ANALISIS PERBANDINGAN WAKTU ENKRIPSI MENGGUNAKAN ALGORITMA MESSAGE-DIGEST 5 (MD5) DAN ALGORITMA AFFINE CIPHER DALAM ENKRIPSI PESAN

Pembimbing :

OLEH:

AIDIL HALIM LUBIS

NIP. 19880527 201903 1 010

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUMATERA UTARA
MEDAN
2020

Judul : ANALISIS PERBANDINGAN WAKTU ENKRIPSI ALGORITMA MESSAGE-DIGEST 5 (MD5) DAN ALGORITMA AFFINE CIPHER DALAM ENKRIPSI PESAN

Nama : Aidil Halim Lubis

NIP : 19880527 201903 1 010

SURAT REKOMENDASI

Saya yang bertanda tangan di bawah ini, menyatakan bahwa penelitian saudara :

Nama	: **Aidil Halim Lubis, M.Kom**
NIP	: 19880527 201903 1 010
Tempat/tanggal lahir	: Binjai, 27 Mei 1988
Jenis Kelamin	: Laki-laki
Agama	: Islam
Pangkat/Gol	: Penata Muda TK.I (III/b)
Unit Kerja	: Fakultas Sains dan Teknologi UIN Sumatera Utara Medan
Judul Penelitian	: Analisis Perbandingan Waktu Enkripsi Menggunakan Algoritma Message-Digest 5 (MD5) dan Algoritma Affine Cipher dalam Enkripsi Pesan

Telah memenuhi syarat sebagai suatu karya ilmiah, setelah membaca dan memberikan masukan saran-saran terlebih dahulu.

Demikian surat rekomendasi ini diberikan untuk dapat dipergunakan seperlunya.

Medan, 10 Desember 2020

Konsultan

Dr. Mhd Furqan, S.Si., M.Comp.Sc

NIP 19800806 200604 1 003

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI ILMU KOMPUTER

AIDIL HALIM LUBIS

Analisis Perbandingan Waktu Enkripsi Menggunakan Algoritma Message-Digest 5 (MD5) dan Algoritma Affine Cipher Dalam Enkripsi Pesan

ABSTRAK

Kriptografi merupakan ilmu yang digunakan untuk menyamarkan pesan yang akan dikirim/transmit oleh pengirim ke penerima pesan. Teknik untuk mengamankan pesan adalah dengan mengenkripsi pesan tersebut menjadi tidak dapat dibaca secara utuh atau informasi sudah diacak. Salah satu algoritma yang cukup banyak digunakan sampai saat ini yaitu algoritma MD5 yang hasil enkripsinya berupa hash dengan panjang 32 bit. Kemampuan algoritma MD5 diakui sampai sekarang masih digunakan dalam melakukan enkripsi pada banyak aplikasi maupun website. Algoritma affine cipher merupakan algoritma dengan tipe kunci simetris yang juga sering digunakan dalam mengamankan pesan. Pada penelitian ini, penulis menganalisis kinerja algoritma MD5 dan juga algoritma affine cipher berupa analisis waktu enkripsi berdasarkan panjang pesan yang akan dienkripsi serta cipherteks yang dihasilkan untuk dari proses enkripsi dari kedua algoritma tersebut.

Kata kunci : enkripsi, affine cipher, MD5, cipherteks

KATA PENGANTAR

Puja dan puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga laporan penelitian ini dapat terselesaikan. Penelitian ini berjudul "Analisis Perbandingan Waktu Enkripsi Menggunakan Algoritma Message-Digest 5 (MD5) dan Algoritma Affine Cipher dalam Enkripsi Pesan". Penyusunan laporan tidak terlepas dari bantuan dan dorongan dari berbagai pihak, baik moril maupun materiil. Untuk itu dengan segala hormat penulis mengucapkan terimakasih kepada rekan-rekan yang telah membantu dan terutama kepada Konsultan yang memberi koreksi dan masukan berharga.

Penulis menyadari bahwa semua yang tertuang dalam laporan ini jauh dari sempurna. Oleh karena itu penulis mengharapkan masukan berupa kritik dan saran demi kesempurnaan penelitian ini. Akhirnya penulis berharap semoga laporan penelitian ini dapat bermanfaat.

Medan, Desember 2020

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dengan semakin berkembangnya pemanfaatan teknologi informasi dalam membantu pekerjaan manusian dalam kehidupan sehari-hari diberbagai jenis kegiatan yang melibatkan komputer sebagai media utamanya. Maka aspek keamanan menjadi sangat penting dalam system informasi, beberapa informasi umumnya hanya ditujukan bagi komunitas pengguna tertentu. Oleh karena itu keamanan sangat dibutuhkan untuk mencegah informasi tersebut digunakan oleh pengguna yang tidak sah. Kejahatan yang sekarang ini sering terjadi dalam penggunaan teknologi informasi yang menjadi ancaman antara lain :

1. Penyusupan kedalam suatu sistem jaringan komputer secara tidak sah tanpa izin ataupun sepengetahuan dari pemilik sistem tersebut.
2. Memasukkan data atau informasi kedalam internet tentang suatu hal yang tidak benar dan melanggar hokum.
3. Pemalsuan dokumen-dokumen penting dan disebarluaskan melalui internet.
4. Melakukan tindakan mata-mata terhadap individu atau organisasi untuk mendapat keuntungan pribadi.
5. Melakukan gangguan, perusakan atau penghancuran data, program komputer atau sistem jaringan komputer yang terhubung ke internet. (Golose, 2006)

Hal ini merupakan ancaman, sehingga banyak pihak yang berusahan untuk menyandikan informasi yang dimiliki sebelum pesan tersebut ditransmit ke orang lain yang memang memiliki hak untuk informasi tersebut. Menyandikan informasi tersebut membuat pesan menjadi tidak memiliki makna yang beraturan dan cenderung suliti dipahami secara langsung serta sulit untuk dipecahkan. Cara yang digunakan untuk hal tersrebut adalah dengan menggunakan kriptografi. Kriptografi merupakan metode untuk mengamankan data, baik berupa teks, maupun gambar. Metode ini dilakukan dengan melakukan penyandian pesan kedalam bentuk yang tidak dipahami oleh orang lain maupun orang yang tidak memiliki hak dalam pesan tersebut. Dalam kriptografi dikelompokan menjadi 2 bagian yaitu kriptografi simetris dan asimetris.

Algoritma MD5 atau *Message Digest 5* adalah algoritma yang dirancang dengan tujuan keamanan. Keluaran hash yang memiliki panjang 128 bit disebut message digest yang berasal dari masukkan dengan panjang yang berbeda (Sallam, et al. 2012).

Kriptografi juga memiliki kerentanan dalam hal pemecahan pesan. Kasgar (2013) meneliti bahwa dalam kriptografi ada beberapa bentuk serangan yang dapat dilakukan untuk memecahkan pesan dari enkripsi kriptografi antara lain serangan brute force atau exhaustive key. Strategi ini digunakan melalui teori data enkripsi oleh penyerang yang mengambil keuntungan dari setiap kelemahan dalam sistem enkripsi sehingga lebih mudah. Dalam serangan brute force secara sistematis memeriksa semua kemungkinan kunci yang muncul sampai kunci yang sebenarnya ditemukan.

Algoritma Affine cipher merupakan sebuah algoritma klasik yang merupakan perluasan dari algoritma Caesar cipher. Pada algoritma ini akan dilakukan subsitusi terhadap plaintext dimana dalam

setiap huruf pada alphabet dipetakan ke dalam bentuk numerik dan dilakukan proses enkripsi dengan menggunakan metode matematika sederhana (Shukla & Verma, 2014).

Berdasarkan dari latar belakang diatas, penulis tertarik untuk melakukan penelitian ini yang akan membandingkan kedua algoritma tersebut dalam melakukan enkripsi pesan. Penulis memberikan judul " Analisis perbandingan waktu enkripsi menggunakan Algoritma Message Digest 5 (MD5) dan Algoritma Affice cipher dalam enkripsi pesan".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang diuraikan bahwa algoritma MD5 dan algoritma Affice cipher adalah algoritma yang mampu melakukan enkripsi pesan. Kedua algoritma tersebut juga memiliki kerentan dalam hal serangan yang dilakukan untuk pemecahan pesan oleh orang yang tidak memiliki hak akses atau kripanalis.

1.3 Batasan Masalah

Agar penelitian ini mendapatkan hasil yang tidak menyimpang dari hal-hal yang telah diuraikan sebelumnya, maka diperlukan sebuah batasan masalah. Selain itu batasan masalah membuat penelitian menjadi lebih terarah sehingga tujuan penlitian dapat tercapai.

Adapun yang menjadi batasan masalah dalam penelitian ini adalah :

1. Proses enkripsi tidak akan dilakukan pada pesan image atau suara.
2. Panjang teks yang akan dienkripsi maksimal 2048 karakter.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah membandingkan waktu yang dibutuhkan untuk melakukan enkripsi pesan menggunakan kedua algoritma tersebut. Sehingga waktu enkripsi tersebut berpengaruh terhadap kerentan serangan pada masing-masing pesan menggunakan enkripsi algoritma kriptografi MD5 dan Affine cipher.

1.5 Manfaat Penelitian

Melalui penelitian ini penulis mengharapkan dapat memahami tentang kriptografi yang nantinya dapat membantu dalam pekerjaan pada bidang cybersecurity dan lainnya. Serta dapat dimanfaat bagi orang lain yang membaca serta memahami penelitian ini untuk bidang-bidang lainnya.

BAB II

TINJAUAN PUSTAKA

2.1 Keamanan

Keamanan merupakan komponen yang vital dalam komunikasi data elektronis. Masih banyak yang belum menyadari bahwa keamanan (*security)* merupakan sebuah komponen penting yang tidak murah. Teknologi kriptografi sangat berperan juga dalam proses komunikasi, yang digunakan untuk melakukan enkripsi (pengacakan) data yang ditransaksikan selama perjalanan dari sumber ke tujuan dan juga melakukan dekripsi (menyusun kembali) data yang telah teracak tersebut. Berbagai sistem yang telah dikembangkan adalah sistem private key dan public key. Penguasaan algoritma-algoritma popular digunakan untuk mengamankan data juga sangat penting. Contoh-contoh algoritma ini antara lain : DES, IDEA, RC5, RSA dan ECC (*Elliptic Curve Cryptography*).

