Terbit online pada laman web jurnal: **https://journal.budiluhur.ac.id/index.php/bit**

JURNAL BIT
(Budi Luhur Information Technology)

Vol. 16 No. 2 (2019) 13 - 22 ISSN Media Elektronik: **2685-127X**

Aplikasi Kriptografi Menggunakan Algoritma RSA Dan Corrected Block TEA (XXTEA)

M. Anif[1)], **Siswanto**[2)], **Muhammad Fikri**[3)], **Gunawan Pria Utama**[4)]
[1)2)3)4)]Teknik Informatika, Fakultas Teknologi Informasi, Universitas Budi Luhur
Jl. Ciledug Raya, Petukangan Utara, Jakarta Selatan, 12260
E-mail : muhammad.anif@budiluhur.ac.id[1)], siswanto@budiluhur.ac.id[2)], fikswznd@gmail.com[3)], gputama@gmail.com[4)]

Abstract

Cryptography is a very important part in the security of digital data exchange in daily activities, so a strong and fast cryptographic algorithm is needed. But sometimes there are security holes caused by users, such as exchanging keywords that are used to open encrypted data on insecure media. Then we need a cryptographic application that can prevent these security holes, thus minimizing security holes caused by users. The cryptographic method needed to support the application is by combining asymmetric and symmetric methods, namely the RSA algorithm and Corrected Block TEA (XXTEA). In this application users can carry out the process of encryption and decryption without the process of exchanging keywords on each file sent, and built using the JavaScript programming language on the NodeJS platform. This application has the ability to encrypt and decrypt text files and binary files.

Keywords: *Data Security, Encryption, Cryptography, RSA, Corrected Block TEA, XXTEA.*

Abstrak

Kriptografi menjadi baigan yang sangat penting dalam keamanan pertukaran data digital dalam kegiatan sehar-hari, sehingga algoritma kriptografi yang kuat dan cepat sangat dibutuhkan. Namun terkadang terjadi celah keamanan yang disebabkan oleh penggunanya, seperti melakukan pertukaran kata kunci yang digunakan untuk membuka data yang sudah terenkripsi pada media yang tidak aman. Maka dibutuhkan aplikasi kriptografi yang dapat mencegah celah keamanan tersebut, sehingga menimalisir celah keamanan yang disebabkan oleh pengguna. Metode kriptografi yang dibutuhkan untuk mendukung aplikasi tersebut adalah dengan penggabungan metode asimetris dan simetris, yaitu algoritma RSA dan *Corrected Block TEA* (XXTEA). Pada aplikasi ini pengguna dapat melakukan proses enkripsi dan dekripsi tanpa melakukan proses pertukaran kata kunci pada setiap *file* yang dikirimkan, dan dibangun menggunakan bahasa pemrogram JavaScript pada *platform* NodeJS. Aplikasi ini memiliki kemampuan enkripsi dan dekripsi file teks maupun file *binary*.

Kata kunci: Keamanan Data, Enkripsi, Kriptografi, RSA, *Corrected Block TEA*, XXTEA.

1. PENDAHULUAN

Pada era digital saat ini pertukaran data atau informasi sudah semakin mudah dilakukan, dengan memungkinkannya pertukaran data atau informasi tanpa melalui media fisik. Hal tersebut berbanding lurus dengan kemudahan pelaku kejahatan dalam melakukan penyimpangan atau penyalahgunaan data atau informasi. Sehingga pada saat ini keamanan data merupakan bagian yang sangat penting dalam proses komunikasi digital. Salah satu teknik keamanan data yang sering digunakan adalah teknik kriptografi, yang berfungsi melakukan penyembunyian pesan dengan cara mengubah data yang asli menjadi data acak menggunakan kata kunci yang sudah ditentukan, sehingga data yang dikirimkan hanya dapat dibaca oleh yang berhak.

PT. Beetlebox Indonesia adalah perusahaan pengembang perangkat lunak untuk pengembangan produk internal, atau produk eksternal yang bersifat pesanan dari *customer*. Perangkat lunak yang dikembangkan sering kali membutuhkan pertukaran data atau informasi antara pengguna ataupun antar perangkat lunak yang digunakan oleh pengguna dan perangkat lunak yang digunakan pada server (*client - server*). Acap kali data yang dikirimkan bersifat sensitif ataupun rahasia. Sehingga teknik kriptografi merupakan bagian yang sangat penting dalam proses pengembangan perangkat lunak, khususnya pada PT. Beetlebox Indonesia. Namun pernah terjadi penyimpangan data atau informasi yang dikirimkan meskipun sudah diamankan dengan teknik kriptografi

Diterima Redaksi : 31-12-2019 | Selesai Revisi : 31-12-2019 | Diterbitkan Online : 01-01-2020

yang kuat. Setelah dilakukan proses penyelidikan hal tersebut terjadi dikarenakan keterlibatan pengguna dalam proses pertukaran kata kunci. Dimana banyak pengguna yang melakukan pertukaran kata kunci melalui media komunikasi yang kurang aman, seperti pada layanan chat *public* atau *email* pribadi, juga kata kunci yang dikirimkan bersifat *plaintext*, tidak dilakukan proses enkripsi, sehingga baik penerima maupun pelaku kejahatan dapat membacanya langsung.

Berdasarkan kejadian tersebut diperlukan penelitian dan uji coba teknik kriptografi tambahan yang memungkinkan pengguna tetap dapat melakukan pertukaran kata kunci pada media yang kurang aman, tanpa membahayakan data yang sudah diamankan sebelumnya, yaitu dengan menggabungkan teknik kriptografi RSA dan Corrected Block TEA. Sehingga diharapkan kasus serupa, atau kasus lain yang disebabkan kecerobohan pengguna tidak terulang kembali.

Permasalahan yang dihadapi adalah data atau informasi yang sebelumnya memiliki celah keamanan yang disebabkan keterlibatan pengguna pada proses pertukaran kata kunci. Tujuan penelitian ini adalah membuat aplikasi yang dapat melakukan enkripsi pada file menggunakan algoritma *RSA* dan *Corrected Block TEA*. Sehingga diharapkan tidak terulang kembali kasus celah keamanan yang disebabkan oleh kelalain pengguna dalam proses pertukaran kata kunci.

Algoritma kriptografi RSA ditemukan oleh tiga orang yang kemudian nama-nama mereka disingkat menjadi RSA. Ketiga penemu itu adalah Ron Riverst, Adi Shamir, dan Leonard Adleman. RSA dibuat di MIT pada tahun 1977 dan dipatenkan oleh MIT (Massachussets Institute of Technology) pada tahun 1983 [1]. Sejak 21 September tahun 2000, paten tersebut berakhir, sehingga saat ini semua orang dapat menggunakannya dengan bebas. RSA merupakan algoritma kriptografi asimetrik yang paling mudah untuk diimplementasikan dan dimengerti.

Algoritma kriptografi RSA merupakan algoritma yang termasuk dalam kategori algoritma asimetri atau algoritma kunci publik. Algoritma kriptografi RSA didesain sesuai fungsinya sehingga kunci yang digunakan untuk enkripsi berbeda dari kunci yang digunakan untuk dekripsi. Algoritma RSA disebut kunci publik karena kunci enkripsi dapat dibuat publik yang berarti semua orang boleh mengetahuinya, namun hanya orang tertentu (si penerima pesan sekaligus pemilik kunci dekripsi yang merupakan pasangan kunci publik) yang dapat melakukan dekripsi terhadap pesan tersebut. Keamanan algoritma RSA didasarkan pada sulitnya memfaktorkan bilangan besar menjadi faktor-faktor primanya [2].

