

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

PROTOKOL KEAMANAN BERBASIS PROXY DALAM JARINGAN PERALATAN BERGERAK

Nama : Sumaryanto
NIM : 23203132



PROGRAM MAGISTER TEKNIK ELEKTRO
BIDANG KHUSUS TEKNOLOGI INFORMASI DIKMENJUR
INSTITUT TEKNOLOGI BANDUNG
2004

Daftar Isi

| | Halaman |
|--|-----------|
| Judul | i |
| Daftar Isi | ii |
| Daftar Gambar | iii |
| Daftar Tabel | iv |
| | |
| Abstrak | 1 |
| | |
| 1. Pendahuluan | 1 |
| | |
| 2. Dasar teori | 3 |
| 2.1 Konsep keamanan | 3 |
| 2.2 Keamanan sistem terdistribusi | 4 |
| 2.2.1 Ancaman dan serangan..... | 4 |
| 2.2.2 Metode penyerangan | 5 |
| 2.2.3 Teknik keamanan | 6 |
| | |
| 3. Protokol keamanan berbasis <i>proxy</i> dalam jaringan peralatan bergerak..... | 9 |
| 3.1 Arsitektur sistem | 9 |
| 3.1.1 Alat-alat | 9 |
| 3.1.2 <i>Proxy</i> | 10 |
| 3.1.3 <i>Server</i> dan jaringan <i>server</i> | 11 |
| 3.1.4 Komunikasi lewat peristiwa | 12 |
| 3.1.5 Penemuan sumber | 12 |
| 3.1.6 Model keamanan | 13 |
| 3.1.7 Inisialisasi alat | 14 |
| 3.1.8 Implementasi alat | 14 |
| 3.2 Protokol peralatan ke <i>proxy</i> | 15 |
| 3.2.1 Komunikasi | 15 |
| 3.2.2 RF protokol | 15 |
| 3.2.3 Keamanan | 17 |
| 3.2.4 Lokasi | 18 |
| 3.3 Protokol <i>proxy</i> ke <i>proxy</i> | 18 |
| 3.3.1 Integrasi SPKI/SDSI | 19 |
| 3.3.2 Protokol | 19 |
| 3.3.3 <i>Additional security consideration</i> | 22 |
| 3.4 Hubungan kerja | 23 |
| 3.4.1 Komunikasi alat ke <i>proxy</i> | 23 |
| 3.4.2 Komunikasi <i>proxy</i> ke <i>proxy</i> | 24 |
| | |
| 4. Evaluasi | 24 |
| 4.1 <i>Memory Requirements</i> | 24 |
| 4.2 <i>Processing Requirements</i> | 25 |
| 4.3 <i>Evaluasi SPKI/SDSI</i> | 25 |
| | |
| 5. Kesimpulan | 26 |
| | |
| 6. References | 27 |

Daftar Gambar

| | Halaman |
|---|---------|
| Gambar 1. Contoh teknologi keamanan sistem terdistribusi..... | 5 |
| Gambar 2. Fungsi f_8 untuk menjamin kerahasiaan data | 7 |
| Gambar 3. Algoritma KASUMI integrasi f_8 dan f_9 | 8 |
| Gambar 4. Arsitektur sistem..... | 9 |
| Gambar 5. Arsitektur sistem komunikasi alat ke <i>proxy</i> | 10 |
| Gambar 6. Model keamanan | 14 |
| Gambar 7. Arsitektur komunikasi | 15 |
| Gambar 8. <i>SPKI/SDSI Proxy to proxy ACL</i> | 22 |
| Gambar 9. <i>Example Layering of Protocol</i> | 23 |

Daftar Tabel

| | Halaman |
|---|---------|
| Tabel 1. <i>Code and data size on the Atmel processor</i> | 25 |
| Tabel 2. <i>Performance of encryption and authentication code</i> | 25 |
| Tabel 3. Analisis protokol <i>proxy</i> ke <i>proxy</i> | 26 |

PROTOKOL KEAMANAN BERBASIS PROXY DALAM JARINGAN PERALATAN BERGERAK

Sumaryanto
NIM 23203132

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

Abstrak :

Protokol keamanan merupakan hal yang sangat penting terutama dalam menjaga kerahasiaan informasi, apalagi jika pengiriman informasi tersebut melalui jaringan peralatan bergerak. Dalam tugas ini diuraikan sebuah penemuan sumber daya dan sistem komunikasi yang dirancang untuk keamanan dan privasi. Semua obyek di dalam sistem tersebut, yaitu peralatan yang dipakai, alat-alat software, dan para pemakai telah menggunakan proxy software yang dipercaya dapat bekerja pada peralatan hardware ataupun pada komputer yang digunakan. Sistem keamanan dan privasi dijalankan dengan menggunakan dua buah protokol terpisah. Sebuah protokol untuk komunikasi alat ke proxy yang aman, dan sebuah protokol untuk komunikasi proxy-proxy yang aman. Dengan menggunakan dua buah protokol terpisah dapat dijalankan sebuah protokol yang canggih untuk pembuktian keaslian sumber daya dan komunikasi dengan alat yang lebih kuat.

Protokol keamanan berbasis proxy menggunakan protokol alat ke proxy untuk alat-alat bergerak yang ringan pada jaringan nirkabel dan protokol proxy ke proxy yang berdasarkan pada SPKI/SDSI (Simple Public Key Infrastructure/Simple Distributed Systems Infrastructure). Penggunaan dua buah protokol yang berbeda dan penggunaan kerangka kerja SPKI/SDKI di dalam protokol proxy ke proxy membuat sebuah sistem otomatisasi yang aman, terukur, efisien, dan mudah untuk dipelihara. Proxy bekerja pada sebuah komputer yang cepat sehingga mampu mengimplementasikan algoritma kriptografi yang canggih.

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

1. Pendahuluan

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 terdapat dalam jaringan alat-alat yang bergerak. Untuk mencapai tujuan dari perhitungan yang ada di mana-mana dan dapat menembus sistem menjadi lebih mungkin sebagaimana jumlah alat-alat perhitungan di dunia yang meningkat dengan cepat. Namun masih ada rintangan-rintangan yang penting atau signifikan untuk diatasi ketika menggabungkan alat-alat yang dapat dipakai dan disimpan ke dalam lingkungan perhitungan yang ada di manapun. Rintangan-rintangan ini meliputi : merancang alat-alat yang cukup cerdas, cepat untuk saling bekerja sama, meningkatkan

performance, dan memungkinkan konektifitas tinggi, serta keamanan dari sistem menjadi faktor kunci. Alat-alat yang hanya mengijinkan akses bagi pemakai yang sah dan juga harus menjaga komunikasi yang aman ketika menstransmisi atau menerima informasi pribadi.

Mengimplementasikan bentuk-bentuk keamanan yang khas untuk komunikasi pribadi menggunakan sebuah infrastruktur *public key*. Sebuah *public key* kriptografi algoritma yang umum digunakan adalah RSA. Untuk membolehkan arsitektur menggunakan sebuah model keamanan *public key* pada jaringan nirkabel, alat-alat yang digunakan menjadi sederhana maka setiap alat diberi sebuah *proxy software*. Semua obyek di dalam sistem itu (alat-alat, peralatan yang dapat dipakai, perangkat *software*, pemakai) telah dihubungkan ke *proxy software* yang telah bekerja dengan baik pada processor yang disimpan pada komputer. *Proxy* yang bekerja pada sebuah processor yang tersimpan pada alat, akan menjamin bahwa alat untuk komunikasi *proxy* adalah aman. Jika alat itu mempunyai kekuatan perhitungan minimal dan komunikasi ke *proxy* melalui jaringan dengan kabel maupun tanpa kabel, komunikasi itu menempel ke sebuah protokol alat ke *proxy*. *Proxy-proxy* saling berkomunikasi dengan yang lainnya menggunakan sebuah protokol *proxy* ke *proxy* yang aman berdasarkan SPKI/SDKI. Menggunakan dua buah protokol yang berbeda memungkinkan kita untuk menjalankan sebuah protokol keamanan yang murah dan protokol yang canggih bagi pembuktian keaslian sumber serta komunikasi pada alat-alat yang lebih kuat.