Dari sisi tindakan pihan yang bertanggung jawab, keamanan jaringan komputer terbagi dua level :

1. keamanan fisik peralatan mulai dari server, terminal/client router sampai dengan cabling.
2. keamanan sistem sekiranya ada penyusup yang berhasil mendapatkan akses ke saluran fisik jaringan komputer. Sebagai contoh, dalam sistem mainframe-dumb terminal di suatu gedung perkantoran, mulai dari komputer sentral sampai ke terminal secara fisik keamanan peralatan dikontrol penuh oleh otoritas sentral. Manakala sistem tersebut hendak diperpanjang sampai ke kantor-kantor cabang di luar gedung, maka sedikit banyak harus menggunakan komponen jaringan komputer yang tidak sepenuhnya dikuasai pemilik sistem seperti menyewa kabel leased line atau menggunakan jasa komunikasi satelit.

Dari sisi pemakaian, sistem keamanan dipasang untuk mencegah :

1. Pencurian
2. Kerusakan
3. Penyalahgunaan data yang dikirim melalui jaringan komputer

Dalam prakteknya, pencurian data berwujud pembacaan oleh pihak yang tidak berwenang atau hak izin akses, biasanya dengan menyadap saluran publik. Teknologi jaringan komputer telah dapat mengurangi bahkan membuang kemungkinan adanya kerusakan data akibat buruknya konektivitas fisik namun kerusakan tetap bisa terjadi karena celah (bug) pada program aplikasi atau ada unsur kesengajaan yang mengarah ke penyalahgunaan sistem.

2.2 Mekanisme Kriptografi

Suatu sistem kriptografi (kriptosistem) bekerja dengan cara menyandikan suatu pesan menjadi suatu kode rahasia yang dimengerti oleh pelaku sistem informasi saja. Pada dasarnya mekanisme kerja semacam ini telah dikenal sejak jaman dahulu. Bangsa Mesir kuno sekitar 4000 tahun yang lalu bahkan telah mempraktekkannya dengan cara yang sangat primitive. Dalam era teknologi informasi sekarang ini, mekanisme yang sama masih digunakan tetapi tentunya implementasi sistemnya berbeda. Beberapa istilah yang umum digunakan dalam pembahasan kriptografi.

2.1 Plainteks

Plainteks (messege) merupakan pesan asli yang dikirimkan dan dijaga keamanannya. Pesan tidak lain dari informasi tersebut.

2.2 Cipher

Cipher merupakan algoritma matematis yang digunakan untuk proses penyandian plainteks menjadi cipherteks.

2.3 Enkripsi

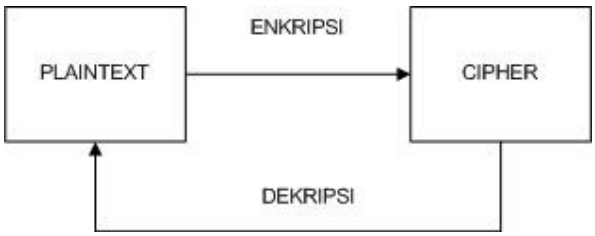
Enkripsi (encryption) merupakan proses yang dilakukan untuk menyandikan plainteks sehingga menjadi cipherteks.

2.4 Dekripsi

Dekripsi (decryption) merupakan proses yang dilakukan untuk memperoleh kembali plainteks dari cipherteks.

2.5 Kriptosistem

Kriptosistem merupakan sistem yang dirancang untuk mengamankan suatu sistem informasi dengan memanfaatkan kriptografi. Urutan-urutan proses kriptografi dapat digambarkan seperti dibawah ini :



Gambar 1. Mekanisme kriptografi

Prosesnya pada dasarnya sangat sederhana. Sebuah plaintext (m) akan dilewatkan pada proses enkripsi (E) sehingga menghasilkan suatu ciphertext (c). kemudian untuk memperoleh kembali plaintext, maka ciphertext (c) melalui proses dekripsi (D) yang akan menghasilkan kembali plaintext (m). secara matematis proses ini dapat dinyatakan sebagai :

$E(m) = c$

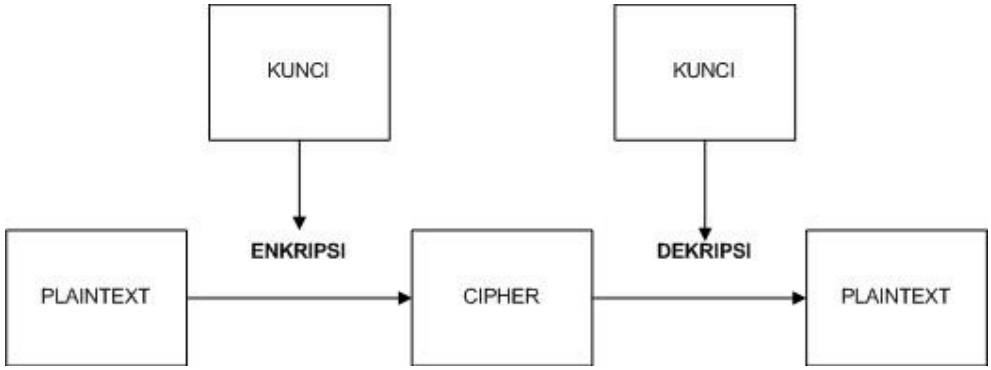
$D(c) = m$

$D(E(m)) = m$

Kriptografi sederhana ini menggunakan algoritma penyandian yang disebut cipher. Keamanannya bergantung pada kerahasiaan algoritma penyandian tersebut, karena itu algoritmanya harus dirahasiakan. Pada kelompok dengan jumlah besar dan anggota yang senantiasa berubah, penggunaannya akan menimbulkan masalah. Setiap ada anggota yang meninggalkan kelompok, algoritma harus diganti (Munir, 2011) karena anggota ini dapat saja membocorkan algoritma tersebut sehingga dapat membahayakan keamanan data.

Kriptografi modern selain memanfaatkan algoritma juga menggunakan kunci (*key*) untuk memecahkan masalah tersebut. Proses enkripsi dan dekripsi dilakukan dengan menggunakan kunci

ini. Setiap anggota memiliki kuncinya masing-masing yang digunakan untuk proses enkripsi dan dekripsi yang akan dilakukannya. Proses tersebut dapat digambarkan seperti gambar berikut.



Gambar 2. Kriptografi berbasis kunci

Mekanisme kriptografi seperti ini dinamakan kriptografi berbasis kunci. Dengan demikian kriptosistemnya akan terdiri atas algoritma dan kunci, beserta segala plaintext dan ciphertextnya. Dengan persamaan matematisnya menjadi seperti berikut,

$$E_e(m) = c$$

$$D_d(c) = m$$

$$D_d(E_e(m)) = m$$

Dengan,

e = kunci enkripsi

d = kunci dekripsi

2.3 Tujuan Kriptografi

Dalam kriptografi ada beberapa hal yang perlu dimiliki oleh sebuah kriptografi dalam mengamankan data yaitu :

a. *Confidentiality*
   Menjamin bahwa data-data tersebut hanya bisa diakses oleh pihak-pihak tertentu saja.
b. *Authentication*
   Baik pada saat mengirim atau menerima informasi, kedua belah pihak perlu mengetahui bahwa pengirim dari pesan tersebut adalah orang yang sebenarnya seperti yang diklaim.
c. *Integrity*
   Aspek ini berhubungan dengan jaminan setiap pesan yang dikirim pasti sampai pada penerimanya tanpa ada bagian dari pesan atau informasi tersebut yang diganti, diduplikasi, dirusak, diubah urutannya dan ditambahkan
d. *Nonrepudation*
   Aspek ini mencegah pengirim dan penerima mengingkari bahwa mereka telah mengirimkan atau menerima suatu pesan atau informasi.

e. *Access control*
   Membatasi sumber-sumber data hanya kepada orang-orang tertentu.
f. *Availability*
   Jika dibutuhkan setiap saat semua informasi pada sistem komputer harus tersedia bagi semua pihak yang berhak atas informasi tersebut.

Tujuan tersebut dapat dicapai tidak hanya bergantung kepada algoritma matematik dan mekanismenya, tetapi ada aspek lain seperti prosedur penyampaian yang digunakan atau perlindungan hukum.

2.4 Kriptografi Simetris dan Asimetris

Berdasarkan jenis kunci yang digunakan dalam proses enkripsi dan dekripsi, kriptografi dapar dibedakan menjadi dua jenis, yaitu kriptografi simetrik dan kriptografi asimetrik. Perbedaan utama diantara keduanya terletak pada sama dan tidaknya kunci yang digunakan dalam enkripsi dengan kunci yang digunakan pada proses dekripsi (Zelvina, at all. 2012).

