IMPLEMENTASI *AFFINE CHIPER* DAN RC4 PADA ENKRIPSI FILE TUNGGAL

Halim Agung[1*], **Budiman**[1]
[1] Program Studi Teknik Informatika, Fakultas Teknologi dan Desain, Universitas Bunda Mulia
Jl.Lodan Raya No.2, Jakarta 14430
[*]Email: hagung@bundamulia.ac.id

Abstrak

Informasi digital yang bersifat pribadi tentunya memiliki kerahasiaan data yang harus tetap dijaga. Salah satu solusi pengamanan informasi yang digunakan adalah teknik pengamanan data menggunakan kriptografi. Kriptografi ialah ilmu untuk menyandikan pesan yang berguna untuk membuat pesan, data, maupun informasi tidak dapat dibaca atau dimengerti oleh orang lain, kecuali untuk penerima yang berhak mengetahuinya. Kriptografi sendiri mempunyai berbagai metode untuk mengacak suatu data atau informasi. Contoh metode kriptografi ialah Affine Cipher dan Rivest Code 4 (RC4). Affine Cipher dan Rivest Code 4 (RC4) merupakan algoritma simetris sehingga kunci yang digunakan pada saat mengenkripsi dan mendekripsi sama. File yang telah dienkripsi berhasil teracak sehingga file tersebut tidak bisa dimengerti, dan hasil dekripsi sama dengan file asli sebelum dienkripsi. Dalam paper ini akan dibahas tentang aplikasi enkripsi dan dekripsi file menggunakan algoritma Affine Cipher dan Rivest Code 4 (RC4). Aplikasi ini dapat dijadikan sebagai salah satu cara mengamankan data. Pembuatan aplikasi ini menggunakan bahasa pemograman PHP dan MySQL. Hasil akhir berupa aplikasi client server, tanpa ada proses instalasi.

***Kata kunci**: Affine Chiper, Kriptografi, Rivest Code 4*

1. PENDAHULUAN

Informasi juga merupakan kebutuhan setiap orang dalam melakukan pekerjaan sehari-hari. Masalah yang muncul ketika informasi tersebut bersifat rahasia, terutama bagi suatu perusahaan, institusi atau organisasi yang mempunyai dokumen-dokumen rahasia dan data-data yang penting. Sehingga perlu menjaga keamanan dari dokumen-dokumen tersebut agar terhindar dari gangguan orang lain. Salah satu cara untuk mengamankan data atau informasi dari tindak kejahatan tersebut adalah menggunakan konsep kriptografi.

Kriptografi merupakan ilmu yang mempelajari mengenai cara mengamankan suatu informasi. Dalam kriptografi terdapat 2 tahap yaitu proses enkripsi dan dekripsi. Enkripsi adalah suatu proses yang dilakukan untuk mengubah pesan asli menjadi *ciphertext*. Sedangkan proses yang dilakukan untuk mengubah pesan tersembunyi menjadi pesan biasa (yang dapat dibaca dan dimengerti) disebut dekripsi. Pesan biasa atau pesan asli disebut plaintext sedangkan pesan yang telah diubah atau disandikan supaya tidak mudah dibaca disebut dengan *ciphertext*.

Sebagai upaya mewujudkan implementasi keamanan data dengan menggunakan metode enkripsi *Affine Cipher* dan *Rivest Code* 4 (RC4) ke dalam suatu aplikasi yang mudah digunakan. Aplikasi ini ditujukan untuk membantu mengatasi masalah keamanan data yang dibuat atau disimpan. Aplikasi ini juga hanya dibatasi dengan mengenkripsi file tunggal.

2. METODOLOGI

Data dan referensi diambil dari beberapa buku yang berhubungan dengan *steganografi* dan pemrograman web dan melakukan studi literatur yang berhubungan dengan metode yang dipakai dalam penelitian yang dilakukan

2.1. KRIPTOGRAFI

Menurut Kromodimoedjo (Kromodimoeljo, 2009), Kriptografi adalah ilmu mengenai teknik enkripsi dimana data diacak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi. Dekripsi menggunakan kunci dekripsi mendapatkan kembali data asli. Proses enkripsi dilakukan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma tidak dirahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan algoritma dianggap sesuatu yang tidak baik. Rahasia terletak di beberapa parameter yang digunakan, jadi kunci ditentukan oleh parameter. Parameter yang menentukan kunci dekripsi

itulah yang harus dirahasiakan (parameter menjadi ekuivalen dengan kunci). Ilustrasi dari penjelasan diatas dapat diliat pada gambar 1.

