RSA Criptography Untuk Autentikasi Pada Network Database

Neo Enriko - NIM 13507104

1) Jurusan Teknik Informatika ITB, Bandung 40132, email: neo_enriko@students.itb.ac.id

Abstract – *Makalah ini membahas perancangan dan desain system untuk halaman log in dan pendaftaran user yang kemudian bisa diimlementasikan ke jaringan computer yang sesungguhnya.*

Perkembangan teknologi telekomunikasi dan penyimpanan data dengan menggunakan komputer telah memberikan kemudahan kepada manusia untuk pengiriman data jarak jauh. Barbagai jenis layanan komunikasi tersedia di internet, diantaranya adalah website, e-mail, milis, mailgroup, situs pertemanan, dan sebagainya.

Dengan meledaknya kuantitas penggunaan layanan komunikasi di internet tesebut, maka permasalahanpun bermunculan, ditambah dengan adanya hacker dan cracker. Sehingga sangat memungkinkan data yang dikirim dapat disadap dan diubah oleh pihak lain.

Dengan menggunakan metode RSA criptosystem, yaitu dengan cara mengenkripsi pesan yang akan dikirim dengan menggunakan kunci public yang telah dibangkitkan oleh komputer server, maka akan memperkuat tingkat keamanan data yang dikirimkan dalam suatu jaringan komputer selama kunci privat terjaga kerahasiaannya.

Untuk kerja sistem server yang baik. Private key yang dibangkitkan selalu berubah-ubah nilainya walaupun terkadang public key yang dihasilkan nilainya sama. Demikian pula nilai chipertext selalu berubah-ubah walaupun input yang diberikan public key selalu sama, hal ini disebabkan karena nilai n yang digunakan untuk komputasi selalu berbeda.

Kata Kunci*: cryptosystem, RSA, dan kunci publik.*

1. PENDAHULUAN

Pada zaman cyber ini perkembangan teknologi informasi dan *datastore* pada computer sangat memungkinkan untuk mengirim data jarak jauh melalui gelombang radio(elektromagnetik) maupun radio lain yang digunakan di masyarakat luas. Karena kecangihan teknologi juga maka sangat memungkinkan untuk pihak lain dapat menyadap dan mengubah data yang kita dikirim dengan tujuan tertentu. Demikian pula pada sistem jaringan komputer local maupun secara luas pada *internet* yang jumlah pemakainya banyak, lengkap dengan tujuan masing-masingnya.

Sangat banyak jenis layanan komunikasi yang tersedia *internet*, seperti *web*, *e-mail*, *milis*, *newsgroup*, dan sebagainya. Seiring dengan banyaknya penggunaan layanan komunikasi di *internet*, maka muncullah berbagai macam permasalahan keamanan dan privasi pada komunikasi tersebut, apalagi ditambah dengan adanya *hacker* dan *cracker*. Ini sangat menimbulkan keresahan pada komunikasi di internet, ketakutan akan keamanan data yang kita kirim.

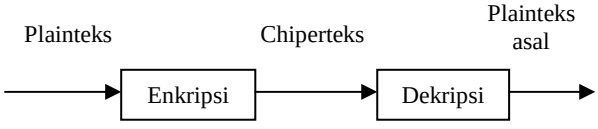
Dalam perkembangan teknologi informasi dibidang *security system*, telah dan sedang dikembangkan cara-cara untuk menangkal berbagai serangan semacam itu. Salah satu cara yang ditempuh mengatasi masalah ini, yang dirasa cukup tangguh adalah dengan menggunakan kriptografi kunci publik *RSA* yang menggunakan alihragam data, dengan langkah-langkah tertentu, dengan proses enkripsi dan dekripsi, sehingga data yang dihasilkan tidak dapat dimengerti oleh pihak ketiga. Hanya orang yang berwenang saja yang bisa mengartikan alihragam data tersebut, sehingga data yang kita kirim lewat internet aman.

Kita dapat menerapkan *public key* RSA pada autentikasi *network database.* Kita akan menganalisa kerja program aplikasi yang menerapkan kriptosistem *public key RSA* menggunakan Bahasa *Java* untuk autentikasi pengguna basis data di jaringan *WAN* atau *Intranet* demi keamanan.

