BAB 2 LANDASAN TEORI

2.1 Otentikasi dan Otorisasi

Otentikasi *(authentication)* menurut Arkills (2003) adalah jaminan kepada suatu entitas bahwa entitas yang lain merupakan dia seperti yang diklaimnya, sedangkan otorisasi *(authorizotion)* lebih menekankan apa saja yang boleh dilihat dan dilakukan oleh suatu identitas.

Banyak sistem otentikasi yang juga menawarkankan fungsi otorisasi, sehingga kedua konsep ini kadang membingungkan. Sebagai contoh pada sistem Unix tradisional file /etc/passwd berisi data otentikasi dan otorisasi. Lebih buruk lagi dengan adanya fungsi direktori. Oleh karena itulah tidak mudah ketika harus membedakan antara konsep-konsep tersebut. Akan tetapi pada intinya otentikasi adalah proses membuktikan siapa yang mengakses, sedangka otorisasi memperhatikan apa saja yang boleh dilakukan oleh *user* yang telah diotentikasi tadi (Arkills, 2003)

Kusdrianto dan Raharjo (2005) menyebutkan bahwa secara umum akses terhadap sumber daya membutuhkan dua hal, otentikasi dan otorisasi. Otentikasi adalah mekanisme untuk mengenali pengguna atau sumber daya. Otentikasi dilakukan dengan tiga hal : (1) sesuatu yang dimiliki, (2) sesuatu yang diketahui, (3) sesutau yang melekat (pada *user*). Contoh dari ketiga hal tersebut antara lain :

1. Sesuatu yang dimiliki : kunci, kartu tanda pengenal, kartu ATM, smartcard, session key, token, digital certificate.
2. Sesuatu yang diketahui : userid dan password, PIN, pass phrase.

3. Sesuatu yang melekat pada seseorang : sidik jari, wajah, retina mata, DNA.

Otorisasi adalah mekanisme untuk memberikan ijin untuk mengakses sumber daya. Biasanya ini terkait dengan klasifikasi sumber daya dan *access control*. Jadi otorisasi mengatakan siapa boleh melakukan apa.

2.2 Kerberos

Kata Kerberos bermula dari mitologi Yunani yang berisi legenda Cerberus. Cerberus adalah penjaga alam neraka yang dikuasai oleh Hades dan istrinya Persephone. Bentuk dari Cerberus memiliki lima kepala, sedangkan Apollodorus menggambarkannya sebagai suatu makhluk yang aneh dengan tiga kepala berbentuk anjing dengan ekor naga. Cerberus sering digambarkan dengan suatu makhluk kejam berkepala tiga. Orang-orang Yunani percaya ketika seseorang meninggal dunia, rohnya akan dikirim ke Hades dan kekal selamnya disana. Bagi orang-orang yang tidak baik selama hidupnya akan menjalani hukuman berat disana. Cerberus sebagai penjaga gerbang Hades, menjamin hanya roh orang-orang yang meninggal yang dapat memasuki wilayah Hades, juga menjamin bahwa roh-roh yang telah masuk kedalamnya tidak akan pernah dapat keluar lagi (Garman, 2003)

Sebagai penjaga gerbang Hades, Cerberus mengotentikasikan pihak ketiga yang dipercaya bersama-sama (*trusted third-party)*. Protokol ini menawarkan otentikasi pada jaringan yang tidak aman. Kerberus Versi 1 sampai 3 digunakan secara internal dalam Project Athena, sedangkan Versi 4 diperuntukkan untuk digunakan secara umum. Oleh karena lingkungan yang berbeda dari Project Athena, maka dibuatlah Versi 5 yang merupakan perbaikan dari Versi 4 tetapi tidak sepenuhnya saling mendukung. Kini yang disebut standar protokol Kerberos adalah Kerberos Versi 5. Selain versi gratisnya oleh MIT, Kerberos tersedia juga dalam bentuk versi komesial misalnya oleh Microsoft dan Sun (Christian, 2004).

Kerberos merupakan salah satu sistem yang menggunakan kriptografi kunci simetri 3DES dan *Advanced Encryption Standart* untuk menjaga informasi yang penting pada jaringan terbuka (Freitag, 2004).

Menurut Garman (2003), tujuan dari sistem Kerberos adalah meningkatkan keamanan dan kenyamanan sekaligus. Kerberos memberikan layanan-layanan yang didefinisikan seperti berikut :

1. Aman
   Kerberos tidak pernah melewatkan *password* dalam bentuk aslinya dalam jaringan.
2. *Single sign on*
   User hanya perlu login sekali untuk mengakses semua sumber daya yang mendukung Kerberos.
3. *Trusted third-party*
   Kerberos menggunakan *centralized authentication server* sebagai pihak ketiga yang dipercaya oleh semua sistem dalam jaringan.
4. *Mutual authentication*
   Dengan adanya *mutual authentication,* bukan hanya *user* yang meyakinkan server bahwa dia adalah *user* yang sebenarnya, tetapi *user* juga dapat percaya bahwa server yang dia hadapi adalah server yang diklaim sebagai server sesungguhnya.

2.2.1 *Principal*, *Instance*, dan *Realm* dari Kerberos

Garman (2003) juga menjelaskan mengenai *realm, principal, dan instance* Kerberos. *Principal* merepresentasikan *user* atau layanan jaringan dari suatu host. Masing-masing *principal* memiliki nama yang unik. Setiap *principal* dimulai dengan *username atau service name. Username dan service name* ini kemudian diikuti oleh *instance* yang sifatnya opsional. *Instance* ini digunakan dalam dua keadaan : untuk

service *principal* dan untuk membuat *principal* khusus yang dimasukkan dalam administrasi. Sebagai contoh, administrator dapat memiliki dua *principal* : satu, *principal* yang digunakan sehari-hari, dan satu lagi *principal* admin digunakan untuk meningkatkan hak akses.

