ISSN : 2548-6985

Perancangan Kios Buku Online Dengan Menerapkan Algoritma MD5 Dalam Pengamanan Record Login

[1)]Amin Setiawan Lahagu
STMIK Budi Darma, Jl. Sisimangaraja No.338 Medan, Sumatera Utara, Indonesia
www.stmik-budidarma.ac.id // Email : lahagua1@gmail.com
,
[2)] Harvei Desmon Hutahaean
STMIK Pelita Nusantara, Jl. Iskandar Muda No. 1 Medan, Sumatera Utara, Indonesia
www.stmik-budidarma.ac.id // Email : harvei.budidarma@gmail.com

ABSTRACT

Online book stall is one form of e-commerce in cyberspace, all the time the sale and purchase transactions of books can be done at any time, but along with the development of online transactions, data controls should also be improved. To avoid data from damage caused by irresponsible people, it is necessary to secure data by applying cryptography to the data.
In modern times there are many methods or algorithms that can be used to protect and control data from various attacks, but each of these algorithms certainly has its own weaknesses and strengths. The need for long-distance transactions and the importance of data security caused developers to continue to develop cryptography. MD5 algorithm is a one-way hash function that is often used in cryptography, especially in web-based transactions. The MD5 algorithm encodes data in a constant form of 32 bits. All processes from sales and purchase transactions in online bookstores will be safe and durable when the bookstore database is designed in such a way. The book stall database records are encoded with the MD5 algorithm, especially for logging records as a gateway for hackers or hackers to infiltrate. All transactions in the online book kiosk system are detected by the database, the resilience of the database from all attacks is the control of the system from all kinds of damage.

Keywords: Online book stall, Cryptography, MD5 hash algorithm.

PENDAHULUAN

Kios buku *online* adalah salah satu bentuk *e-commerce* yang memungkinkan terjadinya pasar di dunia maya, berkembangnya teknologi informasi membuka lapangan kerja baru kepada banyak orang. Kios buku *online* merupakan transformasi dari toko buku konvensional yang terbatas dalam memberikan informasi terhadap persediaan buku yang dimiliki. Kios buku *online* menyediakan buku dalam bentuk *soft copy* ataupun *hard copy*, penyediaan dalam bentuk *soft copy*, selain menawarkan produk yang *portable* juga efisien cenderung murah. Mudahnya melakukan transaksi *online* khususnya kios buku *online* menambah persaingan yang semakin hari semakin sengit, juga kios buku *online* sangat efektif dan selalu mengupdate informasi buku secara aktual.

Seiring berkembangnya transaksi melalui *internet* khususnya penjualan dan pembelian buku secara *online,* kejahatan semakin meningkat mulai dari pencurian *password,* melakukan pengubahan data rahasia bahkan melakukan penipuan dan pemalsuan diri untuk pembobolan kartu belanja. Cracker di dunia maya berusaha masuk ke *database* untuk melakukan pengrusakan. Pengamanan sekarang tidak hanya dibebankan pada *firewall* ataupun pada *instrusion system detection* tetapi pada *database* dengan cara melakukan penyandian terhadap *record database* tersebut sehingga sekalipun seorang cracker masuk ke *database* tetapi tidak bisa membaca isi asli dari *database* tersebut karena dalam bentuk *ciphertext*. Cracker melakukan aksinya dengan mencari *trap-door* dalam *website*, dalam hal ini memanfaatkan *form login* sebagai pintu masuknya.

Pengamanan *database* bisa dilakukan dengan menyandikan *record database* tersebut dengan algoritma *hash message*

digest 5. Fungsi algoritma *md5* terhadap keamanan *login* mencakup aspek *non-repudataion, data integrity, user authetication, dan privacy*. Algoritma *message digest 5* merupakan perbaikan dari algoritma *message digest 4* yang sama-sama menghasilkan *message digest* sepanjang 128 bit. Algoritma *md5* mengolah pesan dalam blok 512 *bit* setelah melewati tahap penambahan *padding bit*, tahap penambahan nilai panjang pesan semula dan tahap inisialisasi *buffer message digest*. Algoritma *message digest 5* merupakan algoritma *hash* satu arah, dimana *ciphertext* tidak bisa dikembalikan ke bentuk asli baik dengan menggunakan algoritma ataupun kunci. *Chiphertext* tidak bisa dimodifikasi, *ciphertext* tidak bisa digunakan untuk *login* ke dalam sistem sehingga membatasi ruang gerak cracker untuk menyerang[3].