Menggunakan pemikiran di atas, sebuah sistem otomatisasi bentuk dasar memungkinkan komunikasi menjadi aman, efisien, akses ke jaringan pada alat-alat bergerak menjadi mudah. Pada sistem ini setiap pemakai menggunakan sebuah kunci yang disebut K21. Kunci tersebut mengidentifikasi pemakai dan lokasi pemakainya. Identitas pemakai dan informasi lokasi ditransmisi dengan aman ke *proxy software* pemakai yang menggunakan protokol alat ke *proxy*. Alat-alat tersebut dapat berupa mobil yang dapat berpindah tempat. Atribut akan mencari semua alat-alat yang dapat dikontrol dan ditampilkan untuk menemukan alat-alat terdekat, atau alat yang paling tepat. Dengan mengeksploitasi SPKI/SDKI sistem keamanan tidak mau kompromi terhadap pemakai baru dan alat-alat yang memasuki sistem itu. Penggunaan dua protokol yang berbeda dan penggunaan kerangka kerja SPKI/SDKI di dalam protokol *proxy* ke *proxy* berakibat pada sebuah sistem otomatisasi yang aman, terukur, efisien, dan mudah untuk dipelihara.

2. Dasar teori

2.1 Konsep keamanan

Keamanan sering dipandang hanyalah merupakan masalah teknis yang melibatkan dapat atau tidaknya tertembusnya 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 :

- **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.
- **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.
- **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.
- **Vandal** : Jenis serangan ini bertujuan untuk merusak sistem, yang sering ditujukan untuk *site-site* besar.
- **Scorekeeper** : Jenis serangan ini hanyalah bertujuan untuk mendapatkan reputasi dengan cara mengacak sistem sebanyak mungkin.
- **Mata-mata** : Jenis serangan ini bertujuan untuk memperoleh data atau informasi rahasia dari pihak kompetitor.

Pendekatan yang digunakan berbagai pihak untuk menerapkan berbagai sistem keamanan adalah :

- **Tanpa keamanan** :
Banyak orang tidak melakukan apa-apa yang berkaitan dengan keamanan (menerapkan keamanan minimal).
- **Keamanan dengan cara penyembunyian** :
Sistem diasumsikan akan lebih aman bila tidak ada orang yang mengetahui mengenai keberadaan sistem itu.

- **Host security :**

Pada pendekatan ini, maka tiap *host* akan dibuat *secure*. Permasalahannya adalah kompleksitas. Pada saat ini suatu organisasi besar memiliki sistem yang *heterogen*, sehingga proses menjadikan tiap *host* menjadi *secure* sangatlah kompleks.

- **Network security :**

Pada pendekatan ini, usaha dikonsentrasikan dengan mengontrol akses ke jaringan pada sistem.

Yang perlu diingat adalah kenyataan bahwa tidak ada suatu modelpun yang dapat memenuhi semua dari kebutuhan sistem keamanan yang kita butuhkan. Sehingga kombinasi dari berbagai pendekatan perlu dilakukan. Perlindungan data adalah hal yang penting dalam masalah keamanan. Sistem keamanan data dapat dikategorikan menjadi :

- **data publik**, yaitu data yang dapat dikomunikasikan dengan siapa saja,
- **data rahasia**, yaitu data yang tidak boleh bocor ketangan yang tidak berhak,
- **sembarang data.**

Beberapa pertimbangan dalam perancangan dan pembahasan sistem keamanan adalah :

- **Confidentiality**, yang akan berkaitan dengan pencegahan akan pengaksesan terhadap informasi yang dilakukan oleh pihak yang tidak berhak.
- **Integrity**, yang akan berkaitan dengan pencegahan akan memodifikasi informasi yang dilakukan oleh pihak yang tidak berhak.
- **Availability**, pencegahan akan penguasaan informasi atau sumber daya oleh pihak yang tidak berhak.

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 ;

- **Leakgace** (kebocoran) : pengambilan informasi oleh penerima yang tidak berhak,
- **Tampering** : pengubahan informasi yang tidak legal,
- **Vandalism** (perusakan) : gangguan operasi sistem tertentu, dimana si pelaku tidak mengharap keuntungan apapun.

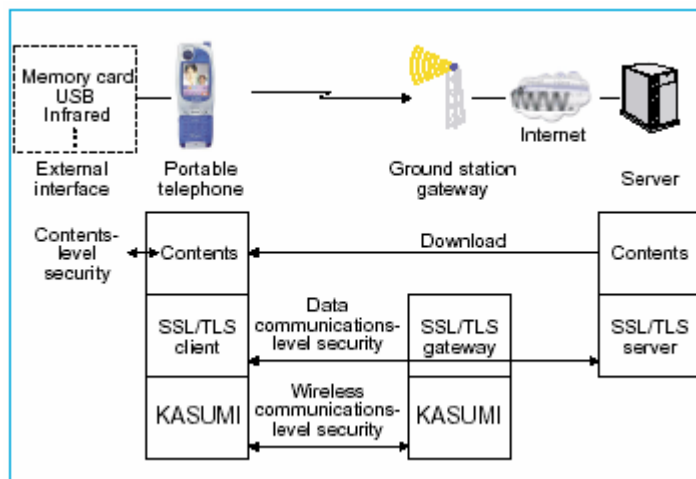
Serangan pada sistem terdistribusi tergantung pada pengaksesan ke saluran komunikasi yang ada atau membuat saluran baru yang menyamarkan sebagai koneksi legal. Jenis serangan sebagai berikut :

- penyerangan pasif, hanya mengamati komunikasi atau data,
- penyerangan aktif, secara aktif memodifikasi komunikasi atau data, pemalsuan atau perubahan *email*, *TCP/IP Spoofing*.

2.2.2 Metode penyerangan

Klasifikasi metode penyerangan adalah :

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



Gambar 1. Contoh teknologi keamanan sistem terdistribusi [12]

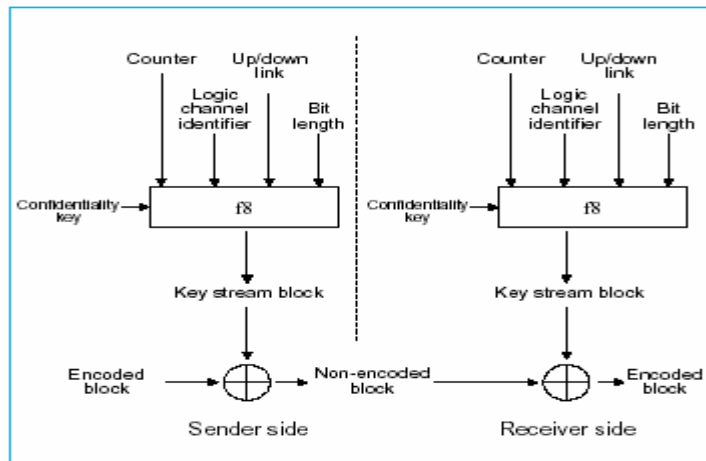
2.2.3 Teknik keamanan

a. Tingkat keamanan autentikasi pada komunikasi nirkabel

Merupakan sebuah teknologi untuk menggunakan fasilitas komunikasi nirkabel. Pemakai merupakan pelanggan yang telah membayar jasa layanan komunikasi dan untuk mendeteksi perubahan data yang tidak sah. Pada Gambar 3 menunjukkan fungsi f_9 yang digunakan untuk menghasilkan kode pengesahan untuk ditambahkan kepada data. Dalam urutan pesanan ketika suatu panggilan diminta, sisi jaringan membuktikan keaslian telepon *mobile* berdasarkan informasi pelanggan yang terdapat dalam *handset*. Jika informasi pelanggan telah dikirim kepada sisi jaringan dalam mode *plain-text*, informasi ini masih memungkinkan untuk disadap orang lain dan dengan sukses memainkan peran menyamar sebagai pelanggan asli. Autentikasi kemudian dilakukan oleh kedua sisi untuk melakukan perhitungan menggunakan informasi pelanggan dan membandingkan hasilnya. Dalam proses perhitungan untuk autentikasi sebuah peralatan bergerak dan jaringan saling berbagi kunci autentikasi itu dan mencegah perubahan data kunci secara tidak sah.