2. KRIPTOGRAFI

Kriptografi merupakan ilmu sekaligus seni untuk menjaga kerahasiaan pesan (data atau informasi) dengan cara menyamarkannya (*to cripyt*) menjadi bentuk tersandi yang tidak bermakna.

Pesan yang dirahasiakan dinamakan *plainteks* (*plainteks*,artinya teks jelas yang dapat dimengerti), sedangkan hasil penyamaran dinamakan *chiperteks* (*chiperteks*, artinya teks tersandi). Proses penyamaran dari plainteks ke chiperteks disebut *enkripsi* (*encryption*) dan proses pembalikan dari chiperteks ke plainteks disebut *dekripsi* (*decryption*). Gambar dibawah ini adalah diagram prosesnya.



Gambar 1 : Proses enkripsi dan dekripsi

Suatu kriptosistem (*cryptosystem* atau seringkali juga disebut *cipher*) adalah algoritma kriptografi, ditambah seluruh kemungkinan *plaintext*, *chipertext* dan kunci-kuncinya.

Setiap kriptosistem mempunyai dua algoritma bagian, yaitu proses enkripsi (proses untuk penyandian data) dan proses dekripsi (proses perhitungan data asli dari

data terenkripsi). Dua-duanya proses ini bisa berbeda, tapi harus saling terkait. Untuk kriptosistem biasanya data yang akan dienkripsi disebut *plaintext*, hasil enkripsi (data yang sudah terenkripsi) disebut *ciphertext*. *Plaintext* dapat berupa aliran *bit*, *file text*, *bitmap*, aliran suara yang digitasi, gambar video digital dan sebagainya. Namun apabila semua pesan tadi berbentuk digital, maka semua pesan tadi dapat dianggap sebagai data *biner*. Dan data biner juga dapat diperlakukan sebagai sistem bilangan *biner*. Karena itulah diperlukan matematika untuk menganalisanya. *Plaintext* dapat dikirim maupun disimpan dalam jaringan.

Dalam perkembangannya, kriptografi sekarang telah digunakan untuk identifikasi pengiriman pesan dengan tanda tangan digital dan keaslian pesan dengan sidik jari digital (*figerprint*), sehingga tingkat keamanan pengiriman data semakin tinggi.

2.1. Notasi Matematis

Jika chiperteks dilambangkan dengan C dan plainteks dilambangkan dengan P, maka fungsi enkripsi E memetakan P ke C,

$$E(P) = C$$

Pada proses kebalikannya, fungsi dekripsi D memetakan C ke P,

$$D(C) = P$$

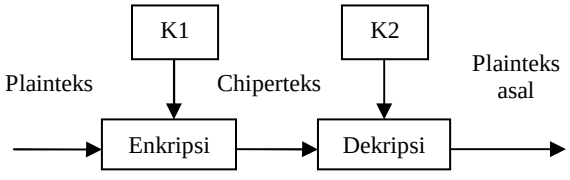
Kekuatan suatu algoritma diukur dari banyaknya kerja yang dibutuhkan untuk memecahkan data chiperteks menjadi plainteksnya. Untuk kriptografi modern, kekuatan algoritma terletak pada kuncinya, yaitu berupa deretan karakter atau bilangan bulat yang Plainteks asal chiperteks enkripsi enkripsi plainteks dijaga kerahasiaannya. Kunci ini dapat dianalogikan dengan penggunaan PIN pada ATM.

Secara matematis, pada sistem kriptografi yang menggunakan kunci K, maka fungsi enkripsi dan dekripsi menjadi,

$$Ek1(P) = C$$
$$Dk2(C) = P$$

dan kedua fungsi ini memenuhi

$$Dk2(Ek1(P)) = P$$



Gambar 2 : Enkripsi dan Dekripsi pada algoritma kriptografi modern

Jika K1 = K2 (yaitu, kunci untuk proses enkripsi sama dengan kunci untuk dekripsi), maka algoritma kriptografinya disebut algoritma simetri. Contoh algoritma yang menjadi standar algoritma simetri adalah DES (Data Encryption Standard). Sebaliknya, apabila K1 tidak sama dengan K2, maka disebut algoritma nirsimetri. Contohnya adalah algoritma RSA (Rivest-Shamir-Adleman).

