Sizzle : Solusi Keamanan untuk Perangkat Keras dengan Kemampuan Terbatas untuk Implementasi pada Wireless Sensor Network

Mukhamad Ikhsan, 13503033
Teknik Informatika, Sekolah Tinggi Elektro dan Informatika
Institut Teknologi Bandung
ikhsan_only@yahoo.co.uk, if13033@students.if.itb.ac.id

Abstraksi : Selama ini algoritma kriptografi kunci publik seperti RSA, El-Gamal dan algoritma lainnya terkenal dengan kebutuhan sumber daya perhitungan yang cukup besar, dan sangat sulit diterapkan dalam sebuah perangkat keras yang kecil yang tidak memiliki kemampuan proses besar dan juga memiliki keterbatasan jumlah memori. Ketika algoritma kriptografi kunci publik yang disebutkan diatas akan diterapkan pada Wireless Sensor Network, algoritma-algoritma tersebut tidak mungkin diimplementasikan, dikarenakan batasan-batasan yang dimiliki oleh arsitektur Wireless Sensor Network yang memiliki karakteristik : kecilnya ukuran paket-paket, kecilnya kamampuan komputasi, kecilnya kapasitas memori, dan kecilnya kernel space dari sistem operasi yang digunakan. Oleh karenanya dikembangkan algoritma elliptic curve cryptography yang mereduksi ukuran kunci yang berukuran sangat besar bagi perangkat-perangkat keras seperti smart card, atau mobile phone. Kini dengan dasar algoritma tersebut dikembangkan sebuah device bernama Sizzle yang bertenaga baterai dengan 8 bit microprocessor, 128 Kb Flash, dan 4 Kb Ram yang mampu mentransfer 1 Kb data melewati protokol SSL (Secure Socket Layer) dalam waktu 0,4 detik. Seiring dengan perkembangan jaman yang akan menuju pada dunia 4G dimana akan diterapkan All IP Environment, maka Sizzle bisa menjadi solusi untuk keamanan pada perangkat-perangkat yang akan digunakan di kemudian hari. Sizzle adalah sebuah ide revolusioner dimana kita dapat secara remote mengontrol sebuah perangkat lewat internet dan memberikan sistem keamanan yang baik, tetapi penelaahan terhadap Sizzle harus dilakukan terlebih jauh lagi terkait dengan penerapannya untuk teknologi masa depan dengan mempertimbangkan banyak hal seperti masalah kompatibilitas dengan perkembangan teknologi pada bidang-bidang lainnya dan apakah Sizzle dapat secara luas digunakan sebagai mobile server yang aman untuk bidang-bidang yang sensitif dalam hal keamanannya seperti e-commerce.

Kata kunci : Sizzle, Cryptography, Wireless Security, Security Protocol, Secure Socket Layer, Transport Layer Security.

Pendahuluan

Pada beberapa tahun belakangan ini, penggunaan internet berkembang begitu cepat diluar perkiraan kita, baik dalam bentuk desktop, server, dan laptop serta beberapa perangkat seperti PDA dan smart phone.

Hal ini akan berkembang pada sebuah trend dimana devices yang kecil seperti sensor, perabotan rumah, alat medis, juga terhubung dengan internet, didukung dengan perkembangan menuju teknologi 4G. Istilah “embedded internet” sering digunakan pada suatu benda sehari-hari yang kita gunakan yang terhubung dengan internet. Perangkat-perangkat dengan kemampuan sensor dan komunikasi akan diimplementasikan dengan kemampuan komputasi yang kini mungkin belum populer, seperti memonitor lingkungan, respon medis yang bersifat darurat, pengaturan di lapangan, atau proses otomatisasi di rumah.

Banyak dari perangkat-perangkat diatas membutuhkan sistem keamanan agar tidak mudah dimanipulasi oleh orang yang tidak berhak. Contohnya, informasi kesehatan pada seorang pasien, hanya boleh diketahui oleh orang-orang yang berhak saja (telah terautentifikasi) dan harus juga terlindung dari modifikasi (integritas) ilegal dan juga menjamin keasliannya (confidentiality) pada saat data-data yang ada pada perangkat tersebut dipertukarkan. Bahkan data-data seperti temperatur dan tekanan udara pun

butuh untuk diamankan. Layaknya pada kasus pabrik bahan kimia, dimana sensor digunakan untuk memantau reaksi-reaksi yang terjadi pada proses manufaktur untuk menghasilkan produk akhir. Tanpa sistem keamana yang mencukupi, seorang bisa saja memanipulasi informasi-informasi yang dihasilkan sensor sehingga dapat menimbulkan bencana yang sangat besar.

Secure Socket Layer (SSL) adalah protokol keamanan yang sangat luas dipakai saat ini di internet. SSL banyak diimplementasikan pada berbagai aplikasi seperti web browser dan banyak digunakan untuk melakukan transaksi-transaksi keuangan yang membutuhkan keamanan, sepertin online banking, stock trading, dan e-commerce.

SSL mengkombinasikan kriptografi kunci publik untuk melakukan distribusi kunci dan autentifikasi dengan kriptografi kunci simetris untuk enkripsi dan integritas data. Kriptografi kunci publik dikenal terlalu berat untuk diimplementasikan di perangkat-perangkat yang kecil. Persepsi ini dihasilkan dari sebuah ekspresimen yang melibatkan bahasa pemograman C yang mengimplementasikan RSA, yang merupakan algoritma kunci publik yang paling populer untuk saat ini.

Eksperimen akhir-akhir ini telah menunjukkan bahwa kita dapat mengoptimasikan agar SSL dapat diimplementasikan pada perangkat-perangkat mini yang telah disebutkan diatas. Dengan penggunaan bahasa pemograman assembler untuk mengimplementasikan RSA ternyata didapatkan hasil yang menunjukan peningkatan kecepatan. Selain itu penggunakan algoritmat Elliptical Curve Cryptography (ECC) juga menghasilkan peningkatan perfoma. Sehingga dapat diimplementasikan sistem keamanan menggunakan SSL pada perangkat dengan spesifikasi 8 bit microprocessor, 128 Kb Flash, dan 4 Kb Ram yang mampu mentransfer 1 Kb data melewati protokol SSL (Secure Socket Layer) dalam waktu 0,4 detik.

Tetapi yang harus dieksplor lagi lebih jauh adalah, apakah tepat solusi Sizzle yang akan kita bahas pada makalah ini, untuk penggunaan di masa yang akan datang. Hal ini membutuhkan pertimbangan-pertimbangan yang didasarkan kepada, perkembangan teknologi internet, perkembangan kemampuan perangkat keras, dan perkembangan aplikasi yang akan muncul di masa yang akan datang sesuai dengan kebutuhan-kebutuhan yang terus meningkat dalam hal teknologi informasi yang berkembang begitu cepat.

Wireless Sensor Networks

Wireless Sensor Network adalah sebuah jaringan wireless yang mengandung beberapa perangkat terdistribusi yang otonom dan menggunakan sensor untuk secara kooperatif memonitor suatu keadaan fisik atau kondisi lingkungan, seperti suhu, suara, getaran, tekanan, gerakan atau polutan diberbagai lokasi yang berbeda.

Perkembangan perangkat-perangkat wireless sensor network asal mulanya dikemabangkan untuk kepentingan militer seperti kamera tersembunyi di arena peperangan. Walaupun begitu, kini wireless sensor network digunakan banyak di dalam kehidupan sipil di berbagai area. Contohnya adalah penerapan untuk memonitor suatu habitat, aplikasi kesehatan, otomatisasi rumah dan kontro lalu lintas.

