Analisis Implementasi Protokol Kriptografi pada *Remote Procedure Call(RPC)*

Sidik Soleman 13508101[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*sidiksoleman@gmail.com*

***Abstract*—RPC atau *Remote Procedure Call* merupakan suatu metode yang memungkinkan untuk memanggil suatu prosedur yang berada pada mesin yang berbeda. RPC biasa dilakukan pada komunikasi sistem terdistribusi. Komunikasi dilakukan oleh klien-server. Proses pemanggilan ini mirip dengan pemanggilan prosedur lokal hanya saja pada RPC klien meminta server untuk mengeksekusi prosedur dan mengirimkan hasil eksekusi tersebut kembali ke klien. RPC memiliki protokol tersendiri yang digunakan untuk mengidentifikasikan pesan saat terjadi proses RPC tetapi protokol tersebut tidak bertanggung jawab terhadap jalur komunikasi yang digunakan. Pada umumnya, RPC berjalan diatas *network layer* dan *transport layer* yang berarti RPC bisa berjalan di atas TCP/IP maupun UDP. RPC juga tidak mendefinisikan secara khusus protokol keamanan komunikasi antara server dan klien saat melakukan pemanggilan prosedur dengan kata lain proses pengamanan komunikasi diserahkan pada pengimplentasi RPC. Hal ini dilakukan agar proses authentifikasi pada RPC tidak bergantung pada satu jenis metode. Jalur komunikasi yang digunakan oleh RPC bukan jalur yang aman, Jika tidak diimplementasikan protokol keamanan pada RPC, hal ini dapat menimbulkan isu keamanan komunikasi pada RPC. Salah satu metode untuk mengamankan jalur adalah dengan menambahkan layer keamanan. Layer keamanan dapat berupa protokol kriptografi. Salah satu protokol kriptografi yang dapat digunakan adalah *secure socket layer* atau SSL.**

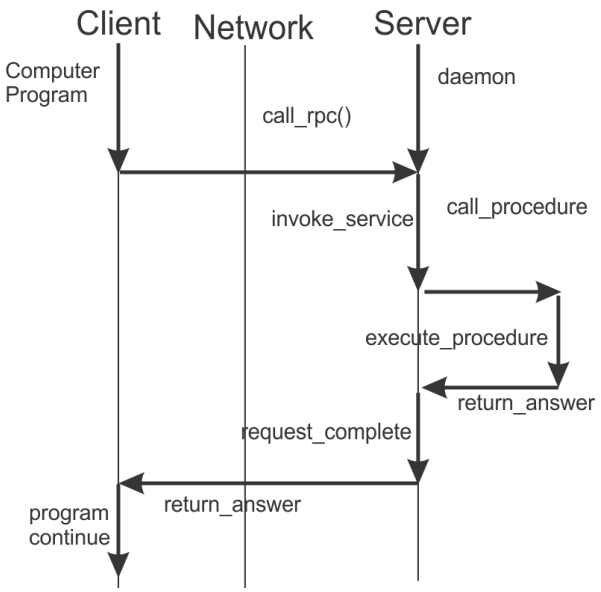
***Keywords*—RPC, SSL, Protokol Kriptografi.**

I. Remote Procedure Call

Remote Procedure Call atau RPC merupakan salah satu metode yang sering digunakan dalam komunikasi sistem terdistribusi. RPC juga merupakan salah satu metode untuk mencapai tujuan terwujudnya *sharing resources* yang merupakan tujuan utama dari sistem terdistribusi. Perbedaan RPC dengan pemanggilan prosedur lokal hanya pada pelaku komunikasi yang terlibat. Jika dalam pemanggilan prosedur lokal pelaku komunikasi adalah mesin komputer itu sendiri sedangkan pada RPC pelaku komunikasi adalah dua buah komputer yaitu klien dan server. Secara sederhana, proses yang terjadi adalah sebuah proses di klien membutuhkan pemanggilan prosedur tetapi prosedur tersebut tidak tersebut tidak berada komputer klien. Oleh karena itu, klien akan melakukan permintaan ke komputer server untuk memanggil prosedur tersebut dan mengeksekusi prosedur tersebut di komputer server. Setelah dilakukan eksekusi, hasil dari eksekusi tersebut selanjutnya dikirimkan kembali ke klien dan terakhir klien akan memberikan pada proses yang meminta pemanggilan prosedur tersebut.

