Diterjemahkan dari bahasa Inggris ke bahasa Indonesia - www.onlinedoctranslator.com

8.1 PENGANTAR KEAMANAN

Kami memulai penjelasan kami tentang keamanan dalam sistem terdistribusi dengan melihat beberapa masalah keamanan umum. Pertama, penting untuk mendefinisikan apa itu sistem yang aman. Kami membedakan keamanan*kebijakan*dari keamanan*mekanisme*, dan lihat sistem area luas Globus yang kebijakan keamanannya telah dirumuskan secara eksplisit. Perhatian kedua kami adalah mempertimbangkan beberapa masalah desain umum untuk sistem yang aman. Terakhir, kami membahas secara singkat beberapa algoritma kriptografi, yang memainkan peran penting dalam desain protokol keamanan.

8.1.1 Ancaman, Kebijakan, dan Mekanisme Keamanan

Keamanan dalam sistem komputer sangat terkait dengan gagasan ketergantungan. Secara informal, sistem komputer yang dapat diandalkan adalah sistem yang kita percayai untuk memberikan layanannya (Laprie, 1995). Seperti disebutkan dalam Bab. 7, ketergantungan mencakup ketersediaan, keandalan, keamanan, dan pemeliharaan. Namun, jika kita ingin menaruh kepercayaan pada sistem komputer, maka kerahasiaan dan integritas juga harus diperhitungkan.**Kerahasiaan**mengacu pada properti sistem komputer dimana informasinya diungkapkan hanya kepada pihak yang berwenang.**Integritas**adalah karakteristik bahwa perubahan terhadap aset suatu sistem hanya dapat dilakukan dengan cara yang sah. Dengan kata lain, perubahan yang tidak patut pada sistem komputer yang aman harus dapat dideteksi dan dipulihkan. Aset utama dari sistem komputer mana pun adalah perangkat keras, perangkat lunak, dan datanya.

Cara lain dalam memandang keamanan dalam sistem komputer adalah kami berupaya melindungi layanan dan data yang ditawarkannya**ancaman keamanan**. Ada empat jenis ancaman keamanan yang perlu dipertimbangkan (Pfleeger, 1997):

1. Intersepsi
2. Interupsi
3. Modifikasi
4. Fabrikasi

Intersepsi mengacu pada situasi di mana pihak yang tidak berwenang memperoleh akses ke layanan atau data. Contoh umum intersepsi adalah ketika komunikasi antara dua pihak didengar oleh orang lain. Intersepsi juga terjadi ketika data disalin secara ilegal, misalnya, setelah membobol direktori pribadi seseorang di sistem file.

Contoh interupsi adalah ketika suatu file rusak atau hilang. Secara umum, interupsi mengacu pada situasi di mana layanan atau data menjadi tidak tersedia, tidak dapat digunakan, hancur, dan sebagainya. Dalam pengertian ini, serangan penolakan layanan yang dilakukan seseorang dengan niat jahat untuk membuat layanan tidak dapat diakses oleh pihak lain adalah a

ancaman keamanan yang tergolong gangguan.

Modifikasi melibatkan perubahan data tanpa izin atau gangguan pada layanan sehingga tidak lagi sesuai dengan spesifikasi aslinya. Contoh modifikasi termasuk menyadap dan kemudian mengubah data yang dikirimkan, merusak entri database, dan mengubah program sehingga secara diam-diam mencatat aktivitas penggunanya.

Fabrikasi mengacu pada situasi di mana data atau aktivitas tambahan dihasilkan yang biasanya tidak ada. Misalnya, penyusup mungkin mencoba menambahkan entri ke file kata sandi atau database. Demikian pula, terkadang ada kemungkinan untuk membobol sistem dengan memutar ulang pesan yang dikirim sebelumnya. Kita akan menemukan contoh-contoh seperti itu nanti di bab ini.

Perhatikan bahwa interupsi, modifikasi, dan fabrikasi masing-masing dapat dilihat sebagai bentuk pemalsuan data.

Sekadar menyatakan bahwa suatu sistem harus mampu melindungi dirinya sendiri terhadap semua kemungkinan ancaman keamanan bukanlah cara untuk membangun sistem yang aman. Hal pertama yang diperlukan adalah uraian persyaratan keamanan, yaitu kebijakan keamanan. A **kebijakan keamanan**menggambarkan dengan tepat tindakan mana yang boleh dilakukan oleh entitas dalam suatu sistem dan tindakan mana yang dilarang. Entitas mencakup pengguna, layanan, data, mesin, dan sebagainya. Setelah kebijakan keamanan ditetapkan, kita dapat berkonsentrasi pada hal tersebut**mekanisme keamanan**dimana suatu kebijakan dapat ditegakkan. Mekanisme keamanan yang penting adalah:

1. Enkripsi
2. Otentikasi
3. Otorisasi
4. Audit

Enkripsi sangat penting untuk keamanan komputer. Enkripsi mengubah data menjadi sesuatu yang tidak dapat dipahami oleh penyerang. Dengan kata lain, enkripsi menyediakan sarana untuk menerapkan kerahasiaan. Selain itu, enkripsi memungkinkan kita memeriksa apakah data telah diubah. Oleh karena itu, ini juga memberikan dukungan untuk pemeriksaan integritas.

Otentikasi digunakan untuk memverifikasi identitas yang diklaim dari pengguna, klien, server, dan sebagainya. Dalam kasus klien, premis dasarnya adalah bahwa sebelum layanan melakukan pekerjaan untuk klien, layanan harus mempelajari identitas klien. Biasanya, pengguna diautentikasi melalui kata sandi, namun ada banyak cara lain untuk mengautentikasi klien.

Setelah klien diautentikasi, penting untuk memeriksa apakah klien tersebut berwenang untuk melakukan tindakan yang diminta. Akses ke catatan dalam database medis adalah contoh umum. Tergantung pada siapa yang mengakses database, izin dapat diberikan untuk membaca catatan, untuk mengubah bidang tertentu dalam catatan, atau untuk

menambah atau menghapus catatan.

Alat audit digunakan untuk melacak klien mana yang mengakses apa, dan dengan cara apa. Meskipun audit tidak benar-benar memberikan perlindungan apa pun terhadap ancaman keamanan, log audit bisa sangat berguna untuk menganalisis pelanggaran keamanan, dan selanjutnya mengambil tindakan terhadap penyusup. Oleh karena itu, penyerang umumnya tidak meninggalkan jejak apa pun yang pada akhirnya dapat mengungkap identitas mereka. Dalam hal ini, akses logging terkadang membuat serangan menjadi bisnis yang lebih berisiko.

Contoh: Arsitektur Keamanan Globus

Gagasan tentang kebijakan keamanan dan peran mekanisme keamanan dalam sistem terdistribusi untuk menegakkan kebijakan tersebut sering kali paling baik dijelaskan dengan melihat contoh nyata. Pertimbangkan kebijakan keamanan yang ditetapkan untuk sistem wilayah luas Globus (Chervenak et al., 2000). Globus adalah sistem yang mendukung komputasi terdistribusi skala besar di mana banyak host, file, dan sumber daya lainnya digunakan secara bersamaan untuk melakukan komputasi. Lingkungan seperti ini juga disebut sebagai grid komputasi (Foster dan Kesselman, 1998). Sumber daya dalam jaringan ini sering kali berlokasi di domain administratif berbeda yang mungkin berlokasi di berbagai belahan dunia.

Karena jumlah pengguna dan sumber daya sangat besar dan tersebar luas di berbagai domain administratif, keamanan sangatlah penting. Untuk merancang dan menggunakan mekanisme keamanan dengan benar, kita perlu memahami apa sebenarnya yang perlu dilindungi, dan asumsi apa yang berkaitan dengan keamanan. Untuk menyederhanakan permasalahan, kebijakan keamanan Globus mencakup delapan pernyataan berikut, yang kami jelaskan di bawah ini (Foster dkk., 1998):

1. Lingkungan terdiri dari beberapa domain administratif.
2. Operasi lokal (yaitu operasi yang dilakukan hanya dalam satu domain) tunduk pada kebijakan keamanan domain lokal saja.
3. Operasi global (yaitu operasi yang melibatkan beberapa domain) mengharuskan inisiator diketahui di setiap domain tempat operasi dilakukan.
4. Operasi antar entitas di domain berbeda memerlukan otentikasi timbal balik.
5. Otentikasi global menggantikan otentikasi lokal.
6. Pengendalian akses terhadap sumber daya hanya tunduk pada keamanan lokal.
7. Pengguna dapat mendelegasikan hak atas proses.
8. Sekelompok proses dalam domain yang sama dapat berbagi kredensial.