Gambar 1. Enkripsi secara umum

2.2. CRYPTANALYSIS

Menurut Kromodimoedjo (Kromodimoeljo, 2009), *Cryptanalysis* adalah teknik untuk mencoba memecahkan enkripsi, biasanya dengan mencari kunci enkripsi. Ada tiga kategori teknik pencarian kunci yang biasanya digunakan untuk kriptografi klasik yaitu *known plaintext attack*, analisa statistik, dan *brute force search*.

2.3. *AFFINE CHIPER*

Menurut Kromodimoedjo (Kromodimoeljo, 2009), Enkripsi yang digunakan Julius Caesar (Caesar cipher) menggunakan transformasi yang sederhana yaitu shift *transformation*. Pembahasan analisa statistik menunjukkan bahwa shift transformation sangat rentan terhadap analisa frekuensi. Untuk mencoba mempersulit analisa frekuensi, enkripsi *affine* menggunakan *affine transformation*, dengan rumus :

$$C = aP + b \pmod{n} \text{ untuk enkripsi} \quad (1)$$

Dan

$$P = a^{-1} C - a^{-1}b \pmod{n} \text{ untuk dekripsi.} \quad (2)$$

Jadi kunci untuk enkripsi *affine* terdiri dari dua parameter: a dan b. Agar a mempunyai *inverse* a-1, a harus mematuhi gcd(a, n) = 1.

2.4. RC4

Menurut Kromodimoedjo (Kromodimoeljo, 2009), RC4 adalah *stream cipher* yang dirancang di RSA *Security* oleh Ron *Rivest* tahun 1987. Pada mulanya cara kerja RC4 dirahasiakan oleh RSA *Security*, akan tetapi ini dibocorkan di *internet* tahun 1994 di milis Cypherpunks. RSA *Security* tidak pernah merilis RC4 secara resmi, akibatnya banyak yang menyebutnya sebagai ARC4 (*alleged* RC4 atau tersangka RC4) untuk menghindari masalah trademark.

RC4 dirancang agar dapat diimplementasikan di *software* secara sangat efisien. Ini membuat RC4 sangat populer untuk aplikasi *internet*, antara lain RC4 digunakan dalam *standard* TLS (*transport layer security*) dan WEP (*wireless equivalent privacy*).

Cara membuat *keystream* dalam RC4 adalah dengan *state automaton* dan terdiri dari dua tahap :

(1) Tahap *key scheduling* dimana *state automaton* diberi nilai awal berdasarkan kunci enkripsi.
(2) Tahap *pseudo-random generation* dimana *state automaton* beroperasi dan outputnya menghasilkan *keystream*.

Tahap pertama dilakukan menggunakan *key scheduling algorithm* (KSA). *State* yang diberi nilai awal berupa *array* yang merepresentasikan suatu permutasi dengan 256 elemen, jadi hasil dari algoritma KSA adalah permutasi awal. Arrayyang mempunyai 256 elemen ini (dengan indeks 0 sampai dengan 255) dinamakan S. Berikut adalah algoritma KSA dalam bentuk *pseudo-code* dimana key adalah kunci enkripsi dan *keylength* adalah besar kunci enkripsi dalam *bytes* (untuk kunci 128 bit, *key length* = 16) :

```
for i = 0 to 255
S[i] := i
j := 0
```

for i = 0 to 255
j := (j + S[i] + *key*[i mod *key length*]) mod 256
swap(S[i],S[j]) (3)

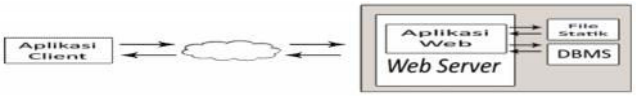
Tahap kedua menggunakan algoritma yang dinamakan *pseudo-random generation algorithm* (PRGA). Setiap putaran, bagian *keystream* sebesar 1 *byte* (dengan nilai antara 0 sampai dengan 255) dioutput oleh PRGA berdasarkan *state* S. Berikut adalah algoritma PRGA dalam bentuk *pseudo-code*:

i := 0
j := 0
loop
i := (i + 1) mod 256
j := (j + S[i]) mod 256
swap(S[i],S[j])
K = S[(S[i] + S[j]) mod 256] (4)

Byte K di-XOR kan dengan *plaintext* untuk menghasilkan *ciphertext* atau di XOR kan dengan *ciphertext* untuk menghasilkan *plaintext*. Permutasi dengan 255 elemen mempunyai 255! kemungkinan. Ditambah dua *indeks* (i dan j) yang masing-masing dapat mempunyai nilai antara 0 dan 255, maka *state automaton* yang digunakan untuk membuat *keystream* mempunyai 255! x2552= 21700 kemungkinan *internal states*. Karena banyaknya jumlah kemungkinan untuk internal *state*, sukar untuk memecahkan RC4 dengan menganalisa PRGA (teknik paling afisien saat ini harus menjajagi >2700 kemungkinan).