Hal lain yang membedakan dengan pemanggilan prosedur lokal adalah adanya penanganan kesalahan yang diakibatkan oleh jaringan. Selain itu, pada RPC tidak ada global variabel karena server dan klien berbeda mesin. RPC biasanya akan lebih lama dibandingkan pemanggilan lokal prosedur.

Secara umum diagram proses komunikasi pada RPC adalah berikut ini :

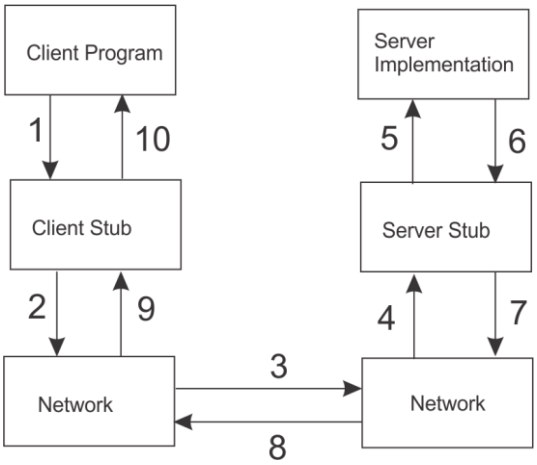


Gambar 1. Skema komunikas RPC

Pada komunikasi RPC didefinisikan protokol RPC yang akan disepakati oleh klien maupun server. Protokol ini dibuat sesederhana mungkin dan paling umum untuk menghindari ketergantungan pada mesin atau teknologi. Protokol ini menambahkan satu bagian penting yang menjadi perantara komunikasi antara server dan klien yaitu *stub*. *Stub* ini akan diletakkan pada server dan klien yang berisi daftar prosedur yang bisa dipanggil atau yang

mungkin dipanggil oleh klien.

Protokol komunikasi pada RPC akan mendefinisikan urutan proses komunikasi yang ada. Proses komunikasi yang ada dapat diilustrasikan sesuai dengan gambar di bawah ini :



Gambar 2. Protokol RPC

Urutan proses komunikasi pada yang didefinisikan pada RPC :

1. Program klien yang sedang berjalan akan membutuhkan suatu prosedur yang harus dieksekusi sehingga program tersebut dapat melanjutkan kembali proses yang sedang berjalan.
2. Program tersebut akan memanggil klien stub untuk melakukan RPC. Selama menunggu proses RPC proses akan berhenti menunggu sampai RPC selesai.
3. Stub akan melakukan pembungkusan terhadap nama prosedur dan parameter yang dilewatkan ke stub. Setelah dilakukan pembungkusan, selanjutnya Stub akan meminta *operating system* untuk mengirimkan paket yang telah dibungkus oleh stub ke server.
4. *Operating system* akan mengirimkan pesan tersebut ke server melalui jaringan komputer.
5. *Operating system* pada server menerima paket yang telah dikirimkan klien, selanjutnya OS akan memanggil server stub dan memberikan paket tersebut ke server stub.
6. *Server stub* selanjutnya akan membongkar pesan dan memerintahkan *Operating System* untuk melakukan eksekusi prosedur yang dipanggil.
7. Setelah selesai melakukan pemanggilan prosedur tersebut, hasil eksekusi diberikan kembali oleh *operating system* ke *server stub*. *Server Stub* selanjutnya membungkus hasil eksekusi tersebut dan memerintahkan kembali *operating system* untuk mengirimkan paket tersebut yang berisi hasil eksekusi prosedur.
8. *Operating system* selanjutnya akan mengirimkan paket melalui jaringan komputer menuju klien yang meminta.
9. *Operatins system* klien menerima kembali paket hasil eksekusi dan memberikan paket tersebut ke klien stub.
10. Klien stub melakukan pembongkaran pesan dan memberikan hasil eksekusi tersebut ke proses yang meminta stub untuk melakukan RPC sebelumnya.
11. Proses pada klien kembali berjalan.