b. Autentikasi data

Dalam teknologi peralatan jaringan bergerak fungsi f_8 digunakan untuk menghasilkan satu rangkaian angka-angka secara acak dan logika eksklusif OR(X-OR) melakukan penjumlahan untuk masing-masing bit data pemakai dan sinyal data untuk melakukan pengundian. Panjang bit untuk *encoding*, *counter*, *uplink*, *downlink*, pengidentifikasi saluran yang logis dan kunci untuk kerahasiaan data dimasukkan kedalam fungsi logika f_8 untuk menghasilkan deretan angka-angka secara acak.



Gambar 2. Fungsi f8 untuk menjamin kerahasiaan data [13]

c. Integritas data

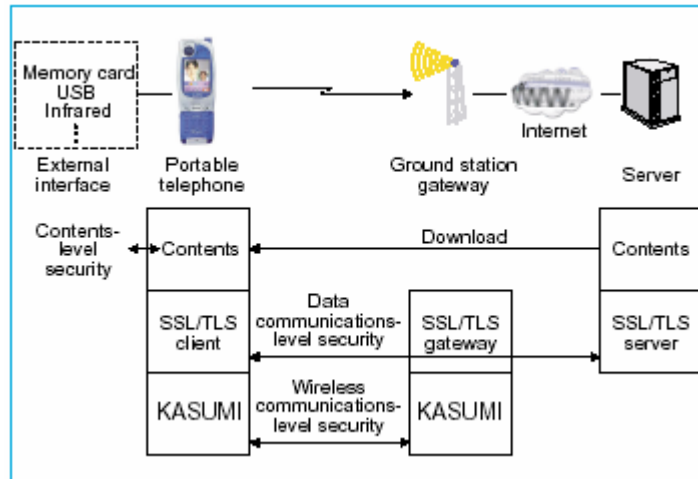
Integritas data mengacu pada teknologi yang mana kode autentikasi ditambahkan ke sinyal data di dalam komunikasi nirkabel dalam rangka mendeteksi perubahan data yang tidak sah. Fungsi f9 digunakan untuk menghasilkan kode autentikasi untuk ditambahkan pada data dalam rangka melihat kemungkinan perubahan data yang tidak sah dan memastikan integritas data. Kunci integritas dimasukkan kedalam fungsi f9 untuk menghasilkan kode pengesahan pesan. Penerima membandingkan kode pengesahan pesan yang dikirim oleh pengirim kepada kode pengesahan pesan yang dihasilkan oleh penerima, hal ini memungkinkan terjadi konfirmasi, ketika kode cocok, berarti menunjukkan tidak ada perubahan data secara tidak sah.

d. Algoritma Enkripsi KASUMI

Algoritma *enkripsi* membentuk inti *kepercayaan* data fungsi f8 dan fungsi integritas data f9 dikenal sebagai algoritma KASUMI. Beberapa kondisi yang harus dipenuhi ketika mengembangkan suatu algoritma *enkripsi* digambarkan oleh *Konsorsium Third Generation Pormerskip Project (3GPP)*, yang meneliti standar teknis untuk peralatan bergerak meliputi :

- keamanan harus dipelihara dibawah spesifikasi terbuka,
- mempraktekkan pembatasan didalam peralatan bergerak yang berarti bahwa algoritma itu harus diterapkan dalam perangkat keras menggunakan tidak lebih dari 10K gerbang,

- mempertimbangkan trafik peralatan bergerak, yang berarti bahwa pengelolaan harus pada 2Mbps.



Gambar 3. Algoritma KASUMI integrasi f8 dan f9 [12]

e. Tingkat keamanan komunikasi data

Komunikasi rahasia untuk peralatan bergerak pada jaringan nirkabel biasanya menggunakan SSL/TLS (*Secure Socket Layer/Transport Layer Security*) yang telah menjadi standar global untuk protokol keamanan antara *web server* dan *browser* pada internet. Fungsi keamanan untuk menjamin legitimasi berbagai isi dan modul yang di *download* dari jaringan itu. Legitimasi mengacu pada data yang telah dihasilkan oleh *provider* yang benar dan yang tidak mempunyai perubahan secara tidak sah setelah data itu dihasilkan.

f. Tingkat keamanan isi

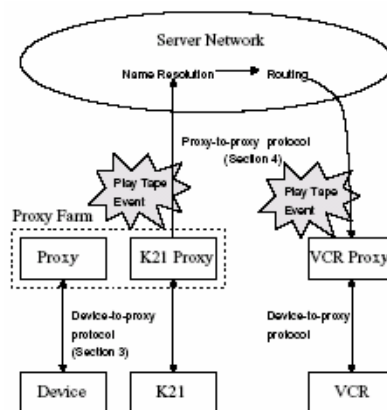
Penelitian tingkat keamanan isi masih dalam tahap awal dan standar-standar yang digunakan belum ditetapkan. Sebuah kartu memori *eksternal* yang disisipkan ke dalam peralatan bergerak telah dikembangkan pada Mitsubishi Electric sebagai *copyright control technology*. Teknologi tersebut dimungkinkan menggunakan teknologi *enkripsi*. Apabila isi yang telah dikemas disimpan secara *eksternal*, mereka di *enkripsi* dengan informasi dikhususkan untuk peralatan bergerak tersebut. Dengan cara yang sama ketika isi dimainkan kembali, isi itu di

deskripsi menggunakan data yang sama. Hal ini mungkin membuat pembatasan *playback* isi ke peralatan bergerak yang tersimpan.

3. Protokol keamanan berbasis proxy dalam jaringan peralatan bergerak

3.1 Arsitektur sistem

Sistem ini mempunyai tiga tipe komponen utama yaitu peralatan, *proxy*, dan *server*. Alat menunjuk pada tipe apapun dari sumber *network* (jaringan) yang terbagi, baik *hardware* maupun *software*. Dapat sebuah printer, kamera keamanan tanpa kawat, lampu, atau sebuah perangkat *software*. Karena protokol komunikasi dan *bandwith* antara alat-alat dapat bervariasi secara luas, maka setiap alat mempunyai sebuah *proxy* yang unik untuk menyatukan *interface*-nya dengan alat-alat lainnya. *Server* menyediakan nama dan fasilitas penemuan terhadap alat-alat yang bermacam-macam. Kami mengambil sebuah koresponden *one-to-one* antara alat-alat dan *proxy*. Kami juga menganggap bahwa semua pemakai dilengkapi dengan kunci K21, yang mana *proxy-proxy*-nya bekerja pada komputer yang dipercayai. Maka sistem kami hanya memerlukan untuk berhadapan dengan alat-alat, *proxy*, dan jaringan server. Sistem yang digambarkan seperti terlihat di Gambar 4 berikut ini :



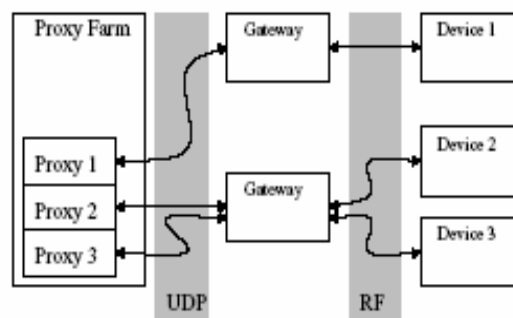
Gambar 4. Arsitektur sistem [6]

3.1.1 Alat-alat

Tiap alat *hardware* atau *software*, mempunyai sebuah *proxy software* yang terpercaya. Pada alat *hardware proxy* tersebut mungkin bekerja pada sebuah processor yang tersimpan dalam alat-alat, atau pada jaringan komputer terpercaya

dengan alat tersebut. Pada sebuah alat *software*, alat itu bisa memasukkan *software-proxy* itu sendiri.