2.2. Pengkodean Teks Dengan Metode Kriptografi Nirsimetris RSA

Kriptosistem *RSA* adalah kriptosistem kunci public yang menyediakan fasilitas enkripsi dan tanda tangan digital (*digital signature*) untuk autentikasi. Algoritma ini diciptakan pada tahun 1977 oleh Ron Rivest, Adi Shamir dan Leonard Adleman yang berasal dari MIT. Karena sulitnya pemfaktoran bilangan yang sangat besar, maka *RSA* dianggap aman.

Algoritma kriptografi *RSA* merupakan algoritma yang termasuk dalam kategori algoritma nirsimetri (juga disebut sebagai algoritma kunci publik), didesain sedemikian sehingga kunci yang digunakan untuk enkripsi berbeda dari kunci yang digunakan untuk dekripsi.

Algoritma disebut kunci publik karena kunci enkripsi, dapat dibuat publik yang berarti semua orang boleh mengetahuinya. Sembarang orang dapat menggunakan kunci enkripsi tersebut untuk mengenkripsi pesan, namun hanya orang tertentu (calon penerima pesan sekaligus pemilik kunci dekripsi yang merupakan pasangan kunci publik) yang dpat melakukan dekripsi terhadap pesan tersebut.

Dalam sistem ini, kunci enkripsi sering disebut kunci publik, sementara kunci dekripsi sering disebut *private/secret key* (kunci rahasia).

Dibawah ini adalah algoritma RSA secara ringkas:

1. Pilih dua buah bilangan prima sembarang, sebut a dan b. Jaga kerahasian angka ini.
2. Hitung $n = a \times b$, besaran n tidak dirahasiakan.
3. Hitung $m = (a-1) \times (b-1)$. Sekali m dihitung, nilai a dan b langsung dihapus, agar tidak ketahuan.
4. Pilih sebuah bilangan bulat untuk kunci public, sebut namanya e, yang relative prima terhadap m. dimana $1<e<m$.
5. Hitung d (kunci dekripsi), dengan $ed \equiv 1 \pmod m$.
6. Kunci umumnya adalah n, e. Kunci dekripsi (rahasia) adalah d.
7. Lakukan ekripsi terhadap pesan rahasia, dengan persamaan $c_i = p_i^e \bmod n$, p_i adalah plainteks dan c_i adalah chiperteks.
8. Proses dekripsi dilakukan dengan persamaan $p_i = c_i^d \bmod n$.

3. CLIENT/SERVER DENGAN JAVA REMOTE METHOD INVOCATION (JAVA RMI)

Istilah *client/server* pertama kali digunakan pada tahun-tahun 1980-an untuk mengacu sebuah *Personal Computer* (PC) pada suatu jaringan. Model *client/server* yang sebenarnya mulai diterima pada akhir tahun 1980-an. Arsitektur perangkat lunak *client/server* adalah infrastruktur modular dan *message-based*. Dalam pemrograman *client/server*, *client* dan *server* dapat ditulis dalam bahasa pemrograman yang sama atau berbeda. Bahasa pemrograman *Java* menyediakan beberapa aplikasi untuk kedua situasi tersebut. Aplikasi *socket* dapat dapat digunakan untuk kedua situasi diatas. *CORBA* (*Common Object Request Broker Architecture*)

digunakan untuk situasi yang kedua. Sedangkan untuk situasi dimana pemrograman dilakukan dengan bahasa yang sama, *Java* menawarkan *RMI*(*Remote Method Invocation*) sebagai alternatif dari *socket*. Tidak seperti *Socket*, *RMI* mengabstrakkan *interface* antara *client* dan *server* menjadi satu pemanggilan prosedur lokal. Dengan menggunakan *RMI*, programmer tidak perlu merancang satu protokol.
Client/server dengan java remote method invocation bagus digunakan untuk applikasi autentikasi log in ke database pada suatu server.

