatan sebesar 8% dari sebelum dimodifikasi yaitu sebesar 63%. Hal ini menunjukkan bahwa *chipertext* yang dihasilkan memiliki tingkat pengacakan yang lebih bagus sehingga aman untuk digunakan.

REFERENSI

[1] D. C. Luminita, "Information security in E-learning Platforms," *Procedia - Social and Behavioral Sciences*, vol. 15, pp. 2689–2693, 2011, https://doi.org/10.1016/j.sbspro.2011.04.171

[2] C. Chazar, "Standar Manajemen Sistem Informasi Berbasis ISO/IEC 27001:2005," *Jurnal Informasi*, vol. 7, no. 2, pp. 48–57, Nov 2015.

[3] D. Abdullah and C. I. Erliana, "Bisnis Rental Mobil Melalui Internet (E-Commerce) Menggunakan Algoritma SHA-1 (Sequre Hash Algorithm-1)," *Speed-Sentra Penelitian Engineering dan Edukasi*, vol. 4, no. 2, pp. 38–45, 2011.

[4] N. Agani, M. Hardjianto, D. Virgian, "Pengamanan Sistem Menggunakan One Time Password Dengan Pembangkit Password Hash SHA-256 dan Pseudo Random Number Generator (PRNG) Linear Congruential Generator (LCG) di Perangkat Berbasis Android,", *Budi Luhur Information Technology*, vol. 13, no . 1, 2016.

[5] S. Bahri, D. Diana, and P. S. Dian, "Studi dan Implementasi Pengamanan Basis Data Menggunakan Metode Enkripsi MD5 (Message-Digest Algorithm 5)," *Jurnal Ilmiah*, vo. 5, no. 1, pp. 1-15, 2012.

[6] D. M., Khairina, "Analisis Keamanan Sistem Login," *Jurnal Informatika Mulawarman*, vol. 6, no. 2: 64-67, 2011.

[7] R. Roshdy, M. Fouad, and M. Aboul-Dahab, "Design and Implementation a New Security Hash Algorithm Based on MD5 and SHA-256," *Intertional Journal of Engineering Sciences & Emerging Technologies*, vol. 6, no. 1, pp. 29-36, 2013.

[8] M. Ichwan, M. Gustian, and N. R. Nurjaman, "Implementasi Keyed-Hash Message Authentication Code pada Sistem Keamanan Rumah," *MIND Journal*, vol. 1, no. 1, pp. 9-18, 2016.

[9] R. Sadikin, *Kriptografi Untuk Keamanan Jaringan dan Implementasinya dalam Bahasa Java*, Yogyakarta: Andi, 2012.

[10] R. Munir, *Pengantar Kriptografi. Departemen Teknik Informatika Institut Teknologi Bandung*, Bandung, 2004.

[11] S. Ramajunam and M. Karrupiah, "Desaigning an algorithm with high Avalanche Effect," *International Journal of Computer Science and Network Security*, vol 11, no. 1, pp.106-111, 2011.

[12] A. Kumar and M. N. Tawiri, "Effective Implementation and Avalanche Effect of AES," *International Journal of Security, Privacy and Trust Management (IJSPTM)*, vol. 1, no. 3/4, pp. 31-35, 2012.

[13] Dafid, "Kriptografi Kunci Simetris Dengan Menggunakan Algoritma Crypton," *@ lgoritma*, vol. 2, no. 3, pp. 20–27, 2006.

[14] Y. Kurniawan, A. S. Ahmad, M. S. Mardiyanto, I. Supriana, and S. Sutikno, "Analisis Sandi Diferensial terhadap AES, DES dan AE1," *Journal of Mathematical and Fundamental Sciences*, vol. 38, no. 1, pp. 73–88, 2006.

[15] A. Karima and M. N. Diyatan, "Algoritma Kriptografi Gost Dengan Implementasi MD5 untuk Meningkatkan Nillai Avalanche Effect," *Techno. Com*, vol. 15, no. 4, pp. 292–302, 2016.