Pada umumnya, proses komunikasi RPC berjalan di atas *transport layer* dan *network layer* dengan kata lain RPC akan berjalan di atas TCP/IP atau UDP. Terkait dengan hal tersebut ada beberapa isu mengenai RPC :

1. Komunikasi pada RPC bisa tidak *reliable* karena komunikasi bisa melalui jalur TCP/IP atau UDP. Oleh karena itu, jalur komunikasi harus disepakati antara kedua komputer yaitu server dan klien.
2. *Passing parameter*, menjadi isu karena komunikasi ini terjadi pada dua komputer yang berbeda sehingga tidak mungkin untuk dilakukan *passing parameter by reference*.
3. Perbedaan mesin juga menjadi isu dalam RPC karena RPC harus bisa berkomunikasi antar komputer apapun mesinnya.

Isu selanjuntya adalah isu keamanan yang akan dibahas lebih lanjut dalam, karena tidak ada penanganan keamanan dalam RPC tidak didefinisikan secara khusus melainkan diserahkan pada pengimplementasi untuk menangani hal tersebut atau tidak.

II. ISU KEAMANAN PADA RPC

Keamanan pada RPC hanya dilakukan dilakukan dengan authentifikasi. Untuk authentifikasi ini RPC menspesifikasikannya pada XDR(*eXternal Data Representation*) *Language*. Secara umum ada tiga *filed* dalam XDR yaitu *AUTH_NONE*, *AUTH_SYS*, dan *AUTH_SHORT* dan mungkin bisa ditambahkan lagi secara manual. *AUTH_NONE* merupakan authentifikasi yang mengizinkan semua untuk melakukan pemanggilan prosedur dalam komputer server atau dengan kata lain apapun klienya akan dilayani. Namun, metode authentifikasi ini masih memungkinkan untuk dilakukan penyerangan. Walaupun sudah dilakukan authentifikasi, baik server dan klien tidak dapat menjamin apakah pihak yang diajak komunikasi tersebut merupakan pihak yang benar-benar ingin diajak berkomunikasi.

Hal ini ditambah lagi dengan jalur komunikasi yang dilalui oleh RPC bukan merupakan jalur yang aman sehingga masih memungkin untuk dilakukan penyerangan atau penyadapan bahkan fabrikasi. Penyerangan bisanya ingin mengakses pada pada mesin server maupun mesin klien. Penyerang bisa saja meminta server untuk melakukan pemanggilan pada prosedur yang memang sebenarnya tidak boleh diakses oleh sembarang klien. Jadi penyerang seolah-oleh berperan menjadi klien.

Penyerang tersebut juga mampu menyerang klien, yaitu dengan melakukan pengubahan pada hasil prosedur yang semestinya diberikan oleh server. Artinya kembalian dari

server dimodifikasi atau bahkan diganti oleh penyerang dan kemudian penyerang tersebut kembali menyerahkan tersebut ke klien. Jadi seolah-olah sekarang penyerang bertindak sebagai sebuah server. Akibatnya klien dapat melakukan eksekusi yang mungkin saja membahanyakan keamana komputer tersebut.

Secara umum, berikut ini adalah risiko dari masing-masing pihak saat berkomunikasi dalam RPC.

1. Resiko Klien

   a. Klien *confidentiality*
      Klien mungkin saja mengirimkan informasi yang sensitive dan informasi ini dapat disadap oleh pihak yang tidak seharusnya mengetahui karena data tersebut dikirimkan melalui jalur komunikasi yang tidak aman. Sebagai contoh, klien mengirimkan sebuah PIN atau password.

   b. Klien *integrity*
      Penyerang bisa mengirimkan pesan kepada klien untuk memasang sebuah program yang akan mengirimkan informasi pada komputer klien pada penyerang dengan berpura-pura menjadi server.

Selain, klien yang mengalami ancaman keamanan, server juga mengalami ancaman keamanan.