4. IMPLEMENTASI RSA CRYPTOGRAPHY

4.1. Desain System Untuk Log in User

Prinsip kerja dari system ini sebenarnya hampir mirip dengan proses pendaftaran *user*. Beberapa perbedaannya yaitu, pertama pada *input*an (*plaintext*) hanya pemakaian *field* userID dan password, kedua terletak pada teks yang dikirim yaitu hanya *chipertext*. Sedangkan perbedaan ketiga terletak pada perintah *query* di *server*.
Prosesnya pertama input, yaitu user ID dan password (*plainteks*), lalu diminta *public key* ke server. Lalu server akan membangkitkan *public key* dan *private key*, dan *public key* akan diberikan ke user. Lalu dilakukan enkripsi plainteks oleh user dengan algoritma RSA menggunakan *public key* yang diberikan server. Lalu teks yan terenkripsi dikirim ke server, dan server akan melakukan dekripsi dengan menggunakan *private key*. Lalu server akan mengecek query user ID dan password, jika teks yang dimasukan user ada dalam database maka *log ini* berhasil, jika tidak ada, maka *log in* berarti gagal.

4.2. Cryptosystem RSA, enkripsi, dekripsi, dan pembangkitan kunci

Input disandikan melalui enkripsi, kemudian hasilnya(*plaintext*) akan dikirimkan ke *server*. Dalam proses ini *input* mengalami *encoding* mulai dari *string* ke *byte array*, kemudian dipecah ke masing-masing posisi *array* untuk diubah ke *string*. Setelah semua mengalami pengubahan, *stringstring* ini kemudian disusun sesuai dengan urutan *array* masing-masing. Langkah selanjutnya seluruh *string* diubah ke bilangan *biginteger*, akhir dari tahapan ini akan mengalami enkripsi (menggunakan *public key* milik *server*) yang akan dihasilkan sebuah *chipertext* yang dikirim ke *server*.
Proses enkripsiya sebagai beriku, plainteks dibaca lalu diubah ke dalam *bit aray* , ambil kode *bit aray* pertama, ubah kode byte ke string, proses semua kode *bit aray*, jika sudah maka dilakukan penggabungan kode *bit aray* pertama, kedua, dan seterusnya. Lalu konversi hasil penggabungan ke *biginteger*. Lalu dilakukan enkripsi dengan kunci pulik(n) maka akan dihasilkan *chiperteks*.
Proses dekripsi pada intinya merupakan kebalikan dari proses enkripsi. Bedanya pada tahap dekripsi, kita menggunakan *private key*. Dalam proses ini, prinsip kerjanya adalah *chipertext* yang dikirim dari *client* akan diubah kembali menjadi teks asalnya (*plaintext*) menggunakan *private key* milik *server*. Dalam proses ini juga mengalami beberapa tahapan, diantaranya: bilangan *biginteger* ke *string*, mencari karakter *ASCII* yang sesuai dengan metode *look-up* kode *ASCII* dan kemudian merangkainya menjadi teks sesuai dengan teks asalnya (*plaintext*).
Pembangkitan kunci (*public key* dan *secret key*), proses ini merupakan proses utama dari kedua proses yang telah dijelaskan diatas karena proses inilah yang digunakan baik untuk mengenkripsi dan mendekripsi.

5.PEMBAHASAN

Pengujian program ini akan menghasilkan suatu *ouput* dari *input* yang diberikan oleh *user*, *output-output* tersebut diantaranya adalah: *private key*, *public key*, *n*, *chipertext* dan *plaintext* hasil dekripsi.

5.1. Media Stand-alone PC Untuk Log in ke Database

Aplikasi *Java* biasa (paket 1) :

Tabel 1. Hasil eksekusi kelas-kelas paket 1 media *stand-alone PC*

No	Plainteks		Key		n	Chiperteks		Timer login(s)
	User ID	Pass	Public	Secret		User ID	Pass	
1	ergi2004	ta	5	*60000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000007972 54008513874 68000000003 24000000000 00000000000 00000000000 00000000000 00026483914 75381005669	*1000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000132 87566808564 57800000000 05600000000 00000000000 00000000000 00000000000 00000441398 57923016761	*6111657332 94486830449 77157738461 90666296981 58363168583 64960590842 69056003606 34062120812 71839524724 77653131166 16023820115 96944570774 05589487567 77515302234 94893944640	*1551222377 49045899396 86674352409 81770333617 10007360008 30929075404 88363740939 32417929172 51586914062 5	4,328