2.5. APLIKASI *WEB*

Aplikasi *web* adalah aplikasi yang disimpan dan dieksekusi di lingkungan *web server*. Setiap permintaan yang dilakukan oleh *user* melalui aplikasi klien (*web browser*) akan direspon oleh aplikasi *web* dan hasilnya akan dikembalikan lagi ke hadapan *user*. Dengan aplikasi *web*, halaman yang tampil di layar *web browser* dapat bersifat dinamis, tergantung dari nilai data atau parameter yang dimasukkan oleh *user*. Komunikasi antara *web browser* dan aplikasi *web* dapat digambarkan seperti berikut (Raharjo dkk, 2010) :



Gambar 2. Komunikasi *web browser* dan aplikasi *web*

Seperti yang tampak pada gambar 2 bahwa aplikasi *web* dapat juga digunakan untuk mengakses *file-file* yang bersifat statis (misal : dokumen, HTML, *file* gambar maupun *file teks*). Berbeda dengan dokumen HTML dan kode pemograman *client-side* lainnya (misal : *JavaScript* dan VB*Script*), kode dari aplikasi *web* tidak dapat dilihat atau dibaca oleh *user*.

3. HASIL DAN PEMBAHASAN

3.1. Analisis Sistem

3.1.1 Analisis Sistem Berjalan

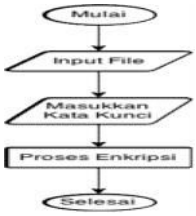
Pembuatan aplikasi enkripsi dimaksudkan untuk memaksimalkan peran algoritma *affine chiper* dan RC4 dalam mengenkripsi *file* tunggal dan didekripsikan kembali dengan benar tanpa bisa diganggu serangan dari luar serta menerapkan algoritma *Affine Cipher* dan *Rivest Code* 4 (RC4) yang digunakan untuk enkripsi dan dekripsi *file*.

Proses utama pada aplikasi ini adalah melakukan enkripsi dan dekripsi *file*. Adapun proses yang harus dilewati dalam melakukan enkripsi *file* sebagai berikut :

(1) Visitor

- Pengguna memasukkan input berupa *file*. *File* yang akan diinput berupa *file teks*, *file* gambar, *file* suara, dan lain sebagainya.

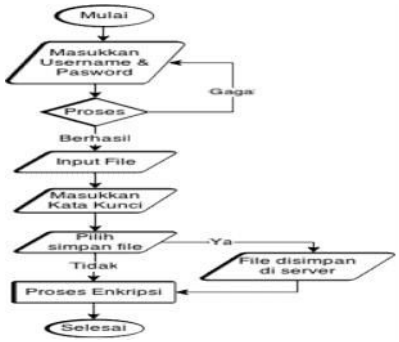
- Pengguna memasukkan kata kunci untuk mengenkripsi.
- Melakukan enkripsi *file* yang telah diinput.
- *File* yang telah dienkripsi menjadi *file* yang tidak terbaca.
- Diagram alir yang menggambarkan proses yang harus dilewati visitor tampak pada gambar 3.



Gambar 3. *Flowchart* Enkripsi Visitor

(2) *User* / Member

- Pengguna harus melakukan *login* terlebih dahulu.
- Setelah berhasil *login*, pengguna memasukkan input berupa *file*.
- Pengguna memasukkan kata kunci untuk mengenkripsi.
- Pengguna memilih apakah *file* hasil enkripsi akan disimpan pada *server*.
- Melakukan enkripsi *file* yang telah diinput.
- *File* yang telah dienkripsi menjadi *file* yang tidak terbaca.
- Diagram alir yang menggambarkan proses yang harus dilakukan oleh user/member dapat dilihat pada gambar 4.