2. Resiko Server

   a. Server *confidentiality*
      Server mungkin mengirimkan informasi yang sensitif kepada klien yang hanya boleh diketahui oleh klien. Namun, karena jalur komunikasi yang tidak aman, ada pihak yang mungkin melakukan *man-middle-attack* sehingga informasi tersebut dapat diketahui oleh pihak ketiga.

   b. Server *integrity*
      Penyerang bisa berpura-pura menjadi seorang klien. Kemudia meminta server untuk memanggil prosedur yang tidak semestinya dipanggil, sehingga hasil dari pemanggilan prosedur tersebut dapat diketahui oleh penyerang. Hal ini berarti penyerang memiliki akses yang penuh terhadap server karena semua prosedur yang telah disediakan untuk klien dapat dipanggila oleh penyerang. Ini juga berarti penyerang dapat mengetahui informasi penting yang dihasilkan dari pemanggilan prosedur yang memungkinan untuk menhasilkan suatu informasi penting yang hanya boleh diketahui oleh server dan klien.

Salah satu cara untuk mengatasi resiko tersebut adalah dengan melakukan authentifikasi pada server maupun klien sehingga dapat dipastikan bahwan informasi tersebut memang benar-benar berasal dari klien maupun server. Ada beberapa metode untuk melakukan authentifikasi untuk meyakinkan apakah benar pihak yang berkomunikas adalah pihak yang benar. Salah metode yang dapat digunakan adalah dengan mengamankan jalur komunikasi tersebut, artinya jalur komunikasi antara server maupun klien harus dipastikan bahwa jalur tersebut merupakan jalur yang aman. Dengan kata lain, tidak mungkin ada pihak ketiga yang bisa melakukan penyerangan, fabrikasi pesan, atau penyadapan.

Oleh karen itu, ada beberapa tujuan yang inign dicapai terkait dengan penamanan komunikasi pada RPC :

1. Jalur komunikasi yang aman antara klien dan server.
   Dengan jalur komunikasi yang aman, akan membuat penyerang akan kesulitan untuk menyadap atau melakukan penyerangan sehingga penyerang tidak dapat mengetahui informasi penting dan begitu pula penyerang tidak dapat memanggil atau memerintah klien untuk menjalankan program tertentu.

2. Klien dan server yang dapat dipercaya
   Sebelum dilakukan pengaman jalur komunikasi, penting untuk mengetahui apakah pihak yang dipercaya merupakan pihak yang benar-benar ingin diajak berkomunikasi. Hal ini perlu dilakukan suatu mekanisme sehingga baik klien maupun server dapat saling percaya.

Untuk melakukan pengaman jalur komunikasi antara server dan klien dapat dengan mengimplementasikan protokol kriptografi. Penempatan protokol kriptografi yaitu dengan menjadi layer tersendiri dalam RPC. Protokol ini akan berkerja terlebih dahulu untuk melakukan proses authentifikasi dan pengamanan jalur sebelum dilakukan proses pemanggilan prosedur antara klien dan server.

Ada beberapa macam teknologi RPC, antara lain *Java RMI, CORBA*, dan *XML-RPC*Pada implementasi RPC digunakan java-RMI(*java-Remote Method Invocation*).

III. PROTOKOL KRIPTOGRAFI

Protokol kriptografi merupakan salah satu aturan yang berguna agar kerbelangsungan suatu komunikasi dapat berjalan secara aman. Aman dalam arti tidak ada pihak yang sebenarnya tidak berhak mengetahui informasi yang mengalir pada komunikasi diketahui.

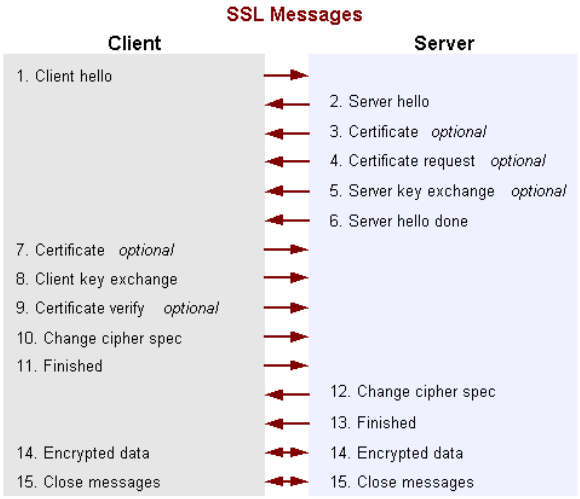
Protokol kriptografi ini biasanya memanfaatkan lebih dari satu algoritma kriptografi. Ada banyak protokol kriptografi yang ada, salah satunya adalah TLS(*Transport Layer Security*) atau SSL(*Secure Socket Layer*) yang merupakan bagian dari *transport layer*.