				76459216002 98422987461 63599999933 09	16274320280 00397063980 81839999998 8903	46968378480 51770092499 47450712589 750		
2	arjuna17	Abg	5	*20000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000010910 92852429609 33999999996 52000000000 00000000000 00000000000 00000000000 00071803156 65493662062 21144288158 62230071357 49199999983 89	*1000000000 00000000000 00000000000 00000000000 00000000000 00000000000 00000000545 54642621480 46699999999 82800000000 00000000000 00000000000 00000000000 00003590157 83274683103 11057214462 48575765715 92129999999 1771	*5941656136 01501869839 49766737459 53890561872 35472187964 91728452682 68211732600 25620291597 96972568592 74420663835 42343925766 73234399019 88777679735 88856864741 40982657061 75822680054 09559120457 83103270105 656	*2714538048 67075641167 83376636690 73533323627 56902050978 67585290876 92577514733 43784145337 40009862360 75269874054 80435399081 88883339964 80870987548 6751	1,641

Kita analisa angka-angka yang tertera pada tabel 1, hasil dari pembangkitan *privatekey* selalu berbeda meskipun kunci pasangannya yaitu *public key* berpeluang muncul dengan nilai yang sama. Sedangkan hasil dari enkripsi yaitu *chipertext* nilainya juga selalu berbeda walaupun *input* yang diberikan nilainya sama, hal ini terjadi karena proses enkripsi memerlukan *parameter-parameter* yaitu nilai *n* dan *public key* itu sendiri. Apabila nilai *n* selalu berubah, maka secara otomatis hasil komputasi enkripsi akan berubah ($chipertext = plaintext^{publickey} \bmod n$) .

Timer log in yaitu waktu yang dibutuhkan untuk koneksi ke *database* pada kejadian saat tombol *log in* ditekan hingga tombol *log in* di lepas, pada Tabel 1 *timer log in*-nya cenderung menurun atau waktunya singkat meskipun *input* yang diberikan nilainya sama. Pada uji coba ke-1 timer *log in*-nya relatif lama dibandingkan untuk uji coba ke-2.

Apikasi *client* menggunakan fasilitas *browser* :

Tabel 2. Hasil eksekusi kelas-kelas paket 2 media *stand-alone PC*

No	Plainteks		Key		n	Chiperteks		Timer login(s)
	User ID	Pass	Public	Secret		User ID	Pass	
1	z enden13	z en	11	*727272727 272727272 727272727 272727272 727272727 272727272 727272727 272727272 837398470 546197498 181818087 709090909 090909090 909090909 090909090 909090909 094718631 601749728 115202009 123524641 542867437 090909090 91	*100000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 015142289 700102155 999999987 080000000 000000000 000000000 000000000 000000000 000523811 845240587 615840276 255998867 182154488 199999998 707	*694882086 148103783 640455191 170858999 471058626 801641495 199018113 219040885 585532585 189675519 951084287 828361535 894482034 888687719 918420634 967739629 474348922 799212598 520086161 770144082 963148812 040242809 56	*945375763 151499479 381593577 958644940 921950375 456750047 044050570 212898004 460855451 957173011 018625701 269088269 842112212 850214016 799366542 500254560 769480244 433461063 149656123 356198914 133426734 800140685 04	6 ,515

2	badhanku1	badhanku	5	*200000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 053905747 270976565 999999989 920000000 000000000 000000000 000000000 000000000 001771309 027516549 868511870 348540116 709717154 799999981 21	*100000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 026952873 635488282 999999994 980000000 000000000 000000000 000000000 000000000 000885654 513758274 934255935 176965345 718407405 699999990 101	*653997738 968731888 408247251 411338491 858209098 487055954 470783640 595933617 401705430 862646772 635044021 968143723 332024189 813280391 931052532 126029852 809382843 339648066 792312357 626894959 255499723 531205380 26	*257231351 171122974 892453614 070978232 589255411 137860116 557442987 057318914 212566686 443531469 851431797 980200838 376446542 056444663 273848394 157929368 053190197 882689852 997191804 127992847 084625808 219976892 96	1 ,578