Username dan *instance* disatukan dalam *realm* sebagai bentuk identitas yang bersifat unik. Kerberos mendefinisikannya sebagai *realm*. Melalui sebuah konvensi, nama *realm* Kerberos diberikan dari nama DNS yang diubah kedalam bentuk huruf kapital. Misalnya Wedgie International yang memiliki domain wedgie.org akan diberikan nama *realm* WEDGIE.ORG.

Walaupun konvensi menetapkan nama *realm* sama dengan nama domain, akan tetapi tetap diperbolehkan apabila hendak menggunakan nama yang lain. Harus diperhatikan pula bahwa nama *realm* bersifat case-sensitif, sehingga wedgie.org tidak sama dengan WEDGIE.ORG

Kasus lainnya menentukan *principal* yang diberikan kepada John Doe, yang bekerja di IT Department di Widgie International. *Principal* yang bisa diberikan jdoe/it.WEDGIE.ORG.

Ini adalah bentuk sederhana dari *principal* yang bisa digunakan, dan merupakan *principal* yang valid baik di Kerberos 4 maupun Kerberos 5. *Principal* tersebut merepresentasikan username dari jdoe, tanpa *instance*, dan sebuah *realm* IT.WEDGIE.ORG.

Mutual authentication di Kerberos mengijinkan otentikasi di kedua belah pihak. Oleh sebab itu bukan hanya *user* yang diberikan *principal*, tetapi host dan server juga. Service *principal* sedikit berbeda dengan *user principal*. Komponen username dalam service *principal* merepresentasikan nama service. Pada host *principal*, komponen username menunjukkan host.

Principal dalam Kerberos 4 tersusun oleh tiga komponen yaitu username, *instance*, dan *realm*. Seperti contoh *user principal* dari John Doe menjadi jdoe.admin@IT.WEDGIE.ORG

Contoh service *principal* dari host unixsvr.it.doesystem.com di *realm* IT.WEDGIE.ORG menjadi host.unixsvr@IT.WEDGIE.ORG.

Service *principal* di Kerberos 4 hanya disusun dati hostname dari server dan tidak ada komponen domain.

Format dari *principal* pada Kerberos 4 adalah sebagai berikut :

user[*.instance*]*@REALM*

service.hostname@*REALM*

Principal dalam Kerberos 5 memiliki komponen dasar yang sama seperti Kerberos 4. Sedikit perbedaan dalam komponen *instance* dimana pada Kerberos 5 dapat berisi beberapa komponen sub-*instance*. Selain itu *principal* dalam Kerberos 5 tidak menggunakan dot(.) untuk memisahkan antara komponen username dan *instance*, tetapi menggunakan forward slash (/) (Garman, 2003).

Contoh *user principal* jdoe/admin@IT.WEDGIE.ORG ini sama dengan contoh pertama pada Kerberos 4 pada sub bab sebelumnya yang menunjukkan format *principal* John Doe dengan *instance* admin.

Pada Kerberos 5, baik host maupun service *principal* memasukkan Fully *Qualified Domain Name* (FQDN) dari host dimana service terinstal. Dengan menggabungkan FQDN pada *principal*, administrator dapat memiliki lebih dari satu host dengan hostname yang sama tetapi domain yang berbeda pada *realm* yang sama. Host/unixsvr.it.wedgie.org@it .wedgie.org merupakan host unixsvr.it.wedgie.org pada *realm* it.wedgie.org.

Format *principal* Kerberos 5 secara umum yaitu component[/component][/component]...*@REALM.*

Ada dua tipe *principal* Kerberos 5 yang bisa digunakan. Tipe pertama yang biasa digunakan yaitu username [/*instance*]*@REALM*, sedangkan tipe berikutnya adalah service/fully-qualified-domain-name*@REALM.*

2.2.2 *Key distribution center* (KDC)

Masih oleh Garman (2003), *Key distribution center* atau disingkat KDC merupakan bagian penting dari sistem Kerberos. KDC terdiri dari tiga komponen logika yaitu database *principal* beserta kunci enkripsinya, *Authentication Server, dan Ticket Granting Server.*

Suatu *realm* harus memiliki paling tidak sebuah KDC. Sebaiknya mesin yang bertindak sebagai KDC terpisah dari mesin tempat diletakkannya aplikasi atau *service* yang ditawarkan untuk alasan keamanan. Lebih baik lagi menempatkan KDC pada tempat yang aman secara fisik.

Masing-masing KDC berisi database dari seluruh *principal* pada suatu *realm.* Sebagian besar *software* KDC juga menyimpan informasi tambahan, misalnya *password* lifetime, last password change, dan yang lainnya.

Hal lain yang tak kalah pentingnya yaitu konsep tiket dalam Kerberos. Secara konseptual, tiket Kerberos adalah struktur data terenkripsi yang dikeluarkan oleh KDC. Tujuan dari tiket ini yang pertama yaitu untuk memastikan kebenaran dari *end participant*. Kedua adalah untuk membangun kunci enkripsi sementara, diantara dua pihak sehingga tercipta komunikasi yang lebih aman.

2.3 LDAP

2.3.1 Pengenalan Direktori

Buku karangan Arkill(2003) yang berjudul *LDAP Directories Explained : An Introduction and Analysis* menjelaskan bahwa sebuah direktori merupakan cara yang lebih efisien untuk mencari dan mengatur informasi. Jika terdapat banyak sumber informasi untuk dicari, dimungkinkan adanya komunikasi antara satu sumber dengan sumber yang lain atau adanya informasi yang belum ter-*update*. Lebih dari itu, berpindah dari satu direktori ke direktori lain adalah hal yang sangat bisa membuat frustasi. Direktori sebaiknya diatur secara terpusat, hal ini penting sebagai sebuah otorisasi sebagai sumber informasi. Dengan cara ini tidak perlu lagi adanya pencarian dalam beberapa sumber untuk suatu informasi dan selanjutnya dapat dipastikan kebenaran informasi yang diperoleh tersebut.