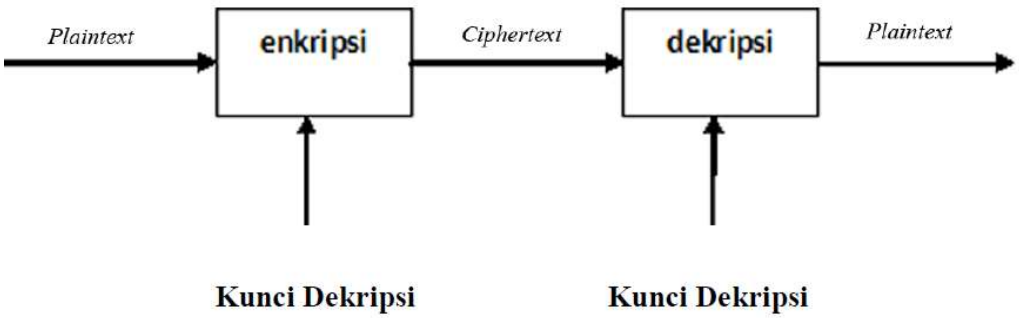
2.4.1 Kriptografi Simetris

Algoritma ini mengasumsikan pengirim dan penerima pesan sudah berbagi kunci yang sama sebelum bertukar pesan. Algoritma simetris *(symmetric algorithm)* adalah suatu algoritma dimana kunci enkripsi yang digunakan sama dengan kunci dekripsi, sehingga algoritma ini disebut juga sebagai *single-key algorithm.* Algoritma ini sering juga disebut dengan algoritma klasik karena memakai kunci yang sama untuk kegiatan enkripsi dan dekripsi. Algoritma ini sudah ada sejak lebih dari 4000 tahun yang lalu.Semua algoritma kriptografi klasik termasuk kedalam sistem kriptografi simetri.Keamanan dari pesan yang menggunakan algoritma ini tergantung pada kunci. Jika kunci tersebut diketahui oleh orang lain maka orang tersebut akan dapat melakukan enkripsi dan dekripsi terhadap pesan (Mollin, 2007).

Algoritma yang memakai kunci simetris diantaranya adalah :

1. *Data Encryption Standard* (DES)
2. *RC2, RC4, RC5, RC6*
3. *International Data Encrytion Algorithm* (IDEA)
4. *Advanced Encryption Standard* (AES)
5. *One Time Pad* (OTP)

Berikut ini adalah ilustrasi penggunaan Algoritma simetris



Gambar 3. Diagram proses enkripsi dan dekripsi

Sebelum melakukan pengiriman pesan, pengirim dan penerima harus memilih suatu kunci tertentu yang sama untuk dipakai bersama, dan kunci ini haruslah rahasia bagi pihak yang tidak berkepentingan sehingga algoritma ini disebut juga algoritma kunci rahasia *(secret-key algorithm).*

Kelebihan Algoritma Simetris :

- Kecepatan operasi lebih tinggi bila dibandingkan denagn algoritma asimetris.
- Karena kecepatan yang cukup tinggi, maka dapat digunakan untuk sistem real-time.

Kelemahan Algoritma Simetris :

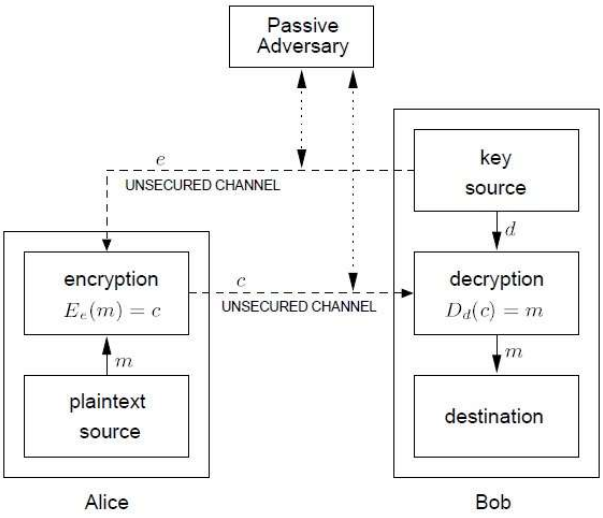
- Untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.
- Permasalahan dalam pengiriman kunci itu sendiri yang disebut " key distribution problem" (Chehal, 2012).

2.4.2 Kriptografi Asimetris

Kriptografi asimetris juga disebut dengan kriptografi *kunci-publik*. Dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi adalah berbeda. Pada kriptografi jenis ini, setiap orang yang berkomunikasi mempunyai sepasang kunci yaitu:

1. Kunci umum (*public key*) : yaitu kunci yang boleh semua orang tahu. Pengirim mengenkripsi pesan dengan menggunakan kunci publik si penerima pesan.
2. Kunci rahasia (*private key*) : yaitu kunci yang dirahasiakan atau diketahui oleh satu orang saja. Hanya penerima pesan yang dapat mendekripsi pesan, karena hanya ia yang mengetahui kunci *private* nya sendiri.

1. Kunci-kunci tersebut berhubungan satu sama lain. Walau kunci publik telah diketahui namun akan sangat sukar mengetahui kunci *private* yang digunakan. Contoh algoritma kriptografi kunci-publik diantaranya *RSA, Elgamal, DSA* dll.



Gambar 4. Diagram proses enkripsi dan dekripsi algoritma asimetris

Kelebihan *Algoritma Asimetris*:

• Masalah keamanan pada distribusi kunci dapat lebih baik

• Masalah manajemen kunci yang lebih baik karena jumlah kunci yang lebih sedikit

Kelemahan *Algoritma Asimetris*:

• Kecepatan yang lebih rendah bila dibandingkan dengan algoritma simetris

• Untuk tingkat keamanan sama, kunci yang digunakan lebih panjang dibandingkan dengan algoritma simetris (Mulya, 2013).

2.5 Keamanan Sistem Kriptografi

Keamanan suatu sistem kriptografi merupakan masalah yang paling fundamental. Dengan menggunakan sistem standar terbuka, maka keamanan suatu sistem kriptografi akan lebih mudah dan lebih cepat dianalisa. Mengingat kenyataan inilah maka sekarang tidak digunakan lagi algoritma rahasia yang tidak diketahui tingkat keamanannya.

Sebuah sistem kriptografi dirancang untuk menjaga plaintext dari kemungkinan dibaca oleh pihak-pihak yang tidak berwenang, yang secara umum dinamakan sebagai penyerang (*attacker*). Penyerang diasumsikan memiliki akses tak terbatas terhadap jalur taka man yang digunakan untuk transaksi ciphertext. Oleh karena itu penyerang dianggap memiliki akses langsung terhadap ciphertext.

Tipe penyerangan paling umum terhadap suatu sistem kriptografi adalah serangan kriptanalisis (*cryptanalysis attack*). Kriptanalisis merupakan ilmu yang mempelajari tentang upaya-upaya untuk memperoleh plaintext dari ciphertext tanpa informasi tentang kunci yang digunakan.

2.6 Algoritma Message Digest 5

MD5 adalah salah satu dari banyak algoritma kriptografi (merupakan salah satu fungsi Hash). *Message Digest* yang didesain oleh Profesor Ronald Rivest dari MIT (Rivest, 1994). MD5 sering digunakan untuk menyimpan password serta digunakan juga dalam digital signature dan certificate. Dalam algoritma MD5 spesifikasi lengkap ada pada RFC (*request for comment*). Dalam hal kerja algoritma MD5 memiliki besar blok sebanyak 512 bit sedangkan ukuran digest adalah 128 bit. Karena word size ditentukan sebesar 32 bit, satu blok terdiri dari 16 word sedangkan digest terdiri dari 4 word. Indeks untuk bit dimulai dari 0. Proses awal dimulai dengan padding sebagai berikut :

1. Bit dengan nilai 1 ditambahkan setelah akhir naskah
2. Deretan bit dengan nilai 0 ditambahkan setelah itu sehingga besar dari naskah mencapai nilai 448 / mod 512 (sedikitnya 0 dan sebanyaknya 511 bit dengan nilai 0 ditambahkan sehingga tersisa 64 bit untuk diisi agar besar naskah menjadi kelipatan 512).
3. 64 bit yang tersisa diisi dengan besar naskah asli dalam bit. Jika besar naskah asli lebih dari 2^{64} bit maka hanya 64 lower order bit yang dimasukkan. Lower order word untuk besar naskah asli dimasukkan sebelum high order word.

Setelah padding, naskah terdiri dari n-word M [0 . . . n -1], dimana n adalah kelipatan 16. Langkah berikutnya dalam preprocessing adalah menyiapkan MD buffer sebesar 4 word, yaitu:

(A, B, C, D)

Dimana A merupakan lower order word. Buffer diberi nilai awal sebagai berikut (nilai dalam hexadecimal dimulai dengan lower order byte).

A : 01 23 45 67

B : 89 ab cd ef

C : fe dc ba 98

D : 76 54 32 10

Proses hashing dilakukan perblok, dimana setiap blok melalui 4 putaran. Proses hashing menggunakan 4 fungsi yaitu F, G, H, dan I yang masing – masing mempunyai input 3 word dan output 1 word :

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

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

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

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

Dimana $\wedge$ adalah bitwise and, $\vee$ adalah bitwise or, $\oplus$ adalah bitwise exclusive or dan $\neg$ adalah bitwise not.

2.7 Algoritma Affine Cipher

Algoritma *affine cipher* merupakan salah satu jenis algoritma kriptografi simetris yang bersifat monoalpabetic subsitutis. Algoritma ini merupakan perluasan dari algoritma *caesar cipher,* yang mengalikan plainteks dengan sebuah nilai dan menambahkannya dengan sebuah pergeseran. Proses enkripsi plaintext menjadi ciphertext dinyatakan dengan fungsi kongruen $C \equiv aP + b \pmod n$ dimana dalam hal ini :

n = ukuran alfabet

m = bilangan bulat yang relatif prima dengan n

b = jumlah pergeseran

P = Plaintext

C = Ciphertext

Berdasarkan fungsi tersebut plainteks dapat dihitung sebagai berikut :

Plainteks : U I N S U

n = 26, a = 5, b = 2

U = 5 * 20 + 2 = 102 mod 26 = 24 → Y

I = 5 * 8 + 2 = 42 mod 26 = 16 → Q

N = 5 * 13 + 2 = 67 mod 26 = 15 → P

S = 5 * 18 + 2 = 92 mod 26 = 14 → O

U = 5 * 20 + 2 = 102 mod 26 = 24 → Y

Cipherteks : YQPOY

Sehingga dari enkripsi diatas didapat cipherteks “YQPOY” yang merupakan pesan sebelumnya berupa plainteks “UINSU” yang dalam hal ini n merupakan ukuran alphabet, a adalah bilangan yang harus relatif prima dengan n, kemudian b adalah jumlah pergeseran. Untuk mendapatkan P (plainteks), solusi kekongruenan tersebut hanya dapat di invers a (mod n) sehingga proses deskripsi pada algoritma affine cipher dapat dilakukan sebagai berikut :

$P \equiv a^{-1} (C-b) \bmod n.$

Berdasarkan persamaan tersebut, maka cipherteks tadi dapat dikembalikan sebagai berikut :

Cipherteks : YQPOY

$n = 26, a = 5, b = 2, a^{-1} = 21$

Y = $21 (24-2) \pmod{26} = 20 \rightarrow$ U

Q = $21 (16-2) \pmod{26} = 8 \rightarrow$ I

P = $21 (15-2) \pmod{26} = 13 \rightarrow$ N

O = $21 (14-2) \pmod{26} = 18 \rightarrow$ S

Y = $21 (24-2) \pmod{26} = 20 \rightarrow$ U

Dekripsi tersebut didapatlah plainteks “UINSU” dari cipherteks “YQPOY” yang merupakan kebalikan dari plainteks.