Kita lihat pada tabel 2, hasil dari pembangkitan *private key* selalu berbeda meskipun kunci pasangannya yaitu *public key* berpeluang muncul dengan nilai yang sama. Sedangkan hasil dari enkripsi yaitu *chipertext* nilainya juga selalu berbeda walaupun *input* yang diberikan nilainya sama, hal ini terjadi karena proses enkripsi memerlukan *parameter-parameter* yaitu nilai *n* dan *public key* itu sendiri. Apabila nilai *n* selalu berubah, maka secara otomatis hasil komputasi enkripsi akan berubah (*chipertext* = *plaintext* ^{publickey} *mod n*) .

Pada Tabel 2 *timer log in*-nya cenderung menurun atau waktunya singkat meskipun *input* yang diberikan nilainya sama. Timer *log in* pada uji coba pertama relatif lama dibandingkan untuk uji coba kedua.

5.2. Media Jaringan Client/Server Untuk Log in ke Database

Apikasi *client* menggunakan fasilitas *browser* (paket client):

Tabel 3. Hasil eksekusi paket *client* media jaringan *client/server* dengan *browser*

No	Plainteks		Key		n	Chiperteks		Timer login(s)
	User ID	Pass	Public	Secret		User ID	Pass	
1	badhanku1	badhanku	5	*200000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 022383881 792367209 999999983 880000000 000000000 000000000 000000000 000000000 000626296 677953272 567978594 839077177 966896631 999998963 57	*100000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 011191940 896183604 999999991 960000000 000000000 000000000 000000000 000000000 000313148 338976636 283989297 420657783 073066676 499999480 979	*709072288 274547091 469669244 524596229 468488104 500731365 033143472 325933507 153085913 039227284 816711268 218465828 905697380 875981997 051721099 960755079 866924091 487185604 575297093 184979484 223113330 835874498 77	*763480282 989232108 831815351 517212940 765485385 814723971 878459937 233643721 265500105 023915658 191060789 109949508 445949812 158199128 545955891 079875634 851641081 238972581 452263312 441174075 539608403 261422784 25	4.11

Hasil dari pembangkitan *private key* yang tertera pada tabel 3 berbeda meskipun kunci pasangannya yaitu *public key*berpeluang muncul dengan nilai yang sama. Sedangkan hasil dari enkripsi yaitu *chipertext* nilainya juga selalu berbeda walaupun *input* yang diberikan nilainya sama, hal ini disebabkan karena proses

enkripsi memerlukan *parameter-parameter* yaitu nilai *n* dan *public key* itu sendiri.

5.3. Media Jaringan Client/Server Untuk Pendaftaran User ke Database

Tabel 4. Hasil eksekusi kelas-kelas paket *client* media jaringan *client/server* dengan *browser*