Setiap alat berkomunikasi dengan *proxy* miliknya sendiri atas protokol yang sesuai bagi alat khusus itu. Suatu kawat printer dalam *ethernet* dapat berkomunikasi dengan *proxy*-nya dengan memakai TCP/IP. Suatu kamera tanpa kawat menggunakan protokol tanpa kawat untuk tujuan yang sama. K21 berkomunikasi dengan memakai protokol khusus alat ke *proxy* yang digambarkan pada Gambar 5 di bawah ini :



Gambar 5. Arsitektur sistem komunikasi alat ke *proxy* [6]

3.1.2 Proxy

Proxy merupakan *software* yang bekerja pada sebuah jaringan komputer yang tampak. Fungsi utama *proxy* adalah untuk membuat keputusan akses kontrol mewakili alat. *Proxy* juga menunjukkan fungsi skunder seperti mengatur tindakan tertulis (*scripted action*) mewakili alat dan *interfacing* dengan suatu *servis directory*.

Proxy memberikan suatu API yang sangat sederhana terhadap alat. Metode *send to proxy ()* dipanggil oleh alat untuk mengirim pesan ke *proxy*. Metode *send to device ()* dipanggil oleh *proxy* untuk mengirim pesan ke alat. Ketika sebuah *proxy* menerima pesan dari *proxy* lain maka *proxy* dapat menterjemahkannya ke dalam format yang dapat dipahami oleh alat tertentu dari *proxy* tersebut. Kemudian *proxy* menyampaikan pesan ke alat. Pada saat *proxy* menerima pesan dari alatnya, *proxy* menterjemahkan pesan tersebut ke dalam format yang dapat dipahami oleh semua *proxy*, kemudian meneruskannya ke *proxy* lainnya. Kapanpun *proxy* menerima pesan sebelum menampilkan terjemahan dan memberikan pesan itu ke alat *proxy* menampilkan akses kontrol.

Untuk administrasi, kita kelompokkan *proxy* oleh administratornya. Seperangkat administrator *proxy* disebut *proxy farm*. Perangkat ini khususnya meliputi *proxy* K21 milik administrator, yang dianggap *proxy* akar dari *proxy farm*. Ketika administrator menambahkan alat baru ke suatu sistem, *proxy* alat itu secara otomatis diberikan ACL (*Access Control List*), *default* (kesalahan), suatu duplikat ACL bagi *proxy* K21 milik administrator. Secara manual administrator dapat mengubah ACL berikutnya jika dikehendaki. Keuntungan yang perlu diperhatikan dari arsitektur berbasis *proxy* adalah dapat menunjukkan masalah virus pada lingkungan komputer yang dapat menembus. *Software scanning* virus yang canggih dapat diinstall dalam *proxy* itu, sehingga dapat membaca dengan cepat kode apapun sebelum di *download* kedalam alat-alat itu.

- **Command event** : digunakan untuk menginstruksikan sebuah peralatan untuk menghidupkan atau mematikan,
- **Error event** : digunakan untuk menyampaikan pada pendengar ketika terjadi kesalahan,
- **Status change event** : men-*generate*, contoh, ketika sebuah peralatan mengubah lokasinya,
- **Query event** : ketika *server* menerima sebuah *query event*, akan menampilkan sebuah DNS (*Domain Name Service*) atau INS (*Intentional Naming System*) *lookup* pada *query* dan kembali ke hasil *lookup* dalam *response event*,
- **Response event** : men-*generate* dalam respon ke *query*.

3.1.3 Server dan jaringan server

Jaringan ini terdiri dari kumpulan nama *independen server* dan *router*. Sebenarnya, tiap-tiap *server* bertindak sebagai nama *server* dan *router*. Hal ini sama dengan nama *resolver* pada INS [11], yang mana peralatan *resolve* menamai alamat IP, tetapi dapat juga mengirimkan *events*. Jika nama tujuan sebuah *event* (peristiwa cocok dengan *proxy-proxy* ganda, jaringan *server* akan mengirimkan *event* itu) ke semua tujuan yang cocok.

Ketika sebuah *proxy* datang secara *online*, *proxy* itu mencatat nama alat yang mewakilinya dengan salah satu *server* tersebut. Ketika sebuah *proxy* memakai satu *server* untuk menampilkan sebuah pandangan pada sebuah nama, *server* itu,

mencari direktorinya untuk semua nama yang cocok dengan nama yang diberikan, dan mengembalikan alamat-alamat IP-nya.

3.1.4 Komunikasi lewat peristiwa

Kita menggunakan sebuah mesin komunikasi berbasis-*event* (*event-based*) pada sistem yang dikembangkan. Karenanya, semua pesan lewat antara *proxy-proxy* yang sinyalnya menunjukkan bahwa beberapa telah terjadi. Contoh, sebuah bolam lampu bisa menghasilkan *light-on* (lampu hidup) dan *light-off* (lampu mati). Untuk menerima pesan-pesan ini *proxy x* dapat menambah sendiri seperti *event-listener* ke *proxy y*. Maka ketika *y* menggerakkan sebuah *event*, *x* akan menerima salinan.

Sebagai tambahan, sistem itu mempunyai beberapa kategori peristiwa *pre-defined* yang menerima perlakuan khusus pada *proxy* atau lapisan *server*. Seorang pengembang juga bisa menentukan *event*-nya sendiri. Secara sederhana jaringan *server* melewati *events* yang ditentukan pengembang melalui tujuannya.

Keuntungan utama dari mesin yang berbasis *event* (*event-based mechanism*) adalah bahwa mesin itu bisa menghapus kebutuhan secara berulang-ulang sebuah peralatan untuk menentukan perubahan dalam statusnya. Ketika sebuah perubahan terjadi, alat itu mengirimkan sebuah *event* kepada semua pendengar. Sistem seperti Sun Microsystem Jini [10] mengeluarkan “*device driver*” (RM1) kepada semua yang ingin mengontrol peralatan yang diberikan. Kemudian mesin itu bisa membuat panggilan lokal pada *device driver*, yang diterjemahkan dalam panggilan RM1 pada peralatan itu sendiri.

3.1.5 Penemuan sumber

Peralatan mesin untuk penemuan sumber sama dengan protokol penemuan sumber yang digunakan oleh Jini. Ketika sebuah peralatan datang secara *online*, alat itu menginstruksikan *proxy*-nya untuk menyiarkan secara berulang-ulang suatu permintaan untuk suatu *server* sub jaringan lokal. Permintaan itu berisi nama alat-alat dan alamat IP dan sisi kiri *proxy*-nya. Ketika *server* ini menerima salah satu permintaan, *server* itu menambah sepasang alamat nama / IP ke direktorinya. *Proxy* itu harus memperbaharui kontraknya secara periodik dengan mengirimkan sepasang alamat nama/IP yang sama ke *server*, sebaliknya *server* itu memindahkannya dari direktori itu.