Selain itu, untuk satu atau lebih sensor yang digunakan, setiap node didalam sensor biasanya dipasang juga radio transceiver atau wireless communications device, mikrokontroler kecil, dan sebuah sumber energi yang biasanya berupa baterai.

Ukuran dari sebuah sensor dapat bermacam-macam, dari seukuran kotak sepatu sampai seukuran kelereng atau bahkan debu. Harga dari sensor pun bermacam-macam dari jutaan rupiah sampai hanya puluhan ribu, tergantung dari ukuran sensor dan tingkat kerumitan dari sensor tersebut. Ukuran dan harga dari sensor menentukan batasan-batasan kemampuan sensor dalam hal kehematan penggunaan energi, memori, kecepatan komputasi, dan bandwith.

Aplikasi pada Wireless Sensor Network

Aplikasi pada wireless sensor network sangat banyak dan bermacam-macam. Biasanya digunakan pada bidang komersil dan industri untuk memonitor data yang mungkin akan lebih sulit dan mahal jika harus mengimplementasikan wired sensor.

Aplikasi-aplikasi ini dapat diimplementasikan pada area-area yang berbahaya atau liar, dimana harus bertahan untuk jangka waktu yang cukup lama (memonitor suatu variabel dalam lingkungan) tanpa harus mengisi ulang atau mengganti sumber energi yang digunakan.

Aplikasi pada umumnya digunakan untuk kebutuhan-kebutuhan monitoring, tracking, dan kontroling. Sebagian aplikasi yang spesifik adalah memonitor habitat, pelacakan suatu objek, kontrol reaktor nuklir, pendeteksian api, memonitor lalu lintas, dan lain-lain. Di beberapa aplikasi pada umumnya, WSN mempunyai beberapa sensor yang disebarkan di suatu area dalam rangka mendapatkan informasi yang luas secara simultan. Biasanya diimplementasikan pada :

- Monitoring lingkungan
- Monitoring habitat
- Pendeteksian akustik
- Pendeteksisan seismik
- Penyadapan di militer
- Pelacakan inventaris
- Monitoring kesehatan
- Smart Spaces
- Monitoring proses

Karakteristik dari WSN

WSN memiliki karakteristik-karakteristik yang berbeda dengan jaringan-jaringan lainnya seperti jaringan kabel.

Karateristik-karakteristik uniknya adalah :

- Sensor yang berukuran kecil
- Sumber energi yang kecil dan terbatas
- Kondisi lingkungan yang keras
- Mobilitas dari node
- Memiliki topologi jaringan yang dinamis
- Heterogenitas dari node
- Pemasangan jaringan untuk skala besar
- Memiliki communication failure dan node failure

Node sensor dapat diibaratkan sebagai sebuah komputer kecil, yang mempunyai interface dan komponen yang sangat minimalis (bisa jadi hanya sebuah mikrokontroler dan sedikit rangkaian elektronik). Biasanya mengandung unit komputasi, dengan keterbatasan kemampuan komputasi dan jumlah memori, memiliki sensor, perangkat komunikasi dan sumber energi.

Sistem-sistem operasi yang digunakan diantara lain : Bertha, BTnut Nut/OS, Contiki, CORMOS (A Communication Oriented Runtime System fo Sensor Network). eCos, EYESOS, MagnetOS, MANTIS (MultimodAI Networks in-situ Sensors), SenOS, SOS, TinyOS, t-Kernel, dan LiteOS.

Bahasa pemograman yang biasanya digunakan antara lain : C@t (Computation at a point in space), DCL (Distributed Compositional Language), galsC, nesC, Protothreads, SNACK, dan SQTL.

Elliptical Curve Cryptography (ECC)

Elliptical Curve Cryptography adalah sebuah algoritma kunci publik yang perhitungannya didasarkan kepada konsep kurva elips pada area yang terbatas. Penggunaan algoritma ini digagas oleh Neal Koblitz dan Victor S Miller pada tahun 1985.

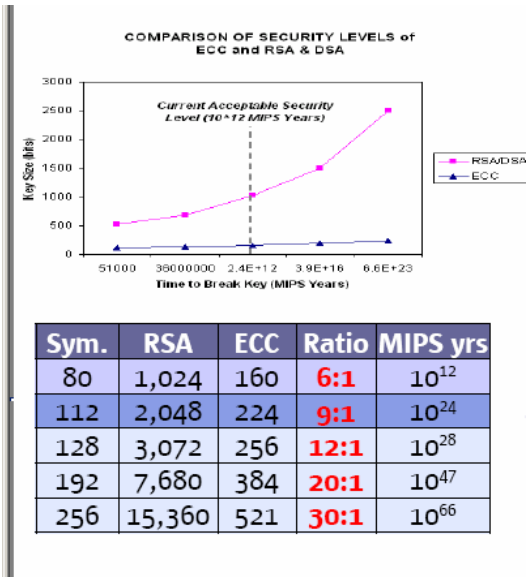
Pengenalan

Konsep dasar dari algoritma kriptografi kunci publik adalah, kesulitan seseorang untuk melakukan perhitungan matematis dalam rangka memecahkan bagaimana suatu bilangan dapat dibentuk tanpa mengetahui beberapa informasi yang dibutuhkan, yang sering kita sebut dengan kunci. Pembuat kunci tetap merahasiakan kunci privat yang terbentuk dan mempublikasikan kunci publiknya. Kemudian dengan kunci-kunci tersebut si pembuat pesan yang asli dapat mengenkripsinya lalu mengirimkannya kepada orang lain, dan hanya si pemilik kuncinyalah yang dapat mendekripsi pesan tersebut.

Pada awalnya kita mengenal algoritma RSA, dimana pada algoritma tersebut dihasilkan dua angka prima yang sangat besar yang digunakan untuk mengenkripsi pesan. Salah satu angkanya digunakan sebagai kunci privat dan satunya lagi digunakan sebagai kunci publik yang dikirimkan bersama pesan yang telah dienkripsi. Kesulitan dalam memfaktorkan suatu bilangan memastikan tidak akan ada satu orangpun mendapatkan kunci privatnya dengan modal kunci publik yang dipublikasikan. Tetapi sayangnya seiring dengan perkembangan kamampuan

komputasi, algoritma kunci publik RSA membutuhkan kunci dengan panjang ratusan bit untuk memastikan keamanannya.

Selain itu ada juga konsep algoritma kunci publik yang dapat digambarkan sebagai berikut :



Sym.	RSA	ECC	Ratio	MIPS yrs
80	1,024	160	6:1	10^{12}
112	2,048	224	9:1	10^{24}
128	3,072	256	12:1	10^{28}
192	7,680	384	20:1	10^{47}
256	15,360	521	30:1	10^{66}

Gambar 1 perbandingan tingkat keamanan antara ECC RSA dan DSA.

Ada suatu persamaan dimana $a^b = c$, permasalahannya adalah mengetahui bilangan b dimana a dan c tidak diketahui. Jika dilihat persamaan ini seharusnya mudah diselesaikan dengan logaritma, tetapi jika area solusi yang ada mempunyai cakupan yang sangat besar, mencari solusinya akan sangat susah. Permasalahan ini biasa disebut permasalahan logaritma diskrit.