Correted Block TEA atau sering disebut dengan XXTEA adalah cipher block yang dirancang untuk memperbaiki kelemahan dalam Block TEA sebelumnya (XTEA). *Corrected Block TEA* adalah operasi jaringan Feistel pada blok yang terdiri dari setidaknya dua buah kata 32 *bits* dan menggunakan kunci 128 *bits* Algoritma *Corrected Block TEA* didesain oleh Roger Needham dan David Wheeler dari Laboratorium Komputer *Cambridge*, Algoritma ini dipublikasikan dalam laporan teknis pada bulan Oktober 1998. *Corrected Block TEA* juga merupakan sebuah algoritma enkripsi efektif yang mirip dengan *DES* yang dapat digunakan untuk aplikasi web yang membutuhkan keamanan. Ketika menggunakan algoritma ini, sebuah perubahan dari data asal akan mengubah sekitar setengah dari data hasil tanpa meninggalkan jejak dimana perubahan berasal [3].

Corrected Block TEA beroperasi pada blok yang berukuran tetap yang merupakan kelipatan 32 *bits* dengan ukuran minimal 64 *bits*. Jumlah dari putaran lengkap bergantung pada ukuran blok, tetapi terdapat minimal 6 (bertambah terus hingga 32 untuk ukuran blok yang lebih kecil). Algoritma ini menggunakan lebih banyak fungsi pengacakan yang menggunakan kedua blok tetangganya dalam pemrosesan setiap kata dalam blok [3].

Penelitian sebelumnya yang berjudul "Perancangan Keamanan Informasi Sistem Pemungutan Suara Elektronik (e-Voting) Menggunakan Kombinasi Algoritma AES dan RSA". Algoritma kriptografi RSA didesain sesuai fungsinya sehingga kunci yang digunakan untuk enkripsi berbeda dari kunci yang digunakan untuk dekripsi. Algoritma RSA disebut kunci publik karena kunci enkripsi dapat dibuat publik yang berarti semua orang boleh mengetahuinya, namun hanya orang tertentu (si penerima pesan sekaligus pemilik kunci dekripsi yang merupakan pasangan kunci publik) yang dapat melakukan dekripsi terhadap pesan tersebut [4].

Penelitian sebelumnya berjudul "Implementasi Algorithma RSA (Rivest Shamir Adleman) dalam Sistem Enkripsi File dan Pengamanan Folder". Karena panjang kunci dalam bit menentukan tingkat kesulitan dan kerumitan penguraian penyandian. Dengan semakin panjang bit maka semakin sukar untuk dipecahkan karena sulitnya memfaktorkan dua bilangan prima acak yang telah dipilih untuk membangkitkan kunci [5].

Penelitian sebelumnya Denver berjudul "Pengamanan Komunikasi Suara Melalui Internet Pada Telepon Seluler dengan Algoritma TEA pada *Platform* Android**".** Tiny Encryption Algorithm (TEA). Algoritma TEA merupakan algoritma cipherblock dengan ciri khas berupa kode yang tidak panjang tetapi kuat, dan cepat. Kekuatan algoritma ini sekompleks Data Encryption Standard (DES) dan kesederhanaannya membuat algoritma ini dapat ditranslasikan pada berbagai bahasa serta digunakan pada berbagai macam alat komputasi [6].

Penelitian sebelumnya berjudul "Perancangan *Secure Login* Website Menggunakan Algoritma Enkripsi XXTEA**".** XXTEA beroperasi pada blok yang

berukuran tetap yang merupakan kelipatan 32 bits dengan ukuran minimal 64 bits. Jumlah dari putaran lengkap bergantung pada ukuran blok, tetapi terdapat minimal 6 (bertambah terus hingga 32 untuk ukuran blok yang lebih kecil). Algoritma ini menggunakan lebih banyak fungsi pengacakan yang menggunakan kedua blok tetangganya dalam pemrosesan setiap kata dalam blok [7].

Penelitian sebelumnya berjudul "Keamanan Pertukaran Kunci Kriptografi dengan Algoritma Hybrid : Diffie-Hellman dan RSA". Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci pribadi. Selama pemfaktoran bilangan besar menjadi faktor-faktor prima belum ditemukan algoritma yang mangkus, maka selama itu pula keamanan algoritma RSA tetap terjamin [8][9].

Penelitian sebelumnya berjudul **"**Cryptanalysis of XXTEA". Correted Block TEA (atau biasa disebut dengan XXTEA) adalah cipher block yang dirancang untuk memperbaiki kelemahan dalam Block TEA sebelumnya (XTEA). XXTEA adalah operasi jaringan Feistel pada blok yang terdiri dari setidaknya dua buah kata 32 -bit dan menggunakan kunci 128-bit [10].

Penelitian sebelumnya dalam Jurnal berjudul "Comparative Analysis of DES, AES, RSA Encryption Algorithms". RSA menggunakan sepasang kunci dari bilangan angka yang sangat besar, n, yang merupakan hasil dari dua buah bilangan prima yang dipilih dengan cara yang khusus[11[.

2. METODE PENELITIAN

Perancangan aplikasi ini menggunakan metode peneltian *waterfall* , di mana langkah-langkahmya sebagai berilut:

2.1 *Requirement Definition*

Proses ini digunakan untuk pencarian kebutuhan diintensifkan dan difokuskan pada program. Untuk mengetahui sifat dari program yang dibuat, maka para pembuat program harus mengerti tentang informasi program. Proses penelitian program ini dilakukan di PT. Beetlebox Indonesia.

2.2 **Sistem dan *Design* Program**

Proses mengubah kebutuhan-kebutuhan tentang *requirement definition* menjadi reprentasi ke dalam bentuk *blueprint* atau *mockup* program sebelum pemrograman dimulai. Desain program harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap requirment definition. Program yang digunakan untuk design disini adalah *Balsamiq Mockups*.

