ISSN : 2407 - 3911

KRIPTOGRAFI RSA PADA APLIKASI *FILE TRANSFER CLIENT- SERVER BASED*

Muhammad Arief[1], Fitriyani[2], Nurul Ikhsan[3]
[1,2,3]Prodi Ilmu Komputasi Universitas Telkom Bandung
Jl. Gegerkalong Hilir, Bandung, Jawa Barat 40152
[1]ariefmuhammad08@gmail.com, [2]fitriyani.telkomuniversity.ac.id,
[3]nurul.ikhsan@yahoo.co.id

Abstrak

Perkembangan teknologi membuat kemudahan dalam berkomunikasi. Kemudahan ini juga membuat mudahnya tersebar data-data privat seseorang. Dibutuhkan suatu pengamanan data. Algoritma RSA merupakan algoritma kriptografi yang memiliki tingkat keamanan tinggi. Kunci RSA dengan panjang 1024 bit dapat memakan waktu ratusan tahun untuk dibobol jika menggunakan metode *brute force*.

Dalam penelitian ini, enkripsi RSA pada file akan diimplementasikan dalam sebuah aplikasi FTP *client*. Saat proses *upload, file* akan dienkripsi sehingga *file* tersebut tidak bisa dibaca sembarang orang. Hanya yang memiliki kunci yang dapat membacanya. Dengan ini dihasilkan mekanisme berbagi *file* yang lebih aman walaupun menggunakan sebuah jaringan publik.

Dari beberapa percobaan, dihasilkan bahwa algoritma RSA dapat digunakan untuk enkripsi dan dekripsi sebuah *file* untuk meningkatkan keamanan pada suatu jaringan publik. Namun dikarenakan penggunaan JVM yang terbatas, ukuran *file* yang dapat dienkripsi juga terbatas.

Kata Kunci : Kriptografi, RSA, FTP

Abstract

Developments in technology make it easy to communicate. This easiness also makes it easy to spread someone's private data. It required a data security. The RSA algorithm is a cryptographic algorithm that has a high security level. RSA key with a length of 1024 bits can take hundreds of years to be cracked if using a brute force method.

In this research, RSA encryption algorithm of files will be implemented on a FTP client application. When the upload process, the file will be encrypted so that the file cannot be read by anyone. Only those who have a key that can read them. With the implementation of this application resulted a secure file sharing mechanism despite using a public network.

From some experiments, resulted that the RSA algorithm can be used to encrypt and decrypt a file to improve the security on a public network. However, due to the limited use of the JVM, the size of files that can be encrypted is also limited

Keywords: Cryptography, RSA, FTP

I. PENDAHULUAN

I.1 Latar Belakang

Perkembangan teknologi belakangan ini tumbuh dengan sangat pesat. Perkembangan teknologi ini banyak melahirkan keuntungan yang mempermudah kelangsungan hidup manusia. Tapi seiring dengan berkembangnya teknologi, muncul pula masalah-masalah baru, mulai dari privasi, keamanan, hingga hak cipta. Hal inilah yang sering dimanfaatkan oknum-oknum yang tidak bertanggung jawab untuk melakukan hal-hal negatif bahkan tindak kejahatan. Mudahnya penyebaran informasi palsu, pembobolan akun bank, dan maraknya pembajakan merupakan contoh sisi gelap dari perkembangan teknologi. Hal ini membuat para pengembang TI memutar otak untuk menangani masalah-masalah tersebut. Yang tidak pernah habis menjadi pembicaraan adalah pengembangan sistem keamanan.

Banyak cara yang dilakukan oleh pengembang TI dalam urusan keamanan, salah satunya menggunakan metode kriptografi. Kriptografi adalah

sebuah ilmu yang mempelajari tentang penyembunyian huruf atau tulisan sehingga membuat tulisan tersebut tidak dapat dimengerti atau dibaca, proses ini disebut dengan enkripsi. Untuk membacanya kembali, dilakukan proses dekripsi atau pengembalian ke tulisan aslinya. Hal ini banyak dimanfaatkan untuk menyamarkan dokumen-dokumen penting sehingga hanya orang-orang tertentu yang dapat membuka dan membacanya. Selain itu membuat dokumen tersebut aman apabila jatuh ke pihak lain. Seiring dengan perkembangannya, kriptografi mulai dimanfaatkan untuk menyamarkan *file-file* non dokumen, seperti, gambar, video maupun suara.