BAB III

METODOLOGI PENELITIAN

Dalam penelitian ini penulis membandingkan proses waktu enkripsi menggunakan algoritma messege digest 5 dan juga algoritma affine cipher. Nantinya dari pesan yang memiliki panjang berbeda akan diketahui tingkat proses enkripsinya.

3.1 Bahan-bahan

Penelitian ini dirancang dengan menggunakan beberapa perangkat lunak dan perangkat keras dalam proses pembuatannya. Adapun perangkat yang digunakan sebagai berikut :

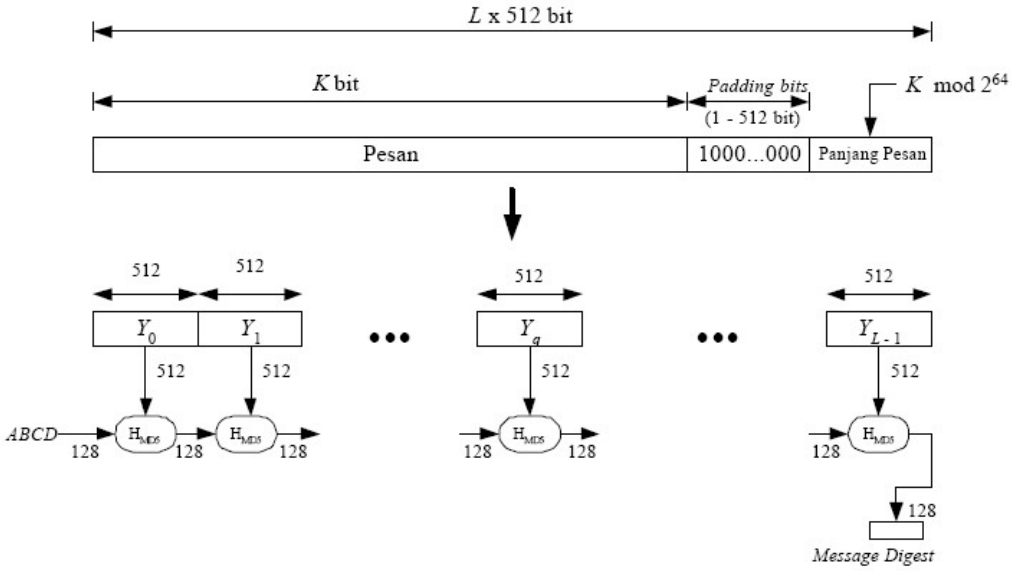
1. Sistem Operasi : Windows 10 Home Edition
2. Bahasa Pemrograman : Python 2.7 dan Jupiter Lab dengan python 3.8
3. Prosesor : Intel Core i5 Generasi 10
4. Memori/RAM : 8 GB DDR 4
5. Storage : SSD NVME

Untuk melakukan analisis waktu enkripsiterlebih dahulu pesan yang akan dienkripsi akan disiapkan dan dilakukan pengenkripsian satu per satu menggunakan algoritma MD5 lalu dilanjutkan dengan menggunakan algoritma affine cipher.

3.2 Analisis Algoritma MD5

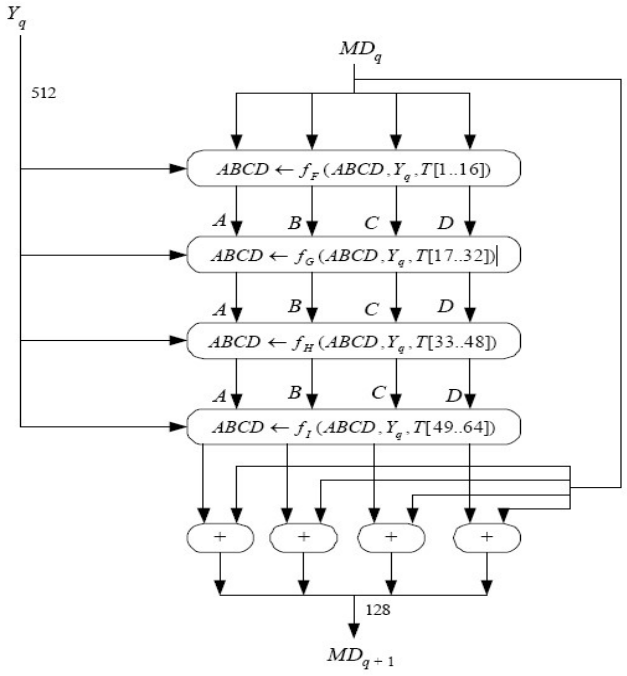
Algoritma MD5 adalah fungsi *hash* satu-arah yang dibuat oleh Ron Rivest. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya mencapai 128 bit. Langkah pembuatan message digest secara umum adalah :

1. Penambahan bit-bit pengganjal (padding bits).
2. Penambahan nilai panjang pesan semula.
3. Inisialisasi penyangga (buffer) message digest.
4. Pengolahan pesan dalam blok berukuran 512 bit.



Gambar 5. Ilustrasi pembuatan algoritma MD5

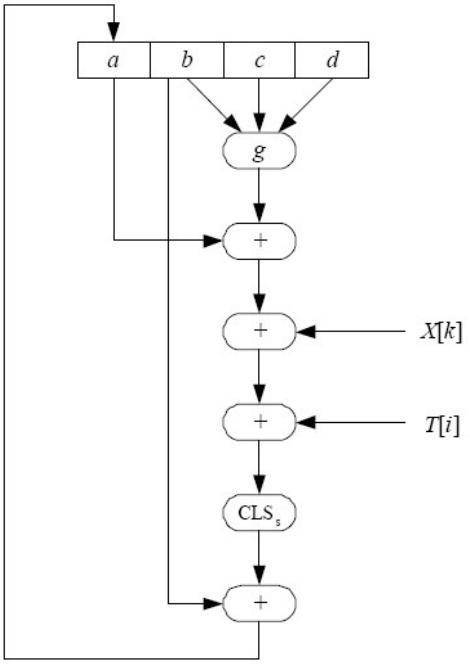
Pada pengolahan pesan dalam blok berukuran 512 bit pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y_0 sampai Y_{L-1}). Setiap blok 512 bit diproses dengan penyangga MD yang menghasilkan keluaran 128 bit, dan ini disebut H_{MD5}.



Gambar 6. Pengolahan blok 512 bit

Dalam proses H_{MD5} terdiri dari 4 buah putaran dan masing-masing putaran melakukan operasi dasar MD5 sebanyak 16 kali. Disetiap operasi dasar memakai sebuah elemen T, sehingga setiap putaran melakukan 16 elemen tabel T. Y_q menyatakan blok 512 bit ke-q dari pesan yang ditambahkan dengan bit-bit pengganjal pada proses pertama dan tambahan 64 bit nilai panjang pesan semula pada proses kedua. MD_q adalah nilai message digest 128 bit dari proses H_{MD5} ke-q. Pada proses awal MDq berisi inisialisasi penyangga MD kemudian fungsi f_F, f_G, f_H, f_Is eperti pada

gambar, masing-masing berisi 16 kali operasi dasar terhadap input, setiap operasi dasar menggunakan elemen tabel T.



Gambar 7. Proses operasi dasar MD5

Dalam operasi dasar MD5 dapat dituliskan dengan persamaan dibawah ini :

$$a \leftarrow b + CLS_s(a + g(b,c,d) + X[k] + T[i]$$

Dimana :

a,b,c,d = empat buah peubah penyangga 32 bit (Nilai penyangga A,B,C,D)

g = salah satu fungsi F,G,H,I

CLS_s = circular left shift sebanyak s bit

X[k] = kelompok 32 bit ke-k dari blok 512 bit message ke-q. nilai k = 0 sd 15

T[i] = elemen table T ke-i (32 bit)

+ = operasi penjumlahan modulo 2^{32}

Fungsi fF, fG, fH, dan fI adalah fungsi untuk memanipulasi masukan a, b, c, dan d dengan ukuran 32-bit. Masing-masing fungsi dapat dilihat pada tabel berikut :

Tabel 1. Fungsi-fungsi dasar MD5

Nama	Notasi	g(b,c,d)

f_F	F(b,c,d)	*(b ∧ c) ∨ (~b ∧ d)*
f_G	G(b,c,d)	*(b ∧ d)∨ (c ∧ ~d)*
f_H	H(b,c,d)	*b ⊕ c ⊕ d*
f_I	I(b,c,d)	*c⊕ (b ∧ ~d)*

Dalam fungsi-fungsi MD5 secara logika merupakan lambang operator logika dimana :

AND = ∧

OR = ∨

NOT = ~

XOR = ⊕

Kemudian nilai T[i] dapat dilihat pada tabel berikut. Tabel ini disusun oleh fungsi 2^{32} x abs(sin(i)), i dalam radian.