Pada umumnya protokol kriptografi dijalankan sebelum terjadi proses transaksi antara dua belah pihak. Hal ini dimaksudkan agar jalur komunikasi antara dua belah pihak menjadi aman. Pada proses pengamanan ini terjadi beberapa proses tapi pada umumnya ada dua tahap proses yaitu pertukaran kunci dan penetuan algoritma yang digunakan untuk proses pengiriman informasi.

Untuk pengamanan jalur komunikasi yang terjadi pada RPC dapat digunakan menggunakan TLS atau SSL. TLS memiliki kelebihan yaitu *interoperability* yang tinggi sehingga dapat digunakan dalam program dengan tidak mengganggu kode yang lain. Selain itu, TLS juga mampu membuat jalur komunikasi yang aman dan terjamin sehingga komunikasi dapat berjalan aman antara dua buah mesin. Oleh karena itu, TLS dapat diterapkan sebagai protokol yang akan mengamankan komunikasi antara klien dan server pada komunikasi *remote method invocation*.

IV. TRANSPORT LAYER SECURITY

TLS atau *transport layer security* adalah protokol kriptografi yang sering digunakan. TLS berjalan di atas TCP/IP layer. Proses yang terjadi pada pembuatan jalur komunikasi yang aman pada TLS adalah berikut ini :



Gambar 3. Protokol SSL

1. Klien akan mengirimkan pesan “hello” ke server dengan menginformasikan versi SSL yang tertinggi yang dimiliki oleh klien.
2. Server membalas dengan pesan ‘hello’ dengan memilih versi SSL tertinggi yang akan digunakan untuk proses komunnikasi selanjutnya.
3. Server memberikan sertifikat ke klien. Sertifikat tersebut dimulai dengan sebuah kunci publik dan diakhiri dengan pihak yang berwenang mengeluarkan sertifikat tersebut. Sertifikat ini digunakan oleh klien untuk mengauthentifikasikan bahwa yang berkomunikasi dengan klien benar-benar server yang dimaksud.
4. Server jika perlu mengauthentifikasikan klien akan meminta sertifikat ke klien.
5. Jika tenyata sertifikat tidak dilakukan pertukarkan maka dilakukan pertukaran kunci dengan menggunakan sebuah algoritma seperti deffie-Hellman untuk bertukar kunci.
6. Server mengatakan jika negosiasi pesan telah selesai dilakukan.
7. Klien akan memberikan sertifikat jika diminta oleh server tapi jika ternyata server tidak meminta sertifikat ke klien, klien akan memberikan sertifikat kosong atau dengan kata lain klien tidak memberikan sertifikat.
8. Klien selanjutnya akan melakukan pertukaran kunci. Kunci ini adalah kunci yang akan digunakan untuk proses komunikasi pertukaran pesan menggunakan algoritma kunci simetri.
9. Jika klien mengirimkan sertifikat sebelumnya, klien akan mengirimkan pesan yang berisi tanda tangan digital kemudian dienkripsi menggunakan kunci private milik klien. Jika tanda tangan digital tersebut sesuai dengan yang ada disertifikat setelah di dekripsi menggunakan kunci publik milik klien yang telah diberikan ke server maka server yakin bahwa tersebut adalah klien yang benar.
10. Kemudian, klien akan mengirimkan pesan yang akan mengatakan bawha server harus berpindah menjadi mode enkripsi.
11. Selanjutnya, klien mengirimkan pesan yang berisi bahwa komunikasi selanjutnya adalah pertukaran pesan dan transaksi pesan dimulai dengan pesan dienkripsi menggunakan algoritma yang telah disepakati.
12. Jika proses transaksi pesan usai, jalur komunikasi akan dilepaskan.