Berdasarkan uraian latar belakang di atas, maka rumusan masalah pada penelitian ini adalah sebagai berikut : Bagaimana proses enkripsi menggunakan algoritma *hash md5*?, Bagaimana merancang antarmuka kios buku *online*?, Bagaimana cara menerapkan algoritma *hash md5* dalam pengamanan kios buku *online*?

Adapun tujuan penelitian ini adalah sebagai berikut :

1. Mengenkripsi *record login database* kios buku menggunakan algoritma *md5*.
2. Merancang antarmuka kios buku *online*.
3. Menerapkan algoritma *md5* untuk pengamanan kios buku *online*.

Adapun manfaat penelitian ini adalah sebagai berikut :

1. Mengamankan kios buku *online* dari berbagai serangan *cracker* atau *hacker*.
2. Meningkatkan kenyamanan pengguna dalam menggunakan kios buku *online*.

LANDASAN TEORI

2.1 Kriptografi

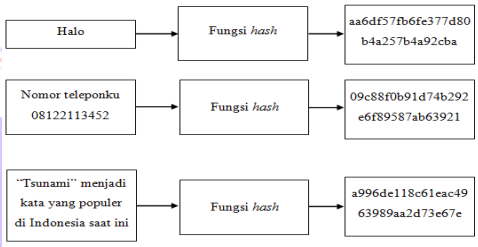
Kriptografi berasal dari bahasa Yunani, menurut bahasa dibagi menjadi dua *kripto* dan *graphia, kripto* berarti *secret* (rahasia) dan *graphia* berarti *writing* (tulisan). Menurut teminologinya kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari satu tempat ke tempat lain[3].

2.1.1 Fungsi *hash*

Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya sembarang dan mengkonversinya menjadi *string* keluaran yang panjangnya tetap (*fixed*), umumnya berukuran jauh lebih kecil daripada ukuran *string* semula. Fungsi *hash* dapat menerima masukan *string* apa saja. Jika *string* menyatakan pesan (*message*), maka sembarang pesan *M* berukuran bebas dikompresi oleh fungsi *hash H* melalui persamaan

$h = H(M)$……(1)

Keluaran fungsi *hash* disebut juga nilai *hash* (*hash-value*) atau pesan ringkas (*message digest*). Pada persamaan (2.1), *h* adalah nilai *hash* atau *message digest* dari fungsi *H* untuk masukan *M.* Dengan kata lain, fungsi *hash* mengkompresi sembarang pesan yang berukuran berapa saja menjadi *message digest* yang ukurannya selalu tetap (dan lebih pendek dari panjang pesan semula). Gambar 1 memperlihatkan contoh 3 buah pesan dengan panjang yang berbeda-beda selalu di-*hash* menghasilkan pesan ringkas yang panjangnya tetap (dalam contoh ini pesan ringkas dinyatakan dalam kode *hexadesimal* yang panjangnya 128 *bit* .



Gambar 1 Contoh hashing beberapa buah pesan[2]

2.1.2 MD5

MD5 adalah salah satu penggunaan fungsi *hash* satu arah yang paling banyak digunakan. *MD5* merupakan fungsi *hash* kelima yang dirancang oleh Ron Rivest dan didefinisikan pada *RFC* 1321. *MD5* merupakan pengembangan dari *MD4* dimana terjadi penambahan satu ronde. *MD5* memproses teks masukan ke dalam *block-block bit* sebanyak 512 *bit*, dibagi ke dalam 32 *bit subblock* sebanyak 16 buah. Keluaran dari *MD5* berupa 4 buah *block* yang masing-masing 32 *bit* yang mana akan menjadi 128 *bit* yang biasa disebut nilai *hash* [4].