Skema algoritma kunci publik sandi RSA terdiri dari tiga proses yaitu, proses pembentukan kunci, proses enkripsi, dan proses dekripsi [12]:

1) Proses Pembentukan Kunci
   1. Memilih dua bilangan prima yang diberi simbol sebagai *p* dan *q*.
   2. Menghitung nilai $n=p.q$ ($n \neq p$,karena jika $n=p$,maka nilai n=p2 dan akan mudah mendapatkan nilai n).
   3. Hitung $\varphi(n) = (p-1)(q-1)$.
   4. Memilih kunci publik *e* yang relatif prima terhadap $\varphi(n)$
   5. Bangkitkan kunci privat dengan persamaan $d.e \equiv 1 \ (mod\ \varphi(n))$, dimana $1<d<\varphi(n)$

   Proses diatas menghasilkan dua buah kunci, yaitu:
   1. Kata kunci publik, pasangan *e* dan *n*
   2. Kata kunci private, pasangan *d* dan *n*
2) Proses Enkripsi
   1. Ambil kunci publik penerima pesan (*e*) ,dan modulus (*n*).
   2. *Plaintext* dinyatakan dengan blok-blok m_1, m_2, .., sedemikian sehingga setiap blok merepresentasikan nilai [0, n-1].
   3. Setiap blok m_i dienkripsikan menjadi blok c_i dengan rumus $C_i = m_i^e \ mod\ n$
3) Proses Dekripsi

   Proses dekripsi hampir sama dengan proses enkripsi, tapi menggunakan pasangan private key
   1. Ambil kunci publik penerima pesan (*d*) ,dan modulus (*n*).
   2. *Plaintext* dinyatakan dengan blok-blok m_1, m_2, .., sedemikian sehingga setiap blok merepresentasikan nilai [0, n-1].
   3. Setiap blok m_i dienkripsikan menjadi blok c_i dengan rumus $C_i = m_i^d \ mod\ n$

Proses pengacakan yang dilakukan dalam satu iterasi *Corrected Block TEA* adalah sebagai berikut:

1. Algoritma akan mengacak blok ke-*r* dari *plaintext*.
2. Proses akan mengambil x_{r-1}, x_{r+1}, *DELTA*, dan kata kunci sebagai input.
3. Pengacakan pertama: $x_{r-1}<<2$ di-*XOR*-kan dengan $x_{r+1}>>5$.
4. Pengacakan kedua: $x_{r-1}>>3$ di-*XOR*-kan dengan $x_{r+1}<<4$.
5. Hasil yang didapat dari tahap 3 dan 4 ditambahkan.
6. Pengacakan ketiga: x_{r-1} di-*XOR*-kan dengan *D* yang merupakan perkalian antara konstanta *DELTA* yang bernilai 0x9E3779B9 dengan jumlah iterasi pertama yang telah dilakukan.
7. Pengacakan keempat: x_{r+1} di-*XOR*-kan dengan salah satu blok kata kunci, yaitu blok ke-(*r XOR D*>>2).
8. Hasil yang didapat dari tahap ke 6 dan 7 ditambahkan.

9. Hasil yang didapat dari tahap 5 dan 8 di-*XOR*-kan.
10. Hasil yang didapat pada tahap 9 ditambahkan ke blok plaintext ke-*r*.

Corrected Block TEA tidak memiliki batas ukuran blok, *Corrected Block TEA* dapat digunakan untuk mengenkripsi satu buah pesan utuh tanpa memerlukan mode operasi cipher. Meski *Corrected Block TEA* dapat mengenkripsi keseluruhan pesan atau file sekaligus, pada implementasinya *Corrected Block TEA* dapat dioperasikan dengan mode operasi untuk *file* yang berukuran sangat besar sehingga tidak bisa dibaca ke dalam memory sekaligus.

2.3 *Implementation* dan *Unit Testing*

Agar dapat dimengerti oleh mesin, dalam hal ini adalah komputer, maka desain sebelumnya harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh komputer, yaitu sebuah program komputer yang menggunakan bahasa pemrograman Javascript berjalan diatas runtime Chrome Javascript, dan untuk tampilan menggunakan HTML dan CSS.

2.4 *Integration* dan Sistem Testing

Fungsi-fungsi program diujicobakan agar program bebas dari error atau bug dan hasilnya sesuai dengan kebutuhan yang didefinisikan sebelumnya. Dalam hal ini sistem dicoba menggunakan NodeJS 0.12 pada Mac OS X 10.9.5.

2.5 Operasi dan Maintenance

Pemeliharaan program sangat diperlukan dalam membangun sebuah sistem. Salah satunya adalah pengembangan program yang bisa dilakukan secara bertahap sesuai dengan kebutuhan yang

.

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Pada kriptografi kata kunci rahasia merupakan hal penting selain kekuatan alogritma kriptografi itu sendiri. Namun sering kali kata kunci kurang mendapat perhatian dalam keamanannya, seperti pada beberapa kasus dibutuhkan untuk membagi kata kunci tersebut kepada orang yang berhak selain si pembuat data. Pada proses pembagian/pertukaran kata kunci tersebut sering kali terdapat kecerobohan yang dilakukan oleh pengguna dikarekan tidak semua pengguna mengetahui potensi celah keamanan yang mungkin terjadi, seperti melakukan pertukaran kata kunci pada media komunikasi yang kurang aman, seperti chat public atau email pribadi, juga kata kunci yang dikirimkan berupa plaintext, sehingga penerima yang berhak maupun pelaku kejahatan dapat dengan mudah membaca dan mengetahuinya.

Pada proses enkripsi yang akan digunakan, dibutuhkan operasi aritmatika bilangan sebesar 16 digit, dikarenakan key size yang akan digunakan sebesar 128 bits. Sedangkan pada komputer yang digunakan masyarakat saat ini menggunakan arsitektur 32 bits dan 64 bits. Sehingga tidak memungkinkan melakukan operasi perhitungan aritmatika dengan teknik biasa digunakan.

Pada sistem yang akan dibangun memerlukan perubahan ekstensi file, yang bertujuan untuk memudahkan pengguna dalam pengenalan file yang sudah dienkripsi. Namun hal tersebut menimbulkan masalah pada sistem, dikarenakan sistem harus mengetahui ekstensi file sebelum dilakukan proses enkripsi.

3.2 Penyelesaian Masalah

Agar pengguna dapat mendistribusikan kata kunci dengan aman, walaupun melalui media komunikasi yang tidak aman, maka diperlukan algoritma tambahan untuk digunakan dalam distribusi kata kunci yang bersifat asimetris, dalam hal ini dipilih menggunakan algoritma RSA.

Sedangkan untuk melakukan proses perhitungan aritmatika sebesar 128 bits, menggunakan fungsi bignum yang sudah terdapat pada library OpenSSL. Kelebihan OpenSSL adalah sudah menjadi bagian dari *runtime engine* akan yang digunakan, Node.js.

Agar ektensi file yang telah dienkripsi dapat dikembalikan pada saat dekripsi, maka setelah proses enkripsi selesai, sistem akan menambahkan informasi ektensi yang digunakan pada bagian awal file yang telah dienkripsi.

3.3 Rancangan Sistem

Metode kriptografi yang dilakukan dalam proses enkripsi dan dekripsi menggunakan dua metode, yaitu *RSA* dan *Corrected Block TEA*. *RSA* digunakan untuk melakukan proses enkripsi pada kata kunci yang akan digunakan pada *Corrected Block TEA* untuk melakukan proses enkripsi pada file.

3.3.1 Registrasi

Proses registrasi akan tampil jika data pengguna, berupa nama, email dan *key* (*public key* dan *private key*) yang akan digunakan oleh kriptografi *RSA* belum tersimpan pada sistem. Pada proses registrasi pengguna diharuskan memberikan data berupa nama, email dan *password*. Setelah data tersebut diberikan, sistem akan membuat *public key* dan *private key* dan menyimpannya bersamaan dengan data yang telah diberikan oleh pengguna. Namun untuk meningkatkan keamanan, sebelum disimpan *private key* diamakan dengan melakukan enkripsi menggunakan metode *Corrected Block TEA* dengan kata kunci sesuai dengan *password* yang telah diberikan pengguna pada saat registrasi, dan sebelum dienkripsi, *private key* di-*padding* dengan *signature* yang sudah ditentukan, agar sistem pada saat proses dekripsi sistem dapat megetahui apakah proses tersebut berhasil atau tidak.