Globus berasumsi bahwa lingkungan terdiri dari beberapa administratif

domain, di mana setiap domain memiliki kebijakan keamanan lokalnya sendiri. Diasumsikan bahwa kebijakan lokal tidak dapat diubah hanya karena domain tersebut berpartisipasi dalam Globus, dan kebijakan Globus secara keseluruhan tidak dapat mengesampingkan keputusan keamanan lokal. Akibatnya, keamanan di Globus akan membatasi dirinya pada operasi yang memengaruhi banyak domain.

Terkait dengan masalah ini adalah Globus berasumsi bahwa operasi yang sepenuhnya bersifat lokal pada suatu domain hanya tunduk pada kebijakan keamanan domain tersebut. Dengan kata lain, jika operasi dimulai dan dilakukan dalam satu domain, semua masalah keamanan akan dilakukan hanya dengan menggunakan langkah-langkah keamanan lokal. Globus tidak akan menerapkan tindakan tambahan.

Kebijakan keamanan Globus menyatakan bahwa permintaan operasi dapat dimulai baik secara global maupun lokal. Pemrakarsa, baik pengguna atau proses yang bertindak atas nama pengguna, harus dikenal secara lokal dalam setiap domain tempat operasi tersebut dilakukan. Misalnya, pengguna mungkin memiliki nama global yang dipetakan ke nama lokal khusus domain. Bagaimana tepatnya pemetaan tersebut dilakukan diserahkan kepada masing-masing domain.

Pernyataan kebijakan yang penting adalah bahwa operasi antar entitas di domain berbeda memerlukan otentikasi bersama. Artinya, misalnya, jika pengguna di satu domain menggunakan layanan dari domain lain, identitas pengguna harus diverifikasi. Yang tidak kalah pentingnya adalah pengguna harus yakin bahwa dia menggunakan layanan yang menurutnya sedang dia gunakan. Kita kembali ke otentikasi secara ekstensif nanti di bab ini.

Dua masalah kebijakan di atas digabungkan ke dalam persyaratan keamanan berikut. Jika identitas pengguna telah diverifikasi, dan pengguna tersebut juga dikenal secara lokal di suatu domain, maka dia dapat bertindak sebagai orang yang diautentikasi untuk domain lokal tersebut. Ini berarti bahwa Globus mengharuskan tindakan otentikasi seluruh sistemnya cukup untuk mempertimbangkan bahwa pengguna telah diautentikasi untuk domain jarak jauh (di mana pengguna tersebut diketahui) ketika mengakses sumber daya di domain tersebut. Otentikasi tambahan dengan domain tersebut tidak diperlukan.

Setelah pengguna (atau proses yang bertindak atas nama pengguna) telah diautentikasi, masih perlu memverifikasi hak akses yang tepat sehubungan dengan sumber daya. Misalnya, pengguna yang ingin memodifikasi file harus diautentikasi terlebih dahulu, setelah itu dapat diperiksa apakah pengguna tersebut benar-benar diizinkan untuk memodifikasi file atau tidak. Kebijakan keamanan Globus menyatakan bahwa keputusan kontrol akses tersebut dibuat sepenuhnya secara lokal dalam domain tempat sumber daya yang diakses berada.

Untuk menjelaskan pernyataan ketujuh, pertimbangkan agen seluler di Globus yang melaksanakan tugas dengan memulai beberapa operasi di domain berbeda, satu demi satu. Agen seperti itu mungkin memerlukan waktu lama untuk menyelesaikan tugasnya. Untuk menghindari komunikasi dengan pengguna yang atas nama agen bertindak, Globus mengharuskan proses dapat didelegasikan sebagian dari hak pengguna. Akibatnya, dengan mengautentikasi agen dan kemudian memeriksa haknya, Globus harus dapat mengizinkan agen untuk memulai operasi tanpa harus menghubungi pemilik agen.

Sebagai pernyataan kebijakan akhir, Globus mengharuskan sekelompok proses berjalan

dengan satu domain dan bertindak atas nama pengguna yang sama dapat berbagi satu set kredensial. Seperti yang akan dijelaskan di bawah, kredensial diperlukan untuk otentikasi. Pernyataan ini pada dasarnya membuka jalan menuju solusi otentikasi yang dapat diskalakan dengan tidak menuntut setiap proses membawa kumpulan kredensial uniknya sendiri.

Kebijakan keamanan Globus memungkinkan perancangnya berkonsentrasi pada pengembangan solusi keamanan secara keseluruhan. Dengan asumsi bahwa setiap domain menerapkan kebijakan keamanannya sendiri, Globus hanya berkonsentrasi pada ancaman keamanan yang melibatkan banyak domain. Secara khusus, kebijakan keamanan menunjukkan bahwa masalah desain yang penting adalah representasi pengguna dalam domain jarak jauh, dan alokasi sumber daya dari domain jarak jauh ke pengguna atau perwakilannya. Oleh karena itu, yang paling dibutuhkan Globus adalah mekanisme autentikasi lintas domain, dan membuat pengguna dikenal di domain jarak jauh.

Untuk tujuan ini, dua jenis perwakilan diperkenalkan. A**proksi pengguna**adalah proses yang diberikan izin untuk bertindak atas nama pengguna untuk jangka waktu terbatas. Sumber daya diwakili oleh proxy sumber daya. A**proksi sumber daya**adalah proses yang berjalan dalam domain tertentu yang digunakan untuk menerjemahkan operasi global pada sumber daya menjadi operasi lokal yang mematuhi kebijakan keamanan domain tertentu. Misalnya, proksi pengguna biasanya akan berkomunikasi dengan proksi sumber daya ketika akses ke sumber daya tersebut diperlukan.

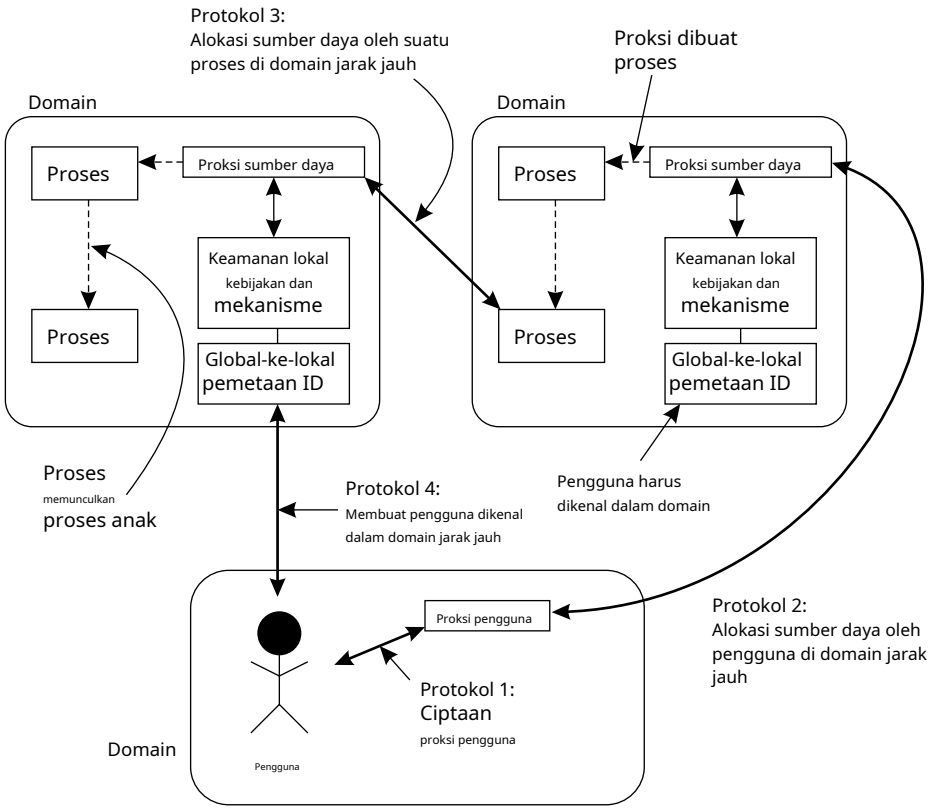
Arsitektur keamanan Globus pada dasarnya terdiri dari entitas seperti pengguna, proksi pengguna, proksi sumber daya, dan proses umum. Entitas-entitas ini terletak di domain dan berinteraksi satu sama lain. Secara khusus, arsitektur keamanan mendefinisikan empat protokol yang berbeda, seperti yang ditunjukkan pada Gambar 8-1 (lihat juga Foster et al., 1998).