Pada model ini, jika sebuah alat secara diam-diam menjadi *offline* atau alamat IP berubah. Kontrak *proxy* tidak akan lama menerima pembaharuan dan *server* akan melihat dengan cepat dan memindahkannya dari *directory* atau membuat kontrak baru dengan alamat IP yang baru. Contoh, bayangkan suatu alat dengan nama **[name=foo]** yang mempunyai *proxy* yang bekerja pada **10.1.2.3 : 4011**. ketika alat itu dinyalakan, alat itu menampilkan *proxy*-nya yang telah datang secara *online*, menggunakan suatu protokol seperti protokol *device-to-proxy*, *proxy* itu mulai mengirimkan paket format *lease-request* (permohonan kontrak) (**[name : foo]10.1.2.3 :4011**) pada sub jaringan lokal.

Jika suatu *server* menerima salah satu paket ini, server akan mengecek *directory*-nya untuk **[name = foo]**. Jika **[name=foo]** tidak ada disana maka *server* membuat suatu kontrak untuknya dengan menambahkan sepasang alamat nama/IP kepada *directory*-nya. Jika **[name=foo]** ada di *directory*, *server* akan memperbaharui kontrak. Beberapa saat atau waktu kemudian alat itu mati. Ketika peralatan itu turun, alat itu membawa *proxy offline* denganya, sehingga paket permintaan kontrak tidak lama lagi mendapatkan kiriman. Kontrak alat itu berhenti mendapatkan pembaharuan. Setelah beberapa saat, periode waktu *pre-defined*, *server* itu mengakhiri kontrak yang tidak diperbaharui dan memindahkannya dari *directory*.

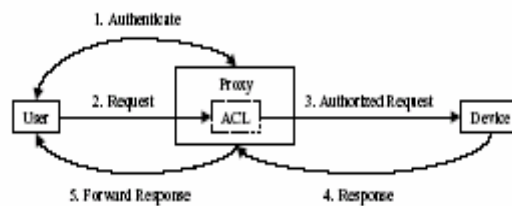
3.1.6 Model keamanan

Proxy dan alat berbagi sebuah kunci rahasia. Kunci rahasia ini, memungkinkan mereka untuk berkomunikasi dengan menggunakan keaslian kunci simetrik dan enkripsi. Pengoperasian kunci simetris memerlukan daya yang jauh lebih sedikit daripada *public-key*, sehingga alat itu dapat melakukan perhitungan dengan sebuah *micro-controller* kecil.

Semua komunikasi melewati *proxy*, sehingga *proxy* itu membuktikan keaslian dan kemudian menyampaikan komunikasi dari pemakai alat itu. Arus komunikasi ditunjukkan pada Gambar 6 dengan langkah seperti diuraikan berikut ini :

- *proxy* dan pemakai saling membuktikan serta saling membuat sebuah saluran komunikasi yang aman,
- mengecek pemakai mengirim permintaannya ke *proxy*,

- *proxy* memeriksa daftar akses kontrolnya untuk membuktikan pada pemakai apakah diijinkan untuk menampilkan permintaan khusus. Jika pengecekan berhasil *proxy* itu mengirimkan pesan pada alat tersebut, sebaliknya *proxy* akan merespon dengan sebuah pesan yang salah,
- alat menampilkan tindakan yang diminta dan mengirim sebuah respon kembali ke *proxy*,
- *proxy* mengirim respon itu kembali ke pamakai.



Gambar 6. Model keamanan [1]

3.1.7 Inisialisasi alat

Ketika sebuah alat diberi inisial, maka alat itu harus menentukan *proxy* dan harus mendapat sebuah kunci rahasia yang dipakai bersama dengan *proxy* tersebut. Ini dilakukan secara fisik menyentuh alat itu ke komputer yang akan menjalankan *proxy*. Ketika alat itu disentuh ke komputer, sebuah *proxy* dibuat dan *proxy* itu kemudian menciptakan sebuah kunci rahasia yang acak dan berbagi bersama alat itu. Inisialisasi ini langsung dan mudah bagi pemakai yang menginisial alat itu. Pemakai tidak perlu menampilkan konfigurasi manual apapun.

3.1.8 Implementasi alat

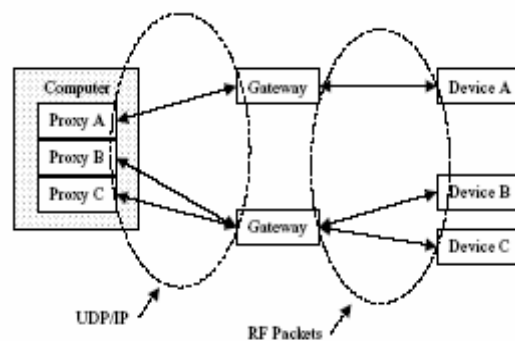
Peralatan-peralatan tanpa kabel dapat menentukan lokasinya dan dapat berkomunikasi dengan aman. Peralatan-peralatan itu juga dapat bertindak sebagai komunikator yang dapat dipakai dengan aman, diantara benda-benda lain, lokasi pemakai ke *proxy-proxy*-nya.

3.2 Protokol peralatan ke *proxy*

3.2.1 Komunikasi

Protokol *device-to-proxy* bervariasi untuk jenis-jenis alat yang berbeda. Secara khusus kita menganggap peralatan ringan dengan koneksi jaringan tanpa kawat yang *low bandwidth* (bandwidth rendah) dan CPU yang lambat, dan peralatan berat dengan koneksi *bandwidth* yang lebih tinggi dan CPU yang lebih cepat. Kita berasumsi bahwa alat-alat berat bisa menjalankan *software proxy* secara lokal yaitu *proxy* untuk sebuah printer dapat bekerja pada sebuah CPU. Dengan *proxy* lokal, protokol yang canggih untuk komunikasi *device-to-proxy* yang aman tidaklah perlu, menganggap bagian-bagian kritis suatu alat merupakan suatu *tampering-resistant*. Untuk alat-alat berat, *proxy* harus bekerja di tempat lain.

Komunikasi RF (*Radio Frequency*) antara alat dan *proxy*-nya dikendalikan oleh sebuah gerbang (*gateway*) yang menterjemahkan komunikasi RF ke dalam paket UDP/IP yang kemudian dikirimkan di atas jaringan ke *proxy*. *Gateway* juga bekerja pada arah yang berlawanan dan mengubah paket UDP/IP dari *proxy* ke paket RF dan mentransmisikan ke alat itu. Suatu tinjauan luas terhadap komunikasi ditujukan pada Gambar 7 di bawah, *proxy* satu untuk setiap tiga alat terpisah. Gambar itu juga menunjukkan bagaimana *gateway* ganda dapat digunakan. Alat A menggunakan suatu *gateway* yang berbeda dengan alat B dan C.



Gambar 7. Arsitektur komunikasi [1]

3.2.2 RF protokol

Semua komunikasi alat dikodekan dengan *proxy*, yang menyimpan kode alat sangat sederhana, karena hanya merespon pesan. Protokol ini juga membantu menjaga saluran RF dari *over-utilized*. Jika peralatan itu mengkodekan komunikasi,

alat ini akan banyak menimbulkan tabrakan transmisi karena alat-alat ganda mengirimkan informasi pada saat yang bersamaan. Karena saluran RF mempunyai *bandwidth* sempit, maka banyak terjadi tabrakan pada transmisi yang dapat menimbulkan *performance* benar-benar menurun dan menyebabkan penundaan yang tidak dapat diterima dalam komunikasi.