Tapi bagaimana jika *file* hasil enkripsi dalam kriptografi ini dikirim melalui sebuah jaringan? Bagaimana jika *file* yang di kirim jatuh ke tangan kriptanalisis? Dalam penelitian ini penulis mencoba membuat sebuah perangkat lunak untuk melakukan pengiriman *file* hasil enkripsi dengan aman. Selain itu penulis juga akan membuat sebuah jaringan FTP *public* sebagai media transfer *file*. Algoritma yang dicoba untuk diimplementasikan oleh penulis adalah kriptografi RSA. Berdasarkan hasil penelitian sebelumnya, kriptografi RSA telah dimanfaatkan dalam pengiriman pesan, seperti layanan *Chat*[7] dan SMS. Disini Penulis mencoba mengimplementasikan kriptografi RSA pada sebuah *file*. Dari penelitian ini diharapkan dapat menjadi solusi untuk keamanan pada pengiriman *file*.

II. KAJIAN LITERATUR

II.1 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani, terdiri dari dua suku kata yaitu *kripto* dan *graphia*. *Kripto* artinya menyembunyikan, sedangkan *graphia* artinya tulisan. Sehingga Kriptografi dapat diartikan sebagai ilmu yang mempelajari tentang penyembunyian huruf atau tulisan sehingga membuat tulisan tersebut tidak dapat dibaca oleh orang yang tidak berkepentingan. Kriptografi mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Algoritma kriptografi merupakan langkah-Iangkah logis bagaimana menyembunyikan pesan dari orang-orang yang tidak berhak atas pesan tersebut.

Algoritma kriptografi terdiri dari tiga fungsi dasar, yaitu:

1. Enkripsi: merupakan hal yang sangat penting dalam kriptografi. merupakan pengamanan data yang dikirimkan agar terjaga kerahasiaannya. Pesan asli disebut *plaintext*, yang diubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan dengan *cipher* atau kode. Sama halnya dengan kita tidak mengerti akan sebuah kata maka kita akan melihatnya di dalam kamus atau daftar istilah. Beda halnya dengan enkripsi. untuk mengubah teks-asli ke bentuk teks-kode kita menggunakan algoritma yang dapat mengkodekan data yang kita inginkan.
2. Dekripsi: merupakan kebalikan dari enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (teks-asli), disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan algoritma yang digunakan untuk enkripsi.
3. Kunci: yang dimaksud di sini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian, kunci privat (*private key*) dan kunci umum (*public key*).

II.2 Kriptografi RSA

Dari banyak algoritma kriptografi asimetris yang ada, algotirma yang paling populer adalah RSA. Algortima RSA dibuat oleh tiga orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976. Nama RSA merupakan singkatan dari nama tiga orang penemunya, yaitu Rivest, Shamir, dan Adleman.

Algoritma RSA melakukan pemfaktoran bilangan yang sangat besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci privat.

Algoritma RSA memiliki besaran-besaran sebagai berikut:

1. p dan q bilangan prima (rahasia)
2. $n = p \cdot q$ (tidak rahasia)
3. $\phi(n) = (p-1)(q-1)$ (rahasia)
4. e (kunci enkripsi) (tidak rahasia)
   Syarat: PBB(e, $\phi(n)$) = 1
5. d (kunci dekripsi) (rahasia)
   d dihitung dari $d \equiv e^{-1} \bmod (\phi(n))$
6. m (plainteks) (rahasia)

7. c (cipherteks) (tidak rahasia)

Pembangkitan kunci :