3.3.2 *Login*

Apabila data pengguna berupa nama, email dan *key* (*public key* dan *private key*) sudah tersimpan pada sistem, maka ketika user pertama kali menjalankan

program akan tampil form login untuk memasukan *password* agar aplikasi tersebut dapat digunakan, *password* yang dimasukan harus sama dengan *password* yang dimasukan pada saat pertama kali daftar.
Setelah pengguna memasukan *password*-nya, sistem akan menggunakannya sebagai kata kunci untuk melakukan proses dekripsi pada data *private key* yang telah terenkripsi pada saat registrasi. Apabila hasil dekripsi mempunyai karakter pada baris pertama dan terakhir yang sesuai dengan yang telah ditentukan oleh sistem, hal itu menandakan proses dekripsi berhasil, dan *private key* yang telah didekripsi akan disimpan pada memory, yang dapat digunakan oleh sistem apabila pengguna ingin melakukan proses dekripsi.

3.3.3 Enkripsi

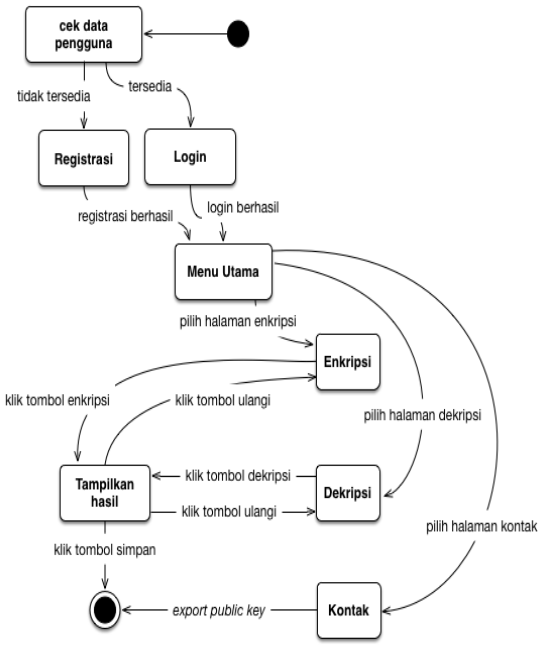
Proses enkripsi dapat dilakukan setelah pengguna berhasil melakukan registrasi atau *login*. Pada proses ini pengguna diharuskan memasukan penerima dan *file* yang akan dienkripsi, sebelumnya data penerima sudah harus tersimpan pada sistem, jika belum pengguna dapat menambahkannya dengan melakukan impor file *bundled key* dengan ekstensi file ".ppk" ke dalam sistem yang sebelumnya sudah diekspor dan diberikan ke pengguna oleh para calon penerima.
Proses enkripsi diawali dengan membuat *string* sebanyak 16 karakter angka dan huruf, lalu disimpan pada *memory*, agar dapat digunakan pada proses selanjutnya. Lalu sistem akan melakukan proses enkripsi pada *key* tersebut menggunakan metode *RSA* berdasarkan *public key* dan *modulus* masing-masing penerima, dah hasil enkripsi tersebut akan digabungkan dengan *id* penerima, yang dipisahkan dengan karaktar titik-koma (;), hasil enkripsi dan penggabungan tersebut disimpan pada memory. Setelah itu sistem akan melakukan proses enkripsi pada *file* menggunakan metode *Corrected Block TEA* dengan *key* yang sebelumnya sudah dibuat secara acak dan sudah tersimpan pada *memory*, dan hasil enkripsi pada file akan disimpan pada *memory*.
Selanjutnya sistem akan membuat file hasil enkripsi tersebut yang diawali dengan *signature file*, diikuti dengan informasi ekstensi *file* asli yang telah dienkripsi, lalu diikuti dengan data ekripsi *password*, lalu terakhir diikuti dengan data hasil enkripsi *file*. Setelah itu sistem akan membuka kotak dialog yang meminta pengguna untuk mementukan lokasi *file* hasil enkripsi yang akan disimpan.

3.3.4 Dekripsi

Proses dekrispi dapat dilakukan setelah pengguna berhasil melakukan registrasi atau login. Pada proses ini pengguna diharuskan memasukan file yang akan didekripsi. Proses dekripsi diawali dengan melakukan validasi file, dengan cara mengecek 3 byte awal, apakah memiliki signature yang sama yang sudah ditentukan oleh sistem, jika ya berari file yang dimasukan merupakan file hasil enkripsi menggunakan sistem yang sama. Selanjutnya sistem akan membaca informasi penerima berupa id dan key yang telah tersimpan pada bagian awal file, dan sistem melakukan validasi apakah pengguna merupakan penerima yang berhak, jika ya, informasi key pada pengguna tersebut disimpan pada memory. Selanjutnya sistem akan melakukan proses dekripsi key tersebut menggunakan metode RSA dengan private key dan modulus pengguna yang telah tersimpan pada sistem, hasil dekrispi tersebut disimpan pada memory. Selanjutnya sistem melakukan proses dekripsi pada file asli menggunakan metode Corrected Block TEA dengan key yang telah didekripsi sebelumnya. Lalu sistem akan membuka kotak dialog untuk menyimpan hasil dekripsi tersebut dengan ekstensi file sama dengan ekstensi file asli, yang informasi tersebut terdapat pada bagian awal file yang dimasukan.

3.3.5 Kontak

Kontak berfungsi sebagai tempat untuk mengatur dan menyimpan data penerima yang akan digunakan pada proses enkripsi. Pada kontak pengguna dapat melakukan import dan export data penerima, dan melakukan penghapusan data penerima. Selain itu pada bagian kontak juga dapat melakukan ekspor bundled key miliknya.
Ketika program digunakan, secara gambar besar proses progam yang dibuat memiliki statechart yang dapat dilihat pada gambar 1.



Gambar 1. Statechart Aplikasi

Berikut ini dekripsi tiap-tiap state yang terdapat state diagaram aplikasi pada gambar 1.

a. State : Cek data pengguna
Deskripsi : Pada state ini aplikasi akan melakukan proses pengecekan pada data pengguna, apakah sudah tersedia atau tidak

b. State : Registrasi
Deskripsi : Pada state ini pengguna melakukan proses entri data pengguna

c. State : Login
Deskripsi : Pada state ini pengguna melkaukan proses login dengan memasukan password

d. State : Menu utama
Deskripsi : Pada state ini pengguna memilih menu untuk mengakses halaman yang lainnya

e. State : Enkripsi
Deskripsi : Pada sate ini pengguna dapat melakukan proses enkripsi dengan mamasukan file dan penerima