2.1.2.1 Cara Kerja *MD5*

MD5 mengolah *block* 512 *bit*, dibagi ke dalam 16 *subblock* berukuran 32 *bit*. Keluaran algoritma diset menjadi 4 *block* yang masing-masing berukuran 32 *bit* yang setelah digabungkan akan membentuk nilai *hash* 128 *bit*. Adapun langkah-langkah pembuatan *message digest* adalah sebagai berikut:

2.1.2.1.1 Penambahan Bit-bit Pengganjal

ISSN : 2548-6985

Pesan ditambah dengan sejumlah *bit* pengganjal sedemikian sehingga panjang pesan (dalam satuan *bit*) kongruen dengan 448 *modulo* 512. Ini berarti panjang pesan setelah ditambahi *bit-bit* pengganjal adalah 64 *bit* kurang dari kelipatan 512. Angka 512 ini muncul karena *md5* memproses pesan dalam blok-blok yang berukuran 512. Pesan dengan panjang 448 *bit* pun tetap ditambah dengan *bit-bit* pengganjal. Jika panjang pesan 448 *bit,* maka pesan tersebut ditambah dengan 512 *bit* menjadi 960 *bit.* Jadi, panjang *bit-bit* pengganjal adalah antara 1 sampai 512. *Bit-bit* pengganjal terdiri dari sebuah *bit* 1 diikuti dengan sisanya bit 0 [2].

2.1.2.1.2 Penambahan Nilai Panjang Semula

Pesan yang telah diberi *bit-bit* pengganjal selanjutnya ditambah lagi dengan 64 *bit* yang menyatakan panjang pesan semula. Jika panjang pesan > 2^{64} maka yang diambil adalah panjangnya dalam *modulo* 2^{64}. Dengan kata lain, jika panjang pesan semula adalah K *bit,* maka 64 *bit* yang ditambahkan menyatakan K *modulo* 2^{64}. Setelah ditambah dengan 64 *bit,* panjang pesan sekarang menjadi kelipatan 512 *bit* [2].

2.1.2.1.3 Inisialisasi Penyangga *MD*

MD5 membutuhkan 4 buah penyangga (*buffer*) yang masing-masing panjangnya 32 *bit*. Total panjang penyangga adalah 4 x 32 = 128 *bit*. Keempat penyangga ini menampung hasil antara dan hasil akhir. Keempat penyangga ini diberi nama *A, B, C,* dan *D*. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi *HEX*) sebagai berikut: *A* = 012345678, *B* = 89ABCDEF, *C* = FEDCBA98, *D* = 76543210 dan beberapa versi *MD5* menggunakan nilai inisialisasi berbeda, yaitu: *A* = 67452301, *B* = EFCDAB89, *C* = 98BADCFE, *D* = 10325467. (Rinaldi Munir, 2006, hal. 221).

2.1.2.1.4 Pengolahan Blok Berukuran 512 *bit*

Pesan dibagi menjadi *L* buah blok yang masing-masing panjangnya 512 *bit* (Y_0 sampai Y_{L-1}). Setiap blok 512 *bit* diproses bersama dengan penyangga *MD* menjadi keluaran 128 *bit,* dan ini disebut proses H_{MD5}. Pada *md5* juga terdapat 4 (empat) buah fungsi *nonliniear* yang masing-masing digunakan pada tiap operasinya (satu fungsi untuk satu *block)*. Adapun fungsi tersebut sebagai berikut:

$F(X,Y,Z) = (X \wedge Y) \vee ((\neg X) \wedge Z)$(2)

$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$(3)

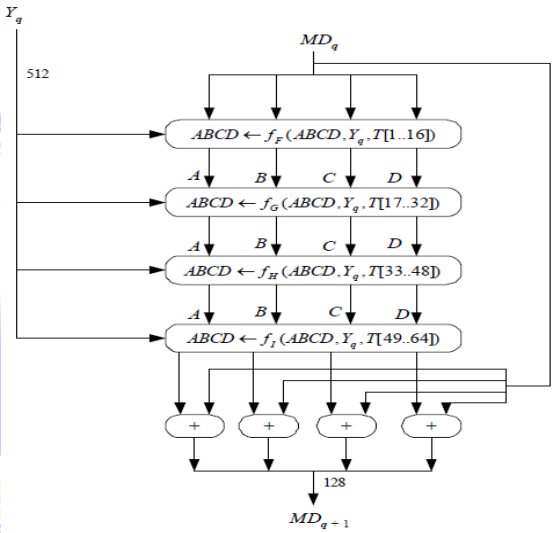
$H(X,Y,Z) = X \oplus Y \oplus Z$(4)

$I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$(5)

Adapun yang dilakukan setelah tahap pengolahan blok 512 *bit* adalah melakukan operasi-operasi sebagai berikut :

FF(a,b,c,d,M_j,s,ti) (6)
GG(a,b,c,d,M_j,s,ti) (7)
HH(a,b,c,d,M_j,s,ti)(8)
II(a,b,c,d,M_j,s,ti) (9)