Protokol pertama menjelaskan dengan tepat bagaimana pengguna dapat membuat proxy pengguna dan mendelegasikan hak kepada proxy tersebut. Secara khusus, agar proxy pengguna dapat bertindak atas nama penggunanya, pengguna memberikan proxy tersebut serangkaian kredensial yang sesuai.

Protokol kedua menentukan bagaimana proksi pengguna dapat meminta alokasi sumber daya di domain jarak jauh. Intinya, protokol memberitahu proksi sumber daya untuk membuat proses di domain jarak jauh setelah otentikasi timbal balik dilakukan. Proses tersebut mewakili pengguna (seperti yang dilakukan proksi pengguna), namun beroperasi di domain yang sama dengan sumber daya yang diminta. Proses ini diberikan akses ke sumber daya yang tunduk pada keputusan kontrol akses lokal pada domain tersebut.

Sebuah proses yang dibuat di domain jarak jauh dapat memulai komputasi tambahan di domain lain. Konsekuensinya, sebuah protokol diperlukan untuk mengalokasikan sumber daya dalam domain jarak jauh seperti yang diminta oleh proses selain proksi pengguna. Di Globus, jenis alokasi ini dilakukan melalui proksi pengguna, dengan membiarkan suatu proses memiliki proksi pengguna terkait yang meminta alokasi sumber daya, yang pada dasarnya mengikuti protokol kedua.

Protokol keempat dan terakhir dalam arsitektur keamanan Globus adalah cara pengguna membuat dirinya dikenal dalam suatu domain. Dengan asumsi bahwa pengguna memiliki akun di suatu domain, yang perlu ditetapkan adalah bahwa kredensial seluruh sistem yang dimiliki oleh proksi pengguna secara otomatis dikonversi ke kredensial yang dikenali oleh domain tertentu. Protokol tersebut mengatur bagaimana pemetaan antara global



Gambar 8-1. Arsitektur keamanan Globus.

kredensial dan kredensial lokal dapat didaftarkan oleh pengguna dalam tabel pemetaan lokal ke domain tersebut.

Rincian spesifik dari setiap protokol dijelaskan dalam (Foster et al., 1998). Masalah penting di sini adalah arsitektur keamanan Globus mencerminkan kebijakan keamanannya seperti disebutkan di atas. Mekanisme yang digunakan untuk mengimplementasikan arsitektur tersebut, khususnya protokol yang disebutkan di atas, merupakan hal yang umum pada banyak sistem terdistribusi, dan dibahas secara luas dalam bab ini. Kesulitan dalam merancang sistem terdistribusi yang aman tidak banyak disebabkan oleh mekanisme keamanan, namun karena menentukan bagaimana mekanisme tersebut akan digunakan untuk menegakkan kebijakan keamanan. Pada bagian selanjutnya, kami mempertimbangkan beberapa keputusan desain ini.

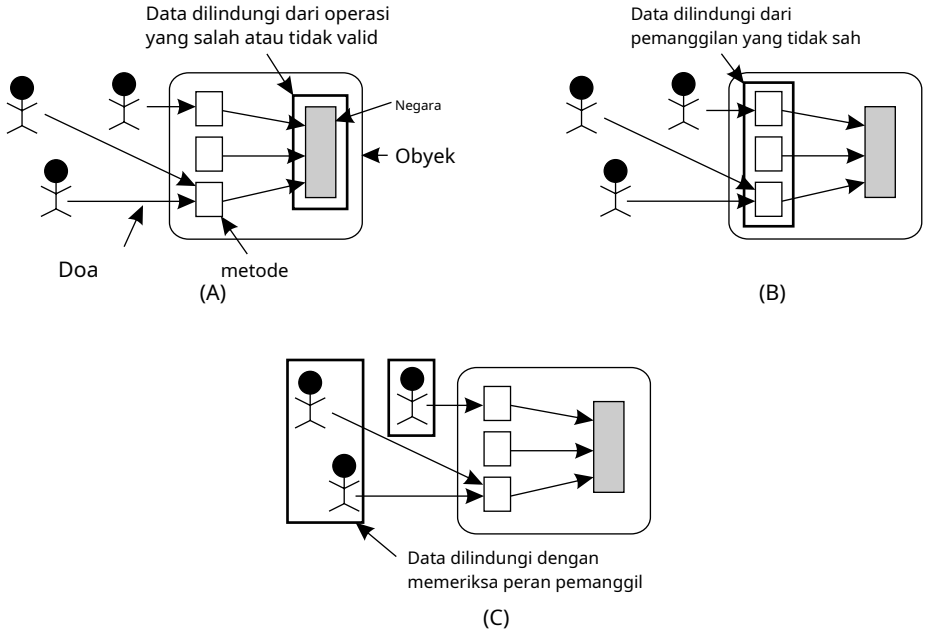
8.1.2 Masalah Desain

Sistem terdistribusi, atau sistem komputer apa pun, harus menyediakan layanan keamanan yang memungkinkan penerapan berbagai kebijakan keamanan. Ada sejumlah masalah desain penting yang perlu dipertimbangkan ketika menerapkan layanan keamanan tujuan umum. Di halaman berikutnya, kami

membahas tiga isu berikut: fokus pengendalian, pelapisan mekanisme keamanan, dan kesederhanaan (lihat juga (Gollmann, 1999).

Fokus Pengendalian

Saat mempertimbangkan perlindungan aplikasi (yang mungkin terdistribusi), pada dasarnya ada tiga pendekatan berbeda yang dapat diikuti, seperti yang ditunjukkan pada Gambar 8-2. Pendekatan pertama adalah berkonsentrasi langsung pada perlindungan data yang terkait dengan aplikasi. Secara langsung, yang kami maksudkan adalah terlepas dari berbagai operasi yang mungkin dapat dilakukan pada suatu item data, perhatian utamanya adalah memastikan integritas data. Biasanya, jenis perlindungan ini terjadi dalam sistem basis data di mana berbagai batasan integritas dapat dirumuskan yang secara otomatis diperiksa setiap kali item data diubah (lihat, misalnya, (Ullman, 1988).



Gambar 8-2. Tiga pendekatan untuk perlindungan terhadap ancaman keamanan. (a) Perlindungan terhadap operasi yang tidak valid (b) Perlindungan terhadap pemanggilan yang tidak sah. (c) Perlindungan terhadap pengguna yang tidak sah.

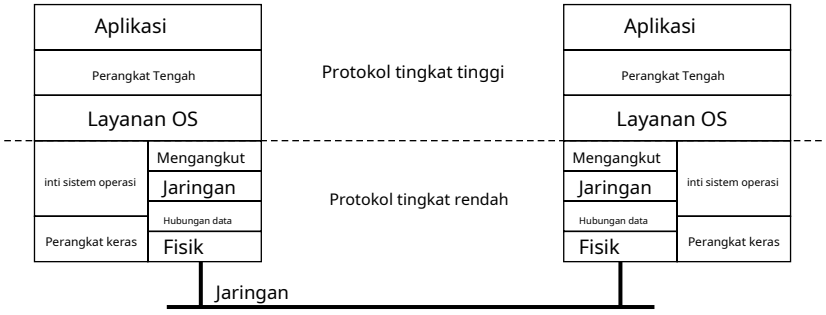
Pendekatan kedua adalah berkonsentrasi pada perlindungan dengan menentukan secara spesifik operasi mana yang dapat dijalankan, dan oleh siapa, kapan data atau sumber daya tertentu akan diakses. Dalam hal ini, fokus pengendalian sangat terkait dengan mekanisme kontrol akses, yang akan kita bahas secara luas nanti di bab ini. Misalnya, dalam sistem berbasis objek, mungkin diputuskan untuk menentukan masing-masing metode itu

tersedia untuk klien yang kliennya diizinkan untuk menggunakan metode itu. Alternatifnya, metode kontrol akses dapat diterapkan ke seluruh antarmuka yang ditawarkan oleh suatu objek, atau ke seluruh objek itu sendiri. Pendekatan ini memungkinkan adanya berbagai rincian kontrol akses.

