

Laporan Tugas Akhir
Mata Kuliah EC 7010 Keamanan Sistem Lanjut
Dosen : Dr. Ir. Budi Rahardjo, M.Sc, Ph.D

PROTOKOL KEAMANAN BERBASIS *PROXY* DALAM JARINGAN *MOBILE DEVICES*

Nama : Sumaryanto
NIM : 23203132



**PROGRAM MAGISTER TEKNIK ELEKTRO
BIDANG KHUSUS TEKNOLOGI INFORMASI DIKMENJUR
INSTITUT TEKNOLOGI BANDUNG
2005**

ABSTRAK

PROTOKOL KEAMANAN BERBASIS *PROXY* DALAM JARINGAN *MOBILE DEVICES*

Sumaryanto
NIM 23203132

Tugas Akhir EC 7010 (Keamanan Sistem Lanjut)
Dosen : Dr. Ir. Budi Rahardjo, M.Sc, Ph.D
Topik ini diajukan dan disetujui pada tanggal 24 September 2004
Diperbaiki pada tanggal 31 Januari 2005
sumaryanta@yahoo.com

Protokol keamanan merupakan hal yang sangat penting dalam sistem informasi terutama untuk menjaga kerahasiaan informasi, terlebih jika pendistribusian informasi tersebut melalui jaringan perangkat bergerak. Dalam tugas ini akan diuraikan sistem komunikasi yang dirancang untuk keamanan data yang dapat dipercaya. Semua obyek di dalam sistem tersebut dirancang menggunakan *proxy software* yang diyakini dapat bekerja pada *hardware* ataupun pada komputer yang digunakan. Sistem keamanan dan privasi dijalankan dengan menggunakan dua buah protokol terpisah. Sebuah protokol digunakan untuk komunikasi perangkat ke *proxy (device-to-proxy)*, dan sebuah protokol lagi digunakan untuk komunikasi *proxy ke proxy (proxy-to-proxy)*. Dengan menggunakan dua buah protokol tersebut maka dapat dijalankan sebuah protokol yang handal untuk pembuktian keaslian informasi.

Untuk perangkat bergerak dan berukuran kecil pada jaringan nirkabel protokol keamanan menggunakan protokol perangkat ke *proxy*. Sedangkan protokol *proxy ke proxy* digunakan pada kerangka kerja *Simple Public Key Infrastructure/Simple Distributed Systems Infrastructure (SPKI/SDSI)*. Penggunaan dua buah protokol yang berbeda membuat sebuah sistem otomatisasi menjadi aman, terukur, efisien, dan mudah untuk dipelihara. *Proxy* bekerja dengan cepat pada sebuah komputer sehingga mampu mengimplementasikan algoritma kriptografi yang dapat dipercaya.

Kata kunci : otorisasi, sertifikat, serangkaian sertifikat, penemuan serangkaian sertifikat, peralatan bergerak, *pervasive*, protokol, *proxy*, keamanan, *wireless*.

KATA PENGANTAR

Segala puji dan syukur kami panjatkan kehadirat Allah SWT yang telah mencurahkan nikmat yang tak terhingga kepada setiap hamba-Nya. Karena pertolongan-Nya semata maka Laporan Tugas Akhir "**Protokol Keamanan Berbasis Proxy dalam Jaringan Mobile Devices**" ini dapat terwujud. Shalawat dan salam semoga tercurah kepada Nabi Muhammad SAW yang telah mendidik umatnya untuk memahami dan melaksanakan aspek lahir dan batin dari setiap ritual ibadah.

Pada kesempatan ini kami sampaikan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu penyelesaian laporan tugas akhir ini. Terima kasih kami sampaikan kepada :

1. Bapak Dr. Ir. Budi Rahardjo, M.Sc, Ph.D sebagai dosen mata kuliah EC-7010 Keamanan Sistem Lanjut, yang telah memberikan kuliah dan bimbingan hingga terselesaikannya tugas ini.
2. Rekan-rekan sejawat sesama mahasiswa S2 TI Dikmenjur yang telah membantu penyusunan laporan tugas akhir ini.
3. Semua pihak yang tidak dapat kami sebutkan satu-persatu, yang telah membantu penyelesaian laporan tugas akhir ini.

Semoga segala dukungan yang diberikan selalu mendapatkan balasan yang melipat dari Allah SWT. Kekurangan-kekurangan dan ketidak-sempurnaan tentu masih melekat dalam laporan ini. Oleh karena itu dengan penuh kerendahan hati kami mohon saran, kritik dan masukan-masukan dalam rangka memperbaiki laporan ini.

Kami berharap semoga laporan tugas akhir ini dapat memberikan kontribusi dan membawa manfaat bagi para pembaca.

Bandung, 31 Januari 2005
Penyusun,

DAFTAR ISI

	Halaman
ABSTRAK	i
KATA PENGANTAR	ii
DAFTAR ISI	iii
DAFTAR GAMBAR	v
DAFTAR TABEL	vi
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Tujuan	2
1.3 Sistematika Pembahasan	2
BAB II KAJIAN TEORI	3
2.1 Konsep Keamanan	3
2.2 Kemanan Sistem Terdistribusi	3
2.2.1 Ancaman dan serangan	3
2.2.2 Metode penyerangan	4
2.2.3 Teknik keamanan	4
2.3 Masalah Keamanan Sistem <i>Wireless</i>	4
2.4 <i>Cryptographic Protocol</i>	5
BAB III PROTOKOL KEAMANAN BERBASIS <i>PROXY</i> DALAM JARINGAN <i>MOBILE DEVICES</i>	6
3.1 Arsitektur Sistem	6
3.1.1 Perangkat sistem	8
3.1.2 <i>Proxy</i>	8
3.1.3 Server dan jaringan server	10
3.1.4 Komunikasi melalui event	10
3.1.5 Penemuan sumber (<i>resource discovery</i>)	10
3.1.6 Model keamanan	11
3.1.7 Inisialisasi perangkat	13
3.2 Protokol <i>device-to-proxy</i> untuk perangkat <i>wireless</i>	13
3.2.1 Komunikasi	13
3.2.2 Protokol RF	14
3.2.3 Keamanan	15
3.2.4 Lokasi	20
3.3 Protokol <i>proxy-to-proxy</i>	20
3.3.1 Integrasi SPKI/SDSI	20
3.3.2 Protokol	21
3.3.3 Pertimbangan Keamanan Tambahan	24

BAB IV ANALISIS DAN EVALUASI	26
4.1. Protokol perangkat ke <i>proxy</i> (<i>devices-to-proxy</i>)	26
4.1.1 Persyaratan memori	26
4.1.2 Persyaratan pemrosesan	27
4.2. SPKI/SDSI	27
BAB V KESIMPULAN DAN SARAN	28
5.1. Kesimpulan	28
5.2. Saran	29
DAFTAR PUSTAKA	30

DAFTAR GAMBAR

	Halaman
Gambar 3.1 Arsitektur Sistem	7
Gambar 3.2 Arsitektur Sistem Komunikasi <i>Device-toProxy</i>	8
Gambar 3.3 Model keamanan	12
Gambar 3.4 Arsitektur komunikasi	14
Gambar 3.5 Authentication	17
Gambar 3.6 Enkripsi kunci simetris	19
Gambar 3.7 Enkripsi kunci asimetris	19
Gambar 3.8 <i>SPKI/SDSI Proxy to proxy ACL</i>	23
Gambar 3.9 <i>Example Layering of Protocol</i>	25

DAFTAR TABEL

	Halaman
Tabel 3.1 Notasi Kriptografi	16
Tabel 3.2 Pihak yang terlibat dalam skenario	18
Tabel 4.1 <i>Code and data size on the Atmel processor</i>	26
Tabel 4.2 <i>Performance of encryption and authentication code</i>	27
Tabel 4.3 Analisis protokol <i>proxy</i> ke <i>proxy</i>	27

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi komunikasi dan penyimpanan data dengan menggunakan komputer memungkinkan pengiriman data jarak jauh yang relatif cepat dan murah. Di lain pihak pengiriman data jarak jauh melalui gelombang radio maupun menggunakan media lain yang digunakan *public* sangat memungkinkan pihak lain dapat menyadap dan mengubah data yang dikirimkan. Perkembangan teknologi internet telah menjadikan salah satu media utama pertukaran informasi. Tidak semua informasi bersifat terbuka untuk umum. Karena Internet merupakan jaringan komputer yang luas dan bersifat publik, maka diperlukan suatu usaha untuk menjamin keamanan dan kerahasiaan pengiriman informasi tersebut. Di satu pihak, telah banyak usaha-usaha untuk menjamin keamanan suatu informasi. Di pihak lain, tetap saja ada orang-orang dengan maksud tertentu yang berusaha untuk menembus sistem keamanan tersebut. Lubang keamanan dapat terjadi di beberapa tempat yaitu : pada sistem operasi, sistem aplikasi, dan pada jaringan komputer [2]. Oleh karena itu, diperlukan suatu sistem yang mampu mengamankan pendistribusian informasi tersebut.

Salah satu cara yang ditempuh untuk mengatasi masalah ini adalah dengan menerapkan kriptografi yang menggunakan transformasi data sehingga data yang dihasilkan tidak dapat dimengerti oleh pihak ketiga. Transformasi ini memberikan solusi pada dua masalah keamanan data yaitu masalah privasi (*privacy*) dan keautentikan (*authentication*). Privasi mengandung arti bahwa data yang dikirimkan hanya dapat dimengerti informasinya oleh penerima yang sah. Sedangkan keautentikan mencegah pihak ketiga untuk mengirimkan data yang salah atau mengubah data yang dikirimkan.

Keamanan informasi merupakan hal yang sangat penting dan utama untuk menjaga kerahasiaan, apalagi informasi tersebut hanya boleh diketahui oleh pihak-pihak tertentu saja dan informasi tersebut berada dalam jaringan alat-alat yang bergerak. Banyak hambatan untuk diatasi ketika menggabungkan alat-alat yang dipakai dan berada dalam lingkungan yang sangat luas. Hambatan tersebut meliputi bagaimana merancang perangkat yang cukup cerdas, cepat untuk saling bekerja sama, meningkatkan *performance*, dan memungkinkan konektivitas yang tinggi, serta keamanan dari sistem menjadi faktor kunci. Perangkat yang hanya mengijinkan akses bagi pemakai yang sah dan juga harus menjaga

komunikasi yang aman ketika menstransmisi atau menerima informasi pribadi.

1.2 Tujuan

Pada tulisan ini akan dipaparkan pengertian umum tentang konsep keamanan, keamanan sistem terdistribusi, masalah keamanan sistem *wireless*, *cryptographic* protokol, dan pembahasan detail tentang protokol keamanan berbasis *proxy* pada jaringan *mobile devices*.