FF(a,b,c,d,M_j,s,ti) menunjukkan a = b + ((a + F(b,c,d) + M_j + ti) <<< s), GG(a,b,c,d,M_j,s,ti) menunjukkan a = b + ((a + G(b,c,d) + M_j + ti) <<< s), HH(a,b,c,d,M_j,s,ti) menunjukkan a = b + ((a + H(b,c,d) + M_j + ti) <<< s), II(a,b,c,d,M_j,s,ti) menunjukkan a = b + ((a + I(b,c,d) + M_j + ti) <<< s). M_j menggambarkan pesan ke-j dari *subblock* (dari 0 sampai 15) dan <<<s menggambarkan *bit* akan digeser ke kiri sebanyak s *bit.*



Gambar 2 Pengolahan blok 512 bit (Proses MD5) [2]

Proses H_{md5} terdiri dari 4 buah putaran, dan masing-masing putaran melakukan operasi dasar *md5* sebanyak 16 kali dan setiap operasi dasar memakai sebuah elemen *T.* Jadi setiap putaran memakai 16 elemen Tabel *T.* Y_q menyatakan blok 512 *bit* ke-q dari pesan yang telah ditambah *bit-bit* pengganjal dan tambahan 64 *bit* nilai panjang pesan semula. MD_q adalah nilai *message digest* 128 bit dari proses H_{md5} ke-*q.* Pada awal proses, MD_q berisi nilai inisialisasi penyangga *MD.*

Keluaran dari *md5* adalah 128 *bit* dari *word* terendah A [2].

PEMBAHASAN

3.1 Analisa Masalah

Pada bagian pembahasan masalah disini, akan dianalisa bagaimana cara kerja algoritma *hash md5* dalam mengenkripsi *record* pada *database* kios buku. *Database* kios buku ini dinamakan kios_data. Kios buku terdiri dari

beberapa tabel yaitu tabel kios_admin, tabel kios_buku, tabel kios_belanja, tabel kios_*member*, tabel kios_*guestbok*, tabel *news*. Maka, proses mengenkripsi *record* pada *database* tersebut dengan menggunakan algoritma *md5,* akan dibagi dalam beberapa tahap yang dijelaskan pada subbab-subbab berikut ini.

3.1.1 Analisis Penambahan *Bit-bit* Pengganjal

Adapun *record* pada *database* yang dimaksud di atas menunjuk pada tabel kios_*admin* seperti pada tabel berikut ini:

Database : kios_data

Tabel 1 Record pada tabel kios_admin

Id	*User*	*Pass*	*Sesi*	*Aktif*
1	AMINSLAHAGU	ADM11	*Hash(User+Pass)*	1

Pada tabel 1 di atas diambil contoh untuk *record* AMINSLAHAGU pada *field user*. Kemudian *record* tersebut diubah ke dalam bentuk *hexadecimal* sebagai berikut ini:

a. *Plaintext* = AMINSLAHAGU
b. Dalam *hexa* = 414D494E534C4148414755
c. Panjang *Plaintext* = 88 *bit*
d. Panjang *bit* pengganjal = (448 *module* 512) – 88 *bit* = 360
e. *Bi –bit* pengganjal dalam *biner* =

01000001 01001101 01001001 01001110
01010011 01001100 01000001 01001000
01000001 01000111 01010101 10000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

f. Maka, *Plaintext* (*P*) =

41 4D 49 4E 53 4C 41 48 41 47 55 80
00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00
00 00

3.1.2 Analisis Penambahan Panjang *Plaintext*

Setelah penambahan *bit-bit* pengganjal pada *plaintext* kemudian dilanjutkan dengan penambahan panjang pesan semula dan diikuti dengan penambahan 64 *bit* bilangan 0 sehingga *plaintext* berjumlah genap 512 *bit* . Adapun langkah-langkah penambahan tersebut sebagai berikut ini.

a. P_{awal}(*Hexa*) =
41 4D 49 4E 53 4C 41 48 41 47 55
b. P_{awal}(*Biner*) =
01000001 01001101 01001001 01001110
01010011 01001100 01000001 01001000
01000001 01000111
01010101
c. Panjang pesan semula = 88 *bit*
d. Maka, P =

41 4D 49 4E 53 4C 41 48 41 47 55 80 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 58 00 00
00 00 00 00 00 00

3.1.3 Inisialisasi Penyangga *Message Digest*

Tahap inisialisasi karena beberapa versi *MD5* memiliki nilai inisialisasi berbeda maka penulis memilih salah satu dari beberapa inisialisasi yang ditentukan, A = 67452301, B = EFCDAB89, C = 98BADCFE, D = 10325476.