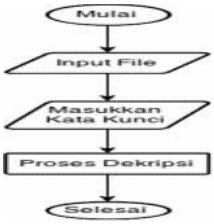


Gambar 4. *Flowchart* Enkripsi Member

Dan proses dekripsi *file* sebagai berikut :

(1) Visitor

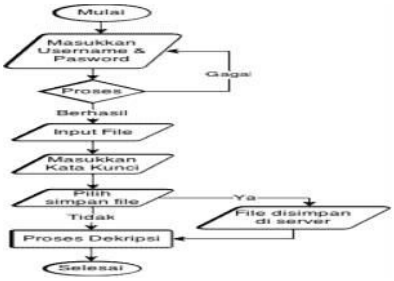
- Pengguna memasukkan input berupa *file* yang telah dienkripsi. *File* yang akan diinput berupa *file teks*, *file* gambar, *file* suara, dan lain sebagainya.
- Pengguna memasukkan kata kunci untuk mengdekripsi.
- Melakukan dekripsi *file* yang telah diinput.
- *File* yang telah didekripsi menjadi *file* asli / *file* yang dapat dibaca.
- Diagram alir proses deskripsi yang dilakukan oleh visitor dapat dilihat pada gambar 5.



Gambar 5. *Flowchart* Dekripsi Visitor

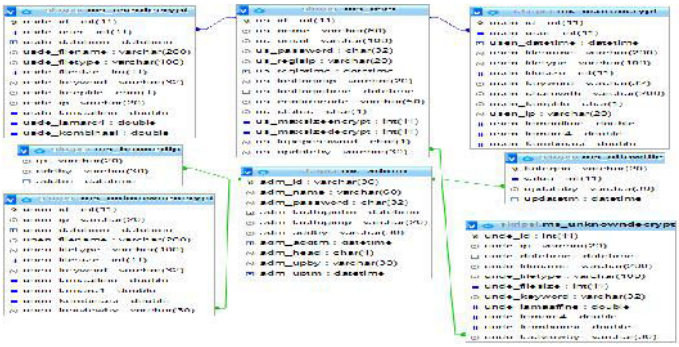
(2) *User* / Member

- Pengguna harus melakukan *login* terlebih dahulu.
- Setelah berhasil *login*, pengguna memasukkan input berupa *file* yang telah dienkripsi.
- Pengguna memasukkan kata kunci untuk mendekripsi.
- Pengguna memilih apakah *file* hasil enkripsi akan disimpan pada *server*.
- Melakukan dekripsi *file* yang telah diinput.
- *File* yang telah didekripsi menjadi *file* asli / *file* yang dapat dibaca.
- Diagram alir proses deskripsi yang dilakukan oleh user dapat dilihat pada gambar 6



Gambar 6. *Flowchart* Dekripsi Visitor

3.1.2 Perancangan Database



Gambar 7. *Entity Relationship Diagram*

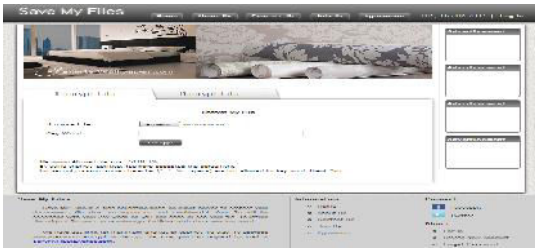
Gambar 7 merupakan tampilah relasi diagram pada database yang dipakai dalam perancangan aplikasi enkripsi *affine chiper* dan RC4.

3.2. Konstruksi Antarmuka

Bagian ini akan menjelaskan implementasi atau konstruksi tampilan dari aplikasi enkripsi *affine chiper* dan RC4.

3.2.1. Tampilan Halaman Utama

Pada saat pengguna mengakses aplikasi maka akan masuk ke halaman utama. Halaman utama terdapat *menu Home, About Us, Contact Us, Join Us, Agreement*, dan *Login*, serta terdapat *form* untuk enkripsi dan dekripsi. Tampilan halaman utama dapat dilihat pada gambar 8.



Gambar 8. Tampilan Halaman Utama

3.2.2. Tampilan *Form* Enkripsi Visitor

Pada *form* enkripsi visitor terdapat input *file* untuk memilih *file* dan *inputkeyword*, dan *buttonencrypt* dapat dilihat pada gambar 9.

Gambar 9. Tampilan *Form* Enkripsi Visitor

3.2.3. Tampilan *Form* Dekripsi Visitor

Pada *form* dekripsi visitor terdapat *input file* untuk memilih *file* dan *inputkeyword*, dan *button decrypt* dapat dilihat pada gambar 10.