Salah satu kegunaan direktori adalah menjadi sarana interaksi langsung ketika mencari informasi secara manual. Aplikasi perangkat lunak dapat mengambil informasi dalam sebuah direktori untuk memberikan suatu informasi yang disajikan dengan lebih baik dalam bentuk layanan-layanan. Layanan tersebut memberikan landasan bagaimana pihak lain dapat berinteraksi dalam dunia digital, mengenalkan *user* antara satu dengan yang lain, mengatur otorisasi, mengijinkan *user* berkomunikasi dengan *user* lain, disamping untuk menjaga informasi itu sendiri. Masing-masing dari layanan dasar yang seringkali disebut infrastruktur ini harus mempunyai informasi tentang identitas dari sumber informasi itu sendiri. Secara tegas dapat dikatakan bahwa ada sebuah keuntungan ketika memilikisebuah aset informasi yang diatur dan didefinisikan dengan jelas apa dan bagaimana metode akses data tersebut.

Sebuah direktori biasanya berisi entri-entri yang statis dengan frekuensi perubahan yang sedikit karena direktori di desain untuk memberikan respon sangat cepat dalam pencarian informasi. Sebuah database sering berisi entri-entri yang frekuensi perubahannya tinggi. Database didesain untuk penyajian data yang mudah, mendukung manipulasi data, serta mendukung proses baca dan tulis data yang terus menerus (Arkills, 2003).

Masing-masing teknologi mempunyai kelebihan dan kekurangan. Database baik pada penyimpanan objek yang memiliki pengurutan dengan banyak cara. Database biasanya mengimplementasikan sebuah mekanisme penguncian untuk mencegah dua sistem dari penulisan informasi yang sama, sedangkan direktori tidak. *Query* yang kompleks biasanya lebih banyak dalam sebuah basisdata daripada direktori. Database mengatur data yang besar dengan sangat baik, dimana direktori tidak didesain untuk tujuan ini. Database mengijinkan seseorang untuk menyimpan prosedur proses yang efisien dari permintaan yang komplek.

Di sisi lain, banyak ahli teknologi informasi mengetahui bahwa pengimplementasian direktori penting. Dalam buku yang sama, diuraikan keuntungan dari direktori yaitu :

1. Membuat administrasi jaringan lebih mudah
   - pengaturan secara terpusat untuk informasi *user*.
   - pengaturan secara terpusat pada konfigurasi mesin oleh komputer.
   - pengaturan terpusat dari akun pengguna.
   - mengurangi biaya dari dukunganpengaturan terpusat yang telah diterapkan.
2. Keseragaman akses untuk sumber daya jaringan :
   - penggunaan penamaan yang seragam.
   - potensial untuk Single SignOn pada sumber-sumber dalam jaringan.

3. Pencarian informasi pada satu tujuan
   - informasi kontak
   - lokasi sumber daya jaringan
   - potensial sebagai katalog sejumlah data
4. Improvisasi pengaturan data
   - meningkatkan konsistensi data yang biasa digunakan secara luas
   - menyediakan pengaturan keamanan terpusat untuk data bisnis yang penting.
   - mengatur data dalam struktur logika.
5. Membantu kelancaran proses bisnis
6. Memberikan tempat penampungan dan pencarian untuk aplikasi dan layanan data.

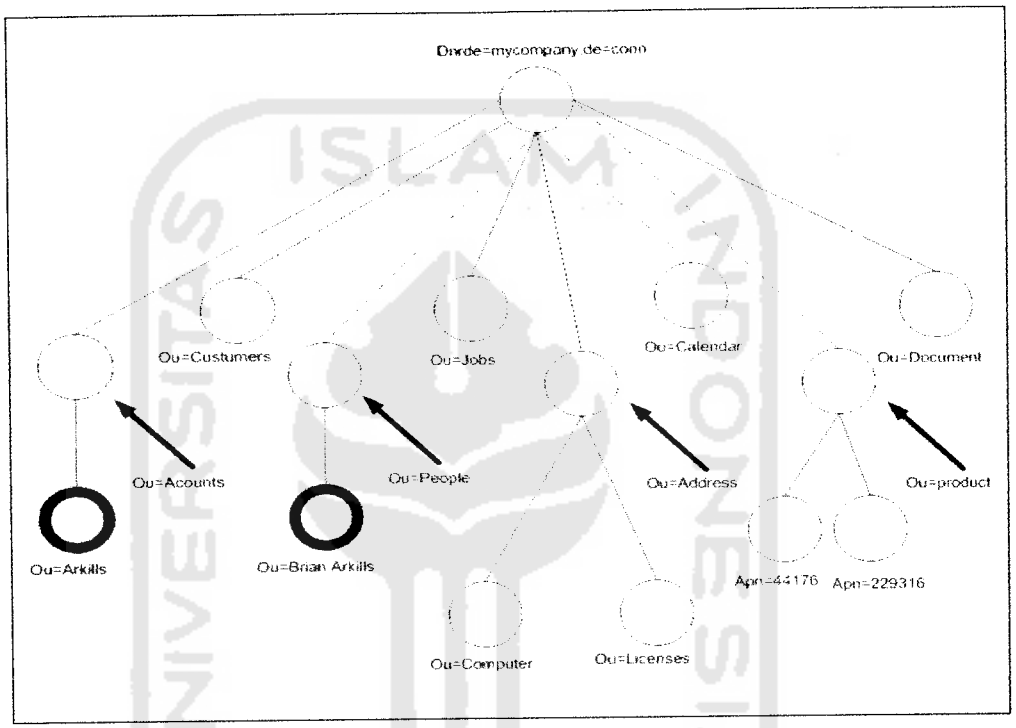
2.3.2 Pengenalan LDAP

2.3.2.1 Namespace

Setiap direktori membutuhkan *namespace* untuk memberikan nama suatu entri. Menurut Arkill (2003) secara umum *namespace* pada LDAP adalah sistem yang digunakan untuk mereferensikan suatu objek dalam direktori LDAP atau dengan kata lain namespace merupakansuatu sistem penamaan.

