Studi Penerapan Kriptografi pada *Mobile Commerce*

Krisantus Sembiring – NIM : 13503121

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13121@students.if.itb.ac.id

Abstrak

Aplikasi e-*commerce* semakin banyak diimplementasikan pada perangkat *mobile* seperti telepon seluler. Aplikasi ini banyak dikembangkan dengan memanfaatkan berbagai *platform* seperti Wap, i-mode dan Java ME. Pada aplikasi seperti ini, masalah keamanan yang muncul meliputi bagaimana menjamin keamanan otentikasi, *non-repudiation*, kerahasian serta integritas data transaksi antara *mobile device* dan web *server* dan juga keamanan data yang tersimpan pada *perangkat mobile*. Untuk mengatasi masalah ini telah dikembangkan solusi keamanan yang memanfaatkan kriptografi seperti *Wireless Public Key Infrastructure* (PKI), *Wireless Transport Layer Security* (WTLS), penggunaaan kriptografi pada level aplikasi atau pada perangkat keras. Oleh karena itu pada makalah ini akan dibahas mengenai permasalahan keamanan yang ada pada *mobile commerce* dan pemanfaatan kriptografi sebagai solusi untuk permasalahan tersebut pada platform yang umum digunakan untuk mengembangkan aplikasi *mobile commerce*. Pembahasan ini akan meliputi kajian bagaimana solusi tersebut dapat menjamin keamanan pada aplikasi *mobile commerce*, bagaimana penggunaannya dan kelemahannya serta usulan yang dapat digunakan untuk mengatasinya.

***Kata kunci**: mobile e-commerce*, WAP, JAVA ME, WTLS, *Wireless PKI, digital signature*

1. Pendahuluan

E-commerce merupakan bisnis proses yang dijalankan melalui melalui internet, misalnya transaksi jual-beli barang dan jasa secara *online*. Bisnis proses ini mungkin dalam bentuk B2B (*Business to Business*) maupun B2C (*Business to Customer*). Defenisi umum ini, tidak membatasi jenis alat yang digunakan oleh *end user* untuk memperoleh akses ke internet. Oleh karena itu, *mobile commerce* merupakan bagian dari *e-commerce*, yang memanfaatkan perangkat *mobile* atau terminal untuk melakukan transaksi bisnis melalui jaringan telekomunikasi *mobile*. Transaksi bisnis ini tidak hanya terbatas pada layanan yang hanya melibatkan komunikasi, transaksi dan hiburan, tetapi juga memungkinkan terjadinya transfer uang. Selain itu, perangkat *mobile* yang digunakan juga tidak terbatas pada telepon seluler, penggunaan penrangkat lain seperti PDA, juga termasuk dalam *mobile commerce.*

Jadi *mobile commerce* merupakan semua transaksi (yang memiliki niai uang) baik secara langsung maupun tidak langsung melalui jaringan komunikasi nirkabel (Stuart J. Barnes).

Pada umumnya jumlah pengguna telepon seluler di berbagai negara lebih banyak dari pada jumlah pengguna internet dengan pertumbuhan yang sangat cepat khusunya di negara berkembang. Dengan semakin tingginya penetrasi telepon seluler dan perangkat mobile lainnya maka lebih banyak jumlah calon pelanggan yang dapat dijangkau. Selain itu, dengan sifat perangkat yang *mobile* maka aplikasi *mobile* dapat digunakan kapan dan dimana pun. Oleh karena itu, layanan *m-commerce* semakin banyak dikembangkan karena memiliki potensi yang sangat besar terutama pada pasar B2C.

Contoh aplikasi telah dikembangkan adalah *mobile banking*, aplikasi untuk transaksi saham, pelelangan barang dan lain sebagainya. Akan tetapi, kesuksesan dari aplikasi ini sangat bergantung pada jaminan keamanan yang juga bergantung pada teknologi yang dimanfaatkan. Hal ini penting, karena umumnya pengguna tidak akan mau menggunakan aplikasi *mobile commerce* sebelum yakin aplikasi tersebut benar-benar aman.

Aspek keamanan pada aplikasi *mobile commerce* meliputi keamanan jaringan telekomukasi nirkabel sebagai media transmisi data, sistem transaksi yang digunakan (seperti sistem

pembayaran), serta keamanan data yang disimpan pada perangkat *mobile*. Untuk itu, kriptografi dan aplikasinya telah digunakan pada berbagai *platform* aplikasi *mobile* karena menyediakan berbagai layanan keamanan yang akan dijelaskan.