Tabel 2. Nilai T[i]

T[1] = D76AA478	T[17] = F61E2562	T[33] = FFFA3942	T[49] = F4292244
T[2] = E8C7B756	T[18] = C040B340	T[34] = 8771F681	T[50] = 432AFF97
T[3] = 242070DB	T[19] = 265E5A51	T[35] = 69D96122	T[51] = AB9423A7
T[4] = C1BDCEEE	T[20] = E9B6C7AA	T[36] = FDE5380C	T[52] = FC93A039
T[5] = F57C0FAF	T[21] = D62F105D	T[37] = A4BEEA44	T[53] = 655B59C3
T[6] = 4787C62A	T[22] = 02441453	T[38] = 4BDECFA9	T[54] = 8F0CCC92
T[7] = A8304613	T[23] = D8A1E681	T[39] = F6BB4B60	T[55] = FFEFF47D
T[8] = FD469501	T[24] = E7D3FBCB	T[40] = BEBFBC70	T[56] = 85845DD1
T[9] = 698098D8	T[25] = 21E1CDE6	T[41] = 289B7EC6	T[57] = 6FA87E4F
T[10] = 8B44F7AF	T[26] = C33707D6	T[42] = EAA127FA	T[58] = FE2CE6E0
T[11] = FFFF5BB1	T[27] = F4D50D87	T[43] = D4EF3085	T[59] = A3014314
T[12] = 895CD7BE	T[28] = 455A14ED	T[44] = 04881D05	T[60] = 4E0811A1
T[13] = 6B901122	T[29] = A9E3E905	T[45] = D9D4D039	T[61] = F7537E82
T[14] = FD987193	T[30] = FCEFA3F8	T[46] = E6DB99E5	T[62] = BD3AF235
T[15] = A679438E	T[31] = 676F02D9	T[47] = 1FA27CF8	T[63] = 2AD7D2BB
T[16] = 49B40821	T[32] = 8D2A4C8A	T[48] = C4AC5665	T[64] = EB86D391

Sebagaimana telah dijelaskian sebelumnya bahwa fungsi fF, fG,fH dan fI melakukan 16 kali operasi dasar. Misalnya notasi berikut ini :

[abcd k s i]

Menyatakan operasi

$$a \; b + ((a + g(b,c,d) + X[k] + T[i]) <<<s)$$

untuk operasi dasar tersebut, <<<s melambangkan operasi *circular left shift* 32 bit, maka operasi dasar pada masing-masing putaran dapat ditabulasikan sebagai berikut :

- Putaran 1 : 16 kali operasi dasar dengan g(b,c,d) – F(b,c,d) dapat dilihat pada tabel berikut

Tabel 3. Rincian operasi pada fungsi F(b,c,d)

No.	[*abcd*	*k*	*s*	*i*]
1	[ABCD	0	7	1]
2	[DABC	1	12	2]
3	[CDAB	2	17	3]
4	[BCDA	3	22	4]
5	[ABCD	4	7	5]
6	[DABC	5	12	6]
7	[CDAB	6	17	7]
8	[BCDA	7	22	8]
9	[ABCD	8	7	9]
10	[DABC	9	12	10]
11	[CDAB	10	17	11]
12	[BCDA	11	22	12]
13	[ABCD	12	7	13]
14	[DABC	13	12	14]
15	[CDAB	14	17	15]
16	[BCDA	15	22	16]

- Putaran 2 : 16 kali operasi dasar dengan g(b,c,d) – G(b,c,d) dapat dilihat pada tabel berikut.

Tabel 4. Rincian operasi pada fungsi G(b,c,d)

No.	[*abcd*	*k*	*s*	*i*]
1	[ABCD	1	5	17]
2	[DABC	6	9	18]
3	[CDAB	11	14	19]
4	[BCDA	0	20	20]
5	[ABCD	5	5	21]
6	[DABC	10	9	22]
7	[CDAB	15	14	23]
8	[BCDA	4	20	24]
9	[ABCD	9	5	25]
10	[DABC	14	9	26]
11	[CDAB	3	14	27]
12	[BCDA	8	20	28]
13	[ABCD	13	5	29]
14	[DABC	2	9	30]
15	[CDAB	7	14	31]
16	[BCDA	12	20	32]

- Putaran 3 : 16 kali operasi dasar dengan g(b,c,d) – H(b,c,d) dapat dilihat pada tabel berikut.

Tabel 5. Rincian Operasi pada fungsi H(b,c,d)

No.	[*abcd*	*k*	*s*	*i*]
1	[ABCD	5	4	33]
2	[DABC	8	11	34]
3	[CDAB	11	16	35]
4	[BCDA	14	23	36]
5	[ABCD	1	4	37]
6	[DABC	4	11	38]
7	[CDAB	7	16	39]
8	[BCDA	10	23	40]
9	[ABCD	13	4	41]
10	[DABC	0	11	42]
11	[CDAB	3	16	43]
12	[BCDA	6	23	44]
13	[ABCD	9	4	45]
14	[DABC	12	11	46]
15	[CDAB	15	16	47]
16	[BCDA	2	23	48]

- Putaran 4 : 16 kali operasi dasar dengan g(b,c,d) – I(b,c,d) dapat dilihat pada tabel berikut.

Tabel 6. Rincian Operasi pada fungsi I(b,c,d)

No.	[*abcd*	*k*	*s*	*i*]
1	[ABCD	0	6	49]
2	[DABC	7	10	50]
3	[CDAB	14	15	51]
4	[BCDA	5	21	52]
5	[ABCD	12	6	53]
6	[DABC	3	10	54]
7	[CDAB	10	15	55]
8	[BCDA	1	21	56]
9	[ABCD	8	6	57]
10	[DABC	15	10	58]
11	[CDAB	6	15	59]
12	[BCDA	13	21	60]
13	[ABCD	4	6	61]
14	[DABC	11	10	62]
15	[CDAB	2	15	63]
16	[BCDA	9	21	64]

Setelah putaran keempat a,b,c dan d ditambahkan ke A,B,C dan D yang selanjutnya algoritma akan memproses untuk blok data berikutnya (Y_{q+1}). Output akhir dari algoritma MD5 adalah hasil penyambungan bit-bit A,B,C dan D. Secara umum dari uraian tersebut fungsi hash MD5 dapat ditulis dalam persamaan matematis berikut :

$$MD0 = IV$$
$$MD_{q+1} = MD_q + fI(Y_q + fH(Y_q + fG(Y_q + fF(Y_q + MD_q))))$$
$$MD = MD_{L-1}$$

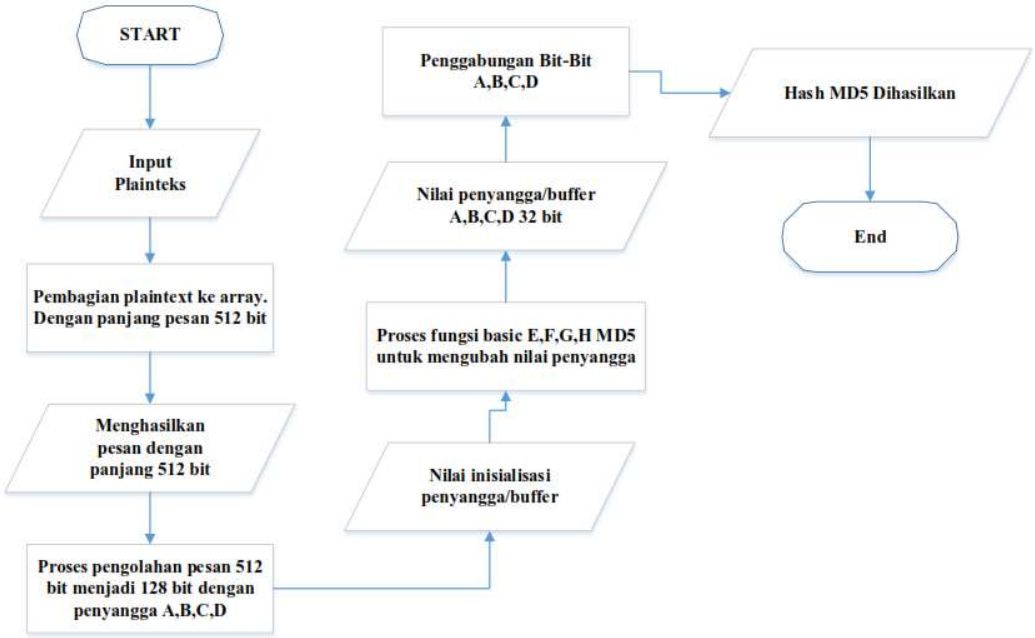
Dimana :

IV = Initial vector dari penyangga ABCD yang dilakukan pada proses inisialisasi penyangga.

Y_q = Blok pesan berukuran 512 bit ke-q

L = Jumlah blok pesan

MD = Nilai akhir message digest



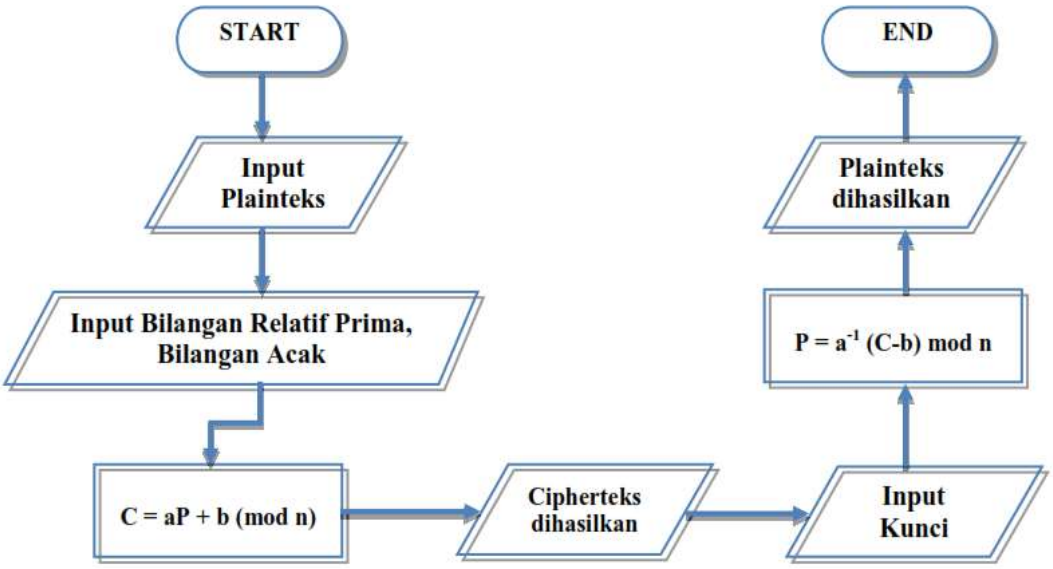
Gambar 8. Proses hash algoritma MD5

3.3 Analisis Algoritma Affine Cipher

Algoritma affine cipher merupakah salah satu algoritma kriptografi simetris yang telah lama digunakan. Kelebihan dari algoritma kriptografi simetris adalah kecepatan dalam proses enkripsi data (Chehal, 2012). Selain itu, kelebihan dari algoritma affine cipher adalah kemudahan dalam proses enkrips, karena hanya melibatkan proses perhitungan matematika sederhana dalam proses enkripsi maupun dekripsi (Mokhtari & Naraghi, 2012). Tahapan enkripsi dalam algoritma affine cipher yaitu :