Dengan protokol yang dimiliki oleh SSL, tujuan untuk mengamankan jalur komunikasi antara server dan klien pada komunikasi RPC dapat dijamin. Karena pada proses komunikasi ini, pesan telah dilakukan enkripsi menggunakan algoritma simetri sehingga pihak yang tidak tahu kunci untuk mendekripsi pesan tersebut tidak dapat mengetahui isi pesan tersebut. Sebaliknya, proses yang sebelumnya saat pertukaran sertifikat adalah mekanisme untuk mengetahui apakah kedua pihak tersebut memang benar-benar pihak yang benar. Dari sertifikat yang dipertukarkan masing-masing pihak ditantang untuk mendekripsi pesan yang dienkripsi menggunakan kunci publik masing-masing pihak. Jika salah satu pihak ternyata tidak mampu mendekripsi pesan tersebut dapat disimpulkan bahwa pihak tersebut bukanlah pihak yang asli. Sehingga proses komunikasi dapat dihentikan dan belum terjadi proses pertukaran data.

V. IMPELENTASI PROTOKOL KRIPTOGRAFI PADA *REMOTE PROCEDURE CALL(RPC)*

Pada implementasi protokol kriptografi di remote procedure call digunakan protokol kriptografi yaitu *SSL(Secure Socket Layer)* atau *TLS(Transport Layer Security)* sedangkan teknologi *remote procedure call* yang digunakan adalah *java-RMI*.

Java-RMI merupakan teknologi objek terdistribusi sehingga memungkin untuk dilakukan pemanggilan prosedur dari mesin yang berbeda. Java-RMI

memungkinkan untuk menggunakan protokol SSL pada RPC sehingga komunikasi yang terjadi pada RPC aman. Untuk merealisasikan hal tersebut, dapat memanfaatkan library yang telah dimiliki oleh java yaitu *SocketFactory*. *SocketFactory* merupakan library yang memungkinkan untuk melakukan penyesuaian komunikasi soket sebelum soket tersebut terbentuk. Dengan ini, protokol SSL dapat disisipkan sebelum soket terbentuk, karena SSL merupakan protokol yang berjalan di atas TCP/IP layer sehingga mungkin untuk dilakukan penyisipan SSL pada bagian ini. Dengan memodifikasi pada SocketFactory, secara otomatis akan memodifikasi network layer yang akan digunakan untuk komunikasi sehingga komunikasi yang berjalan pada network layer berjalan pada jalur komunikasi yang aman. Secara *default*, SokcetFactory akan memiliki koneksi soket secara umum artinya koneksi TCP/IP biasa.

Untuk implementasi SSL pada java-RMI dapat menggunakan library *Java Secure Socket Extentsion(JSSE)*. JSSE merupakan implementasi SSL pada java.

Pada pengimplementasian SSL dan RPC dibuat sebuah program yang akan memanggil sebuah prosedur yang akan mengembalikan sebuah string pada klien ketika klien melakukan RPC.

Hasil Implementasi :

Gambar 4. Server RMI

Saat server RMI berjalan, program akan di-bound pada sebuah *registry*. Registry ini semacam stub yang telah didefinisikan di atas. Untuk mengubah variabel pada RMI digunakan perintah berawalan –Djava. Pengubahan variabel pada sistem *environment* berguna untuk mengizinkan agar semua permintaan dapat dilayani oleh server. Walaupun semua dapat dilayani, tetapi koneksi soket yang akan dibentuk oleh server merupakan SSL sehingga tidak semua klien dapat terhubung dengan server.

Pemanggilan prosedur yang telah ada pada server oleh klien setelah dilakukan akan muncul seperti ini:

Gambar 5.Pemanggilan Pada Klien

Program klien akan memanggil sebuah objek prosedur pada server. Prosedur pada server tersebut akan mengembalikan sebuah string yang kemudian akan ditampilkan oleh program klien.

Pada saat implementasi kedua belah pihak harus mengetahui definisi SSL yang digunakan sehingga keduanya harus mengimplementasikan SSL.