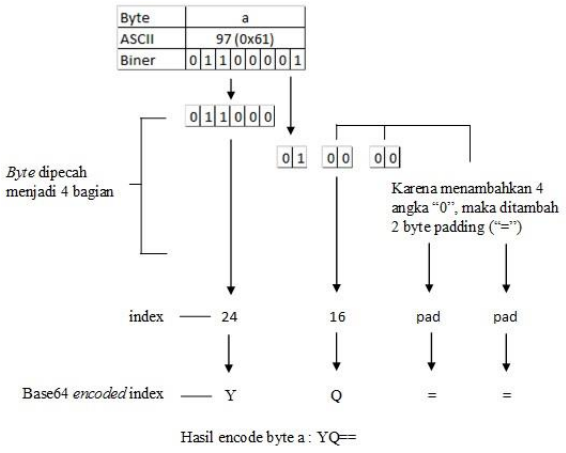
1. Pilih dua bilangan prima, a dan b (rahasia)
2. Hitung $n = a\ b$. Besaran n tidak perlu dirahasiakan.
3. Hitung $\phi(n) = (a-1)(b-1)$.
4. Pilih sebuah bilangan bulat untuk kunci publik, sebut namanya e, yang relatif prima terhadap $\phi(n)$.
5. Hitung kunci dekripsi, d, melalui $ed \equiv 1 \pmod{m}$ atau $d \equiv e^{-1} \bmod (\phi(n))$

Hasil dari algoritma di atas:

1. Kunci publik adalah pasangan (e, n)
2. Kunci privat adalah pasangan (d, n)

II.3 Base 64

Base64 adalah sebuah skema *biner-to-text encoding* yang merepresentasikan data biner dalam format *string* ASCII dengan menerjemahkannya ke dalam representasi base64. Prosesnya dengan mengubah semua *string* ASCII ke dalam sebuah biner yang dapat direpresentasikan dalam indeks Base64. Hal ini membuat *control code* pada ASCII dapat diolah dan ditulis kembali karena sudah berbentuk indeks base64. Proses Base64 akan meningkatkan ukuran *file* sebesar 33%.

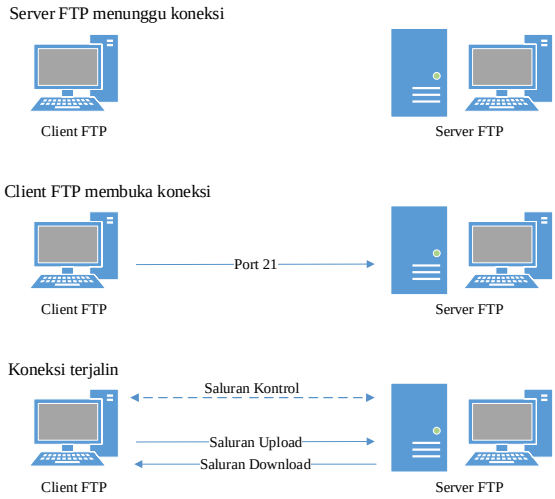


Gambar 1 : Proses encoding Base 64

Base64 *encoding* biasanya digunakan untuk meng-*encode* sebuah data biner yang perlu disimpan atau ditransfer melalui media yang dikhususkan untuk menangani data tekstual. Hal ini untuk memastikan bahwa data tetap utuh tanpa modifikasi selama transportasi.

II.4 File Transfer Protocol

File Transfer Protocol (FTP) adalah suatu protokol yang berfungsi untuk melakukan pertukaran *file* antar mesin dalam suatu jaringan yang menggunakan koneksi TCP. Dua hal yang penting dalam FTP adalah FTP *Server* dan FTP *Client*. FTP *server* adalah suatu *server* yang menjalankan software yang berfungsi untuk memberikan layanan tukar menukar *file* dimana *server* tersebut selalu siap memberikan layanan FTP apabila mendapat permintaan (*request*) dari FTP *client*.