Contoh penerapan kriptografi pada aplikasi *mobile commerce* adalah *Wireless Public Key Infrastructure* (WPKI), penggunaaan kriptografi pada level aplikasi atau penerapan kriptografi pada perangkat keras. *Platform* yang banyak digunakan adalah WAP dan Java ME baik melalui teknologi seperti *Circuit Switched Data (CSD)* atau *General Packet Radio Service (GPRS).* Oleh karena itu, pada makalah ini akan dibahas penerapan kriptografi dalam menjamin keamanan aplikasi *mobile commerce*.

2. Ancaman Keamanan *Mobile commerce*

Terdapat beberapa keamanan aspek keamanan pada aplikasi *mobile commerce* yaitu:

a. Kurangnya kesadaran pengguna akan resiko keamanan. Biasanya pengguna tidak begitu peduli mengenai aksinya misalnya tidak mengunakan pin pada telepon seluler.

b. Pencurian informasi personal yang mungkin tidak disadari pengguna. Hal ini dapat terjadi melalui pencurian perangkat *mobile*, *bluetooth hacking*, *network sniffing* dan sebagainya.

c. Kurangnya kemampuan komputasi pada perangkat *mobile*. Hal ini mengakibatkan terbatasnya penerapan kriptografi yang dapat digunakan.

d. Adanya virus dan aplikasi yang tidak jelas sumbernya dan mungkin berbahaya .

e. Transmisi nirkabel. Hal ini mengakibatkan sinyal transmisi dapat ditangkap oleh siapapun dengan bebas. Seorang penyadap dapat menggunakan radio penerima yang dapat mengintersepsi aliran data dari telepon seluler ke BTS. Penyadap dapat mencoba menginterpretasi data, memodifikasi pesan, menahan pesan tau bahkan mencegah pesan sampai kepada pengguna

f. Serangan *man-in-the-middle-attack*, dimana penyerang dapat mengintersepsi komunikasi antar dua pihak kemudian ”menyerupai” salah satu pihak dengan cara bersikap seolah-olah ia adalah salah satu pihak yang berkomunikasi (pihak yang lainnya tidak menyadari kalau dia berkomunikasi dengan pihak yang salah). Serangan ini bisa saja dilakukan oleh penyerangn yang memiliki akses ke *base station*.

g. Penyerang yang dapat mengintersepsi komunikasi, dapat saja menyimpan pesan tertentu kemudian mengirimkannya kembali di lain waktu. Hal ini memungkinkan penyerang untuk mencoba meyakinkan server bahwa dia adalah seorang pengguna yang terotentikasi.

3. Aspek Keamanan *Mobile commerce*

Hal yang penting pada aplikasi *mobile commerce* (dengan arsitektur umum *client-server*) dari sudut pandang keamanan adalah sebagai berikut:

a. Aplikasi menyediakan layanan yang berdasarkan pendaftaran (*subscription*). Untuk mendaftar user dapat memberikan informasi seperti nomor kartu kredit atau informasi lainnya. Aplikasi harus melindungi informasi ini karena akan di proses di *client* dan akhirnya sampai ke *server*. Jika informasi pelanggan tersebut memenuhi persyaratan yang dibutuhkan maka pelanggan akan terdaftar dan informasi tersebut akan tersimpan pada basis data di *server* yang juga harus dilindungi keamanannya.

b. Pengguna yang sudah terdaftar dapat meminta informasi atau permintaan tertentu dari layanan yang didaftarnya. Oleh karena itu pengguna membutuhkan cara tertentu untuk membuktikan indentitasnya kepada *server* sehingga dapat diverifikasi dengan menggunakan informasi yang tersimpan pada basis data di *server*.

c. Permintaan informasi dari *client* dan respon dari *server* mungkin saja mengandung informasi yang sifatnya sensitif. Contohnya jika seorang pengusaha melakukan transaksi bisnis yang menyangkut rahasia perusahaan dengan memanfaatkan *server*, maka pengusaha tersebut tidak mau orang lain mengetahui isi pesan selama transaksi

dilakukan. Dengan kata lain kerahasian informasi harus dijaga.