Sedangkan algoritma ECC didasarkan pada persamaan kurva yaitu :

$y^2 = x^3 + a\,x + b.$

Jumlah titik pada kurva (semua solusi yang ada pada persamaan dalam bentuk kurva) dapat ditampilkan dalam bentuk grup abelian (dimana titik pada tak hingga ditetapkan sebagai elemen identitas. Jika koordinat x dan y dipilih dari garis kurva pada bagian yang terhingga, maka solusi akan membentuk grup abelian. Permasalahan logaritma diskrit pada suatu persamaan kurva elips akan jauh lebih sulit untuk dipecahkan dibandingkan dengan permasalahan yang didasarkan pada area terhingga. Selain itu kunci yang dihasilkan pada elliptic curve cryptography dapat lebih pendek dibandingkan algoritma yang lainnya dengan tingkat keamanan yang sama.

Sama seperti sistem-sistem kriptografi yang lainnya, tidak ada pembuktian secara matematis yang dapat membuktikan bahwa tingkat kesulitan dari algoritma ECC lebih tinggi dibandingkan algoritma-algoritma lainnya. Walaupun begitu US National Security Agency merekomendasikan ECC sebagai algoritma yang layak dipakai dalam permasalahan keamanan.

Secure Socket Layer / Transport Layer Security

Transport Layer Security (TLS) dan pendahulunya Secure Socket Layer (SSL) adalah sebuah protokol kriptografi yang menyediakan keamanan komunikasi di internet, seperti web browsing, email, internet faxing, dan bentuk-bentuk transfer data lainnya.

Deskripsi

Protokol TLS mengijikan aplikasi-aplikasi client/server saling berkomunikasi dengan mencegah hal-hal seperti eavesdropping, tampering, dan message forgery terjadi. TLS menyediakan autentifikasi end-point, dan privasi komunikasi lewat internet menggunakan kriptografi.

Pada umumnya hanya server yang telah terautentifikasi saja yang dapat berhubungan dengan client yang masih dalam keadaan belum terautentifikasi. Hal ini memastikan bahwa client yakin dengan siapa mereka berkomunikasi. Pada tingkat keamanan yang lebih tinggi, baik client maupun server harus yakin dengan siapa mereka berkomunikasi, oleh karena itu diperlukan proses autentifikasi pada kedua belah pihak tersebut. Keadaan tersebut biasa dikenal dengan mutual authentification. Mutual authentification membutuhkan public key infrastructure (PKI) untuk implementeasinya pada client.

TLS terdiri dari tiga fase :

1. Negosiasis secara peer untuk mengecek dukungan algoritma
2. Enkripsi kunci publik, berdasarkan pertukaran kunci, dan juga autentifikasi berdasarkan sertifikat.

3. Symmetric chiper

Pada fase pertama, client dan server menegosisasikan algoritma kriptografi mana yang akan digunakan. Sampai saat ini dukungan algoritma yang ada adalah :

- Untuk kriptografi kunci publik : RSA, Diffie-Hellman, DSA atau Fortezza.
- Untuk Symmetric Chiper : RC2, RC4, IDEA, DES, Triple DES, AES, atau Camellia.
- Untuk fungsi hash searah : MD2, MD4, MD5 atau SHA

Bagaimana TLS Bekerja

Protokol TLS bekerja dengan cara saling mempertukarkan record. Setiap record sebagian dapat dikompresi, dienkripsi, dan dipaketkan beserta message authentification code (MAC). Setiap record akan mengandung content_type yang akan menentukan tingkatan mana yang dipilih pada protokol yang sedang digunakan.

Ketika koneksi dimulai, record akan membungkus protokol lainnya yaitu protokol handshake dengan content_type 22.

Client kemudian akan mengirimkan dan menerima beberapa struktur handshake sebagai berikut :

- Client akan mengirimkan pesan ClientHello yang menspesifikasikan daftar chiper, metode kompresi, dan versi terbaru dari protokol yang didukung. Selain itu juga mengirimkan byte-byte acak yang akan digunakan kemudian.
- Kemudian akan menerima ServerHello, dimana server akan memilih parameter dari koneksi dari pilihan-pilihan yang ditawarkan oleh client sebelumnya.
- Ketika parameter koneksi telah diketahui, client dan server akan saling mengirimkan sertifikasi (tergantung dari kunci publik chiper yang dipilih). Sertfikasi ini saat sekarang adalah X.509, tetapi ada juga draft yang menspesifikasikan penggunaan sertifikasi berbasiskan OpenPGP.
- Server akan meminta sertifikat dari client, oleh karenanya maka mutually authentification dapat dilakukan.
- Client dan server kemudian menegosiasikan rahasia yang biasa disebut “master secret”, yang kemungkinan menggunakan dari pertukaran hasil dari algoritma Diffie-Hellman, atau menenkripsi dengan sederhana rahasia tersebut dengan kunci publik yang didekripsi oleh kunci private yang didapat secara peer. Semua data nanti didapatkan berdasarkan “master key” yang terbentuk tersebut disertai nilai acak yang dibuat oleh client dan server, yang dikirimkan melalui sebuah desain yang cermat yang disebut dengan fungsi pseudorandom.

TLS atau SSL memiliki bermacam-macam cara dalam melakukan proses keamanannya, yaitu :

- Memberi angka kepada semua record dengan menggunakan angkat yang berurut yang terdapat pada MAC.
- Menggunakan sebuah pesan singkat dengan sebuah kunci (oleh karenanya, hanya dengan kunci seseorang dapat mengetahui MAC). Ini dispesifikasikan pada RFC 2104.
- Proteksi terhadap beberapa serangan yang telah diketahui (termasuk man in the middle attack), dengan memasukkan metode-metode pada protokol-protokol pada versi sebelumnya.
- Pesan yang mengakhiri proses handshake, mengirim nilai has dari semua data yang dipertukarkan oleh kedua belah pihak.
- Fungsi pseudorandom membagi setengah data masukkan dan proses di setiap bagian menggunakan algoritma hash yang berbeda (MD5 atau SHA) kemudian akan mengXOR-kan secara bersama-sama. Hal ini memberi proteksi jika salah satu algoritma berhasil untuk dibobol.

Aplikasi-aplikasi TLS / SSL

TLS berjalan pada layer dibawah protokol aplikasi seperti HTTP, FTP, SMTP, dan NNTP, dan diatas protokol TCP dan UDP,

yang merupakan bagian dari protokol TCP/IP. Selain itu TLS pun dapat menambahkan tingkat keamanan pada protokol apapun yang menggunakan koneksi terpercaya seperti TCP, biasanya diimplementasikan pada HTTP yang kemudian membentuk protokol baru yaitu HTTPS.

HTTPS biasa digunakan untuk World Wide Web yang membutuhkan keamanan seperti halaman electronic commercial (e-commerce). SMTP juga termasuk area dimana TLS berkembang dan ini dispesifikasikan pada RF3207. Aplikasi-aplikasi ini menggunakan sertifikat kunci publik untuk memverifikasi identitas dari setiap end-point.