VI. ANALISIS IMPLEMENTASI

Hasil implementasi memungkinkan pada RPC untuk dilakukan pengaplikasian protokol kriptografi untuk mengamankan jalur komunikasi antara server dan klien.

Pada java-RMI untuk pengimplementasian SSL dapat menggunakan sebuah library yaitu SocketFactory sedangkan untuk implementasi SSL pada java sudah tersedia library-nya yaitu JSSE. Pembuatan sertifikat dapat dilakukan dengan sertifikat sendiri, artinya pada saat dilakukan authentifikasi sertifikat akan merujuk pada sertifikat yang telah disepakati baik oleh server maupun klien.

Dengan mengaplikasikan SSL, jalur komunikasi antara server dan klien aman, sehingga menurunkan potensi untuk terjadi *man-in-middle attack*. Pertukaran kunci pada SSL dilakukan dengan melakukan enkripsi kunci menggunakan kunci publik yang telah diberitahu ke klien oleh klien. Dengan metode ini, seandaninya ada yang berusaha untuk melakukan penyadapan atau penyerangan pesan yang didapat menjadi tidak berarti karena tidak dapat didekripsi. Pesan tidak dapat didekripsi karena pelaku tidak memiliki kunci publik yang dapat digunakan untuk mendekripsi.

Dari penjelasan tersebut maka jalur komunikasi selanjutnya akan aman, sebab kedua belah pihak telah mengetahui kunci masing-masing yang akan digunakan untuk berkomunikasi bertukar pesan menggunakan algoritma simetri biasa. Jikapun ada yang berusaha untuk melakukan penyadapan, pesan yang diperoleh pada saat pertukaran pesan menjadi tidak berarti juga karena untuk mendekripsi pesan tersebut sang penyadap harus mengetahui algoritma yang digunakan kedua belah pihak yaitu server dan klien dan kunci yang digunakan keduanya.

Selanjutnya pada saat pertukaran sertifikat, hal ini akan menjamin bahwa kedua belah pihak benar-benar pihak yang berhak terlibat dalam komunikasi tersebut. Jika pada awal authentifikasi ternyata salah satu ada yang tidak sesuai dengan sertifikat yang diberikan ke server maka komunikasi selanjutnya antara server dan klien tidak dapat terjadi. Hal ini menunjukkan dengan SSL kedua belah pihak mampu untuk mengatahui atau mengauthentifikasi piha yang benar-benar ingin berkomunikasi dengan server maupun klien.

Dengan adanya jalur yang aman dan kemampuan masing-masing pihak untuk melakukan authentifikasi, maka jalur komunikasi pada RPC dapat dikatakan telah aman. Pemanggilan prosedur yang pada server ataupun program yang dijalankan oleh klien tidak akan ada yang mengancam keduanya dengan asumsi bahwa orang yang menjalankan mesin tersebut memang berhak untuk menjalankan atau menggunakan mesin klien dan server.

VII. KESIMPULAN

.

1. Pada RPC dapat digunakan protokol kriptografi untuk mengamankan jalur komunikasi antara server dan klien
2. Java-RMI memiliki library yang memungkinkan untuk pengimplementasian protokol kriptografi pada RPC.
3. Authentifikasi, *integrity*, *confidentiality* merupakan unsur penting untuk mengetahui pihak-pihak yang telibat dalam komunikasi dalam RPC.

VIII.REFERENCES

[1] http://www.ietf.org/rfc/rfc2246.txt.
[2] http://www.ietf.org/rfc/rfc1831.txt
[3] http://download.java.net/jdk7/docs/technotes/guides/rmi/socketfactory/SSLInfo.html
[4] http://download.java.net/jdk7/docs/technotes/guides/security/jsse/samples/index.html
[5] http://www.oracle.com/technetwork/java/jsse-136410.html.
[6] Ninghui Li, Derrick Tong, John C. Mitchell. Securing Java RMI-based Distributed Applications.University Street, West Lafayette, Standford University, Google Inc.

IX. PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 5 Mei 2011

Sidik Soleman
(13508101)