Gambar 2 : Proses Kerja FTP

III. PERANCANGAN SISTEM

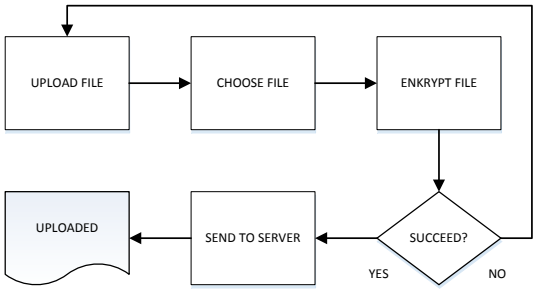
III.1 Desain Umum

Sistem yang akan dibangun pada penelitian ini adalah implementasi algoritma kriptografi RSA pada program *file* transfer sederhana. Program sendiri berupa sebuah FTP *Client* dengan fungsi-fungsi dasar dalam FTP, yaitu:

1. *Login* (*connect* dan *disconnect*)
2. *Upload*
3. *Download*

III.2 Proses Upload

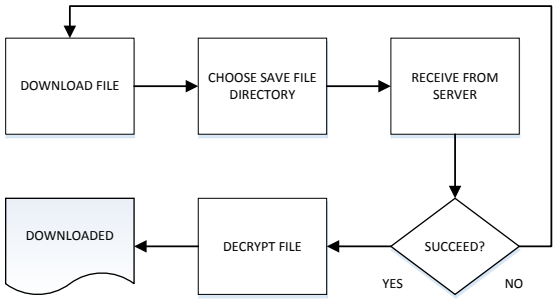
Pada bagian ini dijelaskan bagaimana alur proses *upload* yang dilakukan oleh sistem. Algoritma kriptografi yang dipakai adalah RSA. Enkripsi dilakukan sesaat sebelum *file* akan dikirimkan ke *server*.



Gambar 3 : *Flowchart* Proses *Upload*

III.3 Proses Download

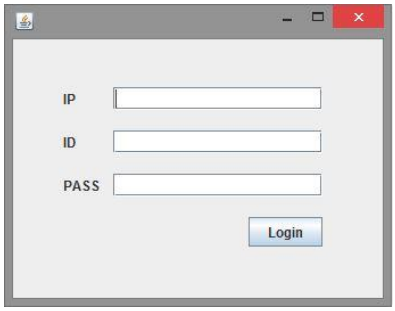
Pada proses *upload*, dilakukan eknripsi pada *file*. Jika *file* di-*download*, hanya akan menghasilkan sebuah *chipper file*. Pada proses *download* ini akan dilakukan deskripsi untuk mengembalikan *file* ke bentuk semula.



Gambar 4 : *Flowchart* Proses *Download*

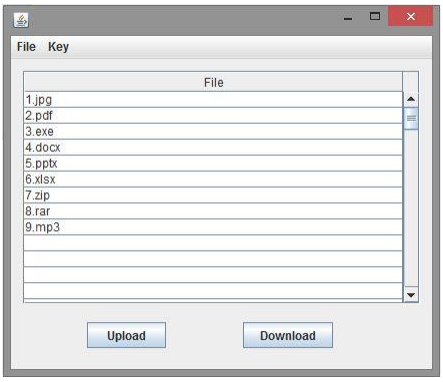
III.4 Tampilan Program

Tampilan Program dibuat berdasarkan fungsi dasar pada sebuah FTP *Client*. Tampilan dibuat sederhana untuk memudahkan pengguna.



Gambar 5 : Tampilan awal

Gambar 5 merupakan tampilan awal program, halaman untuk *login* ke *server*. Disini *user* diminta untuk memasukkan IP *address server*, *id*, dan *password*.



Gambar 6 : Tampilan halaman utama

Gambar 6 adalah tampilan halaman utama, dimana *user* dapat melihat *file* yang ada di dalam *server*. Selain itu, *user* dapat melakukan perintah *upload*, *download*, buat kunci, *load* kunci, serta *disconnect* dari *server*.

IV. ANALISIS DAN PENGUJIAN

IV.1 Batasan Pengujian

Agar pembahasan tidak menyimpang dan meluas, maka pengujian akan dibatasi sebagai berikut:

1. Kriptografi yang digunakan adalah algoritma RSA.
2. Proses enkripsi dan dekripsi dijalankan pada CPU.
3. Aplikasi Transfer *file* berbasis *Client-Server*.
4. Jaringan FTP *public* menggunakan *Filezilla Server*.