Namun, karena *proxy-proxy* itu menginisial semua komunikasi maka *gateway* dapat bertindak sebagai sebuah *arbiter* di atas saluran RF. Ketika *gateway* itu menerima sebuah pesan dari *proxy*, *gateway* tersebut menyiarkannya pada saluran RF. Karena alat akan selalu merespon suatu pesan, *gateway* menunggu 50 ms untuk suatu respon. Selama waktu itu *gateway* tidak akan menstransmisi pesan apapun pada saluran RF, hal itu untuk menghindari tabrakan transmisi antara *gateway* dengan alat. *Gateway* mempunyai RF yang minimal sehingga dapat bertindak sebagai *arbiter* untuk wilayah siarannya.

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

Pada peralatan bergerak akan menyebabkan suatu masalah karena komunikasi harus terjaga. Jika sebuah alat bergerak maka akan keluar dari jangkauan komunikasi dari satu *gateway* dan kemudian masuk ke jajaran yang lain. *Proxy* pada alat itu tidak akan tahu tentang *gateway* yang baru sehingga tidak akan mampu mengontak alat itu. Harus ada sebuah mekanisme bagi alat untuk mengatakan pada *proxy*-nya tentang *gateway* yang baru itu. Sehingga kapanpun alat itu tidak menerima sebuah paket baru dari *proxy*-nya selama 10 detik, alat itu mulai mentransmisi kembali paket terakhir satu kali setiap 4 detik. Paket ini 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 alat komunikasi melalui sebuah saluran yang aman yang mengenkripsi dan meng-autentikasikan semua pesan. Algoritma HMAC-MD5 [5] digunakan untuk pembuktian keaslian dan algoritma RC5 [9] digunakan untuk enkripsi. Kedua algoritma ini menggunakan kunci simetrik, *proxy* dan alat berbagi kunci 128-bit.

- **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 tertentu yang dapat menciptakan suatu MAC khusus atau menguji bahwa sebuah MAC itu benar. HMAC secara esensial menghitung $MAC = H(K, D)$ tetapi penghitungan sebenarnya sedikit lebih kompleks. Karena HMAC menggunakan kunci rahasia, maka hanya seseorang yang tahu bahwa kunci itu dapat membuat MAC atau menguji bahwa MAC itu benar.

HMAC dengan fungsi hash MD5 menghasilkan 16 byte MAC. Delapan byte MAC yang paling signifikan dilampirkan pada akhir tiap-tiap paket. Ini membatasi jumlah data yang harus ditransmisi dengan tiap-tiap paket. Dari segi keamanan, ini mempunyai keuntungan memberikan sedikit informasi pada seorang penyerang, tetapi kerugiannya membiarkan penyerang harus menebak bit lebih sedikit. Hal ini dirasakan merupakan sebuah *tradeoff* yang baik jika ke 16 byte MAC dimasukkan dalam setiap alat paket, bahkan kemudian lebih dari tiap paket akan diautentikasi sebagai pengganti data yang bermanfaat.

- **Enkripsi**

Data dienkripsi dengan menggunakan algoritma enkripsi RC5. RC5 dipilih karena kesederhanaan dan penampilannya. RC5 tidak menghendaki table untuk mempercepat proses karena merupakan XOR utama dan menjalankan pengoperasian. Implementasi RC5 kita berdasar pada kode Open SLL [8]. RC5 merupakan sebuah *cipher blok*, biasanya bekerja dalam 8 byte blok data. Namun dengan mengimplementasikannya dengan menggunakan *mode output feedback* (OFB), RC5 dapat digunakan sebagai suatu chipper arus. Hal ini memungkinkan enkripsi suatu jumlah byte yang berubah-ubah tanpa harus mengenai blok-blok data.

Mode OFB bekerja dengan menggerakkan suatu *bloknote* enkripsi dari sebuah vektor inisial dan kunci. *Bloknote* enkripsi ini kemudian di XORkan dengan data untuk menghasilkan *chipertext*. Karena $X \oplus Y \oplus Y = X$, *chipertext* dapat didekripsikan dengan menghasilkan blok enkripsi yang sama dan meng-XOR-kannya dengan *chipertext*. Karena ini hanya menghendaki enkripsi RC5 untuk menggerakkan blok enkripsi, memisahkan enkripsi dan dekripsi rutin tidaklah dikehendaki. Untuk implementasi, kita menggunakan 16 *round* untuk RC5 dan menggunakan kunci 128-bit yang berbeda enkripsi dan autentikasi.

3.2.4 Lokasi

Lokasi alat ditentukan dengan menggunakan sistem lokasi *cricket* [7]. *Cricket* mempunyai beberapa ciri yang bermanfaat, termasuk privasi pemakai, kontrol disentralisasi, biaya murah, dan penyebaran informasi yang mudah. *Cricket* juga dapat bekerja di dalam ruangan. Privasi pemakai *cricket* berarti bahwa tidak ada layanan pusat dimana lokasi-lokasi disimpan, setiap alat menentukan lokasinya sendiri. Hal ini sampai alat tersebut memutuskan apakah ingin memberitahu yang lainnya letak alat itu.

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

Sistem *cricket* mudah diganggu oleh sumber-sumber *eksternal ultrasound*. Banyak suara yang umum yang dapat menghasilkan *ultrasound* yang dapat membingungkan pendengar *cricket*. Hal ini mungkin untuk menghindari masalah dalam suatu periode yang lebih panjang sebelum membuat keputusan, tetapi dapat menimbulkan penundaan dari 6-10 detik sebelum keputusan dibuat.

3.3 Protokol *proxy* ke *proxy*

SPKI/SDSI [2] merupakan sebuah insfraktuktur keamanan yang dirancang untuk memudahkan pengembangan sistem penghitung yang dapat diukur, aman dan terbagi. SPKI/SDSI menyediakan kontrol akses *fine-grained* dengan menggunakan

suatu arsitek ruangan 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 infrakstruktur global yang hirarkis. Komunitas SPKI/SDSI dibangun dari dasar keatas (*bottom-up*), pada suatu cara yang terbagi, dan tidak menghendaki suatu akar yang terpercaya.

3.3.1 Integrasi SPKI/SDSI

Kita telah mengambil suatu arsitektur *client-server* untuk *proxy-proxy*. Ketika seorang pelaku tertentu, bertindak atas nama alat atau pemakai, membuat suatu permintaan atas nama *proxy* kepada sebuah alat yang diwakili oleh *proxy* lainnya, *proxy* pertama bertindak sebagai *client*, dan *proxy* kedua sebagai *server*. Sumber-sumber pada *server* itu adalah publik yang dilindungi oleh SPKI/SDSI ACL. SPKI/SDSI ACL, terdiri dari daftar peserta, setiap peserta mempunyai subjek (kunci atau kelompok) dan sebuah label yang menentukan seperangkat operasi kunci atau group itu diijinkan tampil. Untuk mencapai sumber yang dilindungi oleh ACL, pemohon harus mencantumkan pada permohonannya sebuah rangkaian sertifikat yang menunjukkan bahwa dia adalah anggota dari sebuah kelompok dalam suatu *entry* pada ACL.

Jika suatu sumber yang diminta dilindungi oleh ACL, permohonan pelaku harus disertai dengan “*proof of authenticity*” (bukti keaslian) yang menunjukkan bahwa itu adalah asli, dan “*proof of authorization*”(bukti otorisasi) yang menunjukkan bahwa pelaku itu berwenang untuk mengajukan permohonan tertentu pada sumber tertentu. Bukti keasliannya adalah terdapat permohonan yang ditandatangani, permohonan harus sama dengan pelaku yang mengesahkan sertifikat-sertifikat itu. Desain sistem dan protokol antara *proxy-proxy* sangat mirip sehingga digunakan pada *Project Geronimo* SPKI/SDSI dimana SPKI/SDSI tergabung dalam Apache dan Nestcape serta digunakan untuk memberikan akses kontrol *client* di atas *web*.