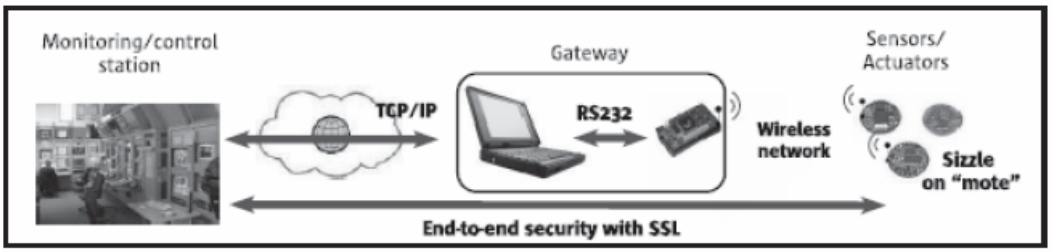
Seiring dengan pertumbuhan jumlah client dan server yang mendukung TLS, tetapi masih kurangnya dukungan untuk penerapan tersebut. Sebagai alaternatif, banyak pengguna mengharapkan dapat menggunakan produk TLS standalone seperti Stunnel. Wrappers seperti Stunnel digunakan agar koneksi yang menggunakan TLS dapat secara seketika dilakukan, dengan menyederhanakan proses koneksi ke port yang tersedia sesuai dengan tujuan secara terpisah. Sebagi contoh, secara default prot TCP untuk HTTPS adalah 443, untuk membedakannya dengan port HTTP pada port 80. Walaupun begitu, pada 1997 Internet Engineering Task Force merekomendasikan setiap aplikasi menggunakan protokol memulai koneksi dengan protokol yang tidak aman kemudian baru ditingkatkan sesuai kebutuhan dibandingkan langsung menawarkan secara langsung koneksi dengan TLS.

TLS juga dapat digunakan untuk melakukan tunneling pada semua jaringan dengan menciptakan VPN, sama juga seperti menggunakan OpenVPN. Banyak vendor kini menggunakan enkripsi dan autentifikasi pada TLS.

Sejarah perkembangan

TLS dikembangkan oleh Netscape, SSL versi 3.0 dikeluarkan pada tahun 1996, yang kemudian dijadikan dasar untuk pengembangan TLS versi 1.0. Standard protokol IETF pertama didefinisikan pada RFC 2246, Visa, Master Card, American Express, dan banyak institusi finansial lainnya menggunakan SSL untuk melakukan transaksi lewat internet.

Pada awal implementasi SSL menggunakan 40 bit kunci simetris karena batasan yang ditetapkan Pemerintahan Amerika Serikat yang tertulis di export of cryptography technology. Pemerintahan Amerika Serikat menilai bahwa panjang kunci 40 bit cukup panjang dan kecil sehingga tidak mudah dibobol dengan metode Brute Force. Setelah beberapa tahun terjadi kontroversi, kebijakan tersebut dihilangkan. Implementasi saat ini lebih banyak menggunakan 128 bit atau lebih panjang lagi untuk kunci simetris.



Gambar 2 Arsitektur berbasis Gateway untuk membuat embedded devices dapat mengakses via internet. Gateway berperan sebagai jembatan antara jaringan TCP/IP yang cepat dengan sekumpulan perangkat sensor yang terkoneksi dengan wireless network yang lebih lambat. Walaupun melalui TCP koneksi berakhir di gateway, yang menggunakan SSL sebagai keamanan dengan konsep end-to-end.

Sizzle

Ketika SSL dilihat sebagai solusi yang baik untuk komunikasi yang aman lewat internet, SSL juga dinilai terlalu berat untuk

diimplementasikan pada perangkat-perangkat kecil karena penggunaan SSL membutuhkan algoritma kunci publik yang dikenal membutuhkan resource process dan memori yang cukup besar.

Sizzle memungkinkan seseorang mengimplementasikan sebuah web server pada sebuah perangkat yang kecil dengan kemampuan terbatas yang dapat digunakan untuk remote monitoring dan kontrol (lihat gambar 2). Konsep gateway antara stasiun monitoring/kontrol dan perangkat yang sedang dimonitor atau dikontrol memiliki beberapa keuntungan, yaitu :

- Gateway berperan sebagai jembatan antara embedded device dan disisi lain dengan internet. Gateway ini melakukan koneksi ke internet dengan jalur berkecepatan tinggi (Ethernet) dan berkomunikasi dengan satu atau lebih embedded device menggunakan jalur nirkabel berkecepatan rendah (seperti IEEE 802.15.4) dan hemat dalam penggunaan tenaga listrik.
- Gateway menyediakan satu titik komando untuk mengontrol semua embedded device. Dapat diimplementasikan pada bermacam-macam mekanisme, termasuk penyaringan berdasarkan alamat, untuk memaksa dilakukan akses yang selektif dari hubungan lewat internet. Gateway juga merupakan tempat yang paling tepat untuk menyimpan log semua interaksi dengan embedded device.
- Gateway dapat bertindak layaknya sebuah peningkat kemampuan proxy. Pada umumnya protokol TCP yang digunakan pada lalu lintas internet (termasuk HTTP dan HTTPS) melakukan interpretasi paket-paket yang hilang sebagai indikasi terjadinya kepadatan pada lalu lintas internet. Hal ini mengakibatkan TCP akan memiliki performa yang sangat jelek ketika koneksi harus berhubungan dengan banyak wireless hop. Salah satu pendekatan yang biasanya untuk menangani permasalahan penurunan performa ini adalah dengan cara dengan cara membagi jalur end-to-end yang terdapat pada jalur-jalur nirkabel. Terminating TCP pada gateway dan menggunakan protokol khusus yang dapat digunakan secara efektif pada embedded device, memiliki beberapa keuntungan diantaranya :
    - Protokol khusus tersebut dapat menyederhanakan proses karena tidak harus mendukung end-to-end congestion control diantara bermacam-macam jalur yang ada.
    - Paket lokal yang dapat diselamatkan meningkatkan performa secara keseluruhan.
    - Dapat disesuaikan untuk menyesuaikan dengan karakteristik kehilangan paket pada suatu jalur.

Ada perbedaan penting antara arsitektur keamanan yang dijelaskan diatas dengan arsitektur lainnya berbasiskan gateway untuk koneksi dengan small device, yang paling terkenal adalah WAP 1.0, dimana gateway mengasumsikan semua lalu lintas data dalam kondisi baik, mendekripsi semua data yang masuk, dan mengenkripsi kembali sebelum data itu dipertukarkan. Sedangkan di arsitektur Sizzle, gateway yang digunakan harus diyakini merupakan gateway yang telah terpercaya, dan menganggap koneksi yang melewatinya adalah koneksi yang legal. Dengan pendekatan ini, keamanan di dukung dengan diterapkannya SSL yang mendukung end-to-end. Semua data tetap dalam bentuk terenkripsi ketika harus melewati gateway, sehingga penyerang tidak dapat melihat atau mengubah isi dari data yang melewati gateway tersebut.

SSL adalah protokol yang elastis yang dapat mendukung banyak algoritma kriptografi dan dapat melibatkan bermacam-macam metode autentifikasi. Sizzle diimplementasikan dengan mengoptimalkan fitur yang disediakan SSL ini. Spesifikasi dari bandwith, memori dan komputasi terlihat seperti dibawah ini :

- Sizzle mengimplementasikan algoritma kriptografi yang relatif kecil. MD5 dan SHA1 adalah algoritma hash yang digunakan untuk mendapatkan master secret dan kunci simteris yang biasanya dipakai pada SSL. Sizzle memilih

mengimplementasikan RC4 untuk enkripsi dikarenakan kecepatannya yang lebih cepat dibanding yang lain, ukuran kode yang kecil, dan cukup dikenal luas digunakan untuk SSL. Sizzle memilih ECDH-ECSA (menggunakan 160 bit curve secp 160r1) untuk pertukaran kunci dikarenakan memiliki efisiensi dalam penggunaan resource dan juga cepat. Selain itu diimplementasikan juga RSA dengan panjang kunci 1024 bit untuk melihat perbandingan performa dan kompatibilitas dengan browser-browser yang ada.