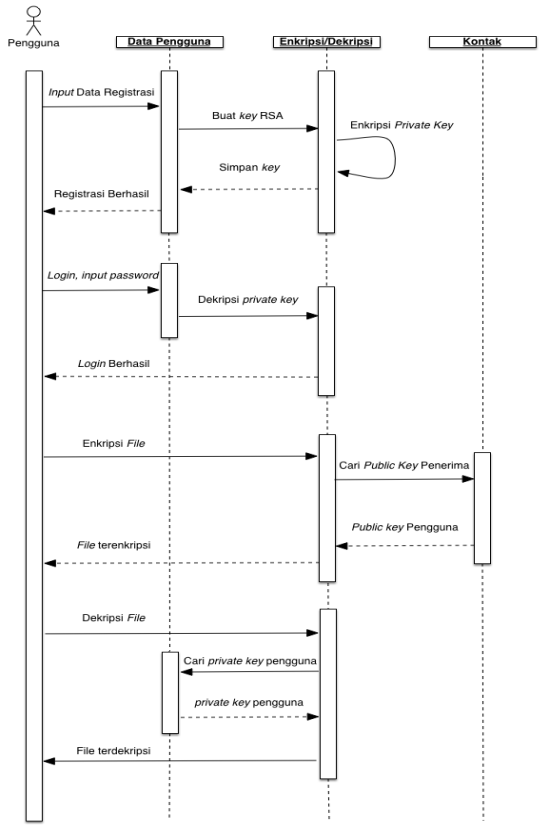
f. State : Deskripsi
Deskripsi : Pada state ini pengguna dapat melakukan proses dekripsi dengan masukan file yang terenkripsi

g. State : Kontak
Deskripsi : Pada state ini pengguna dapat melakukan porses *export public key* miliknya dan *import public key* calon penerima

h. State : Tampilkan Hasil
Deskripsi : Pada state ini aplikasi menampilkan informasi hasil dari proses enkripsi atau dekripsi, dan pengguna dapat melakukan penyimpan file hasil

3.4 Sequence Diagram

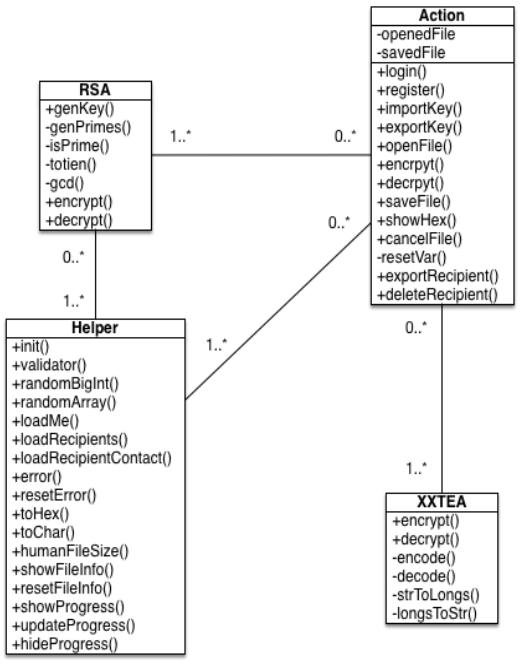
Sequence diagram aplikasi yang akan dibuat dapat dilihat pada gambar 2.



Gambar 2. Sequence Diagram Aplikasi

3.5 Class Diagram

Class diagram aplikasi yang akan dibuat dapat dilihat pada gambar 3.



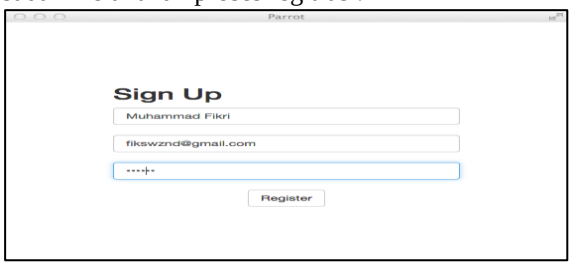
Gambar 3. Class Diagram Aplikasi

3.6 Pengoperasian program

Pengoperasian program ini menggunakan Node.JS jika tidak ada maka harus di install terlebih dahulu.

a. Tampilan Layar Registrasi

Tampilan layar registrasi akan tampil pada saat pengguna menjalankan aplikasi pertama kali dan belum ada data pengguna belum tersimpan pada aplikasi, dapat dilihat pada gambar 4. Sehingga halaman ini tidak akan tampil lagi jika pengguna sudah melakukan proses regitrasi.



Gambar 4. Tampilan Layar Registrasi

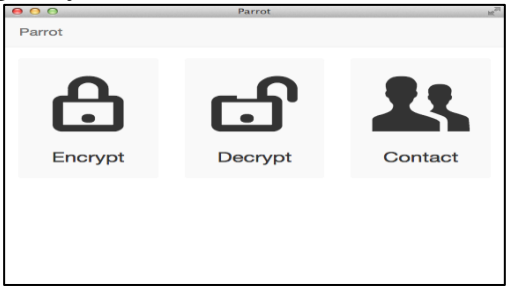
b. Tampilan Layar *Login*

Tampilan layar *login*, dapat dilihat pada gambar 5, akan tampil pada saat pengguna menjalankan aplikasi pertama kali dan data pengguna sudah tersimpan pada aplikasi melalui proses registrasi, dapat dilihat pada gambar 4.

Gambar 5. Tampilan Layar *Login*

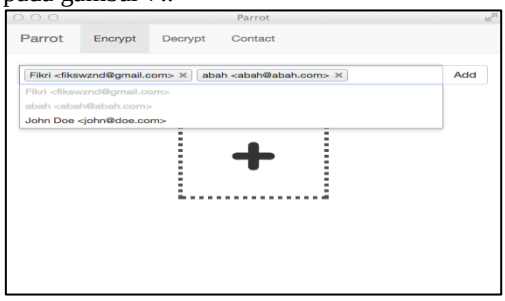
c. Tampilan Layar Menu

Tampilan Layar Menu, dapat dilihat pada gambar 6, akan tampil pada saat pengguna berhasil melakukan proses registrasi, dapat dilihat pada gambar 4 atau *login*, dapat dilihat pada gambar 5. Pada halaman ini pengguna diberi pilihan untuk menuju halaman yang diingikannya.



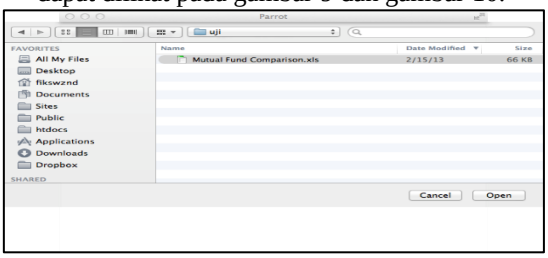
Gambar 6. Tampilan Layar Menu

1) Jalankan aplikasi dan lakukan proses *login*, pada layar *login*, dapat dilihat pada gambar 4. atau registrasi, pada layar registrasi, dapat dilihat pada gambar 5.
2) Pilih menu enkripsi, pada layar menu, dapat dilihat pada gambar 6..
3) Cari dan pilih penerima *file* yang akan dienkripsi, pada layar enkripsi, dapat dilihat pada gambar 7..