3.3.2 Protokol

Protokol diimplementasikan oleh *client* dan *proxy server* terdiri dari empat pesan. Protokol ini ditunjukkan pada Gambar 8 dan berikut adalah penjelasannya :

- a. *Proxy client* mengirim suatu permohonan, tidak asli dan tidak sah, ke *sever proxy*.

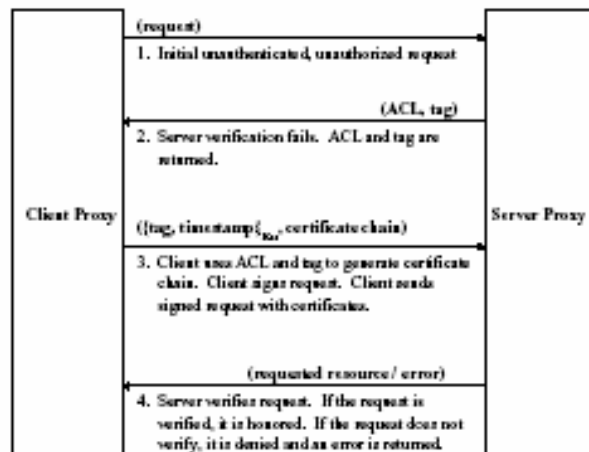
- b. Jika *client* meminta akses ke sebuah sumber terlindungi, *server* merespon dengan ACL yang melindungi sumber tersebut dan label dibentuk dengan permohonan *client*. *Tag* merupakan suatu struktur data SPKI/SDSI yang mewakili suatu permohonan. Ada banyak contoh pada draft SPKI/SPSI IETF (*Internet Engineering Task Force*) [2]. Jika tidak ada ACL yang melindungi sumber yang diminta, permohonan itu segera di terima.
- c. *Proxy client* men-*generate* serangkaian sertifikat yang menggunakan SPKI/SDSI *certificate chain discovery algorithm* [3] (algoritma penemuan serangkaian sertifikat SPKI/SDSI). Seperangkat ini memberikan sebuah bukti otorisasi yang membuktikan bahwa kunci pemakai itu sah untuk mengajukan permohonannya. Algoritma penemuan serangkaian sertifikat itu mengambil *input* ACL dan *tag* dari *server*, kunci publik pemakai, seperangkat sertifikat pemakai dan *timestamp*. Jika hal itu ada, algoritma itu mengembalikan serangkaian sertifikat pemakai yang memberikan bukti bahwa kunci publik pemakai adalah sah untuk menampilkan operasi yang ditentukan dalam *tag* itu, pada saat ditentukan pada *timestamp*. Jika algoritma tidak dapat men-*generate* serangkaian sertifikat karena pemakai tidak mempunyai sertifikat yang penting, atau jika kunci pemakai secara langsung ada pada ACL maka algoritma itu menembalikan serangkaian sertifikat kosong. Klien men-*generate timestamp* dengan menggunakan jam lokalnya.
- d. Klien membuat urutan SPKI/SDSI [2] 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 di-*generate* 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*,
 - mengutip 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 *entry* pada ACL ke kunci dari tanda tangan, melalui serangkaian sertifikat yang disajikan. Serangkaian otorisasi harus mengizinkan klien untuk menampilkan operasi yang diinginkan.

Jika permohonan itu membuktikan keaslian, maka permohonan dikabulkan. Jika tidak membuktikan keaslian, permohonan ditolak dan *proxy server* mengembalikan kesalahan pada *proxy* klien. Kesalahan ini dikembalikan ketika klien menyajikan sebuah permohonan sah yang ditolak.

Protokol dapat dilihat sebagai sebuah protokol *challenge-response* yang khas. Jawaban *server* pada langkah b dari protokol itu merupakan suatu tantangan bagi *server* yang memberikan klien dengan mengatakan “**Anda sedang mencoba mengakses sebuah file yang terlindungi**”. **Buktikan pada saya bahwa anda mempunyai surat mandat untuk menampilkan operasi yang anda minta sumber yang dilindungi oleh ACL ini.** Klien menggunakan ACL untuk membantunya menghasilkan sebuah rangkaian sertifikat, menggunakan algoritma penemuan serangkaian sertifikat dan permohonan yang tertanda 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, permohonan diterima.

Timestamp pada urutan *tag-timestamp* membantu melindungi terhadap jenis-jenis tertentu suatu serangan balik. Contoh, anggap saja *server* mencatat permohonan dan anggap bahwa catatan itu tidak diberikan dengan tepat. Jika seorang musuh mencapai akses ke *logs* (catatan), *timestamp* mencegahnya dari permohonan ulang yang ditemukan pada *logs* dan mencapai akses kesumber terlindungi.



Gambar 8. SPKI/SDSI Proxy to proxy ACL [6]

3.3.3 Additional Security Consideration

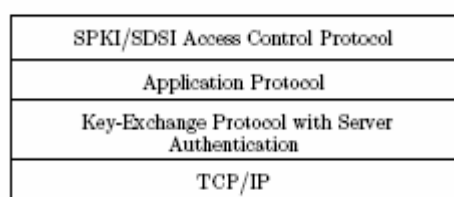
Protokol SPKI/SDSI sebagaimana yang telah diuraikan, menunjukkan masalah penyediaan akses kontrol klien. Protokol tidak meyakini kerahasiaan, membuktikan pada *server*, atau memberikan perlindungan terhadap serangan balik dari jaringan. Protokol SSL saat ini merupakan protokol keamanan yang paling banyak digunakan. Protokol TLS merupakan pengganti SSL. Tujuan utama dari SSL/TLS adalah memberikan kerahasiaan dan integritas data hubungan antara *client* dan *server*, dan memberikan pembuktian keaslian pada *server*. Ada dukungan bagi pembuktian keaslian *client*, tetapi pembuktian itu merupakan pilihan.

Protokol akses kontrol SPKI/SDSI bisa dilapisi di atas sebuah protokol *keyexchange* (pertukaran kunci) seperti TLS/SSL untuk memberikan keamanan tambahan. TLS/SSL saat ini menggunakan x.509 PKI untuk membuktikan keaslian *server*, tetapi dapat juga memakai SPKI/SDSI dengan cara yang sama. SSL/TLS juga memberikan perlindungan terhadap serangan balik dari jaringan, dan perlindungan terhadap serangan *person-in-the-middle*. Dengan pertimbangan-pertimbangan ini, lapisan protokol ditunjukkan pada Gambar 9. Pada gambar tersebut, “**Application Protocol**” menunjukkan 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*. 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 itu. Dengan SSL/TLS dan SPKI/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 mendekati masalah ini, kita mengajukan 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 itu, protokol SPKI/SDSI keluar 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 menampilkan permohonan klien. Jika keamanan tambahan dibutuhkan, protokol secara luas dapat dilapisi diatas 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, hanya terdiri dari *client* dan *server*.



Gambar 9. *Example Layering of Protocol* [6]

3.4 Hubungan kerja

3.4.1 Komunikasi alat ke *proxy*

Resurrecting Duckling merupakan suatu model untuk jaringan *wireless*. Pada model ini, ketika peralatan-peralatan mulai bekerja, mereka harus ditandai sebelum digunakan. Sebuah *master* menandai sebuah alat dan menjadi yang pertama untuk berkomunikasi dengannya. Setelah memberi tanda, sebuah alat hanya mendengarkan

master-nya. Selama proses pemberian tanda *master* tersebut diletakkan pada kontak fisik dengan alat itu dan mereka berbagi sebuah kunci rahasia yang kemudian digunakan untuk pembuktian keaslian dan enkripsi kunci simetris. *Master* itu juga memberi tugas kontrol suatu alat ke alat-alat lainnya sehingga kontrol itu tidak selalu terbatas pada *master* itu saja. Sebuah alat dapat dimatikan oleh *master*-nya kemudian dihidupkan lagi oleh yang baru agar alat itu dapat menukar *master*-nya.