1.3 Sistematika Pembahasan

Sistematika pembahasan dalam tulisan ini meliputi :

a. Bab I Pendahuluan

Meliputi latar belakang, tujuan, dan sistematika pembahasan.

b. Bab II Kajian Teori

Meliputi konsep keamanan, keamanan sistem terdistribusi, masalah keamanan sistem *wireless*, dan *cryptographic* protokol.

c. Bab III Pembahasan

Pembahasan mengenai protokol keamanan berbasis *proxy* dalam jaringan *mobile devices*.

d. Bab IV Analisis

Meliputi analisis persyaratan memori, persyaratan pemrosesan, dan evaluasi SPKI/SDSI.

e. Bab V Kesimpulan dan Saran

BAB II KAJIAN TEORI

2.1 Konsep keamanan

Keamanan sering dipandang hanyalah merupakan masalah teknis yang melibatkan dapat atau tidaknya ditembusnya suatu sistem. Pada pandangan makro keamanan sendiri memiliki konsep yang lebih luas, yang berkaitan dengan ketergantungan suatu institusi terhadap institusi lainnya. Di dalam aplikasinya suatu pembentukan sistem yang aman akan mencoba melindungi adanya beberapa kemungkinan serangan yang dapat dilakukan pihak lain terhadap kita diantaranya [4] :

- a. *Intrusion* : Penyerangan jenis ini seseorang penyerang akan dapat menggunakan sistem komputer yang kita miliki. Sebagian penyerangan jenis ini menginginkan akses sebagaimana halnya pengguna yang memiliki hak untuk mengakses sistem.
- b. *Denial of services* : Penyerangan jenis ini mengakibatkan pengguna yang sah tidak dapat mengakses system. Seringkali orang melupakan jenis serangan ini dan hanya berkonsentrasi pada *intrusion* saja.
- c. *Joyrider* : Penyerangan jenis ini disebabkan oleh orang yang merasa iseng dan ingin memperoleh kesenangan dengan cara menyerang suatu sistem. Rata-rata mereka karena rasa ingin tahu, tetapi ada juga yang menyebabkan kerusakan atau kehilangan data.
- d. *Vandal* : Jenis serangan ini bertujuan untuk merusak sistem, yang sering ditujukan untuk *site-site* besar.
- e. *Scorekeeper* : Jenis serangan ini hanyalah bertujuan untuk mendapatkan reputasi dengan cara mengacak sistem sebanyak mungkin.
- f. Mata-mata : Jenis serangan ini bertujuan untuk memperoleh data atau informasi rahasia dari pihak kompetitor.

2.2 Keamanan sistem terdistribusi

2.2.1 Ancaman dan serangan

Tujuan utama dengan adanya sistem keamanan adalah untuk membatasi akses informasi dan bersumber hanya untuk pemakai yang memiliki hak akses. Ancaman keamanan terdiri dari [4] :

- a. *Leakgace* (kebocoran) : pengambilan informasi oleh penerima yang tidak berhak,
- b. *Tampering* : pengubahan informasi yang tidak legal,

- c. *Vandalism* (perusakan) : gangguan operasi sistem tertentu, dimana si pelaku tidak mengharap keuntungan apapun.

2.2.2 Metode penyerangan

Klasifikasi metode penyerangan adalah [4] :

- a. *Eavesdropping* : mendapatkan duplikasi tanpa ijin,
- b. *Masquerading* : mengirim atau menerima pesan menggunakan identitas lain tanpa ijin mereka,
- c. *Message tampering* : mencegat atau menangkap pesan dan mengubah isinya sebelum dilanjutkan ke penerima sebenarnya.
- d. *Replaying* : menyimpan pesan yang ditangkap untuk pemakaian berikutnya,
- e. *Denial of service* : membanjiri saluran atau sumber lain dengan pesan yang bertujuan untuk menggagalkan pengaksesan pemakai lain.

2.2.3 Teknik keamanan

Teknik keamanan adalah hal penting dalam menjaga kerahasiaan data. Proses enkripsi di dalam teknik keamanan merupakan proses pengkodean pesan untuk menyembunyikan isi. Algoritma enkripsi modern menggunakan kunci kriptografi dimana hasil enkripsi tidak dapat didekripsi tanpa kunci yang sesuai. Kriptografi adalah suatu ilmu yang mempelajari bagaimana membuat suatu pesan menjadi aman selama pengiriman dari pengirim (*sender*) sampai ke penerima (*receiver*) [1]. Pesan tersebut disebut *plaintext*, proses untuk mengubah *plaintext* menjadi suatu bentuk yang tidak dapat dibaca isinya disebut enkripsi. Pesan yang terenkrip disebut *ciphertext*. Proses untuk mengubah *ciphertext* ke pesan aslinya (*plaintext*) disebut dekripsi. Hubungan antara *plaintext*, *ciphertext*, enkripsi dan dekripsi dapat ditulis dalam bentuk sebagai berikut :

- $C = E (M)$ dimana: $C = ciphertext$, $E =$ proses enkripsi, $M = plaintext$.
- $M = D (C)$ dimana: $C = ciphertext$, $D =$ proses dekripsi, $M = plaintext$

2.3 Masalah Keamanan Sistem Wireless

Jaringan dengan sistem nirkabel atau wireless memiliki permasalahan keamanan yang khusus. Beberapa hal yang mempengaruhi aspek keamanan dari sistem wireless atau nirkabel adalah [3] :

- a. Perangkat pengakses informasi yang menggunakan sistem wireless biasanya berukuran kecil sehingga mudah dicuri, maka informasi yang ada di dalamnya dapat jatuh ke orang lain yang tidak berhak menggunakannya.
- b. Penyadapan (*main-in-the-middle attack*) dapat dilakukan lebih mudah karena tidak perlu mencari jalur kabel untuk di-‘tap’. Sistem yang tidak menggunakan pengamanan enkripsi, atau menggunakan enkripsi yang mudah dipecah, akan mudah ditangkap.
- c. Perangkat wireless yang kecil membatasi kemampuan perangkat dari sisi CPU, RAM, kecepatan komunikasi, dan catu daya. Akibatnya sistem keamanan harus memperhatikan batasan ini.
- d. Pengguna tidak dapat membuat sistem pengaman sendiri dan hanya bergantung kepada vendor.
- e. Adanya batasan jangkauan radio dan interferensi menyebabkan ketersediaan servis menjadi terbatas.

2.4 Cryptographic Protocol

Suatu protokol adalah serangkaian langkah yang melibatkan dua pihak atau lebih dan dirancang untuk menyelesaikan suatu tugas [1]. *Cryptographic protocol* adalah suatu protokol yang menggunakan kriptografi. Protokol ini melibatkan sejumlah algoritma kriptografi. Protokol digunakan untuk mengabstraksikan proses penyelesaian suatu tugas dari mekanisme yang digunakan. Protokol komunikasi biasanya sama meskipun diimplementasikan pada PC. Penyerangan terhadap protokol dapat ditujukan pada beberapa hal sebagai berikut [1] :

- a. algoritma *cryptographic* yang digunakan dalam protokol,
- b. teknik *cryptographic* yang digunakan untuk mengimplementasikan algoritma dan protokol,
- c. protokol itu sendiri.

BAB III

PROTOKOL KEAMANAN BERBASIS *PROXY*

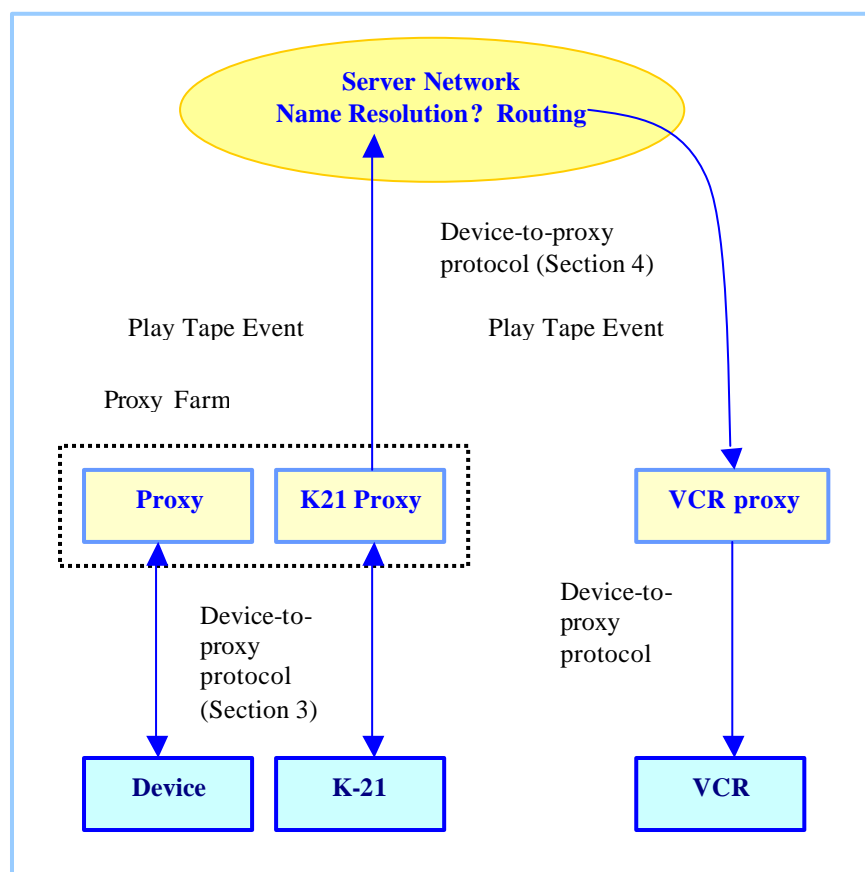
DALAM JARINGAN *MOBILE DEVICES*

3.1 Arsitektur sistem

Untuk mengimplementasikan bentuk keamanan yang dapat dipercaya dalam *personal communication* adalah dengan menggunakan sebuah infrastruktur *public key*. Sebuah algoritma kriptografi *public key* yang umum digunakan adalah Ron Rivest, Adi Shamir, dan Len Adleman (RSA). Agar peralatan yang digunakan dalam sebuah model keamanan *public key* pada jaringan nirkabel menjadi lebih sederhana, maka setiap alat diberi sebuah *proxy software*. Semua obyek di dalam sistem itu (alat-alat, perangkat yang dipakai, perangkat *software*, dan pemakai) dihubungkan ke *proxy software* yang diyakini dapat bekerja dengan baik pada sebuah *processor* yang disimpan di dalam komputer. *Proxy* yang bekerja pada sebuah *processor*, akan menjamin perangkat berkomunikasi dengan aman. Jika perangkat itu mempunyai kemampuan yang minimal dan komunikasi ke *proxy* melalui jaringan dengan kabel maupun tanpa kabel, maka komunikasi itu akan menyatu ke sebuah protokol perangkat ke *proxy*. Berdasarkan SPKI/SDKI *proxy-proxy* saling berkomunikasi dengan aman menggunakan sebuah protokol *proxy* ke *proxy*. Menggunakan dua buah protokol yang berbeda memungkinkan untuk menjalankan sebuah protokol keamanan yang murah dan protokol yang handal untuk pembuktian keaslian sumber serta komunikasi pada perangkat yang dapat dipercaya.