Pendekatan ketiga adalah memusatkan perhatian secara langsung pada pengguna dengan mengambil langkah-langkah dimana hanya orang-orang tertentu yang mempunyai akses terhadap aplikasi, terlepas dari operasi yang ingin mereka lakukan. Misalnya, database di bank dapat dilindungi dengan menolak akses kepada siapa pun kecuali manajemen yang lebih tinggi. Contoh lainnya, di banyak universitas, data dan aplikasi tertentu dibatasi untuk digunakan oleh dosen dan staf saja, sedangkan akses oleh mahasiswa tidak diperbolehkan. Akibatnya, pengendalian difokuskan pada pendefinisian**peran**yang dimiliki pengguna, dan setelah peran pengguna diverifikasi, akses ke sumber daya akan diberikan atau ditolak. Sebagai bagian dari perancangan sistem yang aman, penting untuk mendefinisikan peran yang mungkin dimiliki oleh masyarakat, dan menyediakan mekanisme untuk mendukung kontrol akses berbasis peran. Kita kembali ke peran nanti di bab ini.

Lapisan Mekanisme Keamanan

Persoalan penting dalam merancang sistem yang aman adalah memutuskan pada tingkat mana mekanisme keamanan harus ditempatkan. Level dalam konteks ini terkait dengan pengorganisasian logis suatu sistem ke dalam sejumlah lapisan. Misalnya, jaringan komputer sering kali disusun dalam beberapa lapisan mengikuti beberapa model referensi, seperti yang telah kita bahas di Bab. 2. Dalam Bab. 1, kami memperkenalkan organisasi sistem terdistribusi yang terdiri dari lapisan terpisah untuk aplikasi, middleware, layanan sistem operasi, dan kernel sistem operasi. Menggabungkan organisasi berlapis jaringan komputer dan sistem terdistribusi, secara kasar mengarah pada apa yang ditunjukkan pada Gambar 8-3.

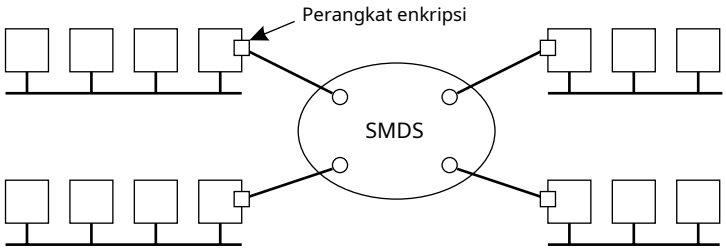


Gambar 8-3. Organisasi logis dari sistem terdistribusi menjadi beberapa lapisan.

Intinya, Gambar 8-3 memisahkan layanan tujuan umum dari layanan komunikasi. Pemisahan ini penting untuk memahami lapisan keamanan dalam sistem terdistribusi dan, khususnya, gagasan tentang kepercayaan. Perbedaan antara

kepercayaan dan keamanan itu penting. Suatu sistem aman atau tidak (dengan mempertimbangkan berbagai ukuran probabilistik), namun apakah klien menganggap suatu sistem aman, itu adalah masalah kepercayaan (Pfleeger, 1997). Di mana mekanisme keamanan lapisan ditempatkan bergantung pada kepercayaan klien terhadap seberapa aman layanan di lapisan tertentu.

Sebagai contoh, pertimbangkan sebuah organisasi yang berlokasi di lokasi berbeda yang terhubung melalui layanan komunikasi seperti**Layanan Data Multi-megabit Beralih**(**SMDS**). Jaringan SMDS dapat dianggap sebagai tulang punggung tingkat link yang menghubungkan berbagai jaringan area lokal di lokasi yang mungkin tersebar secara geografis, seperti ditunjukkan pada Gambar 8-4 (untuk informasi lebih lanjut tentang SMDS lihat (Klessig dan Tesink, 1995).



Gambar 8-4. Beberapa situs terhubung melalui layanan tulang punggung area luas.

Keamanan dapat diberikan dengan menempatkan perangkat enkripsi di setiap router SMDS, seperti juga ditunjukkan pada Gambar 8-4. Perangkat ini secara otomatis mengenkripsi dan mendekripsi paket yang dikirim antar situs, namun tidak menyediakan komunikasi aman antar host di situs yang sama. Jika Alice di lokasi*A* mengirim pesan ke Bob di lokasi*B*, dan dia khawatir pesannya akan disadap, dia setidaknya harus percaya bahwa enkripsi lalu lintas antarsitus berfungsi dengan baik. Artinya, misalnya, dia harus memercayai administrator sistem di kedua situs untuk mengambil tindakan yang tepat terhadap gangguan pada perangkat.

Sekarang anggaplah Alice tidak mempercayai keamanan lalu lintas antarsitus. Dia kemudian dapat memutuskan untuk mengambil tindakan sendiri dengan menggunakan layanan keamanan tingkat transportasi seperti SSL. SSL adalah singkatan dari**Lapisan Soket Aman**dan dapat digunakan untuk mengirim pesan dengan aman melalui koneksi TCP. Kami akan membahas detail SSL di Bab. 11 ketika membahas keamanan di Web. Hal penting yang perlu diperhatikan adalah SSL memungkinkan Alice mengatur koneksi aman ke Bob. Semua pesan tingkat transport akan dienkripsi—dan juga pada tingkat SMDS, tapi hal itu tidak menjadi perhatian Alice. Dalam hal ini, Alice harus menaruh kepercayaannya pada SSL. Dengan kata lain, dia yakin SSL aman.

Dalam sistem terdistribusi, mekanisme keamanan sering kali ditempatkan pada lapisan middleware. Jika Alice tidak mempercayai SSL, dia mungkin ingin menggunakan layanan RPC lokal yang aman. Sekali lagi, dia harus memercayai layanan RPC ini untuk melakukan apa yang dijanjikannya, seperti tidak membocorkan informasi atau mengautentikasi klien dan server dengan benar.

Layanan keamanan yang ditempatkan di lapisan middleware sistem terdistribusi hanya dapat dipercaya jika layanan yang mereka andalkan untuk mengamankannya memang aman. Misalnya, jika layanan RPC aman sebagian diimplementasikan melalui SSL, maka kepercayaan terhadap layanan RPC bergantung pada seberapa besar kepercayaan seseorang terhadap SSL. Jika keamanan SSL tidak dipercaya, maka tidak ada kepercayaan terhadap keamanan layanan RPC.

Distribusi Mekanisme Keamanan

Ketergantungan antar layanan mengenai kepercayaan menimbulkan pengertian a **Basis Komputasi Tepercaya**(**TCB**). TCB adalah kumpulan semua mekanisme keamanan dalam sistem komputer (terdistribusi) yang diperlukan untuk menegakkan kebijakan keamanan. Semakin kecil TCBnya, semakin baik. Jika sistem terdistribusi dibangun sebagai middleware pada sistem operasi jaringan yang ada, keamanannya mungkin bergantung pada keamanan sistem operasi lokal yang mendasarinya. Dengan kata lain, TCB dalam sistem terdistribusi dapat mencakup sistem operasi lokal di berbagai host.

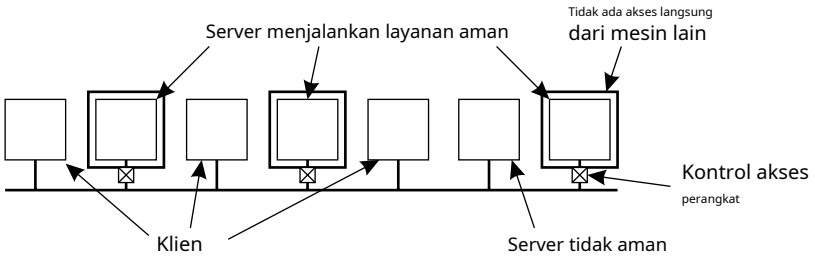
Pertimbangkan server file dalam sistem file terdistribusi. Server tersebut mungkin perlu bergantung pada berbagai mekanisme perlindungan yang ditawarkan oleh sistem operasi lokalnya. Mekanisme tersebut tidak hanya mencakup mekanisme untuk melindungi file terhadap akses oleh proses selain dari server file, namun juga mekanisme untuk melindungi server file agar tidak dirusak secara jahat.