- Paket chiper yang ada di Sizzle tidak membutuhkan untuk mengirimkan pesan ServerKeyExchange, dan server ECC atau RSA mengirimkan kunci publik dalam sertifikat. Hasilnya sertifikat ECC memiliki panjang 222 byte, sedangkan sertifikat RSA memiliki panjang 423 byte.
- Sizzle menggunakan 4 byte session identifier berbeda dengan Apache atau server lainnya yang memiliki kebutuhan skalabilitas yang lebih luas yang menggunakan 32 byte untuk session identifier. Walaupun ada keterbatasan memori Sizzle harus memastikan untuk dapat menyimpan informasi beberapa koneksi pada session cache, dan 4 byte dianggap cukup untuk mengidentifkasi setiap cache session secara unik. Disamping mengimplementasikan session reuse, Sizzle juga mengimplementasikan HTTPS yang diharapkan dapat mengurangi overhead dari kriptografi kunci publik.
- Sizzle tidak mengirimkan pesan CertificateRequest yang dimana secara otomatis akan menghilangkan pesan Certificate/CertificateVerify yang membuat kebutuhan untuk memparsing kode setiap sertifikat yang dilakukan di Sizzle. Client tetap dapat diautentifikasi dengan menggunakan password melalui SSL seperti yang dilakukan pada online banking atau e-commerce.
- Informasi statis seperti kunci private server, sertfikat yang berhubungan, dan informasi statis lainnya berhubungan dengan aplikasi yang spesifik pada halaman web disimpan di program memori, karena akan mengurangi penggunaan memori dibandingkan kalau disimpan di data memori. Baik Atmel Atmega dan MSP430 memungkinkan pemogram menspesifikasikan bagian memori yang sensitif dan menolak akses pembacaan.
- Sizzle hanya mendukung satu satus koneksi SSL dalam satu waktu. Jika ada sebuah koneksi baru berusaha untuk melakukan koneksi sedangkan koneksi sebelumnya masih dalam keadaan terbuka, tetapi dalam keadaan idle, maka koneksi yang terdahulu harus ditutup terlebih dahulu baru kemudian koneksi yang baru dilakukan.
- Diterapkan pula metode untuk melakukan penghematan penggunaan memori dengan cara mengurangi ukuran dari HTTP header. Tidak seperti fitur-fitur lainnya yang disebutkan diatas, fitur ini diimplementasikan di web browser bukan di server Sizzle. Contohnya browser Mozzila, mengirimkan lebih dari 400 bytes untuk header untuk setiap permintaan HTTP. Hal ini memperlihatkan terjadinya overhead yang sangat besar untuk komunikasi jika hal ini diterapkan di Sizzle. Oleh karena itu diterapkan satu fitur di Mozilla pada Networking Configuration Panel untuk mendukung pengiriman header tertentu seperti Accept, Accept-Language, Accept-Encoding, dan lain-lain akan diabaikan oleh Sizzle. Penggunaan fitur ini dapat mengurangi penggunaan untuk header menjadi sekitar 100 byte.

Gambar 3 Perangkat Wireless yang telah diimplementasikan ECC (a) thermostat (b) jam pemonitor kesehatan (c) alat pembaca kartu

Sizzle telah dicoba untuk diimplementasikan pada device untuk sensor dan web browser digunakan untuk pemonitoran dan pengontrolan aplikasi. Banyak device dapat dikoneksikan pada sebuah gateway, dan pada secure web server dimana setiap device dikoneksika kepada port TCP yang berbeda di gateway.

Protokol SSL didesain untuk mendukung layanan byte-stream dua arah, yang biasanya didukung oleh TCP. Tetapi sayangnya, TinyOS yang digunakan dalam Sizzle tidak mendukung mekanisme transfer data tersebut, sehingga TinyOS dibuat sedemikian rupa agar mendukung mekanisme transfer data tersebut.

Prototipe pertama penggunaan mekanisme transfer data adalah dengan menggunakan perintah stop-and-go pada protokol berbasiskan ACK yang sederhana. Walaupun begitu, karena TinyOS menggunakan paket yang berukuran tetap, mengirim ACK untuk setiap paket data yang dikirimkan akan mengurangi performa hanya menjadi setengah dari efektifitas keluaran sebelumnya. Oleh karena itu dikembangkan sebuah prototipe yang mengirimkan ACK hanya pada blok pertama dan terakhir saja dari paket data yang dikirimkan. Paket-paket diantaranya menggunakan skema berbasis NACK dimana hanya pakey yang hilang saja yang secara eksplisit diberi sinyal. Karena penggunaan metode ini, maka dibutuhkan tiga paket sebagai kontrol yaitu : (i) NEWCONNECTION yang mengindikasikan bahwa gateway telah menerima koneksi TCP (ii) DISCONNECT yang memberitahukan gateway untuk memutuskan koneksi (iii) READY yang mengindikasikan bahwa perangkat siap untuk menerima pesan baru dari gateway. Semua tiga paket kontrol ini dikirim secara eksplisit pada ACK.

Pesan aplikasi dipecah-pecah kedalam paket-paket TinyOS dan dikirim melalui jalur nirkabel tanpa harus menggunakan TCP atau IP header. Setiap paket TinyOS menggunakan ukuran 28 byte, tetapi dengan skema arsitektur diatas, digunakan 6 byte dari tiap paket tersebut untuk menyimpan berbagai informasi seperti alamat balikan dan angka-angka yang berurut untuk penomoran. Pada sebuah contoh 286 byte record yang mengandung pesan-pesan ServerHello, Certificate, dan ServerHelloDone dipecah-pecah menjadi 13 (286/22) paket TinyOS. Walaupun begitu ketika record tersebut dikirim lewat jalur nirkabel, record juga terkirim beserta 20 byte TCP header dan 20 byte IP header.

Pengujian Sizzle

Pengujian sizzle dilakukan dengan TinyOS 1.1.10.

Penggunaan Memori

Dalam menentukan berapa besar penggunaan memorid pada berbagai perangkat yang menggunakan Sizzle diterapkan dengan menggunakan objdump. Lamanya run time ditentukan dengan mengisi memori yang dialokasikan untuk stack program dengan byte-byte yang memiliki nilai tertentu dan kemudian dicek seberapa banyak overwrite terjadi selama program dieksekusi.

	FLASH	RAM (static)	RAM (max. stack size)	
			w/ ECC	w/ RSA
Mica2dot	48,882	3,094	656	950
Mica2	49,116	3,094	656	950
MicaZ	48,516	3,040	629	925
TelosB	40,988	2,780	651	853

Gambar 4 Memori yang digunakan oleh termostat wireless yang menggunakan Sizzle. Total RAM yang digunakan dengan RSA berjalan mendekati batas 4096 byte pada perangkat-perangkat Mica, membuat ECC superior sebagai aternatif.