Namespace diorganiasasikan dalam sebuah hirarki sehingga pengaturannya dapat didelegasikan pada beberapa titik dalam struktur hirarki. Hirarki adalah turunan dari namespace yang memberiakn arti efektif pada delegasi koorperatif dari manajemen (Arkill, 2003). Hal ini merupakan keutungan yang signifikan dari LDAP dibanding database dan hal tersebut dijadikan salah satu faktor utama dalam memutuskan bagaimana data diorganisasikan dalam suatu direktori. Banyak dari direktori yang digunakan untuk membagi namespace, sebagai contoh yang terjadi misalnya sebuah standar internet yaitu DNS *(Domain Name System)*. DNS secara alami didefinisikan secara hirarki. *Namespace* dari LDAP juga berupa hirarki

sehingga hampir sama dengan DNS. Banyak direktori LDAP menggunakan *namespace* dari DNS, sehingga *namespace* dari LDAP bekerja seperti DNS. Hal ini membantu membuat LDAP lebih atraktif dan memberikan kontribusi pengembangan masa depan dari direktori LDAP yang terintegrasi secara global.



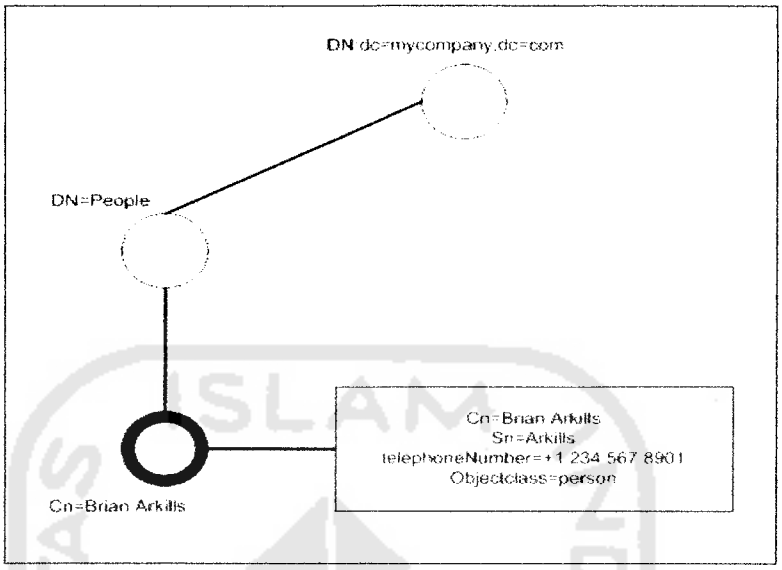
Gambar 2.1 Contoh Hirarki pada *Namespace* LDAP (Arkills, 2003)

Gambar 2.1 menunjukkan sebuah versi sederhana dari direktori mycompany.com. Nama dari direktori utama dikenal sebagai basis DN (*Distinguish Name*). Direktori utama tidak membutuhkan entri. Server berbasis DN biasanya mencocokan nama DNS dari server direktori dan menggunakan atribut *Domain Component* (DC) untuk merepresentasikan zona DNS. Tetapi bagaimanapun juga, server yang berbasis DN tidak perlu disamakan dengan nama server DNS. Server direktori yang berbasis DN mungkin berbeda untuk kepentingan fleksibilitas yang lebih besar dalam membuat desain dari sebuah arsitektur direktori yang terdistribusi di atas beberapa server direktori. Fleksibilitas untuk membuat sebuah direktori yang

terdistribusi menggunakan *namespace* adalah kunci keuntungan LDAP dibanding database.

Masing-masing entri dalam direktori mempunyai sebuah nama unik yang diketahui sebagai *Distinguish Name(*DN*).* Masing-masing entri juga mempunyai sebuah nama lokal yang disebut dengan *Relative Distinguish Name(*DN*).* Masing-masing entri juga mempunyai sebuah nama lokal yang disebut dengan *Relative Distinguish Name* / RDN (Arkill, 2003). RDN unik untuk semua entri dalam *container* tersebut. Sebuah *container* dapat diumpamakan seperti sebuah direktori atau folder dalam sebuah system file. Menurut Arkills (2003), RDN adalah tipe atribut dan bagian nilainya.