Oleh karena itu, sistem terdistribusi berbasis middleware memerlukan kepercayaan pada sistem operasi lokal tempat mereka bergantung. Jika kepercayaan seperti itu tidak ada, maka sebagian fungsi sistem operasi lokal mungkin perlu dimasukkan ke dalam sistem terdistribusi itu sendiri. Pertimbangkan sistem operasi mikrokernel, di mana sebagian besar layanan sistem operasi dijalankan sebagai proses pengguna normal. Dalam hal ini, sistem file, misalnya, dapat diganti seluruhnya dengan sistem file yang disesuaikan dengan kebutuhan spesifik sistem terdistribusi, termasuk berbagai langkah keamanannya. Perhatikan bahwa pendekatan ini secara bertahap dapat menggantikan sistem terdistribusi berbasis middleware dengan sistem terdistribusi*Pengoperasian*sistem seperti yang telah kita bahas di Bab. 1.

Konsisten dengan pendekatan ini adalah memisahkan layanan keamanan dari jenis layanan lainnya dengan mendistribusikan layanan ke berbagai mesin berbeda bergantung pada keamanan yang diperlukan. Misalnya, untuk sistem file terdistribusi yang aman, server file dapat diisolasi dari klien dengan menempatkan server pada mesin dengan sistem operasi tepercaya, yang mungkin menjalankan sistem file aman khusus. Klien dan aplikasinya ditempatkan pada mesin yang tidak tepercaya.

Pemisahan ini secara efektif mengurangi TCB ke sejumlah kecil mesin dan komponen perangkat lunak. Dengan selanjutnya melindungi mesin tersebut dari serangan keamanan dari luar, kepercayaan secara keseluruhan terhadap keamanan sistem terdistribusi dapat ditingkatkan. Mencegah klien dan aplikasi mereka mengakses langsung ke layanan penting diikuti dalam**Antarmuka yang Dikurangi untuk Komponen Sistem yang Aman**(**RISSC**) pendekatan, seperti yang dijelaskan dalam (Neumann, 1995). Dalam pendekatan RISSC, setiap server yang kritis terhadap keamanan ditempatkan pada mesin terpisah

diisolasi dari sistem pengguna akhir menggunakan antarmuka jaringan aman tingkat rendah, seperti yang ditunjukkan pada Gambar 8-5. Klien dan aplikasinya berjalan pada mesin yang berbeda dan dapat mengakses server aman hanya melalui antarmuka jaringan ini.



Gambar 8-5. Prinsip RISSC diterapkan untuk mengamankan sistem terdistribusi.

Kesederhanaan

Masalah desain penting lainnya yang terkait dengan penentuan lapisan mana yang akan menempatkan mekanisme keamanan adalah kesederhanaan. Merancang sistem komputer yang aman umumnya dianggap sebagai tugas yang sulit. Konsekuensinya, jika seorang perancang sistem dapat menggunakan beberapa mekanisme sederhana yang mudah dipahami dan dipercaya untuk bekerja, maka akan semakin baik.

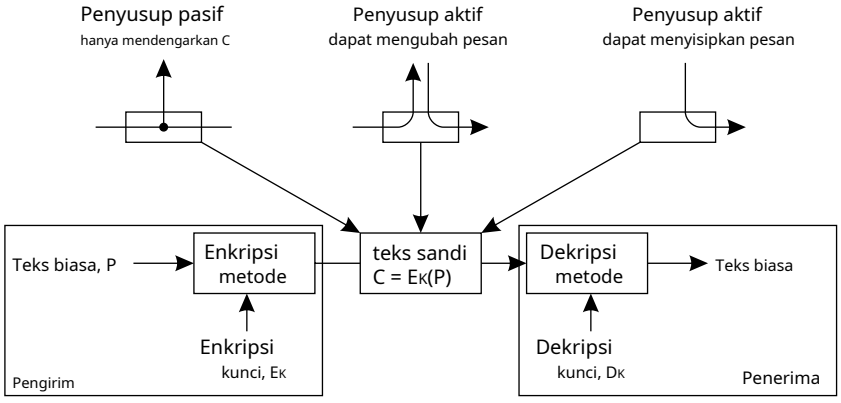
Sayangnya, mekanisme sederhana seringkali tidak cukup untuk menerapkan kebijakan keamanan. Pertimbangkan kembali situasi di mana Alice ingin mengirim pesan kepada Bob seperti yang dibahas di atas. Enkripsi tingkat tautan adalah mekanisme sederhana dan mudah dipahami untuk melindungi terhadap intersepsi lalu lintas pesan antarsitus. Namun, diperlukan lebih banyak lagi jika Alice ingin memastikan bahwa hanya Bob yang akan menerima pesannya. Dalam hal ini, layanan autentikasi tingkat pengguna diperlukan, dan Alice mungkin perlu mengetahui cara kerja layanan tersebut agar dapat memercayai layanan tersebut. Oleh karena itu, otentikasi tingkat pengguna mungkin memerlukan setidaknya pemahaman tentang kunci kriptografi dan kesadaran akan mekanisme seperti sertifikat, meskipun banyak layanan keamanan yang sangat otomatis dan tersembunyi dari pengguna.

Dalam kasus lain, aplikasi itu sendiri pada dasarnya rumit dan penerapan keamanan hanya akan memperburuk keadaan. Contoh domain aplikasi yang melibatkan protokol keamanan kompleks (seperti yang akan kita bahas nanti di bab ini) adalah sistem pembayaran digital. Kompleksitas pembayaran digital sering kali disebabkan oleh banyaknya pihak yang perlu berkomunikasi untuk melakukan pembayaran. Dalam kasus ini, mekanisme dasar yang digunakan untuk mengimplementasikan protokol harus relatif sederhana dan mudah dipahami. Kesederhanaan akan berkontribusi pada kepercayaan yang akan diberikan oleh pengguna akhir ke dalam aplikasi dan, yang lebih penting, akan berkontribusi untuk meyakinkan perancang bahwa sistem tidak memiliki celah keamanan.

8.1.3 Kriptografi

Dasar keamanan dalam sistem terdistribusi adalah penggunaan teknik kriptografi. Ide dasar penerapan teknik ini sederhana. Pertimbangkan pengirimnya *S* ingin menyampaikan pesan *M* ke penerima *R*. Untuk melindungi pesan terhadap ancaman keamanan, pengirim terlebih dahulu **mengenkripsi** itu menjadi pesan yang tidak dapat dimengerti *M'*, dan selanjutnya mengirim *M'* ke *R*. *R*, pada gilirannya, harus **mendekripsi** pesan yang diterima ke dalam bentuk aslinya *M*.

Enkripsi dan dekripsi dilakukan dengan menggunakan metode kriptografi yang diparameterisasi dengan kunci, seperti yang ditunjukkan pada Gambar 8-6. Bentuk asli pesan yang dikirimkan disebut **teks biasa**, ditampilkan sebagai *P* pada Gambar 8-6; bentuk terenkripsi disebut sebagai **teks sandi**, diilustrasikan sebagai *C*.



Gambar 8-6. Penyusup dan penyadap dalam komunikasi.

Untuk mendeskripsikan berbagai protokol keamanan yang digunakan dalam membangun layanan keamanan untuk sistem terdistribusi, penting untuk memiliki notasi yang menghubungkan teks biasa, teks tersandi, dan kunci. Mengikuti konvensi notasi umum, kami akan menggunakan $C = E_K(P)$ untuk menunjukkan bahwa ciphertext *C* diperoleh dengan mengenkripsi teks biasa *P* menggunakan kunci *K*. Juga, $P = D_K(C)$ digunakan untuk menyatakan dekripsi ciphertext *C* menggunakan kunci *K*, menghasilkan teks biasa *P*.