Dari gambar 4 menunjukan pengukuran pada perangkat termostat. Angka-angkat tersebut telah termasuk jumlah resource yang digunakan oleh TinyOS, oleh layer communication, algoritma kriptografi, SSL dan protokol HTTP, dan beberapa aplikasi spesifik. Aplikasi spesifik yang dijalankan secara kasar dapat dikatakan sebesar 8 Kb (tidak tergantung dimana aplikasi itu di implementasikan). Terlihat bahwa masih banyak ruang dalam memori Flash untuk mengimplementasikan aplikasi-aplikasi yang lebih kompleks.

Prosesor MSP430 mempunyai kemampuan untuk beroperasi pada mode 16 bit dibandingkan Atmel Atmega yang hanya dapat beroperasi pada mode 8 bit saja, menghasilkan kode kriptografi yang ukurannya lebih kecil, tetapi kapasitas maksimal memori yang digunakan untuk menyimpan kode pada perangkat-perangkat berbasis MSP lebih kecil dibandingkan perangkat-perangkat yang berbasiskan Atmel Atmega. Persentase total penggunaan memori flash untuk kriptografi sekitar 15%(17-20Kb) untuk Atmel dan sekitar 35%(12-14Kb) untuk MSP.

	Atmel ATmega (128KB max)	MSP430 (40KB max)
RSA	6,444 + 1,349	3,180 + 1,350
ECC	5,348 + 268	3,328 + 268
SHA1	2,046	1,486
MD5	2,442	1,742
RC4	228	216
SSL	7,556	5,848

Gambar 5 Penggunaan memori flash untuk kode kriptografi pada Atmel Atmega dan MSP. Angka setelah tanda plus adalah penggunaan memori tambahan yang digunakan untuk menyimpan sertifikat, kunci dan variabel konstan lainnya.

Handshake, Kecepatan Transfer Data

Pengujian kecepatan dilakukan dengan memisahkan kecepatan handshake pada SSL dengan transfer data, karena adanya perbedaan ukuran data untuk aplikasi-aplikasi spesifik yang dijalankan.

Ada empat buat counter yang terdapat di Atmel Atmega, sedangkan hanya ada dua buah counter yang terdapat di MSP430. Digunakan angka hasil count untuk menghitung putaran CPU yang dihasilkan dari proses komputasi terhadap algoritma kriptografi.

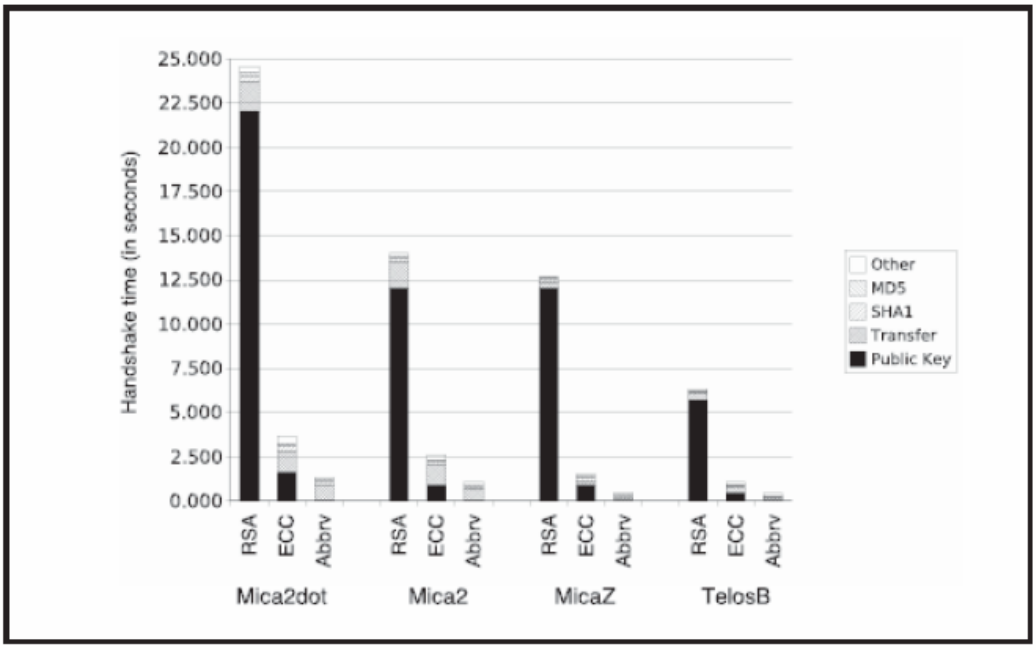
Waktu transmisi dari setiap pesan yang terkirim dihitungan dengan cara mengehitung delay yang terjadi antara pengiriman/penerimaan paket pertama dengan pengiriman/penerimaan acknowledge terakhir. Delay untuk semua pesan dijumlahkan untuk menentukan lama transfer data secara keseluruhan. Total waktu yang digunakan untuk handshake atau transfer data juga dihitung di gateway dengan mengabaikan delay antara pengiriman paket pertama pada ClientHello atau permintaan HTTP dan penerimaan pesan pada paket terakhir atau pesan Finished.

Hasil perhitungan waktu dapat dilihat pada gambar 6. Seperti yang diharapkan, RSA-based Handshake adalah yang terlambat dan abbreviated handshake adalah yang tercepat.

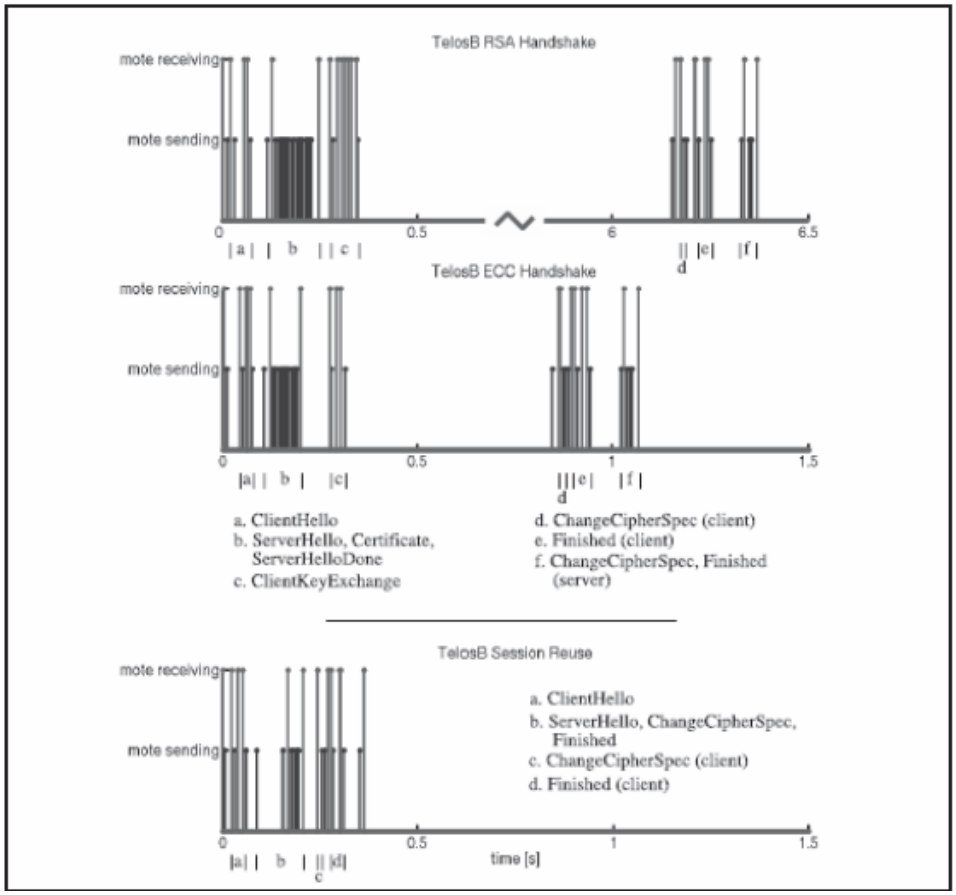
RSA-based Handshake memerlukan waktu sekitar 6– 25 detik tergantung dari platform yang digunakan pada perangkat. Sedangkan ECC-based handshake memerlukan waktu sekitar 1-4 detik, dan abbreviated handshake memerlukan waktu sekitar 0,5-1,5 detik.