Berdasarkan pemikiran di atas, maka sebuah sistem otomatisasi dasar memungkinkan komunikasi menjadi lebih aman, efisien, dan akses ke jaringan pada perangkat bergerak menjadi mudah. Pada sistem ini setiap pemakai menggunakan sebuah kunci yang disebut K-21. Kunci tersebut mengidentifikasi pemakai dan lokasi pemakainya. Identitas pemakai dan informasi lokasi ditransmisi dengan aman ke *proxy software* pemakai yang menggunakan protokol perangkat ke *proxy (device-to-proxy)*. Alat-alat tersebut dapat berupa *mobile* yang dapat berpindah tempat. Atribut akan mencari semua perangkat yang dapat dikontrol dan ditampilkan untuk menemukan perangkat yang terdekat, atau perangkat yang paling tepat. Dengan menggunakan SPKI/SDKI maka sistem keamanan tidak mau kompromi terhadap pemakai baru dan perangkat yang memasuki sistem tersebut. Penggunaan dua buah protokol yang berbeda dan penggunaan SPKI/SDKI di dalam protokol *proxy* ke *proxy* menjamin sebuah sistem otomatisasi dasar menjadi aman, terukur, efisien, dan mudah untuk dipelihara.

Arsitektur sistem mempunyai tiga komponen utama yaitu perangkat, *proxy*, dan *server*. Perangkat menunjuk pada tipe apapun dari jaringan sumber yang terbagi, baik *hardware* maupun *software*. Perangkat tersebut dapat berupa sebuah printer, kamera keamanan nirkabel, lampu, atau sebuah *software*. Karena protokol komunikasi antara perangkat dan *bandwidth* sangat besar variasinya, maka setiap perangkat mempunyai sebuah *proxy* yang unik untuk menyatukan *interface*-nya dengan perangkat yang lain. *Server* menyediakan fasilitas nama untuk bermacam-macam perangkat. Sebuah koresponden *one-to-one* antara perangkat dan *proxy* serta semua pemakai perangkat telah dilengkapi dengan kunci K-21, di mana *proxy-proxy*-nya bekerja pada komputer yang dapat dipercaya. Hal ini menjadikan sistem hanya memerlukan perangkat, *proxy*, dan server. Sistem ini digambarkan seperti terlihat di Gambar 3.1 berikut ini :

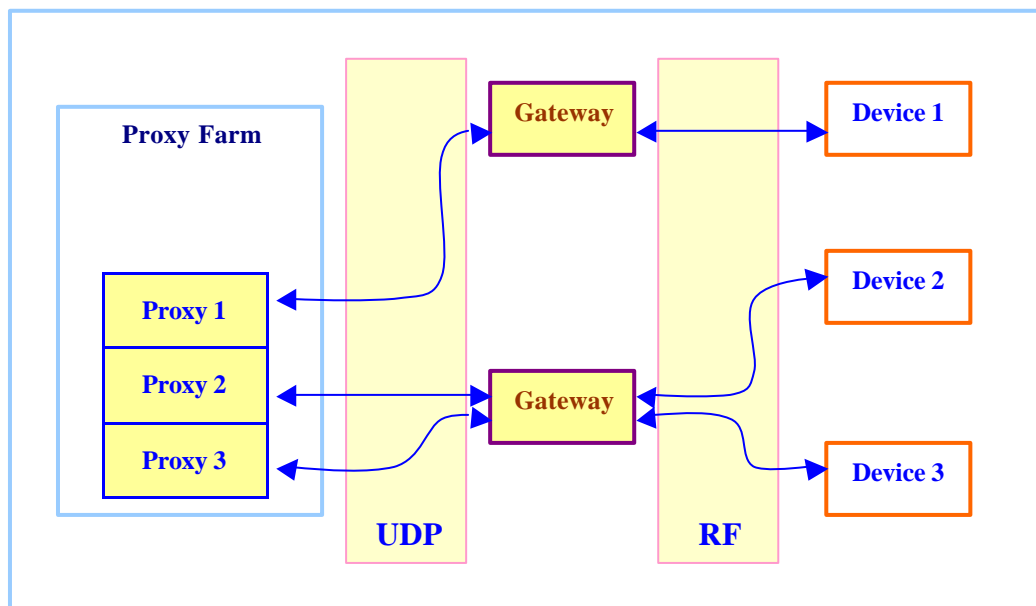


Gambar 3.1 Arsitektur sistem [6]

3.1.1 Perangkat Sistem

Tiap *hardware* atau *software*, mempunyai sebuah *proxy software* yang dapat dipercaya. Pada *hardware*, *proxy* bekerja pada sebuah *processor* yang tersimpan di dalamnya, atau pada jaringan yang terhubung dengan perangkat tersebut. Pada *software*, *proxy software* secara otomatis dapat dimasukkan dalam *software* tersebut.

Setiap perangkat berkomunikasi dengan *proxy* miliknya sendiri sesuai dengan protokol yang dimiliki perangkat itu. Suatu nirkabel printer dalam *ethernet* dapat berkomunikasi dengan *proxy*-nya menggunakan sebuah TCP/IP. Suatu kamera nirkabel menggunakan sebuah protokol nirkabel untuk tujuan yang sama. K-21 berkomunikasi dengan memakai protokol khusus perangkat ke *proxy* yang ditunjukkan pada Gambar 3.2 di bawah ini :



Gambar 3.2 Arsitektur sistem komunikasi *device-to-proxy* [6]

3.1.2 Proxy

Proxy merupakan *software* yang bekerja pada sebuah jaringan komputer yang tampak. Fungsi utama *proxy* adalah mewakili perangkat untuk membuat keputusan kontrol akses. *Proxy* juga memiliki fungsi skunder seperti mengatur tindakan tertulis (*scripted action*) mewakili perangkat dan *interfacing* dengan suatu layanan *directory*. *Proxy* memberikan suatu *Application Programming Interface* (API) yang sangat sederhana

terhadap sebuah perangkat. Metode *send to proxy* () digunakan oleh perangkat untuk mengirim pesan ke *proxy*. Metode *send to device* () digunakan oleh *proxy* untuk mengirim pesan ke perangkat. Ketika sebuah *proxy* menerima pesan dari *proxy* yang lain, maka *proxy* tersebut menterjemahkannya ke dalam format yang dapat dipahami oleh masing-masing *proxy*. Kemudian *proxy* menyampaikan pesan ke perangkat. Setiap saat *proxy* menerima pesan, *proxy* menterjemahkan pesan tersebut ke dalam format yang dapat dipahami oleh semua *proxy*, kemudian meneruskannya ke *proxy* yang lain. Pada saat *proxy* menerima pesan sebelum menampilkan terjemahan, dan memberikan pesan tersebut ke perangkat, *proxy* melakukan pengecekan akses kontrol.

Untuk kemudahan administrasi, *proxy* dikelompokkan oleh administrasinya. Seperangkat administrator *proxy* disebut *proxy farm*. Perangkat ini khususnya meliputi *proxy* K-21 milik administrator, yang dikenal dengan *proxy* akar (*root proxy*) dari *proxy farm*. Ketika administrator menambahkan perangkat baru ke dalam sistem, secara otomatis *proxy* di dalam perangkat tersebut diberi ACL (*Access Control List*) *default*, yang berupa duplikat ACL bagi *proxy* K21 milik administrator. Secara manual administrator dapat mengubah ACL jika dikehendaki. Keuntungan dari arsitektur berbasis *proxy* adalah dapat menampilkan masalah virus yang dapat menembus jaringan komputer. *Software scanning* virus yang canggih dapat diinstall dalam *proxy* tersebut, sehingga dapat membaca dengan cepat kode apapun sebelum di *download* kedalam perangkat-perangkat tersebut. Beberapa kode yang dimaksud adalah sebagai berikut :

- **CommandEvent** : digunakan untuk mengaktifkan atau mematikan sebuah instruksi ke peralatan.
- **ErrorEvent** : digunakan untuk menyampaikan sebuah pesan pada pendengar ketika terjadi kesalahan,
- **StatusChangeEvent** : digunakan untuk memberi tanda, ketika sebuah perangkat mengubah lokasinya,
- **QueryEvent** : ketika *server* menerima sebuah *QueryEvent* maka server akan memiliki DNS (*Domain Name Service*) atau INS (*Intentional Naming System*) pada *QueryEvent* tersebut dan merespon hasil pengecekan tersebut, dalam *QueryEvent*.
- **ResponseEvent** : menghasilkan sebuah respon *QueryEvent*.

3.1.3 Server dan jaringan server

Jaringan server terdiri dari kumpulan beberapa *server* dan *router*, dimana tiap-tiap *server* bertindak baik sebagai *server* maupun *router*. Sama halnya dengan penentuan nama yang terdapat pada INS, dimana peralatan diberi alamat IP, ini juga dapat meroutekan event. Jika nama tujuan sebuah *event* sesuai untuk *proxy* yang lebih dari satu maka jaringan server akan mengirimkan event tersebut ke semua alamat yang sesuai.

Ketika sebuah *proxy* dalam keadaan *online*, maka *proxy* tersebut akan mencatat nama perangkat yang mewakili salah satu *server* tersebut. Ketika sebuah *proxy* memakai satu *server* untuk melakukan pemeriksaan atas sebuah nama, maka *server* tersebut mencari nama pada direktorinya yang sesuai dengan nama yang diberikan, dan mengembalikan alamat-alamat IP-nya.

3.1.4 Komunikasi melalui event

Sistem yang dikembangkan menggunakan sebuah mekanisme komunikasi berbasis peristiwa (*event-based*). Semua pesan yang melalui *proxy-proxy* menunjukkan adanya signal bahwa beberapa peristiwa telah terjadi. Contoh, sebuah bola lampu bisa menghasilkan *light-on* (lampu hidup) dan *light-off* (lampu mati). Untuk menerima pesan-pesan tersebut *proxy x* dapat menambah sendiri perintahnya ke *proxy y*. Maka ketika *y* menggerakkan sebuah *event*, *x* akan menerima salinan. Sebagai tambahan, sistem tersebut mempunyai beberapa kategori *pre-defined* event peristiwa sebelum didefinisikan, yang menerima perlakuan khusus baik *proxy* maupun lapisan *server*.