Gambar 10. Tampilan *Form* Dekripsi Visitor

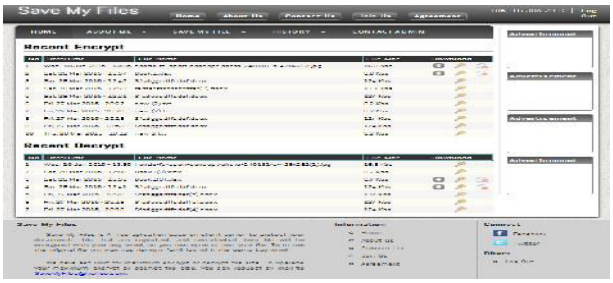
3.2.4. Tampilan Form *Login*

Pada *form login* yang ada pada gambar 11 terdapat *input email*, *input password* serta *button login*.

Gambar 11. Tampilan *Form Login*

3.2.5. Tampilan Halaman Utama *User*

Pada halaman utama *user* yang ada pada gambar 12 terdapat menu tambahan antara lain : *menu my profile, change password, encrypt file, decrypt file, history encrypt, history decrypt* dan *contact admin*



Gambar 12. Tampilan Halaman Utama *User*

3.2.6. Tampilan *Form Encrypt File User*

Pada *form encrypt file user* yang ada pada gambar 13 terdapat *inputfile, keyword, share keyword*, pilihan *keep file* dan *button encrypt*

Gambar 13. Tampilan *Form Encrypt File User*

3.2.7. Tampilan *Form Decrypt File User*

Pada *form decrypt file user* yang ada pada gambar 14 terdapat *input file*, *keyword*, pilihan *keep file* dan *button decrypt*.

Gambar 14. Tampilan *Form Decrypt File User*

3.3. Pengujian Aplikasi

Pada tahap pengujian enkripsi dan dekripsi ini dilakukan pada semua tipe *file*. Apabila telah terenkripsi akan menjadi *file* tidak dapat dibaca, dan jika didekripsi akan kembali menjadi *file* asli.

3.3.1. Pengujian Enkripsi

Pada tahap ini, penulis akan melihat apakah aplikasi dapat melakukan enkripsi pada tipe *file* yang berbeda dan apakah ukuran hasil enkripsi akan berubah atau tidak yang hasilnya dapat dilihat pada tabel 1 dan tabel 2.

Tabel 1. Tabel Informasi *File* Sebelum Enkripsi

No	File	Ukuran
1.	hsh.mp3	3,801.0 KB
2.	crypto-book-complete.pdf	2,386.4 KB
3.	wrar521.exe	1,718.8 KB
4.	video.mp4	1,193.0 KB
5.	TxtTipeFile.txt	0.3 KB
6.	sendproses.php	1.2 KB
7.	sad.png	14.8 KB
8.	NEWS0001533.pdf	60.6 KB

Tabel 2. Daftar Informasi *File* Sesudah Enkripsi

No	File	Ukuran
1.	hsh.mp3	3,801.0 KB
2.	crypto-book-complete.pdf	2,386.4 KB
3.	wrar521.exe	1,718.8 KB
4.	video.mp4	1,193.0 KB
5.	TxtTipeFile.txt	0.3 KB
6.	sendproses.php	1.2 KB
7.	sad.png	14.8 KB
8.	NEWS0001533.pdf	60.6 KB

Proses enkripsi berjalan lancar, semua tipe *file* berhasil dienkripsi tanpa ada kesalahan aplikasi. Semua *file* sudah berubah ke dalam data yang tersandikan. Untuk lama waktu yang digunakan tergantung pada besar kecilnya ukuran *file* yang dienkripsi, semakin besar ukuran *file* maka semakin lama proses enkripsinya. Waktu yang ditampilkan dalam hitungan detik. Tabel 3 menunjukan ukuran *file* dan waktu enkripsi.