1. Tentukan panjang urutan alphabet yang digunakan.
2. Konversi plainteks menjadi bilangan bulat dari 1 sesuai dengan urutan dalam alphabet.
3. Pilih sebuah bilangan bulat yang relatif prima dengan nilai terbesar dari urutan alphabet.
4. Pilih sebuah bilangan bulat acak sebagai jumlah dari pergeseran cipherteks.
5. Lakukan proses perhitungan untuk mendapatkan cipherteks dengan rumus $C \equiv aP + b \pmod{n}$.
6. Lalu dihasilkan cipherteks.

Sebagai contoh plainteks yang digunakan adalah PENELITIAN, selanjutnya akan dipilih dua bilangan yaitu a = 15 dan b = 13. Panjang urutan alphabet terbesar yang digunakan adalah 26.



Gambar 9. Proses Enkripsi dan Dekripsi pada algoritma Affine cipher

Tabel 7. Proses Enkripsi pesan "penelitian" pada affine cipher

Plainteks		Nilai a (a relatif prima dengan n)	b (jumlah pergeseran cipherteks)	n (panjang alphabet yang digunakan)	Cipherteks	
P	15	17	5	26	A	0
E	4	17	5	26	V	21
N	13	17	5	26	S	18
E	4	17	5	26	V	21
L	11	17	5	26	K	10
I	8	17	5	26	L	11
T	19	17	5	26	Q	16
I	8	17	5	26	L	11
A	0	17	5	26	F	5
N	8	17	5	26	S	18
Cipherteks yang dihasilkan adalah AVSVKLQLFS						

Pada proses dekripsi cipherteks, rumus yang digunakan adalah $P \equiv a^{-1}$ (C-b) mod n dimana a^{-1} adalah invers perkalian a modulus n yang dapat memenuhi persamaan sebagai berikut : $1 = a^{-1} \bmod n$

Tabel 8. Proses dekripsi pada algoritma Affine cipher

Cipherteks		Nilai a^{-1} (Nilai invers perkalian a mod m =1)	b (jumlah pergeseran cipherteks)	n (panjang alphabet yang digunakan)	Cipherteks	
A	0	17	5	26	P	15
V	21	17	5	26	E	4
S	18	17	5	26	N	13
V	21	17	5	26	E	4
K	10	17	5	26	L	11
L	11	17	5	26	I	8
Q	16	17	5	26	T	19
L	11	17	5	26	I	8
F	5	17	5	26	A	0
S	18	17	5	26	N	8
Cipherteks yang dihasilkan adalah PENELITIAN						

3.4 Rancangan Penelitian

Pada penelitian ini akan dilakukan proses enkripsi plainteks yang memiliki panjang tertentu menggunakan algoritma MD5 dan algoritma affine cipher sehingga menghasilkan cipherteks dari masing-masing algoritma. Penelitan ini akan menganalisa waktu yang dibutuhkan oleh masing-masing algoritma dalam melakukan enkripsi plainteks hingga menjadi sebuah cipherteks.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil dan Implementasi

Pada tahapan ini penuis akan melakukan penelitian terhadap proses enkripsi menggunakan algoritma MD5 menggunakan beberapa pesan yang akan di enkripsi. Serta pesan-pesan tersebut juga akan dienkripsi dengan menggunakan algoritma affine cipher.

4.2 Enkripsi menggunakan algoritma MD5

Pada bagian ini penulis akan melakukan percobaan terhadap enkripsi menggunakan algoritma MD5. Percobaan pada algoritma ini dilakukan dengan melakukan enkripsi sebanyak 3 pesan untuk mendapatkan cipherteks atau hash dalam algoritma MD5.

4.2.1 Percobaan pertama enkripsi menggunakan algoritma MD5

Untuk percobaan yang pertama ini pada algoritma MD5 plainteks yang digunakan adalah “medan”. Plainteks tersebut akan diubah menjadi bilangan hexadesimal yang terlebih dahulu dibagi menjadi bentuk blok dimana setiap blok terdiri dari 4 karakter. Untuk blok yang kurang dari 4 karakter akan ditambahkan padding byte. Sehingga byte yang terbentuk pada blok pertama yaitu “meda” menjadi 6164656d dan blok kedua setelah ditambahkan padding byte “n” menjadi 0000806e. Setelah itu pesan dimasukkan dan diinisalisasi dalam 4 penyangga MD dan kemudian diolah melalui 4 buah putaran yang masing-masing memiliki 16 kali operasi dan setiap operasi memakai elemen T sehingga hasil yang didapat :

Tabel 9. Hasil putaran 1 dan putaran 2 Message Digest plainteks “medan”

Putaran 1		Putaran 2	
Operasi	Hash	Operasi	Hash
1	57529da5	1	b190c37b
2	8cb3fa33	2	f4ed5cae
3	5bb842a6	3	24c361f7
4	c267e23c	4	dc86691f
5	81eba009	5	654e100a
6	7c1c5af3	6	544027f0
7	cda9679b	7	b2a2449b
8	41ac65e9	8	80c18a63
9	d408a41d	9	c30595c8
10	7fab792d	10	348342fb
11	4fddbc96	11	4f06bdba
12	bd086133	12	bfba97e1
13	5eff0ffa	13	4dc26dbe
14	28be799e	14	a7f04a1c
15	3ccda1c8	15	329cb1b5
16	806a006e	16	b49bab05

Tabel 10. Hasil putaran 3 dan putaran 4 Message Digest plainteks "medan"

Putaran 3		Putaran 4	
Operasi	Hash	Operasi	Hash
1	afdb25cb	1	9368c8c1
2	a3dbe894	2	94ac85a8
3	1d0d4169	3	cc0c0237
4	d4ef08a0	4	c1235e32
5	c232264b	5	d20b0f3a
6	1b702627	6	19db1894
7	b105479c	7	70a8f57c
8	318342a2	8	abfe4f81
9	9dbfc54a	9	ee6857b9
10	195fd96f	10	12a7112b
11	f00814c6	11	22527768
12	355d8e7a	12	0f35e651
13	6f4b23df	13	ca11e47d
14	40b0e131	14	724a1efe
15	1f030b81	15	951d6201
16	46188cd8	16	8115b044

Kemudian dilakukan operasi penambahan sehingga didapat :

a = (a + AA) & 0x0ffffffff ;

b = (b + BB) & 0x0ffffffff ;

c = (c + CC) & 0x0ffffffff ;

d = (d + DD) & 0x0ffffffff ;

Tabel 11. Operasi hasil penambahan dan konversi

Nilai	32 bit	8 char hex	Hex LSB
a	827787134	3157077e	7e075731
b	1893948365	70e35bcd	cd5be37000000000
c	769146623	2dd83eff	ff3ed82d00000000
d	-2105773196	827c7374	74737c8200000000

Hasil akhir : lsb_hex(a) + lsb_hex(b) + lsb_hex(c) + lsb_hex(d) = 7e075731cd5be370ff3ed82d74737c82

4.2.2 Percobaan kedua enkripsi menggunakan algoritma MD5

Untuk percobaan yang kedua menggunakan algoritma MD5, plainteks yang digunakan adalah kata "Universitas". Plainteks ini akan diubah menjadi bilangan hexadesimal yang terlebih dahulu dibagi menjadi bentuk blok dimana setiap blok terdiri dari 4 karakter. Untuk blok yang kurang dari 4 karakter maka akan ditambahkan padding byte. Sehingga byte yang terbentuk pada blok pertama yaitu "Univ" menjadi 76696e55 kemudian "ersi" menjadi 69737265 dan blok berikutnya setelah ditambahkan padding byte "tas" menjadi 80736174. Setelah itu pesan dimasukkan dan diinisialisasi dalam 4 penyangga MD dan kemudian diolah

melalui 4 buah putaran yang masing-masing memiliki 16 kali operasi dan setiap operasi memakai elemen T sehingga hasil yang didapat :

Tabel 12. Operasi putaran 1 dan 2 Message Digest dari plainteks "Universitas"

Putaran 1		Putaran 2	
Operasi	Hash	Operasi	Hash
1	d9d711af	1	53be3553
2	8e97b474	2	9f8694f0
3	248c02cf	3	3cf476cb
4	1b6fdb71	4	18a27d73
5	1313ad1a	5	749a02df
6	037addf4	6	2e04808b
7	b362ad9b	7	c5fb60f6
8	33e9b81a	8	67105245
9	3783b072	9	35f67994
10	50b7949c	10	91ed15c6
11	d37562f7	11	3c0188f3
12	ee81e27f	12	7c47902a
13	941e2839	13	0046e9ad
14	aab10994	14	2daea04c
15	cd618593	15	0b051429
16	e395a2d8	16	dcd88606

Tabel 13. Operasi putaran 3 dan 4 Message Digest dari plainteks "Universitas"