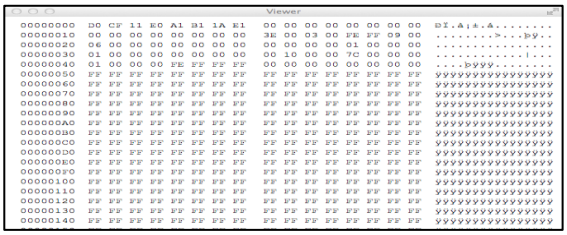


Gambar 7. Tampilan Uji Coba Memilih Penerima

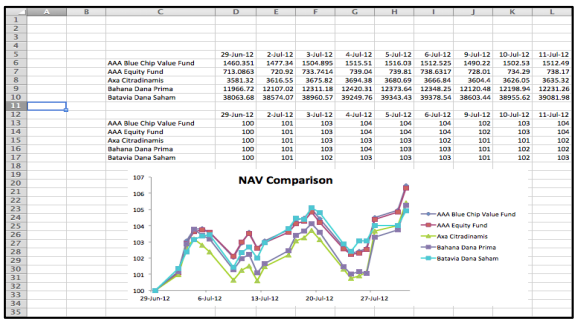
4) Buka file yang akan dienkripsi dengan mengklik tanda "+", pada bagian tengah layar enkripsi, dapat dilihat pada gambar 7, lalu akan muncul *file browser*, dapat dilihat pada gambar 8, dalam hal ini akan menggunakan *file format MS. Excel 97* dengan nama *file* "*Mutual Fund Comparison.xls*", tampilan isi file tersebut dapat dilihat pada gambar 9 dan gambar 10.



Gambar 8. Tampilan Uji Coba Memilih *File*

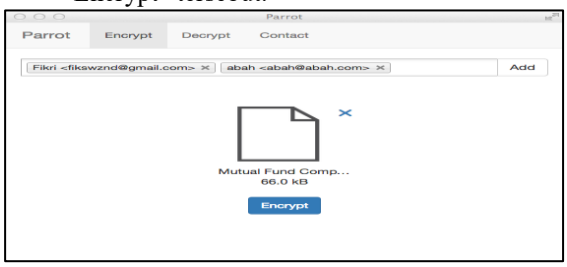


Gambar 9. Tampilan Uji Coba Data yang Akan pada *Viewer* Aplikasi



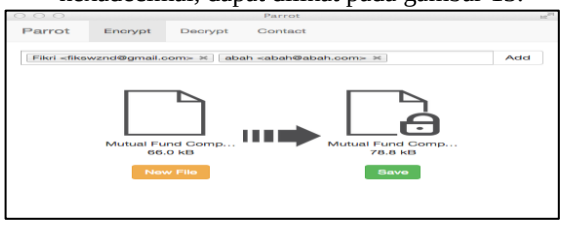
Gambar 10. Tampilan Uji Coba Data yang Akan Dienkripsi pada *MS. Excel*

5) Informasi file dan tombol "*Encrypt*" akan tampil jika file telah berhasil di-*input*, dapat dilihat pada gambar 11. Lalu klik tombol "Encrypt" tersebut.

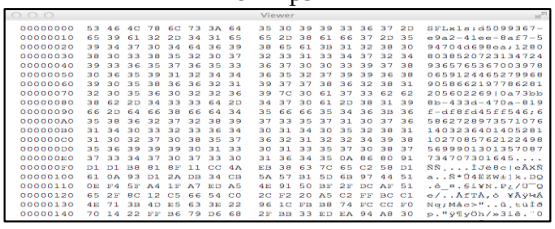


Gambar 11. Tampilan Uji Coba *File* sudah di-*input* untuk dienkripsi

6) Setelah proses enkripsi selesai, akan tampil informasi file hasil enkripsi, berserta tombol "*Save*", dapat dilihat pada gambar 12. Lalu klik gambar file pada halaman tersebut untuk melihat tampilan isi file dalam format hexadecimal, dapat dilihat pada gambar 13.

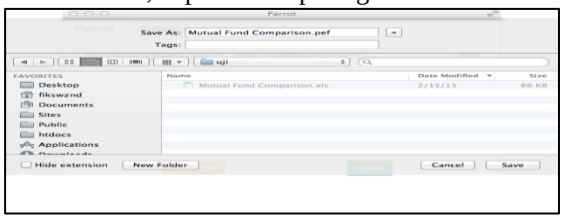


Gambar 12. Tampilan Uji Coba *File* sudah diproses enkripsi



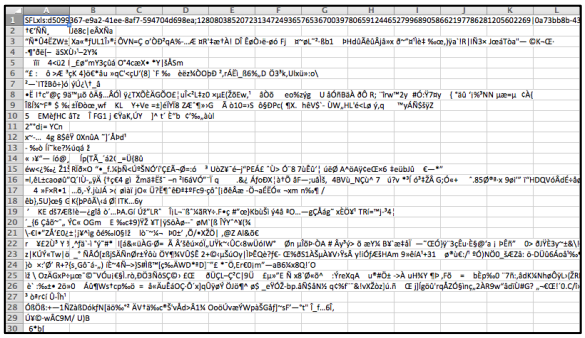
Gambar 13. Tampilan Data Uji Coba pada *Viewer* Aplikasi dalam format *Hexadecimal*

7) Klik tombol "*Save*", pada halaman hasil enkripsi, dapat dilihat pada gambar 12, lalu akan muncul *file browser* untuk menyimpan *file* tersebut, dapat dilihat pada gambar 14.



Gambar 14. Tampilan Uji Coba Menyimpan *File* Hasil Enkripsi

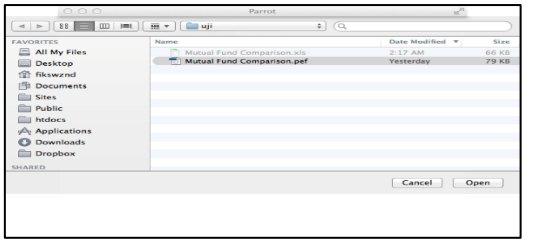
Jika *file* yang sudah dienkripsi dibuka menggunakan *MS. Excel*, akan tampil karakter yang tidak dapat dibaca dapat dilihat pada gambar 16, yang berbeda dengan *file* sebelum dienkripsi, dapat dilihat pada gambar 10.



Gambar 16. Tampilan Uji Coba Data Hasil Enkripsi pada *MS.Excel*

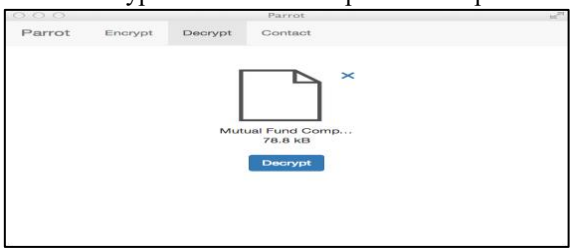
3.7 Pengujian Dekripsi

1) Jalankan aplikasi dan lakukan proses *login*, pada layar *login*, dapat dilihat pada gambar 5. atau registrasi, pada layar registrasi, dapat dilihat pada gambar 4.
2) Pilih menu dekripsi, pada layar menu, dapat dilihat pada gambar 6.
3) Pilih *file* yang akan didekripsi dengan klik tanda "+" pada bagian tengah layar dekripsi, dapat dilihat pada gambar 12.
4) Lalu akan muncul tampilan *file browser* untuk membuka file yang akan didekripsi, dapat dilihat pada gambar 17. Pada pengujian ini menggunakan *file* yang telah dienkripsi sebelumnya pada saat pengujian enkripsi, yaitu *file* dengan nama "*Mutual Fund Comparison.pef*"