Dari grafik pada gambar 6 juga terlihat efek dari penambahan kecepatan CPU dan wireless network pada waktu handshake. Contohnya, pada RSA handshake mendapatkan keuntungan paling besar pada peningkatan kecepatan CPU, sedangkan Abbreviated handshake mendapatkan keuntungan paling besar ketika terjadi peningkatan performa wireless network. Lain yang terjadi pada ECC handshake, peningkatan performa pada dua bagian tersebut keduanya membuat turunnya waktu yang dibutuhkan untuk handshake.

Pada gambar 7 kita dapat melihat lebih detail lagi perbandingan antara tiga tipe handshake tersebut. Grafik tersebut diambil dari perangkat yang menggunkan platform Telos, tetapi perangkat-perangkat lainnya yang menggunakan platform yang berbeda memiliki karakteristik yang sama dengan Telos.



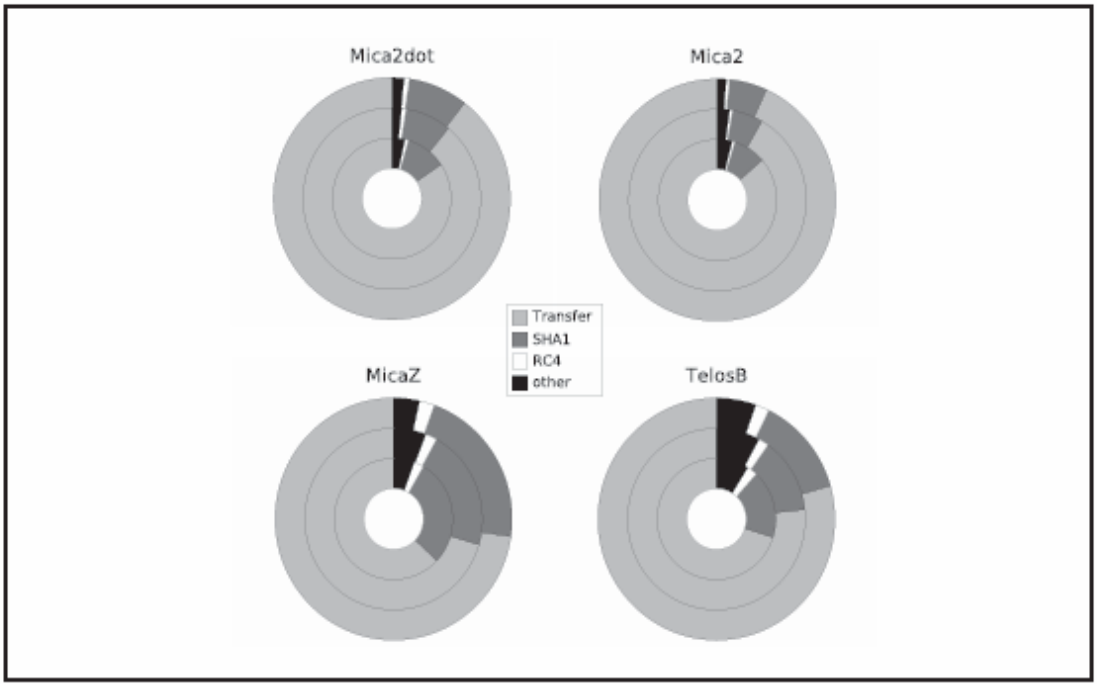
Gambar 6 Waktu yang dibutuhkan untuk melakukan tiga jenis SSL Handshake (i) RSAI-based full handshake (ii) ECC-based full handshake (iii) abbreviated handshake pada setiap perangkat.



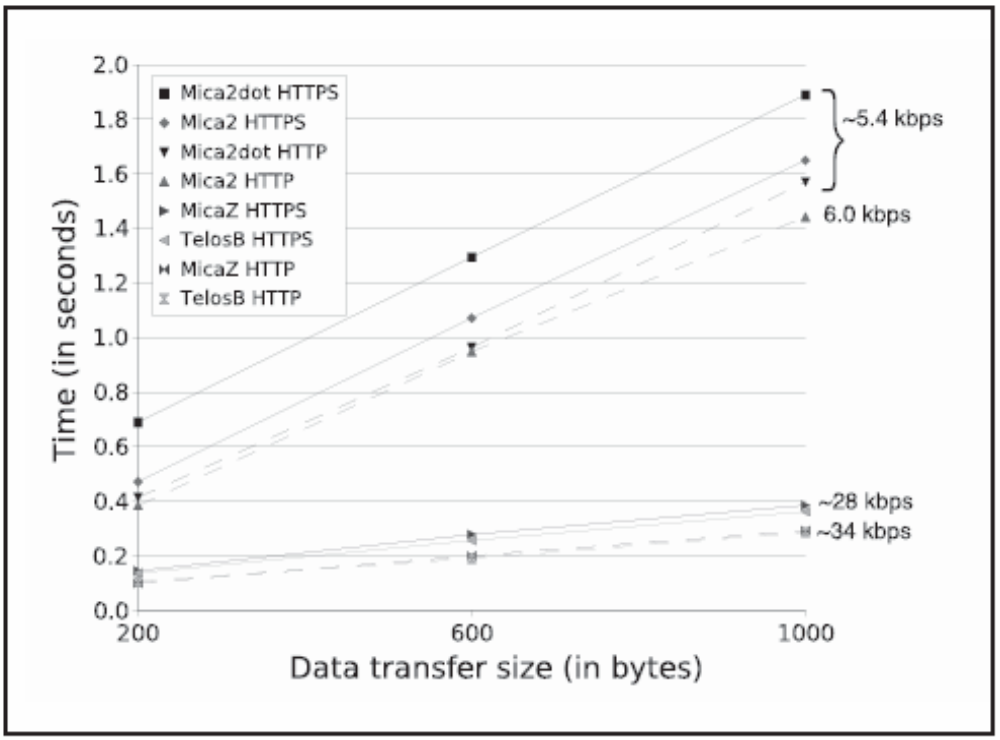
Gambar 7 Pertukaran paket-paket padaTinyOS untuk karateristik SSL handshake yang berbeda, operasi kunci publik.

Kemudian dilakukan juga pengumpulan data yang ditransferkan menggunakan HTTP dengan dan tanpa SSL, untuk tiga jenis ukuran halaman web, yaitu : 100, 500, dan 900 byte. Karena browser telah diset untuk mengurangi ukuran dari HTTP header, maka total data yang dikirim adalah 200, 600, dan 1000 byte.