3.1.4 Pembagian *Plaintext* dalam L buah *block* ($Y_0 – Y_{L-1}$)

Pada tahap ini *padding bit* dimasukkan ke dalam *array*:

X[0] = 4e494d41
X[1] = 48414c53
X[2] = 80554741
X[3] = 00000000
X[4] = 00000000
X[5] = 00000000
X[6] = 00000000
X[7] = 00000000
X[8] = 00000000
X[9] = 00000000
X[10] = 00000000
X[11] = 00000000
X[12] = 00000000
X[13] = 00000000
X[14] = 00000058
X[15] = 00000000

3.1.5 Proses Operasi *MD5*

Adapun persamaan untuk proses *MD5* tersebut adalah =

Round 1: a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s)

Fungsi yang digunakan pada masukan F(b,c,d) = (b ∧ c) ∨ ((~b) ∧ d)

Operasi 1:

FF (a, b, c, d, x[0], 0xD76AA478)

FF : a = 67452301, b = efcdab89, c = 98badcfe, d = 10325476, x0 = 4e494d41,
s = 7, ti = d76aa478

F = (efcdab89 ∧ 98badcfe) ∨ ((~efcdab89) ∧ 10325476)
= (88888888) ∨ (10325476)
= 98badcfe

a + F(b,c,d) + x0 + ti = 67452301 + 98badcfe + 4e494d41 + d76aa478
= 225b3f1b8 *mod* 2^{32}
= 25b3f1b8 = 0010 0101 1011 0011 1111 0001 1011 1000
CLS_s(a + F(b,c,d) + x0 + ti) = CLS_7(25b3f1b8)
= 1101 1001 1111 1000 1101 1100 0001 0010
= d9f8dc12
= b + d9f8dc12
= (efcdab89 + d9f8dc12) *mod* 2^{32}
= c9c6879b
Ciphertext = ABCD = 0fe6de04 749d3c8d f173111f 7f548a0a

ALGORITMA DAN IMPLEMENTASI

4.1 Algoritma

Algoritma merupakan langkah-langkah yang tersusun secara sistematis untuk menyelesaikan suatu masalah atau menerangkan sesuatu hal yang akan dilakukan. Algoritma digunakan untuk menganalisa serta menjelaskan urutan dan hubungan antara kegiatan yang akan ditempuh. Dalam skripsi ini penulis membuat algoritma sebagai berikut.

4.1.1 Algoritma Pengamanan *Record*

Adapun algoritma pengamanan *record* menggunakan algoritma *md5* di bawah ini, dalam mengenkripsi suatu *plaintext* bahwa semua penyangga adalah 32 *bit* dan penjumlahan dalam *modula* 2^32.

Input = P_i ← *Plaintext*
= A ← Penyangga
= B ← Penyangga
= C ← Penyangga
= D ← Penyangga
Output = C_i ← *Ciphertext*
Proses =

```
for (i = 0; 1 <= 63; i++) {
    if (0 · i · 15) {
f = (b and c) or ((not b) and d);
g = i;
    }
    else if (16 · i · 31) {
f = (d and b) or ((not d) and c);
g = (5xi + 1) mod 16;  }
    else if (32 · i · 47) {
f = b xor c xor  d;
        g = (3xi + 5) mod 16; }
    else if (48 · i · 63) {
f = b xor c xor  (b or (not d));
        g = (7xi) mod 16; }
    temp = d;
d = c;
c = b;
b = ((a + f + T[i] +X(g)) <<< s[i]) + b
    a = temp}
```

4.2 Implementasi

Pengujian sistem merupakan proses mencoba aplikasi apakah telah sesuai dengan rancangan sebelumnya, sehingga bisa dipastikan apakah hasil dari pada eksekusi program telah sesuai dengan *output* dari tujuan perancangan, melakukan pengecekan dari proses enkripsi yang dilakukan apakah sesuai dengan perhitungan algoritma *md5*.