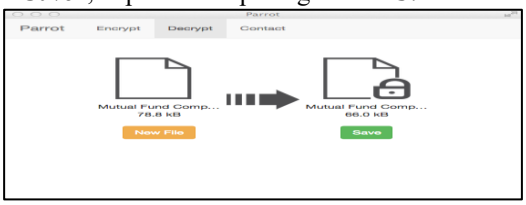
Gambar 17. Tampilan Uji Coba Membuka *File* yang Akan Didekripsi

5) Informasi *file* yang akan didekripsi dan tombol "Decrypt" tampil jika *file* berhasil di-*input*, dapat dilihat pada gambar 18. Lalu klik tombol "Decrypt" untuk memulai proses dekripsi



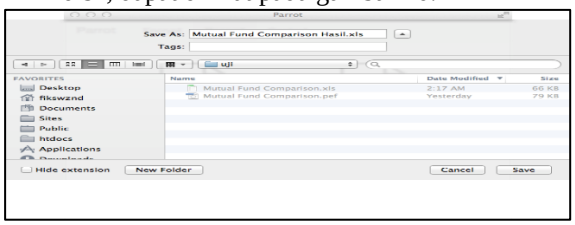
Gambar 18. Tampilan Uji Coba *File* Berhasil di-*input* untuk didekripsi

6) Setelah proses dekripsi selesai akan tampil informas *file* hasil dekripsi, beserta tombol "*Save*", dapat dilihat pada gambar 19.



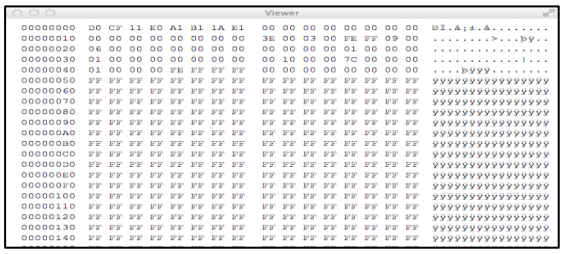
Gambar 19. Tampilan Uji Coba *File* Sudah Diproses Dekripsi

7) Klik tombol "*Save*", maka akan muncul *file browser* untuk menyimpan hasil dekripsi pada disk, dapat dilihat pada gambar 20.

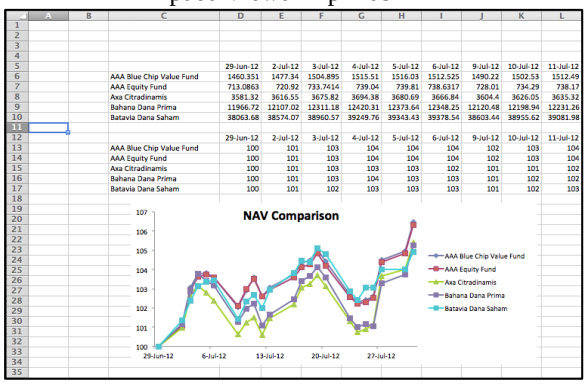


Gambar 20. Tampilan Uji Coba Menyimpan *File* Hasil Dekripsi

Pada file yang telah diproses dekripsi dan disimpan pada *disk*, dilakukan perbandingan dengan file asli sebelum dilakukan proses enkripsi menggunakan *viewer* pada aplikasi, dapat dilihat pada gambar 21 dan dibuka dengan MS. Excel, pada gambar 22 hasilnya identik, tidak ada perubahan. File sebelum enkripsi dapat dilihat pada gambar 13 dan 14.



Gambar 21. Tampilan Uji Coba File Hasil Dekripsi pada *Viewer* Aplikasi



Gambar 22. Tampilan Uji Coba File Hasil Dekripsi pada *MS.Excel*

3.8 Hasil Pengujian

Berdasarkan hasil pengujian yang dilakukan menggunakan 10 file text dan 10 file binary, maka didapatkan hasil seperti tabel 1 dan tabel 2.

a. Pengujian Enkripsi

Tabel 1. Tabel Hasil Pengujian Enkripsi

No	Sebelum Enkripsi		Sesudah Enkripsi		Persentase Perubahan	Waktu
	Nama File	Ukuran File	Nama File	Ukuran File		
Input File Binary						
1	Mutual Fund Comparison.xls	66 kB	Mutual Fund Comparison.pef	78.7 kB	16.14%	326 ms
2	beetlebox_credential.pdf	1433 kB	beetlebox_credential.pef	2135 kB	32.88%	2144 ms
3	Archive.zip	376 kB	Archive.pef	566 kB	33.57%	746 ms
4	overlay.png	323.9 kB	overlay.pef	491.1 kB	34.05%	726 ms
5	Jr Commercial.mov	6435 kB	Jr Commercial.pef	9659 kB	33.38%	11516 ms
6	Shop Directories.xls	57.9 kB	Shop Directories.pef	65.7 kB	11.87%	270 ms
7	loop.mp4	48.7 kB	loop.pef	70.1 kB	30.53%	279 ms
8	beacon.keynote	1299 kB	beacon.pef	1860 kB	30.16%	2019 ms
9	PIM Merchants.xls	87.6 kB	PIM Merchants.pef	99.5 kB	11.96%	363 ms
10	Beetlebox Timeline.numbers	95 kB	Beetlebox Timeline.pef	142.3 kB	33.24%	341 ms
	Rata-Rata				**26.78%**	**1873 ms**
Input File Text						
1	interest.html	376.4 kB	interest.pef	376.7 kB	0.08%	375 ms
2	PE-MKTBL-02C.txt	0.925 kB	PE-MKTBL-02C.pef	1.2 kB	22.92%	212 ms
3	Profile 1.json	1576.2 kB	Profile 1.pef	1596.4 kB	1.27%	1701 ms
4	config.log	12.4 kB	config.pef	12.7 kB	2.36%	218 ms
5	train.txt	2842.1 kB	train.pef	2842.3 kB	0.01%	1476 ms
6	rural.txt	1808 kB	rural.pef	1808.2 kB	0.01%	911 ms
7	masima.sql	29279.8 kB	masima.pef	29333.5 kB	0.18%	11295 ms
8	EastAsianWidth.txt	797.2 kB	EastAsianWidth.pef	797.5 kB	0.04%	518 ms
9	LineBreak.txt	854.9 kB	LineBreak.pef	855.1 kB	0.02%	609 ms
10	register_user_20080223.txt	2573.2 kB	register_user_20080223.pef	2575.3 kB	0.08%	1502ms
	Rata-Rata				**2.70%**	**1882 ms**