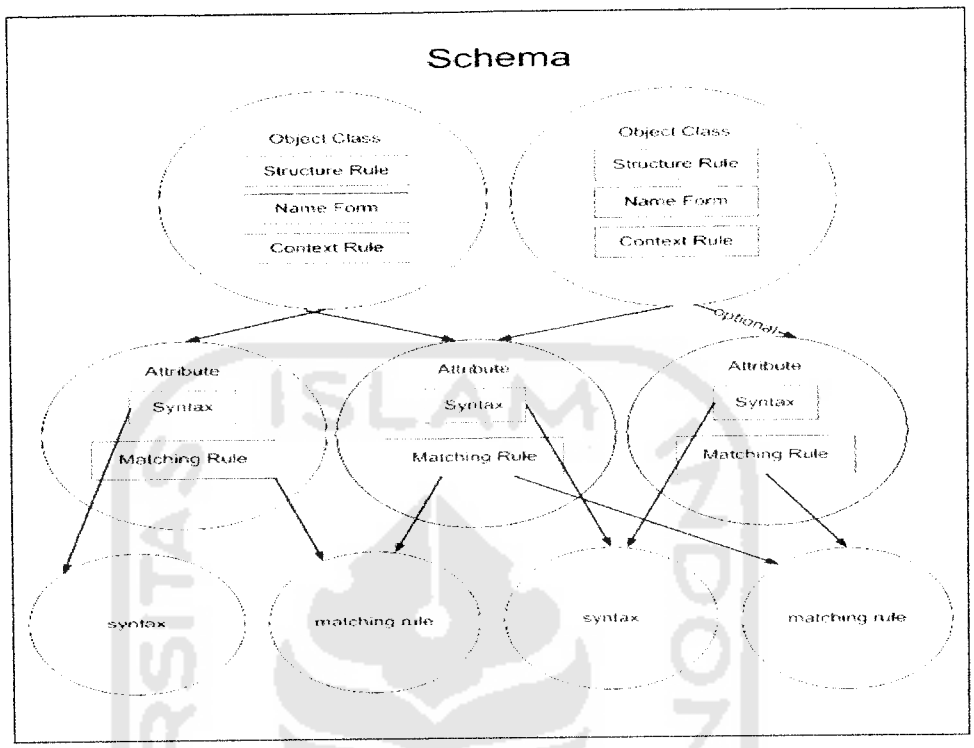
Entri DN dari person seperti yang ditunjukkan oleh gambar 2.2 dapat menjadi cn=Brian Arkills, ou=peole, de=mycompany, dc=com, Perlu diperhatikan bahwa masing-masing komponen RDN di sini termasuk keduanya yaitu atribut dan nilainya. Sebagai contoh, komponen tunggal cn=Briian Arkills mempunyai atribut cn dan nilai atribut Brian Arkills. Nilai atribut tanpa tipe atribut tidak sesuai untuk entri distinguish, karena nilainya mungkin mengacu ke tipe atribut yang berbeda pada banyak entri. Catatan bahwa cn=Brian Arkills harus unik dari semua entri dalam *container* ou=People agar termasuk kualifikasi RDN.



Gambar 2.2 Contoh RDN (Arkills, 2003)

2.3.2.2 Schema

Arkills (2003), *schema* mendefinisikan aturan-aturan yang mengatur hal-hal apa saja yang bisa dilakukan oleh direktori LDAP. *Schema* memegang peranan penting, tetapi *user* tidak perlu tahu apa saja dilakukan oleh *schema.* Di samping itu, *schema* juga menentukan tipe data entri yang bisa dibuat. Memodifikasi schema dapat meningkatkan fungsionalitas direktori. Memodifikasi tersebut misalnya menambahkan tipe baru pada objek atau membuat tipe baru pada atribut.



Gambar 2.3 Diagram Konsep Schema Arkills (2003)

Schema terdiri dari beberapa. Gambar 2.3 menggambarkan bagaimana masing-masing elemen schema saling berelasi. Dari gambar tersebut bisa diketahui bahwa terdapat interdependensi antar elemen, dalam faktanya masing-masing elemen bisa saja bergantung pada beberapa elemen yang lain.

Kelas objek mendefinisikan tipe-tipe entri yang diperbolehkan dalam sebuah direktori. Kelas objek berisi :

1. *content rule* : mendefinisikan atribut-atribut pada kelas objek.

2. *structure rule* : menjelaskan bagaimana masing-masing kelas objek dapat berpartisipasi dalam *name space.*

3. *name form* : menjelaskan atribut apa saja yang busa digunakan untuk memberikan nama entri pada kelas objek.

4. *attribute* : mendefinisikan jenis informasi yang diasosialisasikan dengan masing-masing kelas objek, dan kemudian di dalam entri.

5. *attribute type* : didefinisikan melalui sintaks, matching rules, dan informasi operasional yang lain.

6. *syntax* : menentukan bagaimana nilai data direpresentasikan.

7. *matching rule* : menentukan bagaimana membandingkan nilai-nilai data tersebut pada operasi LDAP.

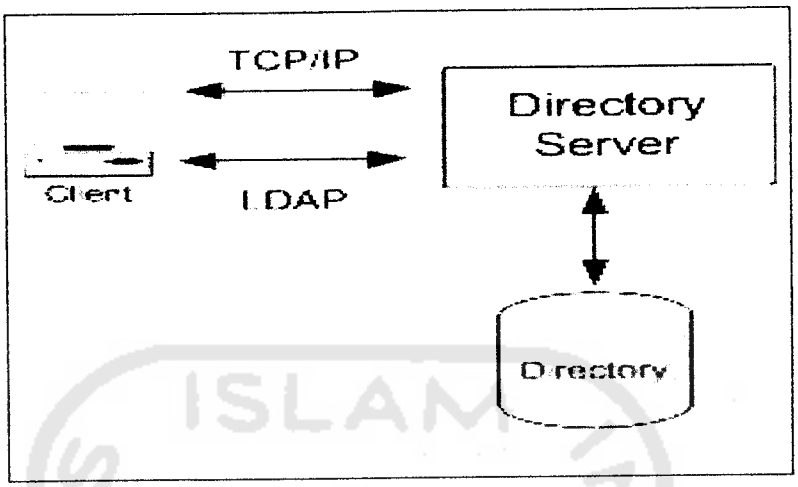
2.3.2.3 Interaksi *client*

1. Model *client*-server

LDAP menggunakan TCP/IP untuk berkomunikasi. Sebuah *client* agar dapat tersambung ke sebuah direktori LDAP. LDAP mengurangi kelebihan beban untuk membuat sebuah sesi yang mengijinkan banyak operasi dari sesi *client* yang sama. Hal tersebut juga memberikan trafik yang efisien dari penempatan karena hampir semua data yang disimpan dalam direktori adalah berupa informasi teks. LDAP menggunakan BER *encording* untuk mengacak nilai data atribut yang melewati antara server dan *client*. Ini berupa metode pengacakan komplek yang diadopsi dari X.500 (Priantoro, 2005).