5. Jaringan yang digunakan adalah LAN (*Local Area Network*).
6. Tidak ada *folder* dalam *Database Server*.
7. Tidak dilakukan proses *brute-force* pada *file* hasil enkripsi.
8. JVM maksimal pada setiap percobaan adalah 2048 *megabytes*.

IV.2 Percobaan *File*

Sebagai percobaan awal, dipilih 19 *file* berbeda dengan ukuran acak untuk di-*upload* dan di-*download* menggunakan program FTP *Client* yang telah dibuat. Percobaan dibagi berdasarkan jaringan yaitu *localhost* dan LAN.

Dari hasil percobaan didapat perubahan ukuran yang terjadi pada file yang dienkripsi.

Tabel 1 : Tabel Perubahan Ukuran File

File	Size (KB)	
	Plain	Chipper
1.jpg	82	4536
2.pdf	302	16683
3.exe	2040	112853
4.docx	269	14861
5.pptx	1151	63645
6.xlsx	16	854
7.zip	2384	136659
8.rar	1158	66348
9.mp3	4018	222283
10.wmv	25631	Gagal
11.flv	7402	404640
12.mp4	16873	Gagal
13.wav	19599	Gagal
14.png	24546	Gagal
ekstra10.wmv	2208	126513
ekstra11.flv	1262	72270
ekstra12.mp4	2713	155497
ekstra13.wav	479	27400
ekstra14.png	554	31754

Hasil dari percobaan *file* yang telah dilakukan pada *localhost* dan LAN mengalami keterbatasan *resource memory*, pada Tabel 1 dapat dilihat bahwa file no. 10, 12, 13, 14 mengalami *exception java.lang. Out Of Memory Error* karena *heap space* yang sudah tidak memadai untuk menjalankan JVM. Hal ini terjadi karena pada setiap percobaan hanya dialokasikan memori JVM sebesar 2048 *megabytes*. Solusi yang dapat dilakukan adalah menggunakan komputer yang memiliki memori lebih besar dengan pengaturan JVM diatas 2048 *megabytes*.

Seperti terlihat pada Tabel 1, semua jenis *file* bisa dienkripsi dan didekripsi dengan batasan seperti dijelaskan diatas.

IV.3 Keamanan

Percobaan selanjutnya yang akan dilakukan adalah menguji keamanan dari sistem yang telah dirancang. Pengujian dilakukan dengan 2 cara yaitu dengan men-*download file* melalui FTP client lain dan dengan melakukan dekripsi menggunakan kunci RSA yang berbeda.

Hasil percobaan ini dapat dilihat pada Tabel 2. Dari Tabel 2 terlihat bahwa *file* yang didownload menggunakan FTP *client* lain tidak dapat dibuka atau dibaca.

Tabel 2 : Hasil Percobaan Download File

File	Percobaan	
	User 1	User 2
1.jpg	Tidak Terbaca	Tidak Terbaca
2.pdf	Tidak Terbaca	Tidak Terbaca
3.exe	Tidak Terbaca	Tidak Terbaca
4.docx	Tidak Terbaca	Tidak Terbaca
5.pptx	Tidak Terbaca	Tidak Terbaca
6.xlsx	Tidak Terbaca	Tidak Terbaca
7.zip	Tidak Terbaca	Tidak Terbaca
8.rar	Tidak Terbaca	Tidak Terbaca
9.mp3	Tidak Terbaca	Tidak Terbaca

Selanjutnya dilakukan percobaan mendekripsi *file* dengan kunci yang berbeda dengan kunci saat enkripsi.