Keuntungan utama dari sebuah mekanisme berbasis *event* (*event-based mechanism*) adalah bahwa dapat mengurangi kebutuhan secara berulang-ulang pada peralatan untuk menentukan perubahan statusnya. Sebaliknya jika terjadi sebuah perubahan, maka perangkat itu akan mengirimkan sebuah *event* kepada semua pendengar. Sistem seperti Sun Microsystem Jini [7] mengeluarkan *device-driver* (RM1) kepada semua yang ingin mengontrol peralatan yang diberikan. Kemudian mekanisme itu bisa membuat panggilan lokal pada perangkat *driver*, yang diterjemahkan dalam panggilan RM1 pada peralatan itu sendiri.

3.1.5 Penemuan sumber (*resource discovery*)

Mekanisme untuk penemuan sumber daya sama dengan protokol penemuan sumber yang digunakan oleh Jini. Ketika sebuah perangkat dalam keadaan *online*, maka perangkat

itu menginstruksikan *proxy*-nya untuk menyiarkan secara berulang-ulang suatu permintaan ke *server* pada sub jaringan lokal. Permintaan itu berisi nama perangkat dan alamat IP serta *port proxy*-nya. Ketika *server* ini menerima salah satu *request*, maka *server* tersebut akan menginformasikan nama, alamat/IP, *port* ke *proxy*, hal ini berarti menambahkan nama, alamat/IP, *port* ke direktorinya. Maka *proxy* harus memperbaharui informasinya secara periodik dengan mengirimkan seperangkat alamat nama/IP yang sama ke *server*, kalau tidak maka *server* itu akan menghapus alamat nama/IP dari direktorinya.

Pada model ini, jika sebuah perangkat secara diam-diam menjadi *offline* atau alamat IP berubah, maka informasi *proxy* tidak akan diperbaharui dan *server* secara cepat akan mencatat dan menghapus dari *directory* atau membuat informasi baru dengan alamat IP yang baru. Contoh, bayangkan suatu perangkat dengan nama **[name=foo]** yang mempunyai *proxy* bekerja pada **10.1.2.3 : 4011**. Ketika perangkat tersebut dinyalakan, maka perangkat tersebut akan menginformasikan kepada *proxy*-nya bahwa dia telah *online*, dengan menggunakan suatu protokol seperti protokol perangkat ke *proxy*. *Proxy* tersebut mulai mengirimkan paket-paket (*lease-request*) dalam bentuk (**[name : foo]10.1.2.3 :4011**) pada sub jaringan lokal.

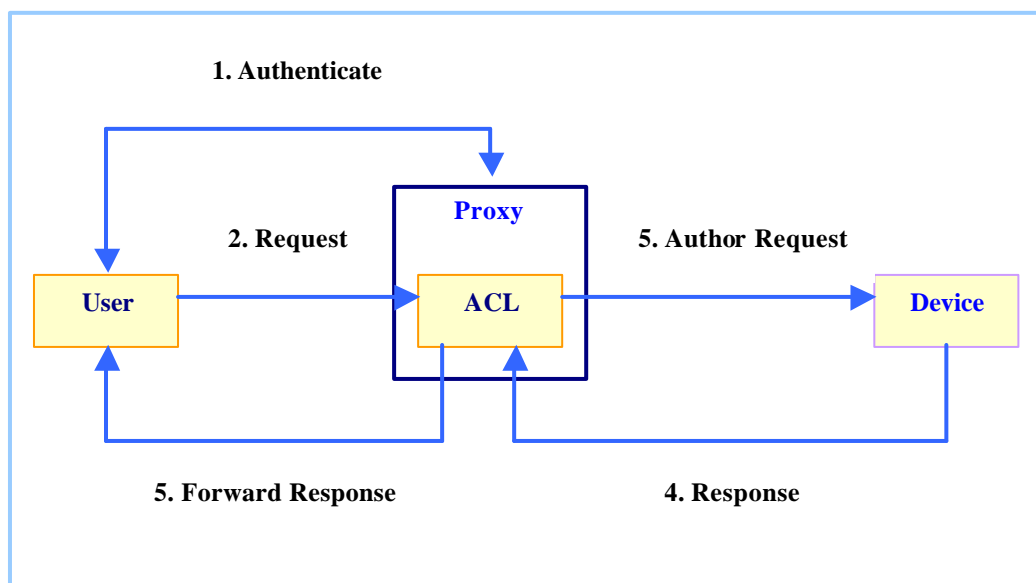
Jika suatu *server* menerima salah satu paket tersebut, server akan mengecek *directory*-nya untuk **[name = foo]**. Jika **[name=foo]** tidak dijumpai maka *server* akan membuat suatu informasi baru untuknya dengan menambahkan seperangkat nama dan alamat IP kepada *directory*-nya. Jika **[name=foo]** terdapat di *directory*, maka *server* akan memperbaharui informasinya. Umpamakan beberapa saat kemudian perangkat tersebut akan mati. Ketika perangkat tersebut mati, maka menyebabkan *proxy offline*, sehingga tidak lagi dapat mengirimkan paket permintaan informasi. Informasi *device* tersebut tidak lagi diperbaharui. Setelah beberapa saat, sesuai periode waktu yang telah ditentukan, maka *server* tersebut mengakhiri informasi yang tidak diperbaharui dan menghapusnya dari *directory*.

3.1.6 Model keamanan

Proxy dan perangkat berbagi sebuah kunci rahasia. Kunci rahasia tersebut, memungkinkan *proxy* dan perangkat saling berkomunikasi dengan menggunakan keaslian kunci simetrik dan enkripsi. Pengoperasian kunci simetris hanya memerlukan sumber daya yang jauh lebih sedikit dibandingkan dengan *public-key*, sehingga perangkat tersebut dapat melakukan perhitungan dengan sebuah *micro-controller*. Semua komunikasi yang terjadi

akan melewati *proxy*, sehingga *proxy* itu dapat membuktikan keaslian informasi dan kemudian menyampaikan hasil pembuktian tersebut ke pemakai perangkat itu. Aliran komunikasi ditunjukkan pada Gambar 3.3 di bawah, dengan langkah-langkah seperti diuraikan berikut ini :

- a. *proxy* dan pemakai saling membuktikan keaslian informasi dan saling membuat sebuah saluran komunikasi yang aman,
- b. pemakai mengecek permintaannya yang dikirim ke *proxy*,
- c. *proxy* memeriksa daftar akses kontrolnya untuk menentukan apakah pemakai diijinkan untuk menampilkan permintaannya. Jika pengecekan berhasil maka *proxy* itu akan mengirimkan pesan pada perangkat tersebut, sebaliknya jika tidak diijinkan maka *proxy* akan merespon dengan sebuah pesan kesalahan,
- d. perangkat akan melakukan tindakan yang diminta dan mengirim sebuah respon kembali ke *proxy*,
- e. *proxy* mengirim kembali respon tersebut ke pemakai.



Gambar 3.3 Model keamanan [9]

3.1.7 Inisialisasi perangkat

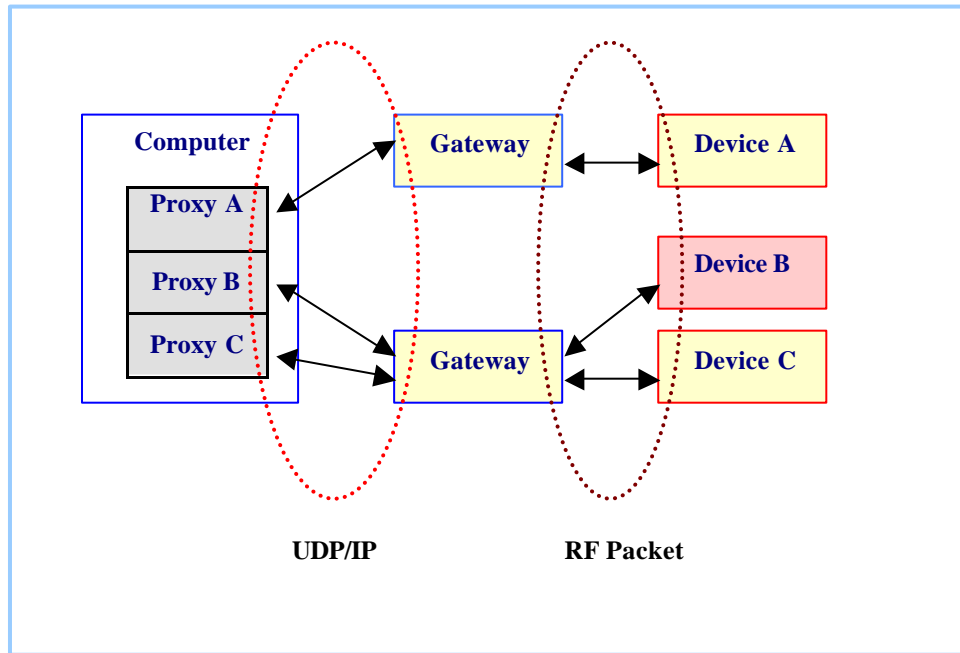
Ketika sebuah perangkat diberi inisial, maka perangkat tersebut harus menentukan *proxy*-nya dan harus mendapat sebuah kunci rahasia yang dipakai bersama dengan *proxy* tersebut. Hal ini dilakukan secara fisik dengan menghubungkan perangkat tersebut ke komputer yang akan menjalankan *proxy*. Ketika perangkat tersebut dihubungkan ke komputer, maka dibuat sebuah *proxy* dan kemudian *proxy* akan menciptakan sebuah kunci rahasia yang acak dan membaginya bersama perangkat itu. Inisialisasi ini terjadi secara langsung dan mudah bagi pemakai yang ingin memberi inisial terhadap perangkat itu. Pemakai tidak perlu menampilkan konfigurasi manual apapun.

3.2 Protokol *device-to-proxy* untuk perangkat *wireless*

3.2.1 Komunikasi

Protokol perangkat ke *proxy* sangat bervariasi untuk jenis-jenis perangkat yang berbeda. Secara khusus kita membahas perangkat ringan dengan koneksi jaringan nirkabel yang mempunyai *bandwith* rendah (*low bandwith*) dan CPU dengan kecepatan rendah, serta perangkat berat dengan koneksi *bandwith* yang lebih tinggi dan CPU yang lebih cepat. Kita berasumsi bahwa perangkat berat tersebut dapat menjalankan *software proxy* secara lokal, sebagai contoh *proxy* untuk sebuah printer dapat bekerja pada CPU printer. Dengan *proxy* lokal tersebut, tidak diperlukan protokol yang canggih untuk menjaga keamanan komunikasi perangkat ke *proxy* dengan asumsi bahwa bagian kritis dari perangkat adalah anti keretakan. Untuk perangkat-perangkat besar, *proxy* harus bekerja di tempat lain.