Dari gambar 8 ditunjukan bahwa ketika menggunakan RC4 dan SHA1, transfer rate lebih dibatasi oleh proses komunikasi dibandingkan dengan kecepatan komputasi. Grafik pie tersebut juga mengindikasikan bahwa RC4 secara signifikan lebih cepat dibandingkan dengan SHA1dan karena ada overhead, ketika initial setup, untuk kedua algoritma menyebabkan kedua algoritma tersebut mempunyai waktu transfer lebih tinggi untuk data aplikasi yang berukuran lebih kecil.



Gambar 8 Grafik Pie ini menunjukan bagaimana waktu dialokasikan untuk mengirimkan bermacam-macam ukuran data aplikasi melewati SSL yang menggunakan jalur terproteksi menggunakan RC4 untuk enkripsi dan SHA1 untuk integritas data. Dari dalam ke luar lingkaran secara berurutan, data yang ditransferkan adalah 200, 600, dan 1000 byte.



Gambar 9 Waktu yang dibutuhkan untuk komunikasi dan memproses HTTP request dan response sebagai fungsi terhadap ukuran data yang digunakan.

Pada gambar 9 dapat kita lihat waktu yang dibutuhkan mentransfer bermacam-macam ukuran aplikasi data menggunakan dan tanpa menggunakan SSL. Ternyata kita mendapatkan fungsi linear antara ukuran terhadap waktu, sedangkan gradien dari persamaan garis tersebut menunjukkan efektifitas dari transmission rate.

Proses tambahan dibutuhkan untuk SSL menghasilkan penurunan efektifitas sebesar 10% pada efektifitas data rate untuk Mica2 dan Mica2Dot (dari 6 ke 5,4 Kbit/s) dan penurunan sekitar 20% untuk MicaZ dan Telos (dari 34 ke 28 Kbits/s).

Aplikasi-aplikasi lainnya pada umumnya yang terdapat pada perangkat yang diuji hanya mentransfer data dalam ukuran yang sangat kecil pada waktu tertentu, dan kondisi tersebut menyebabkan delay untuk penggunaan SSL tetapi dapat diabaikan (30-300 ms).

Kemungkinan Perbaikan

Transmisi data mengkonsumsi banyak energi pada perangkat dibandingkan dengan operasi komputasi. Sebagai contoh, mentrasmisikan 1 bit data pada Mica2 mengkonsumsi energi sama seperti CPU yang melakukan 2000 cycle. Untuk menurunkan konsumsi energi yang dapat diasumsikan karena mekanisme handshake pada SSL, maka dapat dibuat sebuah protokol yang mengurangi jumlah data yang ditransmisikan antara wireless hop.

Kesimpulan

Sizzle untuk pertama kalinya membawa fungsionalitas SSL pada internet untuk dapat diimplemetasikan pada sebuah perangkat yang memiliki keterbatasan dalam komputasi, memori dan sumber daya.

Gambar 10 Sizzle Secure Web Server Terkecil di dunia

Sizzle mengoptimasi kriptografi kunci publik yang digunakan dengan menggunakan kunci publik yang lebih pendek tetapi tanpa harus mengurangi efektifitas sistem keamanan end-to-end. Untuk saat ini Sizzle dapat dijalankan pada Mica2Dot yang merupakan secure web server terkecil di dunia, baik dalam bentuk fisik maupun tingkat kehematan dalam menggunakan resourcenya. Sekarang dimungkinkan bagi kita untuk memasang secure web server pada perangkat-perangkat mini yang biasa kita gunakan dalam kehidupan sehari-hari yang digunakan unuk melakukan pengukuran atau otomatisasi.

Tetapi dari ide-ide revolusioner tersebut Sizzle masih memiliki beberapa isu yang harus diselesaikan, diantaranya :

- Jamming, karena komunikasi antara gateway dan node-node menggunakan jaringan wireless, maka kemungkinan penyerangan dengan penghancuran sinyal di udara dapat membuat semua sensor yang dipasang tidak dapat berkomunikasi dengan gateway. Walaupun dapat digunakan sebuah frekuensi rahasia yang digunakan antara gateway dan sensor, sehingga pihak luar tidak dapat menginterupsi komunikasi karena tidak mengetahui pada frekuensi berapa Sizzle berkomunikasi.
- Keamanan layanan infrastruktur (routing, lokalisasi, sinkronisasi waktu). Untuk suatu node yang jauh dari jangkauan gateway, untuk dapat berkomunikasi dibutuhkan sebuah route agar informasi sampat di gateway. Masalahnya belum ada mekanisme yang aman antara stasiun-stasiun yang melakukan rebroadcasting untuk melakukan komunikasi. Karena tidak dibutuhkan kunci pada saat terjadinya routing tersebut.

Daftar Pustaka

[1] Deng, Jing. Han, Richard. Mishra, Shivakant. “Secure Code Distribution in Dynamically Programmable Wireless Sensor Networks”. Department of Computer Science, University of Colorado at Boulder. 2005.

[2] Gupta Vipul, Wurm Michael, Zhu, Yu. Millard Matthew. Fung, Stephen. Gura, Nils. Eberle, Hans. Chang Shantz, Sheueuling. “Sizzle : A Standard-based end-to-end security architecture for the imbedded internet”. Sun Microsystem Laboratories, 2005.

[3] Gupta Vipul, Wurm, Micahel. Zhu, Yu. Millard, Matthew. Stephen, Fung. Nils, Gura. Eberle, Hans. Chang Shantz, Sheueuling. “Sizzle : SSL on Motes”. Sun Microsystem Laboratories, 2005.

[4] Government Information Technology Executive Council (GITEC), Washington USA. “Government Information Technology Issues 2006 A View to the Future”. http://www.gitec.org.

[5] Piotrowski, Krzysztof. Langendoerfer, Peter. Peter, Steffen. “How Public Key Cryptography Influences Wireless Sensor Node Lifetime”. Im Technologiepark 25, Frankfurt Germany. 2006.

[6] Sastry, Naveen. “A Critical Look at Sensor Network Security”. Barkeley 2005.

[7] Sun Microsystem Whitepaper. “Extending Internet Connectivity to Smart Dust”. Sun Microsystem Inc, http://research.sun.com/projects/crypto diambil pada 15 Desember 2005.

[8] Sun Microsystem Whitepaper. “Elliptic Curve Cryptography, The Next Generation of Internet Security”. Sun Microsystem Inc, http://research.sun.com/projects/crypto diambil pada 15 Desember 2006.

[9] Vanstone, Scott. “Deployment of Elliptic Curve Cryptography”. University of Waterloo, 2005.

[10] Wang, Haodong. Sheng, Bo. Li, Qun. “Elliptic Curve Cryptography-Based Access Control in Sensor Networks”. Department of Computer Science, College of William and Mary, Williamburg USA.

[11] Zeneida, Benenson. Freiling, Felix. Hammerschimidt, Ernest. Lucks, Stefan. Pimeinidis, Lexi. “Authenticated Query Flooding in Sensor Networks”. Department of Computer Science, Aachen University Germany.

[12] Zhao, Meiyuan. “Performance Evaluation of Distributed Security Protocols Using Discrete Event Simulation”. Department of Computer Science, Darmoute College, Hanover, New Hampshire. 2005.

[13] Zinaida, Benenson. Gedicke, Nils. Raivio, Ossi. “Realizing Robust User Authentification in Sensor Networks”. Department of Computer Science, Aachen University Germany. 2005.