4.2.1 Tampilan *Record Login User*

Adapun pada gambar di bawah terlihat ada *field user, pass,* dan *sesi* masing-masing memiliki panjang yang tetap karena disandikan dengan algoritma *MD5*.

	mid	user	pass	sesi	nama
	3	c9d2cce909ea37234be8af1a1f958805	707bedd98d8abd4346de94fffa35b5c5	5850ce4501dddf8f0fc5fcc1b2e293d8	Annisa Wulandari
	7	ee11cbb19052e40b07aac0ca060c23ee	5f4dcc3b5aa765d61d8327deb882cf99	3ae9ea5fe7ad5bf652c51f43da57422c	sdasfas
	17	0fe6de04749d3c8df173111f7f548a0a	73acd9a5972130b75066c82595a1fae3	e21ed2761975351469bfc2550b79df00	AMIN SETIAWAN LAHAGU

Gambar 3. Tampilan Record Login User

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah menyelesaikan penelitian ini, penulis berkesimpulan sebagai berikut:

1. Panjang atau pendek dari *plaintext* yang akan disandikan tidak berpengaruh terhadap kecepatan karena setiap *plaintext* harus dalam bentuk *block-block* 512 *bit*. Penyandian menggunakan algoritma *md5* membutuhkan 64 operasi dan setiap operasi mengalami pergeseran secara sikuler. Penyandian secara manual harus memiliki ketelitian yang tinggi karena kesilafan 1 *bit* saja menyebabkan kesalahan fatal kecuali diulangi dari awal. Struktur algoritma yang digunakan tidak terlalu rumit, cukup dengan pemahaman operasi sitem bilangan.
2. Alat bantu untuk merancang kios buku bisa dilakukan dengan menggunakan *UML* dan *DFD.* Untuk perancangan juga diperlukan konsistensi terhadap tampilan,

struktur dari *database* disesuaikan dengan ukuran data yang akan disimpan, dalam hal ini *md5* memerlukan ukuran *record* yang tetap.
3. Dengan memanfaatkan fungsi yang didukung oleh pemrograman *php* maka proses penyandian dalam pengamanan *record login* sangat mudah dilakukan. *Record* yang sudah dienkripsi memiliki panjang karakter yang tetap yaitu 32 karakter.

5.2 Saran

Untuk menyempurnakan penelitian ini dan benar-benar bermanfaat maka ada beberapa saran penulis adalah sebagai berikut:

1. Algoritma *md5* adalah salah satu fungsi *hash* satu arah dari berbagai algoritma *hash* yang lain, algoritma *md5* menghasilkan *ciphertext* sepanjang 128 *bit.* Untuk pengembangan selanjutnya diperlukan algoritma yang lebih kuat seperti *SHA* yang menghasilkan *ciphertext* lebih panjang yaitu 160 *bit*, *SHA* memerlukan 5 penyangga dibanding *md5* yang memerlukan 4 penyangga saja.
2. Pengamanan sistem bisa dikembangkan dengan menerapkan pada *record* gambar misalnya untuk pengamanan kartu belanja.
3. Untuk pengamanan web yang lebih aman maka diperlukan penyandian terhadap seluruh *record* dari *database*, tidak hanya pada *record login* saja.

DAFTAR PUSTAKA

1. Dony Ariyus, “Keamanan Data dan Komunikasi”, Penerbit Graha Ilmu, Yogyakarta, Edisi 1, 2006
2. Rinaldi Munir, “Kriptografi”, Penerbit Informatika Bandung, Bandung, 2006
3. Dony Ariyus, Rum Andri K.R, “Komunikasi Data”, Penerbit Andi, Yogyakarta, Edisi 1, 2008
4. Kusrini, “Konsep dan Aplikasi Sistem Pendukung Keputusan”, Penerbit Andi, Yogyakarta, Edisi 1, 2007
5. Aghus Sofwan etc., “Aplikasi Kriptografi dengan Algoritma Message Digest 5 (MD5)”, 2006, Jurnal Teknik Elektro Fakultas Universitas Diponegoro
6. Jogiyanto H.M, “Analisa dan Sistem Informasi”, Penerbit Andi, Yogyakarta, 1999
7. Ibnu Aqil, “Sistem Informasi Alumni Program Diploma Pada Bina Sriwijaya Palembang Berbasis Web”, 2010, Jurnal Akademi Manajemen Informatika dan Komputer Bina Sriwijaya Palembang
8. http://www.duniailkom.com/pengertian-dan-fungsi-php-dalam-pemograman-web/, tanggal akses 26 Maret 2015
9. http://www.kamusilmiah.com/it/mengenal-apa-itu-html-hyper-text-markup-language/