Komunikasi menggunakan RF (*Radio Frequency*) antara perangkat dengan *proxy* dikendalikan oleh sebuah gerbang (*gateway*) yang menterjemahkan komunikasi RF tersebut ke dalam paket UDP/IP yang kemudian dikirimkan melalui jaringan ke *proxy*. *Gateway* juga bekerja pada arah yang berlawanan dan mengubah paket UDP/IP dari *proxy* ke paket RF serta mentransmisikan ke perangkat tersebut. Sebuah gambaran dari komunikasi ditunjukkan pada Gambar 3.4 di bawah ini. Setiap satu komputer menjalankan tiga buah *proxy* untuk tiga perangkat terpisah. Gambar 3.4 itu juga menunjukkan bagaimana sebuah *gateway* ganda dapat digunakan. Perangkat A menggunakan suatu *gateway* yang berbeda dengan perangkat B dan C.



Gambar 3.4 Arsitektur komunikasi [6]

3.2.2 Protokol RF

Semua perangkat komunikasi pada saluran RF dikodekan dengan menggunakan *proxy*. Kode disimpan dalam perangkat yang sangat sederhana dan hanya berfungsi untuk merespon pesan. Protokol ini juga membantu menjaga saluran RF dari penggunaan yang berlebihan. Jika komunikasi mengkodekan perangkat itu, maka perangkat banyak menimbulkan tabrakan pada saluran transmisi karena perangkat-perangkat mengirimkan informasi pada saat yang bersamaan dan *bandwidth* saluran RF rendah. Akibat terjadi tabrakan pada saluran transmisi maka dapat menimbulkan penurunan *performance* pada saluran tersebut. Akibatnya terjadi penundaan pengiriman informasi, dimana hal tersebut tidak dikehendaki dalam komunikasi.

Namun, karena *proxy-proxy* dapat menginisialisasi semua komunikasi maka sebuah *gateway* dapat bertindak sebagai pemisah pada saluran RF. Ketika sebuah *gateway* itu menerima sebuah pesan dari *proxy*, maka *gateway* tersebut menyiarkannya pada saluran RF. Karena perangkat selalu merespon pesan, *gateway* akan menunggu selama 50 ms untuk merespon pesan tersebut. Selama waktu tersebut *gateway* tidak akan menstransmisikan pesan apapun pada saluran RF, hal ini untuk menghindari terjadinya tabrakan pada saluran transmisi antara *gateway* dengan perangkat. *Gateway* hanya dapat bertindak sebagai pemisah di wilayah siarannya karena RF yang dimiliki sangat lemah.

Untuk mendapatkan komunikasi yang *reliable*, *proxy* akan mentransmisi paket yang sama secara berulang-ulang sampai *proxy* tersebut menerima sebuah respon. Setiap paket mempunyai urutan nomor dan perangkat akan merespon sesuai dengan urutan nomor tersebut untuk menyatakan bahwa perangkat itu telah menerima sebuah paket. Pada saat perangkat menerima sebuah paket, maka segera mulai mencari kembali paket-paket yang lain sesuai dengan urutan nomor berikutnya. Jika perangkat menerima paket dengan urutan nomor tidak sesuai dari nomor yang diharapkan, maka perangkat tersebut akan kembali mengirim respon sebelumnya. Dalam hal ini, perangkat hanya akan memproses tiap-tiap paket sebanyak satu kali.

Masalah yang timbul pada peralatan bergerak adalah komunikasi harus selalu terjaga. Jika sebuah perangkat bergerak keluar dari jangkauan komunikasi dari sebuah *gateway* kemudian perangkat tersebut akan masuk ke wilayah yang lain. *Proxy* pada perangkat tersebut tidak mengetahui tentang *gateway* yang baru sehingga tidak mampu menghubungi perangkat tersebut, maka harus ada sebuah mekanisme bagi perangkat tersebut untuk mengatakan pada *proxy*-nya tentang keberadaan *gateway* yang baru. Dengan demikian jika perangkat tersebut tidak menerima sebuah paket baru dari *proxy*-nya selama 10 detik, maka perangkat itu akan mulai mentransmisikan kembali paket terakhir satu kali setiap empat detik. Paket tersebut akan disampaikan ke *proxy* melalui *gateway* yang baru, dan dari *header* paket itu *proxy* dapat menentukan alamat *gateway* yang baru tersebut.

3.2.3 Keamanan

Proxy dan *device* berkomunikasi melalui sebuah saluran yang aman yang mengenkripsi dan meng-autentikasi semua pesan yang ada. Algoritma HMAC-MD5 digunakan proses autentikasi dan algoritma RC5 digunakan untuk proses enkripsi [6]. Kedua algoritma tersebut menggunakan kunci simetrik, *proxy* dan perangkat berbagi kunci 128-bit.

3.2.3.1 Autentikasi

HMAC (*Hash Message Authentication Code*) menghasilkan sebuah MAC yang dapat memvalidasi keaslian dan integritas suatu pesan. HMAC menggunakan kunci rahasia, dan karenanya hanya orang yang tahu kunci tersebut yang dapat menghasilkan suatu MAC khusus atau menguji bahwa sebuah MAC itu benar. Inti HMAC adalah menghitung

$MAC=H(K,D)$, akan tetapi penghitungan yang sebenarnya sedikit lebih kompleks. Karena HMAC menggunakan kunci rahasia, maka hanya seseorang yang tahu kunci tersebut yang dapat menghasilkan MAC atau menguji bahwa MAC itu benar.

HMAC dengan fungsi *hash* MD5 menghasilkan 16 byte MAC. Delapan byte MAC yang paling signifikan disertakan pada tiap-tiap paket terakhir. Hal ini menyebabkan jumlah data yang harus ditransmisikan pada tiap-tiap paket menjadi sangat terbatas. Dari segi keamanan, hal ini memberikan sedikit informasi pada seorang penyerang yang baginya adalah merupakan sebuah keuntungan, tetapi kerugiannya adalah membiarkan penyerang harus menebak jumlah bit yang lebih sedikit. Hal ini merupakan sebuah *tradeoff* jika ke 16 byte MAC dimasukkan dalam setiap paket perangkat, bahkan kemudian lebih dari tiap paket akan diautentikasi sebagai pengganti data yang bermanfaat. Berikut dijelaskan proses autentikasi dengan server. Untuk memudahkan penjelasan, maka dibuat notasi kriptografi pada Tabel 3.1 di bawah ini :

Tabel 3.1 Notasi Kriptografi [4]

NOTASI	KETERANGAN
K_A	<i>A secret key</i>
K_B	<i>B secret key</i>
K_{AB}	<i>Secret key shared between A dan B</i>
K_{Apriv}	<i>A's private key (known only to A)</i>
K_{Apub}	<i>A's public key (published by A for all to read)</i>
$\{M\}_K$	<i>Message M encrypted with key K</i>
$[M]_K$	<i>Message M signed with key K</i>

- **Proses *authentication* dengan server**

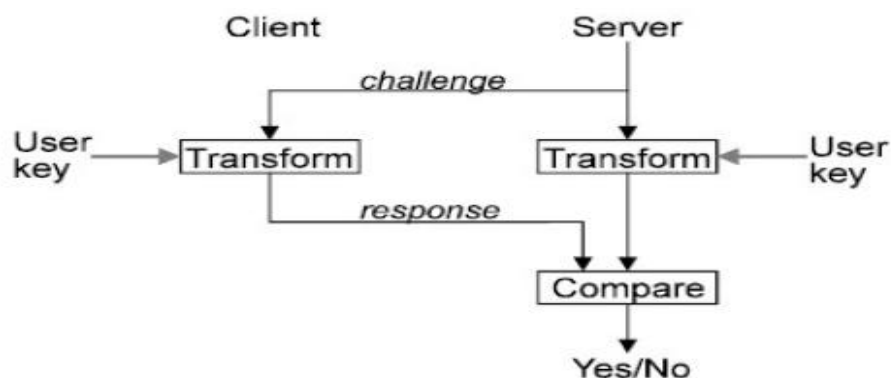
1. A ingin mengakses file yang disimpan oleh B pada file server. S adalah *authentication* server yang memberikan A dan B *password* dan menyimpan *secret key* untuk semua pemakai dalam sistem.
2. S mengetahui kunci A (K_A) dan B (K_B). Kunci ini dikenal dengan nama tiket, yaitu item yang terenkrip yang dikeluarkan oleh *authentication* server yang berisi identitas pemakai dan *shared key* yang dihasilkan untuk *session* komunikasi.
3. A mengirim sebuah pesan (tidak terenkrip) ke S untuk meminta tiket mengakses B ($A \rightarrow S : A, B, N_A$).
4. S mengirim pesan ke A yang terenkrip dengan K_A yang berisi tiket yang dienkrip dengan K_B dan kunci baru K_{AB} untuk digunakan ketika berkomunikasi dengan B.

$\{\{\text{Tiket}\}_{K_B}, K_{AB}\}_{K_A}$ (S ? A : $\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_B}\}_{K_A}$).

5. A mendekrip response menggunakan K_A . Jika A memiliki kunci yang benar, dia mendapat tiket yang valid untuk mengakses B dan kunci enkripsi untuk berkomunikasi dengan B.
6. A mengirim tiket ke B bersamaan dengan identitas A dan meminta B untuk mengakses file $\{K_{AB}, A\}_{K_B}$.
7. B dekrip tiket menggunakan kuncinya K_B . Kemudian B mendapat identifikasi A yang sah dan *shared key* baru K_{AB} untuk berinteraksi dengan A. B akan mengenkrip pesan saat itu (M_B) dengan kunci K_{AB} . $\{M_B\}_{K_{AB}}$.
8. A mengirim balasan ke B dengan menggunakan kunci K_{AB} . $\{M_{B-1}\}_{K_{AB}}$.