Arkills (2003) menyatakan bahwa set dari operasi LDAP saling berhubugan dengan set dari standar *application programming interface* (API) dalam bahasa yang berbeda. API adalah sebuah set dari fungi-fungsi ini memberikan sebuah level yang lebih tinggi dalam penyampaiannya dengan menyembunyikan kode-kode yang susah didalamnya. Beberapa API berupa sumber yang tertutup *(closed source)*, artinya kodc-kode dari fungsi tersebut disembunyikan dari semua orang. Lainnya adalah kode yang terbuka (*open source*), dalam arti bahwa setiap orang dapat melihat detail dari code tersebut. API dan LDAP semuanya adalah *open source* (Arkills,2003). Sebagai contoh fungsi dari LDAP API, yang membuat operasi penambahan entri. Ada standar fungsi LDAP yaitu ldap_add() dalam standar API bahasa C dimana sebuah aplikasi akan menggunakannya untuk meminta server melakukan operasi penambahan. Semua aplikasi yang menggunakan API dari LDAP untuk berinteraksi dengan server LDAP disebut juga LDAP-*enabled.*

Gambar 2.4 menggambarkan server LDAP pada lingkungan TCP/IP. Gambar tersebut menunjukkan interaksi antara klien dengan directory server melalui TCP/IP yang kemudian directory server tersebut berinteraksi terhadap direktori LDAP sesuai perintah yang diberikan oleh klien.



Gambar 2.4 Server LDAP pada TCP/IP Environment (Volgmaier, 2004)

2. Client

Client LDAP dapat berupa perangkat lunak tunggal (standalone) dimana seseorang berinteraksi dengan mengetikkan perintah yang dibutuhkan, ataupun dapat berupa bagian mengetikkan perintah yang dibutuhkan, ataupun dapat berupa bagian yang terintegrasi dengan perangkat lunak yang mempunyai operasi otomatis dan perintah yang digunakan tersembunyi dari pengguna. Sebagai contoh sistem operasi Windowa 2000 telah mengintegrasikan fungsi-fungsi klien LDAP ke dalam beberapa bagain dari aplikasi ini. Dalam windows 2000, dapat dilakukan pemilihan opsi pencarian dari menu start, dan mencari data people (Outlook,Situs, dll) atau printer dalam Microsoft Active Directory (atau dengan kata lain server LDAP). Ada banyak situs LDAP-enabled yang memberikan sebuah interface tunggal (seringkali disebut portal) untuk seseorang yang menggunakan pencarian dan pemutahiran entri dalam server LDAP dari perusahaannya. Sebagai tambahan ke situs tersebut, hampir semua browser yang modern mendukung protokol LDAP dan mampu sebagai klien untuk mengambil informasi dari LDAP. Fleksibilitas dari integrasi adalah salah satu alasan utama kenapa banyak perusahaan perangkat lunak yang mengadopsi protokol LDAP. Keindahan dari LDAP standar terbuka menjadi dikenal direalisasikan bahwa setiap

klien LDAP atau aplikasi LDAP-enabled dapat dengan sukses berkomunikasi dengan setiap server LDAP, tidak berdasar pada bagian sistem operasi klien ataupun server. Standar terbuka, model multipleplatform membuat integrasi mudah, tetapi sebuah pasar dapat membuat integrasi menjadi sulit. Ini berarti bahwa implementasi dari LDAP adalah komplek, lingkungan sistem operasi yang tidak homogen secara signifikan lebih mudah daripada mengimplementasikan teknologi lain.

3. Operasi

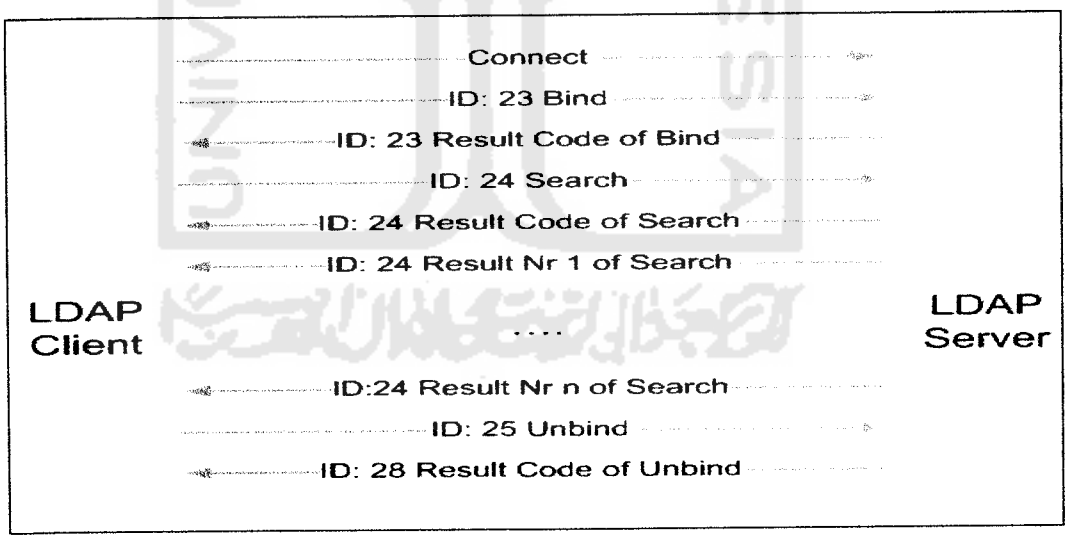
Ada sepuluh operasi dalam LDAP. Terbatasnya jumlah dari operasi sangat penting, pada program client interaksi dengan direktori sangat lebih mudah daripada program client yang berinteraksi dengan teknologi lain yang hampir sama. Operasi-operasi tersebut dapat digolongkan dalam tiga kategori dasar, seperti ditunjukkan dalam tabel 2.1.