Tabel 3 : Hasil Percobaan Kunci

File	Kunci Asli	Kunci Percobaan	Hasil Percobaan
1.jpg	1024	512	Corrupted
2.pdf	1024	64	Corrupted
3.exe	512	256	Corrupted
4.docx	512	32	Corrupted
5.pptx	256	128	Corrupted
6.xlsx	256	16	Corrupted
7.zip	128	64	Corrupted
8.rar	128	1024	Corrupted
9.mp3	64	32	Corrupted

Dari hasil percobaan keamanan dapat dilihat bahwa *file* yang telah di-*upload* menggunakan program yang dibuat tidak dapat dibuka selain di-*download* menggunakan program itu sendiri, jadi

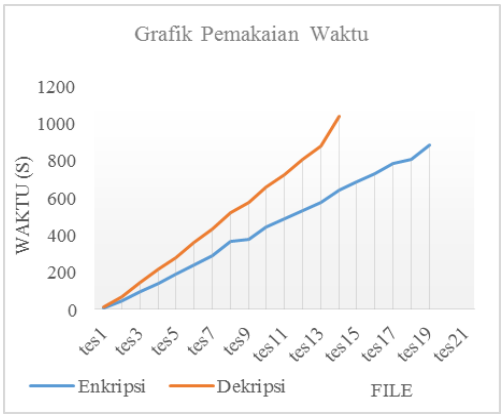
dapat dikatakan aman secara sederhana. Program FTP lain atau FTP dari Windows dapat men-*download file* yang disimpan pada *database server* tetapi *file* tersebut tidak akan bisa diabaca atau dibuka.

Pada Tabel 3 dapat dilihat bahwa penggunaan kunci berbeda menyebabkan *file* menjadi rusak atau *corrupted.*

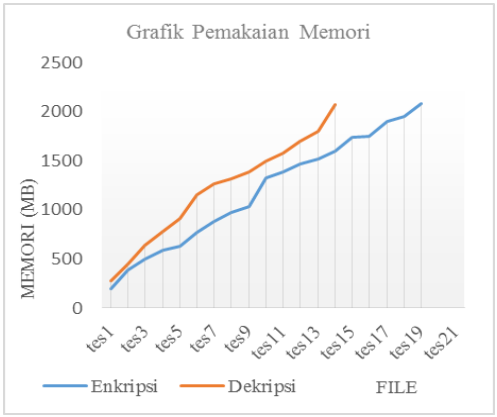
IV.4 Waktu dan Resource

Pengujian dilakukan dengan menghitung memori dan waktu enkripsi dan dekripsi 21 *file* .txt dengan ukuran 100 *kilobytes* hingga 10000 *kilobytes* menggunakan kunci 128 bit.

Hasil dari pengukuran waktu dan resource memori dapat dilihat pada Gambar 7 dan Gambar 8. Pada percobaan tersebut terjadi beberapa kegagalan. Untuk *file* tes15 hingga tes19 mengalami kegagalan pada saat dekripsi, sedangkan tes20 dan tes21 gagal saat enkripsi dan dekripsi.



Gambar 7 : Grafik Pemakaian Waktu



Gambar 8 : Grafik Pemakaian Memori

Sama halnya dengan percobaan *file*, terjadi peningkatan ukuran *file* yang signifikan pada saat *file* di enkripsi. Peningkatan ukuran file dapat dilihat pada Tabel 4.

Tabel 4 : Tabel Perubahan Ukuran File

File (.txt)	Ukuran (KB)		Kenaikan Ukuran Relatif
	Plain	Chipper	
tes1	100	5723	57,23
tes2	500	28656	57,312
tes3	1000	57352	57,352
tes4	1500	86055	57,37
tes5	2000	114767	57,3835
tes6	2500	143469	57,3876
tes7	3000	172173	57,391
tes8	3500	200883	57,39514286
tes9	4000	229583	57,39575
tes10	4500	258288	57,39733333
tes11	5000	286999	57,3998
tes12	5500	315689	57,398
tes13	6000	344409	57,4015
tes14	6500	373108	57,40123077
tes15	7000	401806	57,40085714
tes16	7500	430507	57,40093333
tes17	8000	459231	57,403875
tes18	8500	487925	57,40294118
tes19	9000	516647	57,40522222
tes20	9500	Gagal	Gagal
tes21	10000	Gagal	Gagal
		Rata-rata	57,38045715