Putaran 3		Putaran 4	
Operasi	Hash	Operasi	Hash
1	881ddb35	1	6bc592bc
2	8f1d13dc	2	282a8a0c
3	53576856	3	32bcb9a2
4	e7eef305	4	f294df15
5	073af9e2	5	9521943a
6	016cae55	6	4ea1b7d6
7	59d4da20	7	b7933e91
8	dfd7f2be	8	a1659fff
9	5437e9f9	9	8f229dc1
10	b468439e	10	f6a5b8ac
11	17e6b1ea	11	ea2f1ec4
12	c054be7f	12	e88e97d2
13	dec52258	13	8287ee74
14	b71fa77d	14	6b162740
15	215c8894	15	d51c70f5
16	260385d6	16	1969c0ed

Kemudian dilakukan operasi penambahan sehingga didapat :

a = (a + AA) & 0x0ffffffff ;

b = (b + BB) & 0x0ffffffff ;

c = (c + CC) & 0x0ffffffff ;

d = (d + DD) & 0x0ffffffff ;

Tabel 14. Operasi hasil penambahan dan konversi

Nilai	32 bit	8 char hex	Hex LSB
a	-372436619	e9cd1175	7511cde9
b	154627190	09376c76	766c370900000000
c	1842826739	6dd74df3	f34dd76d00000000
d	2068347830	7b487bb6	b67b487b00000000

Hasil akhir : lsb_hex(a) + lsb_hex(b) + lsb_hex(c) + lsb_hx(d) =
7511cde9766c3709f34dd76db67b487b

4.2.3 Percobaan ketiga enkripsi menggunakan algoritma MD5

Untuk percobaan yang ketiga menggunakan algoritma MD5 plainteks yang digunakan adalah pesan “Informatika”. Plainteks ini kemudian akan diubah menjadi bilangan hexadesimal yang terlebih dahulu dibagi menjadi bentuk blok, dimana setiap blok terdiri dari 4 karakter. Untuk blok yang kurang dari 4 karakter maka akan ditambahkan padding byte. Sehingga byte yang terbentuk pada blok pertama yaitu “Info” menjadi 6f666e49 kemudian “rmat” menjadi 74616d72 dan blok berikutnya setelah ditambahkan padding byte “ika” menjadi 80616b69. Setelah itu pesan dimasukkan dan diinisialisasi dalam 4 penyangga MD dan kemudian diolah melalui 4 putaran yang masing-masing memiliki 16 kali operasi dan setiap operasi memakai element T sehingga hasil yang didapat :

Tabel 15. Operasi putaran 1 dan 2 Message Digest dari plainteks “Informatika”

Putaran 1		Putaran 2	
Operasi	Hash	Operasi	Hash
1	58570bac	1	09067937
2	ede88f10	2	cf7ce0b7
3	b77f9e13	3	5aa8ad35
4	44675b15	4	313720db
5	25c41192	5	4e6e05d9
6	2508ce50	6	4e63d052
7	0f80587d	7	b4a5f0bf
8	899a150b	8	8887c3cd
9	ec5c76e9	9	a37e5952
10	4a7e3aa7	10	25d5fec5
11	222ded5e	11	5c1d7223
12	fe664a27	12	503a9a1f
13	12e692e4	13	ff9629f3
14	749e1a8c	14	bb02f5f1
15	7c74e1a7	15	84be56b5
16	bfa48dc0	16	3a647670

Tabel 16. Operasi putaran 3 dan 4 Message Digest dari plainteks "Informatika"

Putaran 3		Putaran 4	
Operasi	Hash	Operasi	Hash
1	90f7fd00	1	47a8a045
2	a749b885	2	59c8939f
3	9315b8ba	3	79c42993
4	9d043390	4	7722055d
5	d40cb9e5	5	74f1c2eb
6	01e4a84f	6	31c15095
7	28397b0c	7	5cf77405
8	7b6645b9	8	21035cc4
9	71a3400d	9	39b6a62a
10	b78d9404	10	ea617554
11	f4cf4e75	11	ee98d8f6
12	9228b673	12	4fa75179
13	60507af4	13	dbf13ff1
14	65d7581d	14	8f4964e0
15	c5df043e	15	0362641d
16	522acb8d	16	c49eced1

Kemudian dilakukan operasi penambahan sehingga didapat :

a = (a + AA) & 0x0ffffffff ;

b = (b + BB) & 0x0ffffffff ;

c = (c + CC) & 0x0ffffffff ;

d = (d + DD) & 0x0ffffffff ;

Tabel 17. Operasi hasil penambahan dan konversi

Nilai	32 bit	8 char hex	Hex LSB
a	1127637746	433662f2	f2623643
b	-1267959206	b46c7a5a	5a7a6cb400000000
c	-1675804389	9c1d411b	1b411d9c00000000
d	-1619281578	9f7bb956	56b97b9f00000000

Hasil akhir : lsb_hex(a) + lsb_hex(b) + lsb_hex(c) + lsb_hex(d) = f2623643a7a6cb41b411d9c56b97b9f

4.3 Proses enkripsi menggunakan Algoritma Affine Cipher

Pada bagian ini penulis akan melakukan enkripsi beberapa pesan yang sama seperti enkripsi menggunakan algoritma MD5. Pesan-pesan tersebut akan dienkripsi dengan menggunakan algoritma Affine cipher. Hasil cipherteks dari enkripsi pada setiap percobaan akan menjadi bahan proses analisa untuk tahap selanjutnya.

4.3.1 Percobaan pertama enkripsi menggunakan algoritma Affine cipher plainteks "medan"

Pada percobaan pertama, plainteks yang digunakan adalah "medan". Plainteks ini akan diinisiaslisasi berdasarkan urutan berdasarkan alphabet. Kemudian untuk nilai a = 5 yang merupakan bilangan prima dan nilai b = 7. Hasilnya dapat dilihat pada tabel berikut :

Tabel 18. Enkripsi pesan menggunakan algoritma Affine cipher

Plainteks		Nilai a (a relatif prima dengan n)	b (jumlah pergeseran cipherteks)	n (panjang alphabet yang digunakan)	Cipherteks	
M	12	5	7	26	P	15
E	4	5	7	26	B	1
D	3	5	7	26	W	22
A	0	5	7	26	H	7
N	13	5	7	26	U	20
Cipherteks yang dihasilkan adalah PBWHU						

4.3.2 Percobaan kedua enkripsi menggunakan algoritma Affine cipher plainteks "Universitas"

Pada percobaan pertama, plainteks yang digunakan adalah "Universitas". Plainteks ini akan diinisiaslisasi berdasarkan urutan berdasarkan alphabet. Kemudian untuk nilai a = 11 yang merupakan bilangan prima dan nilai b = 13. Hasilnya dapat dilihat pada tabel berikut :

Tabel 19. Enkripsi pesan menggunakan algoritma Affine cipher

Plainteks		Nilai a (a relatif prima dengan n)	b (jumlah pergeseran cipherteks)	n (panjang alphabet yang digunakan)	Cipherteks	
U	20	11	13	26	Z	25
N	13	11	13	26	A	0
I	8	11	13	26	X	23
V	21	11	13	26	K	10
E	4	11	13	26	F	5
R	17	11	13	26	S	18
S	18	11	13	26	D	3
I	8	11	13	26	X	23
T	19	11	13	26	O	14
A	0	11	13	26	N	13
S	18	11	13	26	D	3
Cipherteks yang dihasilkan adalah ZAXKFSDXOND						

4.3.3 Percobaan ketiga enkripsi menggunakan algoritma Affine cipher plainteks "Informatika"

Pada percobaan pertama, plainteks yang digunakan adalah "Informatika". Plainteks ini akan diinisiaslisasi berdasarkan urutan berdasarkan alphabet. Kemudian untuk nilai a = 17 yang merupakan bilangan prima dan nilai b = 7. Hasilnya dapat dilihat pada tabel berikut :

Tabel 20. Enkripsi pesan menggunakan algoritma Affine cipher

Plainteks		Nilai a (a relatif prima dengan n)	b (jumlah pergeseran cipherteks)	n (panjang alphabet yang digunakan)	Cipherteks	
I	8	17	7	26	N	13
N	13	17	7	26	U	20
F	5	17	7	26	O	14
O	14	17	7	26	L	11
R	17	17	7	26	K	10
M	12	17	7	26	D	3
A	0	17	7	26	H	7
T	19	17	7	26	S	18
I	8	17	7	26	N	13
K	10	17	7	26	V	21
A	0	17	7	26	H	7
Cipherteks yang dihasilkan adalah NUOLKDHSNVH						

4.4 Analisa Waktu

Pada penelitian ini akan dilakukan proses perbandingan waktu yang diperlukan pada proses enkripsi yang dihasilkan oleh algoritma MD5 dan juga algoritma affine cipher. Dari enkripsi tersebut akan dibandingan waktu yang dihasilkan oleh kedua algoritma sehingga dapat dianalisis.

4.4.1 Analisa waktu enkripsi dengan algoritma MD5

Pada tahap ini akan dilakukan proses analisa waktu terhadap algoritma MD5 dalam menghasilkan hash hasil enkripsi. Hasil dari analisa waktu ini akan menampilkan grafik dari waktu yang diperlukakan oleh algoritma MD5. Analisa waktu ini dilakukan sebanyak 10 kali percobaan dengan panjang pesan yang berbeda.

Tabel 21. Analisa waktu enkripsi pada algoritam MD5

Percobaan ke-	Plainteks	Hash MD5	Waktu (ms)
1	Medan	7e075731cd5be370ff3ed82d74737c82	
2	Universitas	7511cde9766c3709f34dd76db67b487b	
3	Informatika	f26236435a7a6cb41b411d9c56b97b9f	
4	Islam	2ee6986e22831d3d91b0971635ac4f7c	

5	Indonesia	4647d00cf81f8fb0ab80f753320d0fc9	
6	Kemerdekaan	6e3f468b377bcd0e5d6234848c35cee3	
7	Kampus	1104af4715935c05dddca1a1ba284350	
8	Sumatera utara	234fa575b347313f5780a3867e40d285	
9	Edukatif	08de02c74854332b7855b8bb0a03e6b1	
10	Ilmu komputer	8c05ef480a50f8626abfd044199f41d8	

Berdasarkan tabel diatas dapat dilihat hasil enkripsi beberapa pesan menggunakan algoritma MD5 memiliki waktu yang berbeda pada setiap prosesnya. Dan bentuk cipher atau hash yang dihasilkan oleh algoritma MD5 sangat berbeda-beda.