No	Plainteks			Key		n	Chiperteks			Timer
	User ID	Pass	Tgl Lahir	Public	Secret		User ID	Pass	Tgl Lahir	login(s)
1	ergi2004	ta	1011965	5	*800000000 000000000 000000000 000000000 000000000 000000000 000000000 192865417 727893095 999999986 400000000 000000000 000000000 000000000 000000000 011623673 298980081 076489233 923334115 939692631 999999038 21	*100000000 000000000 000000000 000000000 000000000 000000000 000000000 024108177 215986636 999999998 320000000 000000000 000000000 000000000 000000000 001452959 162372510 134561154 242827582 214060242 699999879 607	*658252420 753572474 608934816 263358008 853249594 120518685 181127038 119494042 470901478 258210273 004193857 623657941 296440544 114552140 770141448 255392673 095236339 964692575 969182977 071415297 383779398 558060661 49	*415283095 540857006 616097440 756412743 269657462 803839902 346917141 208043080 003505536 386948782 222006613 990168253 365737875 903957768 443678867 407426099 153323474 902258266 101188539 681226626 548556327 468414720 80	*915504535 907269337 172110270 631815405 643523964 465746792 336154069 447257586 433227977 418128925 004974064 486778380 994407084 613282385 567602936 678388818 141846475 395676273 623661707 604663649 129334252 164746798 08	3
2	arjuna17	Abg	31101971	11	*727272727 272727272 727272727 272727272 727272727 272727272 727272727 272727273 084111244 953373490 909090913 309090909 090909090 909090909 090909090 909090909 108543453 748872641 159242290 324752659 315681087 999999919 71	*100000000 000000000 000000000 000000000 000000000 000000000 000000000 000000000 049065296 181088855 000000000 600000000 000000000 000000000 000000000 000000000 002424724 890469988 159395814 924560020 274015035 099999989 019	*379936325 265575516 970833071 039310993 191085062 284878733 864558841 103959100 718041525 479621997 426893567 101105280 405973050 409458929 462430804 578303439 638525764 239786247 053843888 414260226 811038042 502562684 26	*236677013 057702050 111897384 103720152 430678089 462815965 835638590 110876539 647283880 544745180 184305683 744591533 312146286 354561927 444739721 098768848 181582947 303391596 285561610 444162821 995560849 993531339 0	*236677013 057702050 111897384 103720152 430678089 462815965 835638590 110876539 647283880 544745180 184305683 744591533 312146286 354561927 444739721 098768848 181582947 303391596 285561610 444162821 995560849 993531339 0	1,172

Kita lihat pada tabel 4, hasil dari pembangkitan *private key* berbeda meskipun kunci pasangannya yaitu *public key* berpeluang muncul dengan nilai yang sama. Sedangkan hasil dari enkripsi yaitu *chipertext* nilainya juga selalu berbeda walaupun *input* yang diberikan nilainya sama, hal ini disebabkan karena proses enkripsi memerlukan *parameter-parameter* yaitu nilai *n* dan *public key* itu sendiri.

6. KESIMPULAN

1. RSA sangat aman untuk diterapkan pada kriptosistem autentikasi jaringan ke database. Karena untuk memecahkan pesan yang sudah berupa suatu *chipertext* yang diketahui, penyerang (hacker) harus dapat mengetahui *private key* yang telah dibangkitkan di *server*. Sedangkan *private key* sendiri tidak akan bisa didapatkan oleh hacker.
2. *Private key* yang dibangkitkan pada server nilainya selalu berubah-ubah walapun terkadang *public key* yang dihasilkan nilainya sama. Sehingga hal ini

menyebabkan data autentikasi pada network database kita sangat aman dari hacker.

3. Nilai *chipertext* selalu berubah-ubah walaupun *input* yang diberikan dan *public key* selalu sama karena nilai n yang digunakan untuk komputasi selalu berbeda.
4. Perbedaan waktu akes antara proses *log in* dan mendaftar ke *database* baik *stand alone PC* ataupun jaringan *client/server*, tidak terlalu mencolok, tetapi pada prose-proses eksekusi program berikutnya waktu yang dibutuhkan relatif cepat terhadap proses-proses sebelumya, dengan catatan kondisi *running server* yang sama.
5. Penerapan RSA untuk kriptosistem pada autentikasi jaringan ke database sangat aman terhadap pencurian data dari pihak yang tidak berwenang, karena keunikan yang dimiliki oleh RSA cryptosystem.

DAFTAR REFERENSI

[1] Rinaldi Munir, *Matematika diskrit*, Prodi Teknik Informatika,2008.

[2] Knudsen, Jonathan H, *Java Cryptography, First Edition*, O'Reilly.M, 1998.

[3] URL : http://javaoop.wordpress.com/2007/11/08/mengenal-konsep-oop/ Tanggal akses : 26 Desember 2008 20:15

[4] URL : www.benpinter.net/article.php?story=20030818005713433 Tanggal akses : 27 Desember 2008 15.30

[5] Moss, Jeff. *Stealing The Network, how to own the box*, 2003, Rockland: Syngress Publishing.

[6] Rasyid, Moh Iqbal, *Tinjauan mengenai kepopuleran teknik penyandian RSA*, Skripsi, Depok: Universitas Gunadharma, 1999.

[7] Ramachandran, Jay, *Designing Security Architecture Solutions*, 2002, John Wiley & Sons.