Kembali ke contoh kita yang ditunjukkan pada Gambar 8-6, saat mentransfer pesan sebagai ciphertext *C*, ada tiga serangan berbeda yang perlu kita lindungi, dan enkripsi dapat membantu. Pertama, sebuah **pengacau** dapat mencegat pesan tanpa pengirim atau penerima menyadari bahwa penyadapan sedang terjadi. Tentu saja, jika pesan yang dikirimkan telah dienkripsi sedemikian rupa sehingga tidak dapat dengan mudah didekripsi tanpa memiliki kunci yang tepat, intersepsi tidak ada gunanya: penyusup hanya akan melihat data yang tidak dapat dipahami. (Omong-omong, fakta bahwa sebuah pesan sedang dikirim terkadang cukup bagi penyusup untuk menarik kesimpulan. Misalnya, jika selama krisis dunia, jumlah lalu lintas ke Gedung Putih tiba-tiba turun menjadi nol sementara jumlah lalu lintas pergi ke tempat tertentu

gunung di Colorado meningkat dengan jumlah yang sama, mungkin ada informasi berguna untuk mengetahui hal itu.)

Jenis serangan kedua yang perlu ditangani adalah memodifikasi pesan. Memodifikasi teks biasa itu mudah; memodifikasi ciphertext yang telah dienkripsi dengan benar jauh lebih sulit karena penyusup harus mendekripsi pesan terlebih dahulu sebelum dapat memodifikasinya secara berarti. Selain itu, ia juga harus mengenkripsinya kembali dengan benar atau jika tidak, penerima akan menyadari bahwa pesan tersebut telah dirusak.

Jenis serangan ketiga adalah ketika penyusup memasukkan pesan terenkripsi ke dalam sistem komunikasi, mencoba membuat R percaya pesan-pesan ini berasal S. Sekali lagi, seperti yang akan kita lihat nanti di bab ini, enkripsi dapat membantu melindungi terhadap serangan semacam itu. Perhatikan bahwa jika penyusup dapat mengubah pesan, dia juga dapat menyisipkan pesan.

Ada perbedaan mendasar antara sistem kriptografi yang berbeda, berdasarkan apakah kunci enkripsi dan dekripsinya sama atau tidak. Di sebuah **kriptosistem simetris**, kunci yang sama digunakan untuk mengenkripsi dan mendekripsi pesan. Dengan kata lain,

$$P = D_K(E_K(P))$$

Kriptosistem simetris juga disebut sebagai sistem kunci rahasia atau kunci bersama, karena pengirim dan penerima diharuskan berbagi kunci yang sama, dan untuk memastikan perlindungan berfungsi, kunci bersama ini harus dirahasiakan; tidak ada orang lain yang diizinkan melihat kuncinya. Kami akan menggunakan notasi tersebut $K_{A,B}$ untuk menunjukkan kunci yang dibagikan oleh A Dan B.

Dalam sebuah **kriptosistem asimetris**, kunci untuk enkripsi dan dekripsi berbeda, tetapi bersama-sama membentuk pasangan yang unik. Dengan kata lain, ada kunci tersendiri K_E untuk enkripsi dan satu untuk dekripsi, K_D, seperti yang

$$P = D_K(E_K(P))_E$$

Salah satu kunci dalam sistem kriptografi asimetris dijaga kerahasiaannya, sedangkan kunci yang lain dibuat publik. Oleh karena itu, kriptosistem asimetris juga disebut sebagai **publik-sistem kunci**. Berikut ini, kami menggunakan notasi K^+_A untuk menunjukkan kunci publik milik A, Dan K^-_A sebagai kunci pribadinya yang sesuai.

Mengantisipasi diskusi rinci mengenai protokol keamanan nanti di bab ini, kunci enkripsi atau dekripsi mana yang benar-benar dipublikasikan bergantung pada cara kunci tersebut digunakan. Misalnya, jika Alice ingin mengirim pesan rahasia kepada Bob, dia harus menggunakan kunci publik Bob untuk mengenkripsi pesan tersebut. Karena Bob adalah satu-satunya yang memegang kunci dekripsi pribadi, ia juga satu-satunya orang yang dapat mendekripsi pesan tersebut.

Di sisi lain, misalkan Bob ingin mengetahui dengan pasti bahwa pesan yang baru saja diterimanya benar-benar berasal dari Alice. Dalam hal ini, Alice dapat merahasiakan kunci enkripsinya untuk mengenkripsi pesan yang dikirimnya. Jika Bob berhasil mendekripsi pesan menggunakan kunci publik Alice (dan teks biasa dalam pesan tersebut memiliki informasi yang cukup untuk membuatnya bermakna bagi Bob), dia tahu bahwa pesan tersebut harus

berasal dari Alice, karena kunci dekripsi terikat secara unik pada kunci enkripsi. Kami kembali ke algoritma tersebut secara rinci di bawah.

Salah satu penerapan terakhir kriptografi dalam sistem terdistribusi adalah penggunaan**fungsi hash**. Fungsi hash*H*mengambil pesan*M*dengan panjang sewenang-wenang sebagai input dan menghasilkan string bit*H*memiliki panjang tetap sebagai keluaran:

$$H = H(M)$$

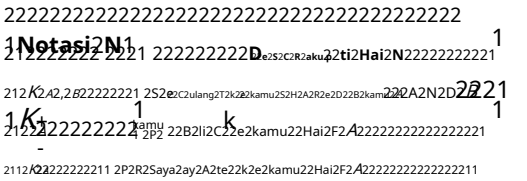
Sebuah hash*H*sebanding dengan bit tambahan yang ditambahkan ke pesan dalam sistem komunikasi untuk memungkinkan deteksi kesalahan, seperti pemeriksaan cyclic-redundancy check (CRC).

Fungsi hash yang digunakan dalam sistem kriptografi memiliki sejumlah properti. Pertama, memang demikian**fungsi satu arah**, artinya secara komputasi tidak mungkin untuk menemukan masukan*M*yang sesuai dengan keluaran yang diketahui*H*. Di sisi lain, komputasi*H*dari*M*gampang. Kedua, mereka memiliki**ketahanan benturan yang lemah**properti, artinya diberi masukan*M*dan keluaran terkaitnya *H*=*H*(*M*), secara komputasi tidak mungkin untuk menemukan masukan lain yang berbeda*M*′ ≠*M*, seperti yang*H*(*M*) =*H*(*M*′).Terakhir, fungsi hash kriptografi juga memiliki **ketahanan benturan yang kuat**harta benda, yang artinya, bila diberikan saja*H*, secara komputasi tidak mungkin menemukan dua nilai masukan yang berbeda*M*Dan*M*′ ,seperti yang*H*(*M*) =*H*(*M*′).

Properti serupa berlaku untuk fungsi enkripsi apa pun*E*dan kunci yang digunakan. Selanjutnya, untuk fungsi enkripsi apa pun*E*, secara komputasi tidak mungkin untuk menemukan kuncinya*K*ketika diberi teks biasa*P*dan teks sandi yang terkait *C*=*EK*(*P*). Begitu pula yang dianalogikan dengan ketahanan tumbukan, ketika diberi teks biasa*P* dan sebuah kunci*K*, tidak mungkin menemukan kunci lain*K*′seperti yang *EK*(*P*) =*EK*′(*P*).

Seni dan ilmu dalam merancang algoritma untuk sistem kriptografi memiliki sejarah yang panjang (Kahn, 1967), dan membangun sistem yang aman seringkali sangat sulit, atau bahkan mustahil (Schneier, 2000). Pembahasan algoritma ini secara rinci berada di luar cakupan buku ini. Namun, untuk memberikan kesan kriptografi dalam sistem komputer, sekarang kami akan menyajikan secara singkat tiga algoritma yang mewakili. Informasi rinci mengenai algoritma ini dan algoritma kriptografi lainnya dapat ditemukan di (Kaufman et al., 1995; Menezes et al., 1996; dan Schneier, 1996).