4.4.2 Analisa waktu enkripsi dengan algoritma Affine cipher

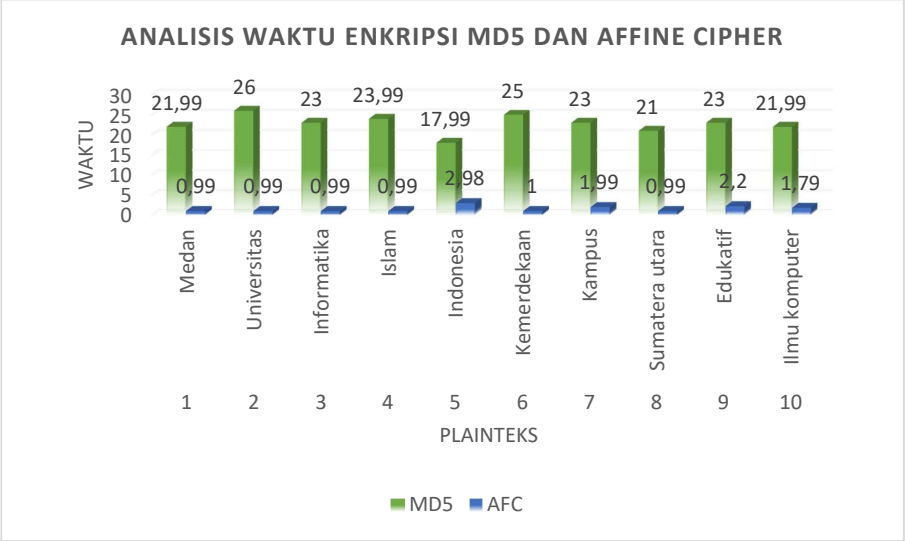
Pada tahapan ini akan dilakukan proses analisa waktu proses enkripsi beberapa pesan menggunakan algoritma affine cipher. Analisa waktu enkripsi dengan algoritma affine cipher ini dilakukan sebanyak 10 kali percobaan dengan panjang karakter yang berbeda. Untuk nilai a = 17 dan b = 13. Hasilnya dapat dilihat pada tabel berikut.

Tabel 22. Analisa waktu enkripsi pada algoritma Affine cipher

Percobaan ke-	Plainteks	Cipherteks	Waktu (ms)
1	Medan	jdmna	0.99
2	Universitas	patgdqhtynh	0.99
3	Informatika	taurqjnytbn	0.99
4	Islam	thsnj	0.99
5	Indonesia	Tamradhtn	2.98
6	Kemerdekaan	bdjdqmdbnna	1.00
7	Kampus	bnjiph	1.99
8	Sumatera utara	hpjnydqnpynqn	0.99
9	Edukatif	dmpbnytu	2.20
10	Ilmu komputer	tsjpbrjipydq	1.79

4.4.3 Analisa perbandingan waktu enkripsi algoritma MD5 dan Affine cipher

Pada bagian didapatlah hasil setelah dilakukan percobaan enkripsi sebanyak 10 pesan menggunakan algoritma MD5 dan algoritma Affine cipher. Kedua algoritma tersebut menghasilkan cipherteks yang berbeda dan juga waktu proses yang berbeda pula. Waktu yang didapat kemudian dibandingkan sehingga diperoleh grafik seperti berikut.



Gambar 10. Analisa waktu enkripsi algoritma MD5 dan Affine cipher

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam penelitian ini penulis memberikan beberapa kesimpulan dari hasil yang diperoleh antara lain :

1. Pada penelitian ini, algoritma MD5 memiliki proses enkripsi yang panjang dilihat dari metode yang digunakan step by step dalam melakukan enkripsi pesan untuk menghasilkan cipherteks, dalam hal ini disebut dengan hash. Kemudian waktu yang dibutuhkan dalam proses enkripsi pada masing-masing plainteks yang memiliki panjang yang berbeda-beda. Panjang hash yang dihasilkan oleh algoritma MD5 yaitu 32 bit dan tidak membutuhkan kunci untuk melakukan enkripsinya yang tentunya berbeda dengan proses enkripsi pada algoritma affine cipher.
2. Pada algoritma Affine cipher hasil enkripsi plainteks dilakukan dengan panjang pesan yang berbeda-beda didapat cipherteks yang memiliki panjang sesuai dengan panjang dari plainteksnya. Kemudian waktu yang dibutuhkan untuk enkripsi lebih cepat atau sangat berbeda dengan algoritma MD5. Pada enkripsi menggunakan algoritma affine cipher pemilihan kunci yaitu bilangan relatif prima untuk nilai a dan juga nilai b sebagai pergeseran sangat penting untuk diperhatikan agar cipherteks yang dihasilkan berjalan dengan baik.

5.2 Saran

Pada penelitian ini penulis memiliki saran untuk keberlanjutan dari penelitian ini antara lain :

1. Untuk menjaga keamanan dari cipher yang dihasilkan oleh kedua algoritma tersebut yaitu algoritma MD5 dan algoritma Affine cipher perlu diperlakukan perluasan hash hasil dari algoritma MD5 yang semula 32 bit bisa lebih ditingkatkan lagi. Sedangkan untuk algoritma Affine cipher yaitu memperpanjang kunci agar lebih sulit untuk dipecahkan.
2. Untuk penelitian lebih lanjut dapat ditingkatkan agar proses enkripsi tidak sebatas teks tapi juga bisa melakukan enkripsi terhadap jenis file lainnya seperti suara, image dan video.

DAFTAR PUSTAKA

Chehal, R. & Singh, K. 2012. Efficiency and security of data with symmetric encryption algorithms. *International Journal of Advanced Research in Computer Science dan Software Engineering* (8) : 472-475

Goyal, S.,Manna, M., & Goyal, A. 2014. Dictionary attack on MD5 hashed key password. *The International Journal of Science & Technology* (12): 130-134

Golose, P.R. 2006. *Perkembangan cybercrime dan upaya penanganannya di Indonesia oleh Polri.* Buletin hukum perbankan dan kebanksentralan, Agustus (2): 29-47

Kasgar, A.K., Dhariwal, M.K., Tantubay, N., & Malviya, H. 2013. A Review paper of message digest 5 (MD5). *International journal of modern engineering & management research* (4): 29-35

Mokhtari, M., & Naraghi, H. 2012. Analysis and desain of affine cipher and hill cipher. *Journal of mathematic research* (1): 67-77

Schneier, Bruce. 1996. *Applied cryptography, Second Edition. : Protocols, Algorithm and Source Code ini C*. Wiley computer publishing

Sreeraj, C., Kumar, K. S., & Nanda, K. R. 2012. Architectural design of MD5 controller IP core. *International journal of scientific & engineering research* (12): 1-5

Shukla, S., & Verma, P.K. 2014. Implementation of affine substituin cipher with keyed transposition cipher for enhancing data security. *International journal of advanced research in computer science and software engineering* (1): 236-240

Zelvina, A., Efendi, S & Arisandi, D.2012. Perancangan aplikasi pembelajaran kriptografi kunci publik El Gamal untuk mahasiswa. *Journal dunia teknologi informasi* (1): 56-62

LAMPIRAN

Kode sumber algoritma MD5 dengan bahasa pemrograman python.

```
import math
import random
import hashlib
import time
#=============Algoritma MD5============================="
kata=raw_input("Input Plaintext = ")

def algoritmaMD5(n):
   m = hashlib.md5()
   m.update(n)
   word=m.hexdigest()
   return word
#=================================================="
def MD5(plain):
   start=time.time()
   md5=algoritmaMD5(plain)
   print "Hasil Hash MD5 = ",md5, " = ",len(str(md5))," Karakter"
   end=time.time()
   result=end-start
   print "Waktu yang dibutuhkan MD5: ",result
```

Kode sumber algoritma Affine Cipher dengan bahasa pemrograman python.

```
import time
start = time.time()
def egcd(a, b):
        x,y, u,v = 0,1, 1,0
        while a != 0:
                q, r = b//a, b%a
                m, n = x-u*q, y-v*q
                b,a, x,y, u,v = a,r, u,v, m,n
        gcd = b
        return gcd, x, y
```

```
def modinv(a, m):
    gcd, x, y = egcd(a, m)
    if gcd != 1:
        return None # modular inverse does not exist
    else:
        return x % m
# affine cipher encrytion function
# returns the cipher text
def affine_encrypt(text, key):
    '''
    C = (a*P + b) % 26
    '''
    return ''.join([ chr((( key[0]*(ord(t) - ord('A')) + key[1] ) % 26)
                        + ord('A')) for t in text.upper().replace(' ', '') ])
# affine cipher decryption function
# returns original text
def affine_decrypt(cipher, key):
    '''
    P = (a^-1 * (C - b)) % 26
    '''
    return ''.join([ chr((( modinv(key[0], 26)*(ord(c) - ord('A') - key[1]))
                        % 26) + ord('A')) for c in cipher ])
# Driver Code to test the above functions
def main():
    # declaring text and key
    text = 'UINSU'
    key = [5, 2]
    # calling encryption function
    affine_encrypted_text = affine_encrypt(text, key)
    print('Encrypted Text: {}'.format( affine_encrypted_text ))
    # calling decryption function
    print('Decrypted Text: {}'.format
    ( affine_decrypt(affine_encrypted_text, key) ))
if __name__ == '__main__':
    main()
```

```
end = time.time()
result = end - start
print("Waktu enkripsi = ",result)
```