- **Proses Authentication dengan public key**

1. Dalam hal ini B diasumsikan menghasilkan *public key* dan *private key*. A mengakses layanan distribusi kunci untuk mendapatkan *public key certificate* yang mengandung *public key* B. Disebut *certificate* karena adanya tanda tangan oleh yang berwenang. Setelah dicek tanda tangannya, A akan membaca *public key* B (K_{bpub}) dari sertifikat.
2. A membuat *shared key* baru (K_{AB}) dan mengenkripsinya menggunakan K_{bpub} dengan algoritma *public key*. A mengirim hasilnya ke B, dengan suatu nama unik yang menunjukkan pasangan *public/private key*. Sehingga A mengirim *keyname*, $\{K_{AB}\}_{K_{bpub}}$ ke B.
3. B menggunakan *private key* K_{bpriv} untuk mendekrip K_{AB} .
4. Ilustrasi di atas menggambarkan kriptografi *public key* untuk mendistribusikan *shared secret key*. Teknik ini disebut dengan istilah *Hybrid Cryptographic Protocol*.



Gambar 3.5 Authentication [4]

3.2.3.2 Enkripsi

Data dienkripsi dengan menggunakan algoritma enkripsi RC5. RC5 dipilih karena kesederhanaan dan *performance*-nya. RC5 tidak memerlukan tabel untuk mempercepat proses, karena merupakan gerbang utama dalam menjalankan operasinya. Implementasi RC5 berdasar pada kode Open SLL [8]. RC5 merupakan sebuah *cipher blok*, yang biasa bekerja dalam 8 byte blok data. Namun dengan mengimplementasi hanya menggunakan model *output feedback* (OFB), RC5 dapat digunakan sebagai aliran *chipper* (*stream chipper*). Hal ini memungkinkan proses enkripsi dengan jumlah byte yang berubah-ubah tanpa mengkhawatirkan besarnya blok-blok data.

Model OFB bekerja dengan menghasilkan sebuah blok enkripsi dari sebuah vektor inisial dan kunci. Blok enkripsi ini kemudian di XOR-kan dengan data untuk menghasilkan *chipertext*. Karena $X \oplus Y \oplus Y = X$, *chipertext* dapat didekripsi dengan menghasilkan blok enkripsi yang sama dan meng-XOR-kannya dengan *chipertext*. Karena hanya membutuhkan enkripsi RC5 untuk menghasilkan blok enkripsi, maka pemisahan enkripsi dan dekripsi tidak dipersyaratkan. Untuk implementasi, kita menggunakan 16 *round* untuk RC5 dan menggunakan kunci 128-bit yang berbeda pada proses enkripsi dan autentikasi. Untuk memudahkan penjelasan, maka dibuat notasi kriptografi pada Tabel 3.2 di bawah :

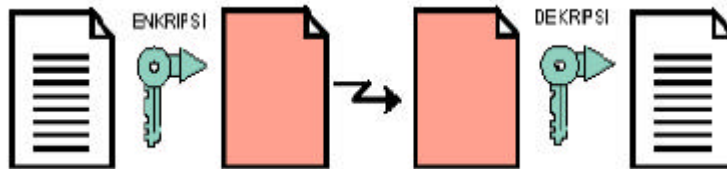
Tabel 3.2 Pihak yang terlibat dalam skenario

KODE DAN NAMA	KETERANGAN
A : Ari	Pihak Pertama
B : Bona	Pihak kedua
C : Cici	Pihak ketiga

- **Enkripsi Simetris**

Teknik kriptografi yang menggunakan kunci simetris adalah yang paling umum dipergunakan. Kunci untuk melakukan proses enkripsi sama dengan kunci untuk melakukan proses dekripsi. Jadi misalkan Ari ingin mengenkrip suatu pesan dan mengirimkannya ke Bona, maka baik Ari maupun Bona harus mempunyai kunci yang sama persis. Siapapun yang memiliki kunci tersebut termasuk pihak-pihak yang tidak diinginkan bisa mendapatkan pesan aslinya dari *ciphertext*. Problem yang paling signifikan disini adalah bukan pada masalah pengiriman *ciphertext*-nya, melainkan masalah

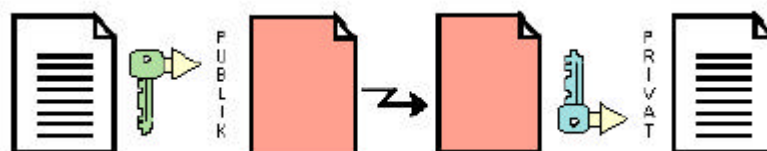
bagaimana menyampaikan kunci simetris tersebut kepada pihak yang diinginkan. Teknik kriptografi menggunakan kunci simetri seringkali disebut juga sebagai *secret key cryptography*.



Gambar 3.6 Enkripsi kunci simetris [4]

- **Enkripsi Asimetris**

Jika teknik kriptografi menggunakan kunci simetris, memakai kunci yang sama untuk melakukan proses enkripsi dan dekripsi, maka teknik kriptografi menggunakan kunci asimetris memerlukan sepasang kunci untuk enkripsi dan dekripsi. Kunci publik dapat diketahui oleh semua orang sedangkan kunci privat hanya boleh diketahui oleh satu orang saja, yaitu orang yang berhak memilikinya. Dengan demikian maka Cici dapat yakin bahwa hanya Ari yang dapat membuka pesan yang dikirim oleh Cici kepada Ari, jika Cici mengenkrip pesan tersebut menggunakan kunci publik milik Ari. Demikian juga Bona dapat yakin bahwa pengirim pesan adalah benar-benar si Cici, jika Cici mengenkrip pesan tersebut menggunakan kunci privat miliknya sendiri, karena hanya kunci publik milik Cici yang dapat membuka pesan tersebut, dengan asumsi bahwa kunci publik yang dipakai adalah benar-benar milik Cici. Masalah yang timbul adalah bagaimana caranya agar Ari bisa mendapatkan kunci publik Cici. Masalah tersebut timbul karena Ari harus benar-benar yakin bahwa kunci publik yang didapat adalah benar-benar milik Cici.



Gambar 3.7 Enkripsi Asimetris [4]

3.2.4 Lokasi

Lokasi perangkat ditentukan dengan menggunakan sistem lokasi *cricket* [7]. *Cricket* memiliki beberapa fitur yang bermanfaat, yaitu privasi pemakai, kontrol disentralisasi, biaya murah, dan pemanfaatan yang mudah. *Cricket* juga dapat bekerja di dalam sebuah ruangan. Setiap perangkat menentukan lokasinya sendiri. Ini tergantung pada perangkat untuk menentukan apakah perangkat lain dapat mengetahuinya atau tidak.

Pada sistem *cricket*, suar (*beacon*) ditempatkan pada langit-langit ruangan. Secara periodik *beacon-beacon* ini menyebarkan informasi tentang lokasinya (seperti : **Room 4011**) yang dapat didengar oleh pendengar *cricket*. Pada saat yang sama ketika informasi ini disebarkan pada sepektrum RF, *beacon* juga menyebarkan sebuah bunyi *ultrasound*. Ketika seorang pendengar menerima pesan RF, maka ia mengukur waktu sampai menerima bunyi *ultrasound*. Dengan menggunakan perbedaan waktu, pendengar dapat menentukan jarak ke *beacon*. *Beacon* menentukan lokasinya dengan menggunakan informasi yang datang.

3.3 Protokol *proxy-to-proxy*

SPKI/SDSI [5] merupakan sebuah infrastruktur keamanan yang dirancang untuk memfasilitasi pengembangan sistem yang *scalable*, aman dan sistem terbagi. SPKI/SDSI menyediakan kontrol akses jaringan yang halus dengan menggunakan suatu arsitektur ruangan dengan nama lokal dan model yang sederhana, fleksibel, serta model kebijakan yang dapat dipercaya. SPKI/SDSI merupakan infrastruktur kunci publik dengan desain *egaliter*. Pelaku-pelaku itu adalah kunci publik dan setiap kunci publik adalah sebuah otoritas sertifikat. Setiap pelaku bisa mengeluarkan sertifikat pada basis yang sama seperti pelaku lainnya. Tidak ada infrastruktur global secara hirarkis. Komunitas SPKI/SDSI dibangun dari bawah ke atas (*bottom-up*), secara terdistribusi dan tidak membutuhkan *root* yang harus dipercaya.

3.3.1 Integrasi SPKI/SDSI

Sistem ini menggunakan suatu arsitektur *client-server* untuk *proxy-proxy*. Ketika seorang pelaku tertentu, bertindak atas nama perangkat atau pemakai, dan membuat suatu permintaan melalui sebuah *proxy* kepada sebuah perangkat yang diwakili oleh *proxy* lainnya, maka *proxy* pertama bertindak sebagai *client*, dan *proxy* kedua sebagai *server*. Sumber-sumber pada *server* tersebut dilindungi baik oleh publik atau oleh SPKI/SDSI

ACL. SPKI/SDSI ACL, terdiri dari daftar masukan, setiap daftar masukan mempunyai subjek (kunci atau kelompok) dan sebuah label yang menentukan seperangkat operasi kunci atau group yang diijinkan untuk melakukan operasi. Untuk memperoleh akses menuju sumber yang dilindungi oleh ACL, maka pemohon harus mencantumkan pada permohonannya sebuah rangkaian sertifikat yang menunjukkan bahwa pemohon adalah anggota dari sebuah kelompok yang masuk dalam daftar ACL.

Jika suatu sumber yang diminta dilindungi oleh ACL, permohonan pelaku harus disertai dengan “*proof of authenticity*” (bukti keaslian) yang menunjukkan bahwa sumber itu adalah asli, dan “*proof of authorization*”(bukti otorisasi) yang menunjukkan bahwa pelaku itu berwenang untuk mengajukan permohonan tertentu pada sumber tertentu. Bukti autentikasi biasanya berupa *request* permohonan yang ditandatangani dan bukti otorisasi biasanya adalah serangkaian dari sertifikat. Pelaku yang menandatangani *request* harus sama dengan pelaku yang mengotorisasi sertifikat-sertifikat itu. Desain sistem dan protokol antara *proxy-proxy* sangat mirip dengan yang digunakan pada *Project Geronimo* SPKI/SDSI dimana SPKI/SDSI tergabung dalam Apache dan Netscape serta digunakan untuk menyediakan *client* akses kontrol terhadap *web*.