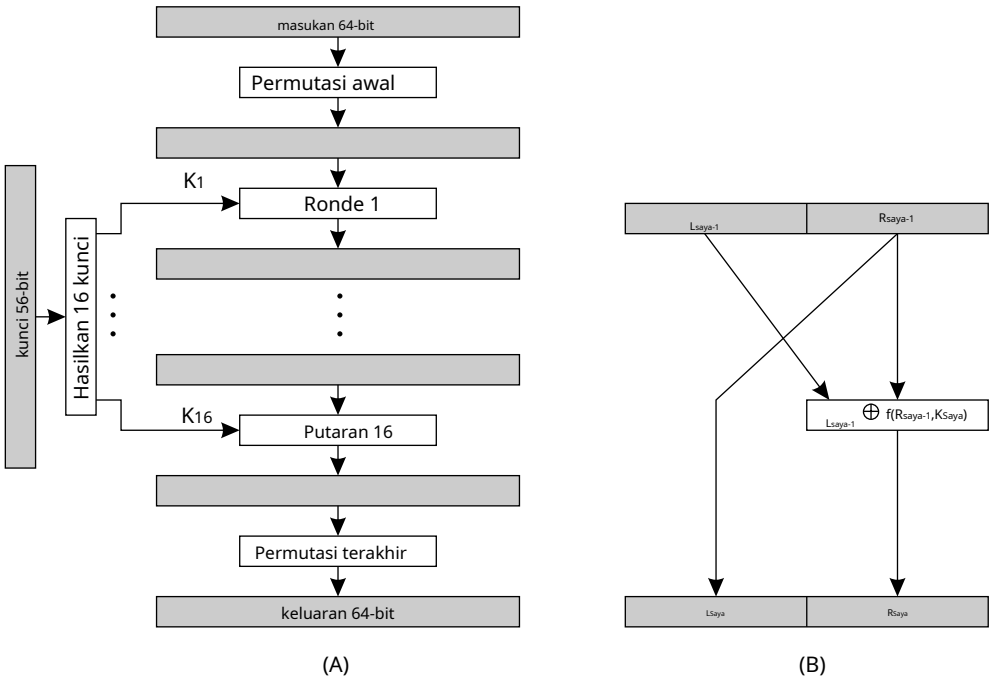
Sebelum kita masuk ke rincian berbagai protokol, Gambar 8-7 merangkum notasi dan singkatan yang kita gunakan.



Gambar 8-7. Notasi yang digunakan dalam bab ini.

Kriptosistem Simetris: DES

Contoh pertama dari algoritma kriptografi adalah**Standar Enkripsi Data**(**DES**), yang digunakan untuk kriptosistem simetris. DES dirancang untuk beroperasi pada blok data 64-bit. Sebuah blok diubah menjadi blok keluaran terenkripsi (64 bit) dalam 16 putaran, di mana setiap putaran menggunakan kunci 48-bit yang berbeda untuk enkripsi. Masing-masing dari 16 kunci ini berasal dari kunci master 56-bit, seperti yang ditunjukkan pada Gambar 8-8(a). Sebelum blok masukan memulai 16 putaran enkripsinya, ia terlebih dahulu mengalami permutasi awal, yang kebalikannya kemudian diterapkan pada keluaran terenkripsi yang mengarah ke blok keluaran akhir.



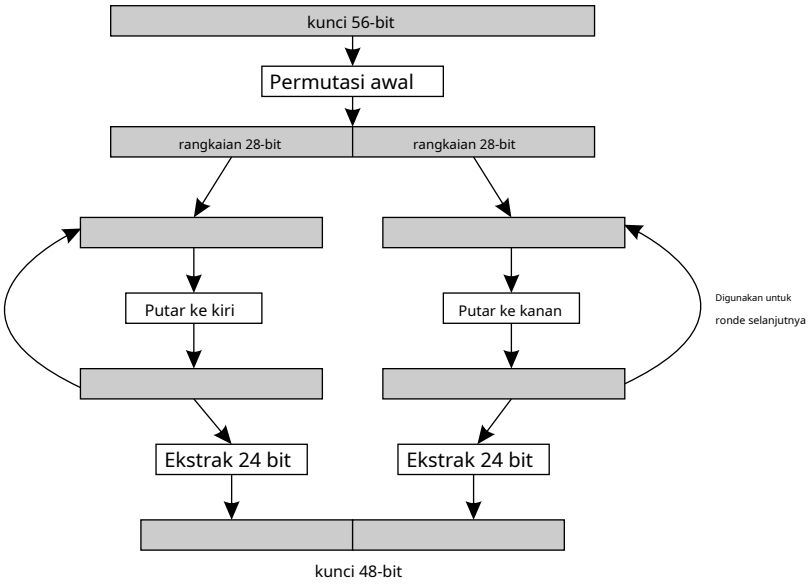
Gambar 8-8.(a) Prinsip DES. (b) Garis besar satu putaran enkripsi.

Setiap putaran enkripsi*Saya*mengambil blok 64-bit yang dihasilkan oleh putaran sebelumnya *Saya*-1 sebagai masukannya, seperti yang ditunjukkan pada Gambar 8-8(b). 64 bit dipecah menjadi bagian kiriL_{Saya-1} dan bagian kananR_{Saya-1}, masing-masing berisi 32 bit. Bagian kanan digunakan untuk bagian kiri pada putaran berikutnya, yaitu,$L_{Saya} = R_{Saya-1}$.

Kerja keras dilakukan dalam fungsi mangkuk*F*. Fungsi ini membutuhkan blok 32-bitR_{Saya-1} sebagai input, bersama dengan kunci 48-bitK_{Saya},dan menghasilkan blok 32-bit yang di-XORL_{Saya-1} untuk menghasilkanR_{Saya}. (XOR adalah singkatan dari eksklusif atau operasi.) Fungsi mangkuk pertama kali diperluasR_{Saya-1}ke blok 48-bit dan melakukan XOR dengannyaK_{Saya}.Hasilnya dipartisi menjadi delapan bagian yang masing-masing terdiri dari enam bit. Setiap bagian adalah

lalu dimasukkan ke tempat lain**S-kotak**, yang merupakan operasi yang mengganti masing-masing dari 64 kemungkinan masukan 6-bit menjadi salah satu dari 16 kemungkinan keluaran 4-bit. Delapan potongan keluaran yang masing-masing terdiri dari empat bit kemudian digabungkan menjadi nilai 32-bit dan diijinkan lagi.

Kunci 48-bitK_{Saya}untuk putaran*Saya*berasal dari kunci master 56-bit sebagai berikut. Pertama, kunci master diubah dan dibagi menjadi dua bagian 28-bit. Untuk setiap putaran, setiap bagian pertama-tama diputar satu atau dua bit ke kiri, setelah itu 24 bit diekstraksi. Bersama dengan 24 bit dari separuh lainnya yang diputar, kunci 48-bit dibuat. Rincian satu putaran enkripsi ditunjukkan pada Gambar 8-9.



Gambar 8-9. Detail pembuatan kunci per putaran di DES.

Prinsip DES cukup sederhana, namun algoritmanya sulit dipecahkan dengan menggunakan metode analitis. Menggunakan serangan brute force hanya dengan mencari kunci yang akan melakukan pekerjaan itu menjadi mudah seperti yang telah ditunjukkan belakangan ini. Menggunakan DES tiga kali dalam mode enkripsi-dekripsi-enkripsi khusus dengan kunci berbeda tampaknya aman untuk saat ini.

Apa yang membuat DES sulit diserang melalui analisis adalah alasan di balik desain tersebut tidak pernah dijelaskan dengan jelas. Sebagai contoh, telah ditunjukkan bahwa menggunakan S-box lain yang saat ini digunakan dalam standar, membuat algoritma secara substansial lebih mudah untuk dipecahkan (lihat Pfleeger, 1997 untuk analisis singkat mengenai DES). Alasan desain dan penggunaan S-box baru dipublikasikan setelah model serangan baru dirancang pada tahun 1990an. DES terbukti cukup tahan terhadap serangan ini, dan para perancangnya mengungkapkan bahwa model yang baru dirancang telah mereka ketahui ketika mereka mengembangkan DES pada tahun 1974 (Coppersmith,

1994).

DES telah digunakan sebagai teknik enkripsi standar selama bertahun-tahun, namun saat ini sedang dalam proses digantikan oleh blok algoritma Rijndael yang berukuran 128 bit. Ada juga varian dengan kunci lebih besar dan blok data lebih besar. Algoritme ini telah dirancang agar cukup cepat sehingga dapat diimplementasikan pada kartu pintar, yang merupakan area aplikasi kriptografi yang semakin penting.