3.4.2 Komunikasi *proxy* ke *proxy*

Teknologi jaringan Jini dari Sun Microsystems merupakan ide dari gedung federasi. Jini menghindari penggunaan *proxy-proxy* dengan anggapan bahwa semua alat dan servis pada sistem itu akan menjalankan Java Virtual Machine. Proyek SIESTA pada Universitas Teknologi Helsinki telah berhasil membangun sebuah *framework* untuk menyatukan Jini dan SPKI/SDSI. Implementasinya memiliki beberapa perhatian masalah *latency* ketika otorisasi baru diberikan. Proyek UC Berkeley's Ninja menggunakan sumber pada suatu *wide area network*. Proyek-proyek lain yang berhubungan adalah Hewlett Packard's Cool Town, IBM's TSPaces dan University of Washington's Portolano.

4. Evaluasi

4.1 *Memory Requirements*

Tabel 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 alat meliputi "*packet and location processing routines*". Komponen kode RF meliputi RF yang mempunyai kebiasaan mentransmisi dan menerima sebagaimana kebiasaan pendengar *cricket*. Komponen yang beraneka ragam merupakan kode yang umum untuk semua komponen lainnya.

Kode alat menghendaki sekitar 12 KB ruang kode dan 1 KB ruang data. Algoritma keamanan, HMAC-MD5 dan RC5 mengambil sebagian besar ruang kode. Kedua-duanya dari algoritma ini telah dioptimalkan di dalam perakitan, yang mengurangi ukuran kode mereka lebih dari separuh. Kode bisa menjadi lebih baik dioptimalkan, tetapi hal ini memberikan pemikiran umum seberapa banyak memori dikehendaki. Ukuran kode yang telah kita dapatkan adalah cukup kecil yang dapat disatukan ke dalam hampir semua alat apapun. Untuk lebih jauh mengoptimalkan

ukuran kode, algoritma autentikasi yang lebih kecil dapat digunakan. Sistem ini menggunakan HMAC-MD5 yang memerlukan 4,6 KB memori. Kemungkinan yang lain akan menggunakan sebuah algoritma autentikasi dengan enkripsi sebagai pengganti fungsi *hash*. Hal ini dapat mengurangi ukuran kode secara signifikan, karena kode RC5 berada dalam enkripsi dan autentikasi. Pengurangan ukuran kode menjadi lebih sedikit dari 8 KB.

Tabel 1. *Code and data size on the Atmel processor* [1]

| Component | Code Size (KB) | Data Size (bytes) |
|----------------------|----------------|-------------------|
| Device Functionality | 2.0 | 191 |
| RF Code | 1.1 | 153 |
| HMAC-MD5 | 4.6 | 386 |
| RC5 | 3.2 | 256 |
| Miscellaneous | 1.0 | 0 |
| Total | 11.9 | 986 |

4.2 Processing Requirements

Algoritma untuk keamanan menempatkan permintaan terbanyak pada alat-alat itu. Tabel 2 merinci rata-rata waktu untuk masing-masing algoritma yang digunakan. RC5 memproses waktu bervariasi secara linier dengan jumlah byte yang di enkripsi atau deskripsi. Disisi lain HMAC-MD5, memerlukan sejumlah waktu yang konstan di atas 56 byte. Hal ini karena HMAC-MD5 dirancang untuk bekerja pada blok data, sehingga waktu yang kurang dari 56 byte harus ditambah. Karena ukuran paket RF dibatasi pada 50 byte, maka analisis HMAC-MD5 yang menggunakan waktu untuk paket ukuran kurang dari atau sama dengan 50 byte.

Tabel 2. *Performance of encryption and authentication code* [1]

| Function | Time (ms) | Clock Cycles |
|----------------------------------|------------------|---------------|
| RC5 encrypt/decrypt (n bytes) | $0.163n + 0.552$ | $652n + 2208$ |
| HMAC-MD5 up to 56 bytes | 11.48 | 45,920 |

4.3 Evaluasi SPKI/SDSI

Protokol yang diuraikan di awal adalah efisiensi. Dua hal pertama protokol itu adalah suatu pasangan permintaan yang standar, tidak ada kriptografi ayng dibutuhkan. Langkah-langkah yang signifikan pada protokol itu adalah langkah dimana suatu

rangkaian sertifikat dibentuk, dan langkah dimana rangkaian itu diuji. Tabel 3 menunjukkan analisis dari kedua langkah tersebut di atas. Tulisan mengenai *Certificate Chain Discovery* pada SPKI/SDSI [4] semestinya berkaitan dengan pembicaraan mengenai analisis penentuan waktu. Waktu CPU merupakan waktu rata-rata yang diukur pada sebuah Sun Microsystem Ultra-1 yang menjalankan SunOS 5.7.

Tabel 3. Analisis protokol proxy ke proxy [6]

| Protocol step | Timing analysis | Approx CPU time |
|----------------------|---|-----------------------------------|
| Cert chain discovery | The worst case 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 $n = 2$. |

5. Kesimpulan

Kita percaya bahwa kecenderungan pada penghitungan *pervasive* meningkatkan perbedaan dan keragaman jaringan dan alat-alat pokoknya. Mengembangkan protokol keamanan yang dapat mengatasi keragaman, alat-alat bergerak yang dibuat jaringan dengan bermacam-macam cara memberikan tantangan yang besar.

Dalam tulisan ini, telah diambil langkah awal untuk memebuhi tantangan tersebut dengan mengamati kebutuhan bagi protokol keamanan ganda, masing-masing karakteristik yang berbeda dan persyaratan secara penghitungan yang berbeda pula. Karena kita telah membicarakan sebuah sistem *prototype* dengan dua protokol yang berbeda, tipe lain dari protokol dapat dimasukkan jika dianggap perlu.

Dua protokol yang telah kita uraikan mempunyai karakteristik yang sangat berbeda, karena mereka menerapkan skenario yang berbeda. Protokol peralatan ke *proxy* dirancang untuk memungkinkan komunikasi data yang aman dari sebuah peralatan yang ringan. 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 sumber daya sistem komunikasi yang aman dan efisien.

6. References

- [1] 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.
- [2] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. Simple Public KeyCertificate. *The Internet Society*, July 1999. See <http://world.std.com/~cme/spki.txt>.
- [3] D. Clarke. *SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI*. Master's thesis, Massachusetts Institute of Technology, 2001.
- [4] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate Chain Discovery in SPKI/SDSI. *Journal of Computer Security*, 2001. To appear.
- [5] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. *Internet Request for Comments RFC 2104*, February 1997.
- [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>
- [7] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. ACM MOBICOM*, August 2000.
- [8] OpenSSL. The OpenSSL Project. <http://www.openssl.org>.
- [9] R. Rivest. The RC5 Encryption Algorithm. In *Proc. of the 1994 Leuven Workshop on Fast Software Encryption*, 2001.
- [10] Sun Microsystems Inc. Jini Network Technology. <http://www.sun.com/jini>.
- [11] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The Design and Implementation of an Intentional Naming System. *Operating Systems Review*, 34(5):186-301, December 1999.
- [12] WCDMA: *Towards IP Mobility and Mobile Internet*, <http://www.techbooks.co.uk/artech/book544.htm>.
- [13] 3GPP confidentiality and integrity algorithms, <http://www.3gpp.org/TB/Other/algorithms.htm>