Tabel 3. Ukuran *File* dan Waktu Enkripsi

No	File	Ukuran	Affine	RC4	Kombinasi
1.	hsh.mp3	3,801.0	12.49449801	3.73303604	16.22754
2.	crypto-book-complete.pdf	2,386.4	8.05017591	2.34973979	10.39992
3.	wrar521.exe	1,718.8	5.83142686	1.68749809	7.518928
4.	video.mp4	1,193.0	4.28993297	1.17905188	5.468988
5.	TxtTipeFile.txt	0.3	0.00266504	0.00111699	0.003784
6.	sendproses.php	1.2	0.00902414	0.00301504	0.012042
7.	sad.png	14.8	0.09776998	0.03141713	0.129191
8.	NEWS0001533.pdf	60.6	0.40029788	0.12548494	0.525786

3.3.2. Pengujian Dekripsi

Pada tahap ini akan dilihat apakah aplikasi dapat melakukan dekripsi, mengubah *file* kebentuk asli. Proses dekripsi berjalan lancar, semua tipe *file* bisa didekripsi ke bentuk semula. Untuk lama waktu yang digunakan tergantung pada besar kecilnya ukuran *file* yang didekripsi. Tabel 4 menunjukan ukuran *file* dan waktu dekripsi.

Tabel 4. Ukuran *File* dan Waktu Dekripsi

No	File	Ukuran	Affine	RC4	Kombinasi
1.	wrar521.exe	1,718.8	5.49403906	2.081985	7.576027
2.	video.mp4	1,193.0	3.72374105	1.5247829	5.248529
3.	TxtTipeFile.txt	0.3	0.00287008	0.00114703	0.00402
4.	sendproses.php	1.2	0.00992799	0.00303984	0.012971
5.	sad.png	14.8	0.09847498	0.03116202	0.12964
6.	NEWS0001533.pdf	60.6	0.19474792	0.0597651	0.254517
7.	kriptografiklasik.ppt	247.5	0.90486407	0.24619293	1.151062
8.	Jempol.gif	5.5	0.03604984	0.01194501	0.047998

4. KESIMPULAN DAN SARAN

4.1. KESIMPULAN

Dari pengujian terhadap penelitian yang dilakukan dapat diambil kesimpulan bahwa :

(1) Ukuran *file* yang semakin besar akan mempengaruhi lama proses enkripsi dan dekripsi. Semakin besar ukuran *file* akan semakin lama proses enkripsi dan dekripsinya.
(2) Algoritma *Affine Chiper* membutuhkan waktu lebih lama proses enkripsi dan dekripsinya dari pada algoritma RC4.
(3) Waktu kombinasi algoritma *Affine Chiper* dan RC4 lebih lama dari pada penjumlahan waktu algoritma *Affine Chiper* dan RC4 itu sendiri.

4.2. SARAN

Berdasarkan penelitian yang diperoleh, ada beberapa saran untuk pengembangan sistem lebih lanjut, antara lain :

(1) Aplikasi dalam penelitian ini berbasis *client server* dapat bervariasi menggunakan bahasa pemograman dan metode yang lain agar dapat memberikan pengamanan yang lebih baik.
(2) Aplikasi dalam penelitian ini hanya dapat memproses 1 *file* dalam 1 kali proses, diharapkan dapat memproses lebih banyak *file* atau 1 *folder* dalam 1 kali proses.
(3) Dengan semakin berkembangnya dunia *mobile*, diharapkan aplikasi dalam penelitian ini dapat dikembangkan dalam bentuk *mobile* dan dapat digunakan dengan teknologi jaman sekarang ini yaitu menggunakan ponsel android.

UCAPAN TERIMA KASIH

Saya ucapkan terima kasih sebesar – besarnya kepada Tuhan Yang Maha Esa yang telah memberikan saya kesempatan dalam penulisan paper ini. Saya juga mengucapkan terima kasih kepada ibu saya yang telah mensupport saya dalam menjalani seluruh kegiatan dalam pekerjaan saya. Terakhir saya juga mengucapkan terima kasih kepada Universitas Bunda Mulia yang telah mempercayakan saya untuk ikut serta dalam seminar SNATIF 2015 ini.

DAFTAR PUSTAKA

Kromodimoeljo, Sentot. (2009). Teori dan Aplikasi Kriptografi. Jakarta : SPK IT Consulting.

Raharjo dan Budi. (2010). WEB (HTML, PHP & MYSQL). Bandung : Modula.

Rizal, Ansar, Suharto. (2011). Implementasi Algoritma RC4 untuk Keamanan Login Pada Sistem Pembayaran Uang Sekolah. Dielektrika, ISSN 2086-9487 Vol. 2 No.2

Wirdasari, Dian. (2008). Prinsip Kerja Kriptografi dalam Mengamankan Informasi, Jurnal SAINTIKOM Vol.5 No.2.