Tabel 2.1 Operasi pada LDAP (Arkills, 2003)

Kategori	**OperasiLDAP**
Operasi sesi client	*Bind, unbind, dan abandon*
Query dan operasi pengambilan	*Search dan compare*
Operasi modifikasi	*Add, modify, modifyRDN dan delete*
Tambahan	Tambahan

Operasi tambahan adalah unik untuk seluruh operasi. Operasi tambahan sebagai tempat untuk emplementasi direktori yang spesifik untuk menambah fungsi dari protokol yang masih mempunyai sebuah sintaks yang didefinisikan terlebih dahulu. Perancang LDAP menunjukkan banyaknya perhatian dengan menambahkan operasi tambahan. Dengan melakukan standarisasi untuk melakukan penambahan fungsi operasional, mereka mengurangi persepsi yang salah dalam jumlah yang terbatas dari operasi. Operasi sesi client membantu untuk mengontrol konteks sesi client server untuk semua *request subsequent* operasi LDAP dari client tersebut. Operasi bind dan unbind mengijinkan client untuk menyambung identitas dengan LDAP. Identitas ini dapat digunakan oleh direktori dalam menentukan hak akses untuk melakukan operasi lain, dan dapat digunakan untuk mengatrol akses ke informasi direktori. Operasi abandon mengijinkan client membatalkan sebuah operasi *request*. Operasi *query* mengijinkan client melihat informasi dalam direktori. Hampir semua pembaca LDAP yang baru membutuhkan pengetahuan bagaimana untuk menggunakan pencarian cerdas pada sebuah direktori. Operasi pencarian adalah operasi yang paling sering dilakukan, keahlian dalam menggunakannya akan lebih bernilai. Operasi pencarian mempunyai banyak parameter, faktanya, parameter-parameter lebih banyak dari operasi lain. Operasi pembandingan mengijinkan sebuah client untuk me-*request* sebuah verifikasi dari informasi yang diasosialisasikan dengan sebuah entri, client mengirimkan nilai dari entri dan server merespon dengan sukses jika cocok atau gagal jika tidak cocok. Operasi modifikasi mengijinkan client untuk mengubah informasi dalam direktori. Operasi ini mungkin terbatas dalam beberapa *instant,* sebagai contoh dalam suatu kasus dari sebuah direktori yang *read-only*. Operasi modifyRDN hanya satu dalam kebutuhan dari kumpulan penjelasan. Operasi modifyRDN mengijinkan client untuk mengubah nama dari sebuah entri dan mungkin memindahkan entri ke kontainer yang berbeda.

Gambar 2.5 mengilustrasikan proses-proses ini. Gambar 2.5 memberikan contoh proses pencarian serta memberikan gambaran apa yang terjadi. Berawal dari client membuka koneksi ke komputer dimana server LDAP terinstal, kemudian client melakukan bind terhadap server LDAP. Proses bind ini mengkualifikasi client menggunakan ID dan password *user* yang valid. Apabila ID dan password *user* tidak server terima, maka server mengasumsikan bahwa client meminta koneksi sebagai *anonymous user, user* dengan level akses terendah jika ada. Server menjawab dengan kode hasil untuk menunjukkan proses bind sukses. Selanjutnya client mengirimkan query ke server, Sekali lagi server mengeksekusi request tersebut dan mengirimkan kembali data yang diminta apabila ada, juga mengirimkan kode hasil dari operasi pada pesan yang terpisah. Apabila server menemukan lebih dari satu hasil, maka akan dikirimkan dengan sejumlah pesan LDAP. Di akhir percakapan, client mengirimkan *request unbind* sehingga server menutup koneksi (Voglmaier, 2004).



Gambar 2.5 Komunikasi Antara Client LDAP dan Server

2.3.2.4 Keamanan

Voglmaier (2004) dalam bukunya yang berjudul *The ABCs of LDAP: How to Install, Run, and Administer LDAP Service,* menyebutkan bahwa sebelum client dapat mengakses data yang ada pada server LDAP, client harus menyelesaikan dua proses terlebih dahulu, yaitu otentikasi dan otorisasi. Dua proses ini sedikit berbeda antara satu sama lain. Fokus kali ini lebih kepada otentikasi oleh karena otorisasi belum terbahas secara tuntas dalam standar.

Otentikasi terjadi ketika client mengidentifikasikan dirinya ke server agar bisa terhubung. Proses ini tergantung dari mekasnisme otentikasi yang digunakan. Cara paling mudah agar terhubung ke server adalah tanpa perlu menunjukkan identitas. Cara ini disebut *anonymous connection* dengan hak akses terendah. Ada beberapa skema otentikasim dari *simple authentication* dengan *user dan password,* hingga otentikasi menggunakan sertifikat. Sertifikat yang dimaksud disini memeberikan jaminan kepada server bahwa client benar-benar merupakan client yang mereka kliam. Sertifikat-sertifikat ini juga mampu menjamiun identitas server bagi client (Voglaier, 2004).

Begitu client dikenali oleh server, client akan mendapatkan hak akses terhadao data. Otorisasi merupakan proses dimana server mengijinkan hak akses yang sesuai kepada client yang sebelumnya telah diotentikasi. Ini berarti *user* dapat membaca dan menulis data dengan batasan-batasan yang tergantung dari level akses yang dimilikinya. Untuk mendefinisikan apa saja yang bisa dilakukan client, server harus memelihara *access control information (ACI).*

Menurut Voglmaier (2004), ada beberapa level otentikasi yang berbeda sehingga muncul beberapa metode yang bermacam-macam pula untuk mengotentikasi client, yaitu :

1. Anonymous Access

   Jenis pertama dari otentikasi adalah tidak ada otentikasi sama sekali, atau bisa juga disebut *anonymous bind* karena server tidak tahu menahu siapa yang sedang meminta koneksi. *Anonymous bind* digunakan oleh data publik semisal buku telepon publik. Konfigurasi server menentukan apakah akses *anonymous* diijinkan atau tidak.