3.3.2 Protokol

Protokol diimplementasikan oleh *client* dan *proxy server* terdiri dari empat pesan. Protokol ini ditunjukkan pada Gambar 3.8 dan berikut adalah penjelasannya :

- a. *Proxy client* mengirim suatu permohonan, tidak diautentikasi dan tidak diotorisasi ke *proxy sever*.
- b. Jika *client* meminta akses ke sebuah sumber daya yang dilindungi, maka *server* akan merespon dengan ACL yang melindungi sumber tersebut dan label dibentuk dengan permohonan *client*. *Tag* merupakan suatu struktur data SPKI/SDSI yang mewakili suatu permohonan. Banyak contoh pada draft SPKI/SPSI *Internet Engineering Task Force (IETF)* [5]. Jika tidak ada ACL yang melindungi sumber yang diminta, maka permohonan tersebut segera di terima.
- c. *Proxy client* membuat serangkaian sertifikat dengan menggunakan algoritma penemuan serangkaian sertifikat SPKI/SDSI [5]. Serangkaian sertifikat ini memberikan sebuah bukti otorisasi yang membuktikan bahwa kunci pemakai itu sah untuk mengajukan permohonannya. Algoritma penemuan serangkaian sertifikat tersebut mengambil *input* ACL dan *tag* dari *server*, kunci publik pemakai, seperangkat sertifikat pemakai dan

timestamp. Jika hal itu terpenuhi, maka algoritma tersebut akan mengembalikan serangkaian sertifikat ke pemakai yang memberikan bukti bahwa kunci publik pemakai adalah sah untuk menampilkan operasi yang ditentukan dalam *tag* itu, pada saat yang ditentukan dalam *timestamp*. Jika algoritma tidak dapat membuat serangkaian sertifikat karena pemakai tidak mempunyai sertifikat yang dibutuhkan, atau jika kunci pemakai secara langsung ada pada ACL maka algoritma itu akan mengembalikan serangkaian sertifikat kosong. Klien membuat *timestamp* dengan menggunakan waktu lokalnya.

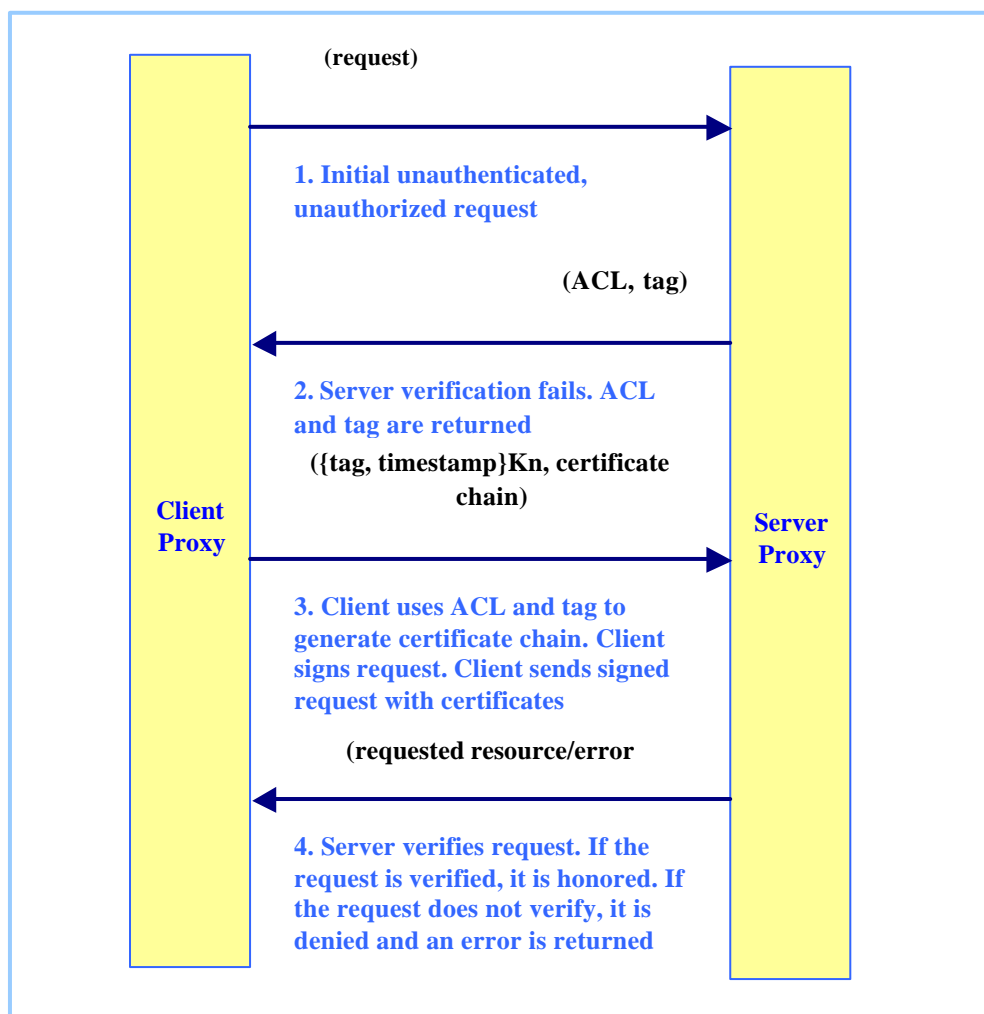
- d. Klien membuat urutan SPKI/SDSI [5] yang berisi *tag* dan *timestamp*. Ini menandakan urutan dengan kunci pribadi pemakai, dan memasukan salinan kunci publik pemakai pada tanda tangan SPKI/SDSI. Kemudian mengirimkan urutan *tag timestamp*, tanda tangan, dan serangkaian sertifikat yang dibuat ke *server*.
- e. *Server* memeriksa permohonan itu dengan :
 - memeriksa *timestamp* pada urutan *tag-timestamp* terhadap waktu lokal *server* untuk meyakinkan bahwa permintaan itu dibuat pada saat itu (masih baru),
 - membuat kembali *tag* dari permohonan klien dan mengecek apakah sama dengan *tag* yang ada pada urutan *tag-timestamp*,
 - mengambil kunci publik dari tanda tangan,
 - menguji tanda tangan pada urutan *tag-timestamp* dengan menggunakan kunci,
 - memvalidasi sertifikat pada serangkaian sertifikat,
 - memeriksa atau menguji bahwa ada serangkaian otorisasi dari sebuah urutan pada ACL ke kunci dari tanda tangan, melalui serangkaian sertifikat yang ditampilkan. Serangkaian otorisasi harus mengizinkan klien untuk menampilkan operasi yang diinginkan.

Jika *request* berhasil diverifikasi maka permohonan akan dipenuhi. Jika tidak berhasil memenuhi verifikasi, maka permohonan ditolak dan *proxy server* mengembalikan kesalahan pada *proxy client*. Kesalahan ini dikembalikan ketika *client* menyajikan sebuah permohonan sah yang ditolak.

Protokol dapat dipandang sebagai sebuah protokol *challenge-response* yang khas. Jawaban *server* pada langkah b dari protokol tersebut merupakan suatu *challenge* bagi *server* yang memberikan tanggapan kepada *client* dengan mengatakan “**Anda sedang mencoba mengakses sebuah file yang terlindungi**”. **Buktikan pada saya bahwa anda**

mempunyai surat mandat untuk menampilkan operasi yang anda minta, dari sumber yang dilindungi oleh ACL ini. Klien menggunakan ACL untuk membantu menghasilkan sebuah rangkaian sertifikat dengan menggunakan algoritma penemuan serangkaian sertifikat. Ini kemudian mengirimkan serangkaian sertifikat dan permohonan yang ditandatangani pada permohonan kedua ke *proxy server*. Permohonan yang tertanda memberikan bukti keaslian, dan serangkaian sertifikat membuktikan otorisasi. *Server* mencoba menguji permohonan kedua, dan jika berhasil, maka permohonan akan diterima.

Timestamp pada urutan *tag-timestamp* membantu melindungi terhadap jenis-jenis suatu serangan balik tertentu. Sebagai contoh, anggap saja *server* mencatat permohonan dan anggap bahwa *logs* itu tidak diberikan dengan tepat. Jika seorang musuh mencapai akses ke *logs* tersebut, maka *timestamp* mencegahnya dari permohonan ulang yang ditemukan pada *logs* dan memperoleh akses menuju *resource* yang dilindungi.



Gambar 3.8 SPKI/SDSI Proxy to proxy ACL [6]

3.3.3 Pertimbangan Keamanan Tambahan (*Additional Security Considerations*)

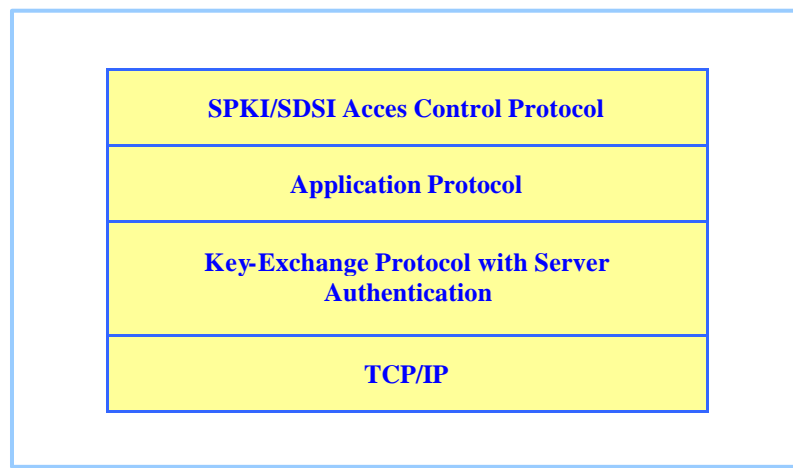
Protokol SPKI/SDSI sebagaimana yang telah diuraikan di awal, membahas masalah tentang penyediaan kontrol akses di klien. Protokol benar-benar tidak menjamin kerahasiaan, pembuktian keaslian pada *server*, dan pemberian perlindungan terhadap serangan balik dari jaringan. Pada saat ini protokol keamanan yang paling banyak digunakan adalah SSL. Protokol TLS merupakan pengganti protokol SSL. Tujuan utama dari SSL/TLS adalah menjaga kerahasiaan dan integritas data pada lalu-lintas antara *client* dan *server*, serta memberikan pembuktian keaslian pada *server*. Ada dukungan bagi pembuktian keaslian *client*, tetapi pembuktian itu merupakan suatu tambahan.

Protokol akses kontrol SPKI/SDSI dapat ditempatkan pada sebuah protokol *keyexchange* (pertukaran kunci) seperti yang terdapat pada TLS/SSL untuk menyediakan keamanan tambahan. TLS/SSL pada saat ini menggunakan x.509 PKI untuk membuktikan keaslian *server*, tetapi dengan cara yang sama dapat juga memakai SPKI/SDSI. SSL/TLS juga memberikan perlindungan terhadap serangan balik dari jaringan, dan perlindungan terhadap serangan *person-in-the-middle*. Lapisan protokol ditunjukkan pada Gambar 3.9, dimana pada Gambar tersebut, “**Application Protocol**” menunjuk pada standar protokol komunikasi antara klien dan *proxy-proxy server*, tanpa keamanan.