b. Pengujian Dekripsi

Tabel 2. Tabel Hasil Pengujian Dekripsi

No	Sebelum Dekripsi		Sesudah Dekripsi		Persentase Perubahan	Waktu
	Nama File	Ukuran File	Nama File	Ukuran File		
	Output File Binary					
1	Mutual Fund Comparison.pef	78.7 kB	Mutual Fund Comparison.xls	66 kB	16.14%	258 ms
2	beetlebox_credential.pef	2135 kB	beetlebox_credential.pdf	1433 kB	32.88%	1425 ms
3	Archive.pef	566 kB	Archive.zip	376 kB	33.57%	417 ms
4	overlay.pef	491.1 kB	overlay.png	323.9 kB	34.05%	419 ms
5	Jr Commercial.pef	9659 kB	Jr Commercial.mov	6435 kB	33.38%	5192 ms
6	Shop Directories.pef	65.7 kB	Shop Directories.xls	57.9 kB	11.87%	224 ms
7	loop.pef	70.1 kB	loop.mp4	48.7 kB	30.53%	227 ms
8	beacon.pef	1860 kB	beacon.keynote	1299 kB	30.16%	1068 ms
9	PIM Merchants.pef	99.5 kB	PIM Merchants.xls	87.6 kB	11.96%	232 ms
10	Beetlebox Timeline.pef	142.3 kB	Beetlebox Timeline.numbers	95 kB	33.24%	243 ms
	Rata-Rata				**26.78%**	**970.5 ms**
	Output File Text					
1	interest.pef	376.7 kB	interest.html	376.4 kB	0.08%	314 ms
2	PE-MKTBL-02C.pef	1.2 kB	PE-MKTBL-02C.txt	0.925 kB	22.92%	208 ms
3	Profile 1.pef	1596.4 kB	Profile 1.json	1576.2 kB	1.27%	697 ms
4	config.pef	12.7 kB	config.log	12.4 kB	2.36%	215 ms
5	train.pef	2842.3 kB	train.txt	2842.1 kB	0.01%	1177 ms
6	rural.pef	1808.2 kB	rural.txt	1808 kB	0.01%	776 ms
7	masima.pef	29333.5 kB	masima.sql	29279.8 kB	0.18%	14905 ms
8	EastAsianWidth.pef	797.5 kB	EastAsianWidth.txt	797.2 kB	0.04%	458 ms
9	LineBreak.pef	855.1 kB	LineBreak.txt	854.9 kB	0.02%	474 ms
10	register_user_20080223.pef	2575.3 kB	register_user_20080223.txt	2573.2 kB	0.08%	1252 ms
	Rata-Rata				**2.70%**	**2047.6 ms**

3.9 Evaluasi Program

a. Kelebihan Program
   1) Program dapat melakukan proses enkripsi rata-rata untuk file binary membutuhkan waktu proses selama 1873 ms dengan prosentase perubahan 26,78%, untuk file text waktu proses selama 1882 ms dengan prosentase perubahan 2,70% dan proses dekripsi rata-rata untuk file binary membutuhkan waktu proses selama 970,5 ms dengan prosentase perubahan 26,78%, untuk file text waktu proses selama 2047,6 ms dengan prosentase perubahan 2,70% tanpa memerlukan pertukaran kata kunci.
   2) Program dapat berjalan pada *multiplatform*, Windows dan Machintosh
   3) Tampilan program sederhana dan jelas, sehingga pengguna dapat menggunakannya dengan mudah.

b. Kekurangan Program
   1) Ukuran file yang dihasilkan pada proses enkripsi lebih besar dari file asli.
   2) Proses enkripsi cukup rumit diawal, dikarena penerima diharuskan mengirimkan *bundled key* terlebih dahulu.

4. KESIMPULAN

Kesimpulan dapat diambil sebagai berikut: penggabungan metode kriptografi asimetris dan simetris dapat dilakukan dengan cara melakukan enkripsi dengan metode asimetris pada kata kunci yang digunakan pada metode simetris, dengan penggabungan kriptografi asimetris dan simetris, pengiriman file kebanyak penerima tanpa melakukan distribusi kata kunci dapat dilakukan, ukuran *file* yang akan dienkripsi mempengaruhi waktu dalam proses enkripsi, semakin besar *file* yang dienkripsi semakin besar pula waktu yang dibutuhkan, peningkatan ukuran *file binary* lebih banyak dibandingkan *file text* pada proses enkripsi, waktu yang dibutuhkan *file binary* lebih singkat dibandingkan *file text* pada proses enkripsi maupun dekripsi.dan waktu yang dibutuhkan *file binary* pada proses dekripsi lebih singkat dibandingkan pada proses enkripsi, namun terjadi sebaliknya pada *file text*, lebih singkat proses enkripsi dibandingkan file dekripsi. Program dapat melakukan proses enkripsi rata-rata untuk file binary membutuhkan waktu proses selama 1873 ms dengan prosentase perubahan 26,78%, untuk file text waktu proses selama 1882 ms dengan prosentase perubahan 2,70% dan proses dekripsi rata-rata untuk file binary membutuhkan waktu proses selama 970,5 ms dengan prosentase perubahan 26,78%, untuk file text waktu proses selama 2047,6 ms dengan prosentase perubahan 2,70% .

Aplikasi ini masih belum sempurna dan masih perlu perbaikan-perbaikan. Beberapa saran yang dapat diberikan untuk meningkatkan kualitas dari aplikasi ini antara lain: ukuran file hasil enkripsi dapat diperkecil dengan menerapkan proses kompresi dan menambahkan layanan *client-server* pada pertukaran *bundled key*, sehingga jika pengguna memiliki akses internet dapat dimudahkan dengan langsung mencari nama penerima tanpa perlu meminta penerima melakukan ekspor *bundled key* terlebih dahulu.

5. DAFTAR PUSTAKA

[1] Ireland, David, *RSA Algorithm*, diakses 3 Mei 2015 http://www.di-mgt.com.au/rsa_alg.html.

[2] Steyn, Barry, *How RSA Works with Example*, diakses 3 Mei 2015 http://doctrina.org/How-RSA-Works-With-Examples.html.

[3] Veness, Chris, Block TEA 'Tiny Encrpytion Algorithm', diakses 14 Mei 2014, (XXTEA) implemented in Javascript, http://www.movable-type.co.uk/scripts/tea-block.html.

[4] Siregar, Fahry Rozy 2014, *Perancangan Keamanan Informasi Sistem Pemungutan Suara Elektronik (e-Voting) Menggunakan Kombinasi Algoritma AES dan RSA.*

[5] Lestari, Puji. 2013, *Implementasi Algorithma RSA (Rivest Shamir Adleman) dalam Sistem Enkripsi File dan Pengamanan Folder.*

[6] Denver. 2013, *Pengamanan Komunikasi Suara Melalui Internet Pada Telepon Seluler dengan Algoritma TEA pada Platform Android*.

[7] Atmojo, Dwi Ardani 2012, *Perancangan Secure Login Website Menggunakan Algoritma Enkripsi XXTEA.*

[8] Wahyuni, Ana. 2011, *Keamanan Pertukaran Kunci Kriptografi dengan Algoritma Hybrid : Diffie-Hellman dan RSA.*

[9] Hendarsyah, Decky & Wardoyo, Retantyo 2012, Implementasi Protokol *Diffie - Hellman* Dan Algoritma RC4 Untuk Keamanan Pesan SMS. Universitas Gajah Mada.

[10] Kurniawan, Agus 2012, *Network Forensics: Panduan Analisis dan Investigasi Paket Data Jaringan Menggunakan Wireshark*. Yogyakarta. Penerbit Andi.

[11] Prajapati, Priteshkumar, et. al. 2014, *Comparative Analysis of DES, AES, RSA Encryption Algorithms.*

[12] Mollin, R. A. 2002, *RSA and Public Key Cryptography*, Boca Raton, Florida, Chapman & Hall/CRC.