2. Basic Authentication

   Selain akses *anonymous,* otentikasi yang juga sederhana adalah *basic authentication* yang biasanya digunakan oleh protokol HTTP. *Client* secara sederhanan mengirimkan *user credential* melalui jaringan. Dalam LDAP ini berarti *client* mengirimkan *distinguished nama dari user dan passwordnya.* Kedunya dikirimkan melalui jaringan dalam bentuk *plaintext* tanpa enkripsi. Metode ini tidak masalah untuk lingkungan yang terjamin kemanannya, tetapi dalam intranet sekalipun hal ini bukanlah ide yang bagus.

   Server mencari atribut *user*Password dalam entri yang sesuai dengan *distinguish name.* Nilai dari atribut ini dicocokkan dengan input yang diberikan *user*. Apabila password tersebut sesuai, koneksi dapat dibangun ke server LDAP. Sekali lagi, ini bukanlah metode paling aman diantara metode-metode yang lain, tetapi dapat juga diterima apabila diimplementasikan di dilingkungan intranet.

3. LDAP over SSL/TLS

   Protokol SSL (Secure Socket *Layer*) mengimplementasikan mekanisme-mekanisme keamanan pada lapisan protokol TCP/IP yaitu antara *transport layer dan aplication layer,* dalam contohnya adalah *layer* dibawah LDAP. SSL didasarkan pada kriptografi public key dan dikembangkan oleh Netscape. Protokol TLS (Transport *Layer* Security) diciptakan dari SSL versi tiga.

   Tujuan dari protokol TLS itu sendiri adalah untuk menyediakan integritas privasi data. Ini berarti protokol TLS menjamin data yang dikirim antara dua

pihak sampai tanpa ada modifikasi dan komunikasi di antara mereka dijamin terenkripsi.

4. Kerberos

LDAP (v2) mendukung mekanisme *bind* berdasar pada Kerberos. Akan tetapi ini tidak dapat didukung secara langsung pada LDAP(v3). Tidak didukung secara langsung di sini artinya bahwa mekanisme bind ini dapat digunakan sebagai mekanisme keamanan ketika ada persetujuan yang terbangun oleh protokol SASL. Kerberos itu sendiri merupakan protokol yang bergantung pada pihak ketiga, *Authentication Server*. Kerberos adalah mekanisme keamanan yang direncanakan bagi lingkungan yang benar-benar tidak aman misalnya saja internet. Kerberos tidak berasumsi mengenai integritas pesan terkirim antara dua pihak yang sedang berkomunikasi. Komunikasi dienkripsikan dan kedua pihak yang berkomunikasi dapat yakin akan identitas pihak lainnya.

Protokol Kerberos sangat stabil dan handal untuk memenuhi kebutuhan keamanan khususnya dalam komunikasi di lingkungan yang tidak aman. Dapat juga digunakan bersama dengan banyak protokol dan merupaka *platform independent.*

5. SASL

*Simple Authentication and Security Layer (*SASL*)* merupakan suatu metode penyediaan layanan otentikasi bagi protokol-protokol yang *connection oriented* seperti misalnya LDAP. Standar SASL didefinisikan pada RCF 2222, SASL. Standar ini memungkinkan klien dan server menyetujui suatu *layer* keamanan untuk enkripsi. Setelah server dank lien terhubung, mereka menyetujui suatu mekanisme keamanan untuk percakapan selanjutnya. Salah satu mekanisme ini adalah Kerberos.

Pada saat buku tersebut ditulis, sejumlah mekanismeyang didukung SASL, meliputi :

- *Anonymous*
- CRAM-MD5
- Digest-MD5
- *External*
- Kerberos (v4)
- Kerberos (v5)
- SecureID
- Secure Remote Password
- S/Key
- X.509

Otentikasi member *credential* kepada server yang dibutuhkan agar *user* diijinkan mengakses server. Selain itu juga untuk memverifikasi identitas dari pihak lain, baik dari sisi server maupun sisi klien. Begitu koneksi terjalin, klien dan server dapat bertukar pesan dengan utuh tanap ada modifikasi ataupun penangkapan pesan dari pihak yang tidak berhak.

Otorisasi atau bisa juga disebut *access control* adalah proses dimana server mengijinkan hak akses yang sesuain bagi *user* yang terkoneksi. Ketika buku tersebut dibuat, standar mengenai otorisasi belum dibahas secara luas. Ini bukan berarti server direktori tidak mendukung otorisasi. *Access Control Information* (ACI) terdapat dalam *access control list (*ACL*).*

Access Control Information dapat berisi spesifikasi-spesifikasi berikut ini:

a. Data Protection : server dapat mengijinkan ataupun manolak akses ke direktori maupun DN.

b. Data Access : server dapat menentukan klien-klien yang memiliki akses ke sistem

- *Anonymous* : semua *user*, tanpa otentikasi

- *Authenticated user* : user-user yang telah diotentikasi oleh sistem.
- *Self* : user yang berhubungan dengan entri tujuan.
- *Distinguished name* : user yang sesuai dengan ekspresi pada distinguished name

c. *Level of access*

- No.access
- Hak untuk *bind*
- Hak untuk eksekusi atau mambandingkan
- Hak untuk eksekusi pencarian
- Hak untuk membaca hasil pencarian
- Hak untuk memodifikasi entri

- *Further Requirement* : server dapat membatasi nomor IP ataupun nama *host* yang dilayani, waktu layanan dapat diijinkan atau ditolak, dan lainnya.