Pada percobaan waktu dan *resource* dapat dilihat bahwa *file* yang telah dienkripsi mengalami peningkatan *size* dari semula. Hal ini karena metode yang dipakai pada penelitian ini adalah dengan cara melakukan enkripsi pada tiap *array byte* menggunakan Base64 *encoding* untuk menjaga keutuhan setiap *byte* pecahan yang dienkripsi. Hal ini membuat ukuran *file* meningkat sekitar 33%. Dari percobaan diatas dihasilkan bahwa proses enkripsi RSA akan meningkatkan ukuran *file* sekitar 57,3804 kali dari ukuran aslinya. Mekanisme ini akan menghasilkan keamanan yang berlipat ganda, dengan bayaran mengorbankan waktu *upload* karena *file* menjadi sangat besar dari semula. Tetapi untuk mendapatkan keamanan yang tinggi maka akan mengorbankan waktu pemrosesan

Pada pengujian ini tidak dilakukan *cracking* pada *chipper file*. Hal ini dikarenakan kurangnya sumber daya untuk melakukan proses *cracking* dan lamanya waktu yang diperlukan jika melakukan proses *brute-force*.

V. KESIMPULAN DAN SARAN

V.1 Kesimpulan

Kesimpulan yang dapat diambil dari penelitian yang telah dilakukan adalah sebagai berikut.

1. Algoritma asimetris seperti RSA dapat diterapkan dalam perangkat lunak FTP *client* dengan mengorbankan waktu *upload* tetapi menghasilkan keamanan yang lebih baik. Selain itu terjadi peningkatan ukuran *file* karena mekanisme base64 dan enkripsi RSA yang diaplikasikan ke setiap *byte* pada *file* yang akan dienkripsi.
2. Proses enkripsi dan dekripsi algoritma RSA dapat di implementasikan ke semua *file*, namun dikarenakan keterbatasan memori JVM yang dapat digunakan maka pada penelitian ini *file* yang dapat di enkripsi berukuran maksimal sekitar 9 *megabytes*.
3. Keutuhan *file* tetap terjaga bila serangkaian proses enkripsi, dekripsi, upload dan download dilakukan dengan program yang telah dibuat. Selain itu, *chipper file* juga dapat di-*download* dan didekripsi menggunakan kunci RSA lain tetapi akan diperoleh *file* seperti aslinya.
4. Berdasarkan hasil analisis, algoritma RSA yang diterapkan pada *file* dalam suatu jaringan komputer dapat meningkatkan keamanan pada *file* walaupun ditransfer dalam jaringan publik.

V.2 Saran

Saran yang dapat diambil dari penelitian yang telah dilakukan adalah sebagai berikut.

1. Untuk pengembangan dapat dibuat program FTP *Client* serupa tetapi berbasis *web based* atau *mobile*.
2. Menambahkan sistem keamanan dengan mekanisme pengecekan keaslian (autentifikasi).
3. Penggunaan GPU *Computing* dalam proses enkripsi dan deskripsi untuk efisiensi waktu dan *resource*.

Untuk mengoptimasikan *resource* dapat dilakukan dengan menggunakan PC dengan *resource* yang lebih besar.

REFERENSI

M. Z. Achmad Zakki Falani, *Sistem Pengaman File dengan Menggunakan Metode RSA Kriptografi & Digital Signature*, 2008.

R. Munir, Kriptografi, Bandung: Informatika Bandung, 2006.

D. Ariyus, Pengantar Ilmu Kriptografi, Yogyakarta: STIMIK AMIKOM Yogyakarta, 2008.

B. Raharjo, "e-Security : Keamaan Teknologi Informasi," Jakarta, 2004.

Apache, "FTP Client (Commons Net 3.3 API)," 25 September 2014. [Online]. Available: http://commons.apache.org/proper/commons-net/apidocs/org/apache/commons/net/ftp/FTPClient.html

Menezes, P. VanOorchot dan S. Vanstone, Handbook of Applied Cryptography, CRC Press, Inc, 1997.

A. Khairan, Analisis dan Implementasi Kriptografi RSA pada Aplikasi Chat Client Server Based, Bandung: Telkom University, 2014.