SSL/TLS membuktikan keaslian *proxy server*. Namun tidak menyebutkan apakah *proxy server* sah untuk menerima permohonan *client*. Sebagai contoh, mungkin masalah *proxy client* menginginkan untuk mencetak sebuah dokumen yang paling rahasia, dan hanya printer-printer tertentu yang mestinya digunakan untuk mencetak dokumen rahasia tersebut. Dengan SSL/TLS dan SDKI/SDSI *Client Access Control Protocol* yang telah kita deskripsikan, *proxy* klien akan tahu bahwa kunci *public proxy* itu dengan *proxy* yang sedang dibatasi sebuah alamat tertentu, dan *proxy server* akan tahu bahwa *proxy* klien berhak untuk mencetaknya. Namun, *proxy* klien masih belum tahu jika *proxy server* berhak untuk mencetak dokumen-dokumen rahasia. Jika *proxy* klien mengirimkan dokumen rahasia tersebut itu untuk dicetak, *proxy server* akan menerima dokumen tersebut dan akan mencetaknya, namun dokumen itu tidak seharusnya dikirim ke *proxy server* ditempat pertama.

Untuk menyelesaikan masalah ini, maka diusulkan perluasan protokol SPKI/SDSI sehingga klien meminta otorisasi dari *server* dan *server* tersebut membuktikan kepada klien bahwa dia berhak untuk menangani permohonan klien sebelum klien mengirim dokumen untuk dicetak. Untuk mengembangkan protokol tersebut, protokol SPKI/SDSI

berjalan dari *proxy client* ke *proxy server*, dan kemudian masuk kearah sebaliknya dari *proxy server* ke *proxy client*. Maka *proxy client* akan memberikan serangkaian sertifikat SPKI/SDSI yang membuktikan bahwa *proxy client* tersebut berhak untuk menerima dan memenuhi permohonan klien. Jika dibutuhkan keamanan tambahan, maka protokol yang diperluas dapat ditambahkan pada SSL/TLS. Catatan bahwa SPKI/SDSI *Access Control Protocol* merupakan sebuah contoh *‘the end to end-argument’*. Keputusan akses kontrol dibuat pada lapisan paling atas, yang hanya terdiri dari *client* dan *server*.



Gambar 3.9 *Example Layering of Protocol* [6]

BAB IV ANALISIS DAN EVALUASI

4.1 Protokol perangkat ke *proxy* (*device-to-proxy*)

4.1.1 Persyaratan memori

Tabel 4.1 menunjukkan kebutuhan memori bagi bermacam-macam komponen *software*. Ukuran kode mewakili memori yang digunakan pada *flash*, dan ukuran data mewakili memori yang digunakan dalam RAM. Komponen fungsionalitas perangkat meliputi rutin pemrosesan paket dan lokasi. Komponen kode RF meliputi rutin RF *transmit* dan RF *receive* sebagai rutin *cricket listener*. Komponen-komponen lain merupakan kode yang umum digunakan untuk semua komponen lainnya.

Kode perangkat membutuhkan kira-kira 12 KB untuk tempat kode dan 1 KB tempat data. Algoritma keamanan HMAC-MD5 dan RC5 mengambil sebagian besar tempat kode. Kedua algoritma tersebut telah dioptimalkan di dalam perakitan, sehingga dapat mengurangi lebih dari 50% ukuran kode yang dibutuhkan. Kode dapat lebih baik dioptimalkan, tetapi hal ini memberikan pemikiran umum seberapa banyak memori yang dibutuhkan. Ukuran kode adalah cukup kecil sehingga dapat disatukan ke dalam semua perangkat apapun. Untuk mengoptimalkan lebih jauh ukuran kode, maka dapat digunakan algoritma autentikasi yang lebih mudah. Sistem ini menggunakan HMAC-MD5 yang memerlukan memori 4,6 KB. Kemungkinan lain adalah menggunakan sebuah algoritma autentikasi dengan enkripsi sebagai pengganti fungsi *hash*. Dimana hal tersebut dapat mengurangi ukuran kode secara signifikan, karena kode RC5 sudah berada dalam enkripsi dan autentikasi. Pengurangan ukuran kode menjadi lebih sedikit dari 8 KB.

Tabel 4.1 *Code and data size on the Atmel processor* [6]

COMPONENT	CODE SIZE (KB)	DATA SIZE (BYTES)
Device Functionality	2.0	191
RF Code	1.1	153
HMAC-MD5	4.6	386
RC 5	3.2	256
Miscellaneous	1.0	0
TOTAL	11.9	986

4.1.2 Persyaratan Pemrosesan

Kebutuhan terbanyak pada perangkat-perangkat yang digunakan adalah algoritma untuk keamanan. Tabel 4.2 menjelaskan waktu rata-rata untuk masing-masing algoritma yang digunakan. Waktu pemrosesan RC5 sangat bervariasi secara linier sesuai dengan jumlah byte yang di enkripsi atau deskripsi. Disisi lain rutin HMAC-MD5, membutuhkan waktu yang konstan hingga 56 byte. Hal ini karena HMAC-MD5 dirancang untuk dapat bekerja pada blok data, sehingga apapun di bawah 56 byte dibentuk blok. Karena ukuran paket RF dibatasi pada 50 byte, maka hanya dianalisis waktu proses HMAC-MD5 untuk paket dengan ukuran kurang dari atau sama dengan 50 byte.

Tabel 4.2 *Performance of encryption and authentication code* [6]

FUNCTION	TIMES (ms)	CLOCK CYCLES
RC5 encrypt/decrypt (n bytes)	$0.163 + 0.552n$	$652n + 2208$
HMAC-MD5 Up to 56 bytes	11.48	45.920

4.2 SPKI/SDSI

Protokol yang telah diuraikan di awal adalah sangat efisien, karena protokol tersebut merupakan suatu permintaan pasangan yang standar, dan tidak ada kriptografi yang dibutuhkan. Langkah-langkah yang signifikan pada protokol tersebut adalah langkah dimana suatu rangkaian sertifikat dibentuk, dan langkah dimana rangkaian itu diuji. Tabel 4.3 menunjukkan analisis dari kedua langkah tersebut di atas. Tulisan mengenai *Certificate Chain Discovery* pada SPKI/SDSI [5] berkaitan dengan analisis penentuan waktu yang digunakan. Waktu CPU merupakan waktu rata-rata yang telah diukur pada Sun Microsystem Ultra-1 yang menjalankan Sun OS 5.7.

Tabel 4.3 *Analisis protokol proxy ke proxy* [6]

PROTOCOL STEP	TIMING ANALYSIS	APPROX CPU TIME
Cert chain discovery	The worst cases is $O(n^3l)$, where n =number of certs, and l =length of longest subject. However, the expected time is $O(nl)$.	330ms, with $n=2$ and $l= 2$.
Chain validation	The worst case is $O(n)$, where n =number of certs.	200ms, with $n=2$.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

1. Mengembangkan protokol keamanan yang dapat mengatasi keragaman, perangkat-perangkat bergerak pada jaringan merupakan tantangan besar yang harus dihadapi.
2. Keamanan informasi merupakan hal yang sangat penting dan utama untuk menjaga kerahasiaan, apalagi informasi tersebut hanya boleh diketahui oleh pihak-pihak tertentu saja dan informasi tersebut berada dalam jaringan perangkat yang bergerak.
3. Tantangan besar yang lain adalah bagaimana merancang perangkat yang cukup cerdas, cepat untuk saling bekerja sama, meningkatkan *performance*, dan memungkinkan konektifitas yang tinggi, serta keamanan dari sistem terjamin. Perangkat hanya mengijinkan akses bagi pemakai yang sah dan juga harus menjaga komunikasi yang aman ketika menstransmisi atau menerima informasi pribadi.
4. Dalam tulisan ini, diambil langkah awal untuk memenuhi tantangan tersebut dengan mengamati kebutuhan suatu protokol keamanan. Analisis sebuah *prototype* sistem menggunakan dua protokol yang berbeda. Jika terdapat tipe protokol lain yang dapat menyempurnakan sistem keamanan maka protokol tersebut dapat dimasukkan.
5. Dua buah protokol yang telah diuraikan mempunyai karakteristik yang sangat berbeda, karena mereka menerapkan skenario yang berbeda pula. Protokol perangkat ke *proxy* dirancang untuk memungkinkan komunikasi data yang aman dari sebuah perangkat yang kecil. Protokol SPKI/SDSI berbasis *proxy* ke *proxy* dirancang untuk memberikan akses kontrol yang fleksibel antar *proxy*. Arsitektur *proxy* dan manfaat dari dua protokol yang berbeda telah menghasilkan penemuan sistem komunikasi yang aman dan efisien.

5.2 Saran

Pembahasan tentang protokol keamanan berbasis *proxy* pada peralatan bergerak yang telah diuraikan di muka, masih jauh dari kesempurnaan. Bagi para pembaca yang berminat dapat melakukan analisa lebih detail.

DAFTAR PUSTKA

- [1] Andi Offset, Wahana Komputer, “*Memahami Model Enkripsi dan Security Data*”, Andi Offset, Yogyakarta, 2003.
- [2] Budi Rahardjo, “*Keamanan Sistem Informasi Berbasis Internet*”, PT Insan Indonesia-Bandung & PT INDOCISC-Jakarta, 1998, 1999, 2000, 2001, 2002.
- [3] Budi Rahardjo, “*Memahami Teknologi Informasi*”, PT Elex Media Komputindo, Jakarta, 2002.
- [4] Budi Susanto, “*Keamanan Sistem Terdistribusi*”, <http://www.ukdw.ac.id>
Dikunjungi 20 September 2004.
- [5] D. Clarke. “*SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI*”, Master’s thesis, Massachusetts Institute of Technology, 2001.
- [6] M. Burnside, D. Clarke, T. Mills, A. Maywah, S. Devadas, R. Rivest. “*Proxy-Based Security Protocols in Networked Mobile Devices*”, MIT Laboratory for Computer Science 200 Technology Square, Cambridge, MA 021139 USA.
<http://citeseer.ist.psu.edu/burnside02proxybased.html>
Dikunjungi 15 September 2004.
- [7] OpenSSL. The OpenSSL Project. <http://www.openssl.org>
Dikunjungi 15 September 2004.
- [8] Sun Microsystems Inc. Jini Network Technology. <http://www.sun.com/jini>.
Dikunjungi 15 September 2004.
- [9] Todd Mills. Matthew Burnside, John Ankorn, Srinivas Devadas. “*A Proxy-Based Architecture for Secure Networked Wearable Devices*”, MIT Laboratory for Computer Science 200 Technology Square, Cambridge, MA 021139 USA.