Kriptosistem Kunci Publik: RSA

Contoh algoritma kriptografi kedua kami banyak digunakan untuk sistem kunci publik: **RSA**, dinamai menurut penemunya Rivest, Shamir, dan Adleman (1978). Keamanan RSA berasal dari fakta bahwa tidak ada metode yang diketahui untuk menemukannya secara efisien**faktor utama**dalam jumlah besar. Dapat ditunjukkan bahwa setiap bilangan bulat dapat ditulis sebagai hasil kali bilangan prima. Misalnya, 2100 dapat ditulis sebagai

$$2100 = 2\times2\times3\times5\times5\times7$$

menjadikan 2, 3, 5, dan 7 sebagai faktor prima pada tahun 2100. Dalam RSA, kunci privat dan publik dibuat dari bilangan prima yang sangat besar (terdiri dari ratusan angka desimal). Ternyata, memecahkan RSA sama dengan menemukan dua bilangan prima tersebut. Sejauh ini, hal ini terbukti tidak mungkin dilakukan secara komputasi meskipun para ahli matematika telah mengerjakan masalah ini selama berabad-abad.

Menghasilkan kunci pribadi dan publik memerlukan empat langkah:

1. Pilih dua bilangan prima yang sangat besar, PDan$Q$.
2. Hitung$N=P\times Q$Dan$z=(P-1)\times(Q-1)$.
3. Pilih nomor$D$itu relatif prima$z$.
4. Hitung jumlahnya$e$seperti yang$e\times D=1 \mod z$.

Salah satu angkanya, katakanlah$D$, selanjutnya dapat digunakan untuk dekripsi, sedangkan$e$ digunakan untuk enkripsi. Hanya satu dari keduanya yang dipublikasikan, bergantung pada algoritma yang digunakan.

Mari kita pertimbangkan kasus dimana Alice ingin menjaga kerahasiaan pesan yang dia kirimkan kepada Bob. Dengan kata lain, dia ingin memastikan bahwa tidak ada orang lain selain Bob yang dapat mencegat dan membacakan pesannya kepadanya. RSA mempertimbangkan setiap pesan$M$menjadi hanya serangkaian bit. Setiap pesan pertama-tama dibagi menjadi blok-blok dengan panjang tetap, di mana setiap blokM_{saya}, ditafsirkan sebagai bilangan biner, harus terletak pada interval$0\le M_{saya}<n$.

Untuk mengenkripsi pesan$M$, pengirim menghitung untuk setiap blokM_{saya}nilai $C_{saya}=M_e{}_{saya} \mod n$), yang kemudian dikirim ke penerima. Dekripsi di penerima sisi terjadi dengan komputasi$M_{saya}=C_D{}_{saya} \mod n$). Perhatikan bahwa untuk enkripsi, keduanya eDan$N$diperlukan, sedangkan dekripsi memerlukan mengetahui nilainya$D$Dan$N$.

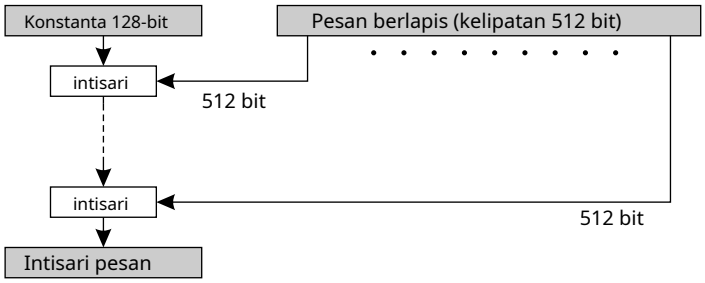
Ketika membandingkan RSA dengan kriptosistem simetris seperti DES, RSA memiliki kelemahan yaitu komputasinya lebih kompleks. Ternyata, mengenkripsi

pesan yang menggunakan RSA kira-kira 100–1000 kali lebih lambat dibandingkan DES, bergantung pada teknik implementasi yang digunakan. Sebagai konsekuensinya, banyak sistem kriptografi menggunakan RSA hanya untuk bertukar kunci bersama dengan cara yang aman, namun lebih sedikit lagi untuk mengenkripsi data ''normal''. Kita akan melihat contoh kombinasi kedua teknik ini nanti di bagian berikutnya.

Fungsi Hash: MD5

Sebagai contoh terakhir dari algoritma kriptografi yang banyak digunakan, kita lihat MD5 (Rivest, 1992).**MD5**adalah fungsi hash untuk menghitung 128-bit dengan panjang tetap **intisari pesan**dari string input biner dengan panjang sembarang. String masukan pertama-tama diisi dengan panjang total 448 bit (modulo 512), setelah itu panjang string bit asli ditambahkan sebagai bilangan bulat 64-bit. Akibatnya, masukan diubah menjadi serangkaian blok 512-bit.

Struktur algoritma ditunjukkan pada Gambar 8-10. Dimulai dengan nilai konstan 128-bit, algoritme melanjutkan*k*fase, di mana*k*adalah jumlah blok 512-bit yang terdiri dari pesan berlapis. Selama setiap fase, intisari 128-bit dihitung dari blok data 512-bit yang berasal dari pesan berlapis, dan intisari 128-bit dihitung pada fase sebelumnya.



Gambar 8-10. Struktur MD5.

Sebuah fase dalam MD5 terdiri dari empat putaran komputasi, dimana setiap putaran menggunakan salah satu dari empat fungsi berikut:

$$F(X,kamu,z) = (x\ DAN\ kamu) ATAU ((BUKAN\ x) DAN\ z)$$
$$G(X,kamu,z) = (x\ DAN\ z) ATAU (kamu\ DAN (BUKAN\ z))$$
$$H(X,kamu,z) = x\ XOR\ y\ XOR\ z$$
$$SAYA(X,kamu,z) = kamu\ XOR (x\ ATAU (BUKAN\ z))$$

Masing-masing fungsi ini beroperasi pada variabel 32-bit*X*,*kamu*, Dan*z*. Untuk mengilustrasikan bagaimana fungsi-fungsi ini digunakan, pertimbangkan blok 512-bit*B*dari pesan empuk yang sedang diproses selama fase*k*. Memblokir*B*dibagi menjadi 16 subblok 32-bit $B_0, B_1, \ldots, B_{15}$. Selama putaran pertama, berfungsi*F*digunakan untuk mengubah empat variabel (dilambangkan sebagai*P* ,*Q*,*R*, Dan*S*, masing-masing) dalam 16 iterasi seperti yang ditunjukkan pada Gambar 8-11

Variabel-variabel ini dibawa ke setiap putaran berikutnya, dan setelah satu fase selesai, diteruskan ke fase berikutnya. Ada total 64 konstanta yang telah ditentukan sebelumnya C_{Saya}. Notasi $X <<< N$ digunakan untuk menyatakan a *putar ke kiri*: bagian yang masuk X digeser N posisi ke kiri, dimana bit yang digeser ke kiri ditempatkan pada posisi paling kanan.

$$P \leftarrow (P + F(Q,R,S) + B_0 + C_1) <<< 7 \qquad P \leftarrow (P + F(Q,R,S) + B_8 + C_9) <<< 7$$
$$S \leftarrow (S + F(P,Q,R) + B_1 + C_2) <<< 12 \qquad S \leftarrow (S + F(P,Q,R) + B_9 + C_{10}) <<< 12$$
$$R \leftarrow (R + F(S,P,Q) + B_2 + C_3) <<< 17 \qquad R \leftarrow (R + F(S,P,Q) + B_{10} + C_{11}) <<< 17$$
$$Q \leftarrow (Q + F(R,S,P) + B_3 + C_4) <<< 22 \qquad Q \leftarrow (Q + F(R,S,P) + B_{11} + C_{12}) <<< 22$$
$$P \leftarrow (P + F(Q,R,S) + B_4 + C_5) <<< 7 \qquad P \leftarrow (P + F(Q,R,S) + B_{12} + C_{13}) <<< 7$$
$$S \leftarrow (S + F(P,Q,R) + B_5 + C_6) <<< 12 \qquad S \leftarrow (S + F(P,Q,R) + B_{13} + C_{14}) <<< 12$$
$$R \leftarrow (R + F(S,P,Q) + B_6 + C_7) <<< 17 \qquad R \leftarrow (R + F(S,P,Q) + B_{14} + C_{15}) <<< 17$$

Gambar 8-11. 16 iterasi selama putaran pertama dalam fase di MD5.

Putaran kedua menggunakan fungsi tersebut G dengan cara yang sama, padahal H Dan $SAYA$ digunakan pada putaran ketiga dan keempat. Setiap langkah terdiri dari 64 iterasi, setelah itu fase berikutnya dimulai, tetapi sekarang dengan nilai itu P, Q, R, Dan S miliki pada saat itu.