Aspek keamanan pada aplikasi *mobile commerce* dengan berbagai ancaman keamanan seperti yang telah dijelaskan diatas dapat diatasi dengan kriptografi karena kriptografi menyediakan beberapa aspek keamanan berikut :

3.1. Kerahasiaan (*confidentiality*)

Layanan yang digunakan untuk menjaga isi pesan dari siapapun yang tidak berhak untuk membacanya. Layanan ini umumnya direalisasikan dengan cara mengenkripsi pesan menjadi bentuk yang tidak dapat dimengerti.

3.2. Integritas Data (*data integrity*)

Layanan yang menjamin bahwa pesan masih asli/utuh atau belum pernah dimanipulasi selama pengiriman. Dengan kata lain aspek keamanan ini dapat diungkapkan sebagai pertanyaan : "Apakah pesan yang diterima masih asli atau tidak mengalami perubahan (modifikasi)?". Istilah lain yang serupa dengan *data integrity* adalah otentikasi pesan (*message authentication*). Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubstitusian data lain kedalam pesan yang sebenarnya.

3.3. Otentikasi (*authentication*)

Otentikasi adalah layanan yang untuk mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication*) dan untuk mengidentifikasi kebenaran sumber pesan (*data origin authentication*). Dua pihak yang saling berkomunikasi harus dapat mengotentikasi satu sama lain sehingga ia dapat memastikan sumber pesan. Pesan yang dikirim melalui saluran komunikasi juga harus diotentikasi satu sama lain sehingga ia dapat memastikan sumber pesan. Dengan kata lain, aspek keamanan ini dapat diungkapkan sebagai pertanyaan : "Apakah pesan yang diterima benar-benar berasal dari pengirim yang benar? ".

Otentikasi sumber pesan secara implisit juga memberikan kepastian integritas data, sebab jika pesan telah dimodifikasi berarti sumber pesan sudah tidak benar. Oleh karena itu layanan integritas data selalu dikombinasikan dengan layanan otentikasi sumber pesan.

3.4. Nirpenyangkalan (*Nonrepudiation)*

Nirpenyangkalan adalah layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan, yaitu pengirim pesan menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan.

4. Kriptografi dan aplikasinya

Kriptografi adalah ilmu dan seni untuk menjaga kerahasian pesan dengan cara mengenkripsinya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya [MUN06]. Berikut ini adalah teknik kriptografi dan aplikasinya yang dapat digunakan dalam menjamin keamanan pada *mobile commerce*:

4.1. Teknik kriptografi

Kriptografi dapat menyediakan aspek keamanan kerahasian, integritas, otentikasi dan nirpenyangkalan. Salah satu teknik kriptografi yang sudah lama dikenal adalah kriptografi kunci simetri yang menggunakan kunci rahasia yang sama untuk mengenkripsi dan mendekripsi pesan. Dengan demikian, dua pihak yang saling berkomunikasi harus saling mempercayai dan merahasiakan kunci rahasia yang digunakan. Hal ini mengakibatkan timbulnya permasalahan bagaimana cara mendistribusikan kunci. Contoh algoritmanya RC4, RC5 untuk *stream cipher* dan AES, IDEA untuk *block cipher*.

Untuk mengatasi permasalahan distribusi kunci tersebut berkembanglah teknik kriptografi kunci publik yang memungkinkan pengguna berkomunikasi secara aman tanpa perlu berbagi kunci rahasia. Pada teknik ini digunakan dua buah kunci yaitu kunci publik dan kunci privat. Kunci publik tidak rahasia dan digunakan untuk mengenkripsi pesan, sedangkan kunci privat bersifat rahasia dan digunakan untuk mendekripsi pesan.

4.2. Pertukaran kunci

Salah satu aplikasi kriptografi kunci publik adalah pertukaran kunci simetri (*session keys*). Contoh algoritmanya adalah RSA dan Diffie-Hellman. Contoh protokol pertukaran kunci dengan menggunakan algoritma Diffie-Hellman adalah sebagai berikut:

a. Pelanggan memilih bilangan bulat acak yang besar x, dan mengirim hasil perhitungannya kepada agen perjalanan *online*:

$$X = g^x \bmod n$$

b. Agen perjalanan *online* menghitung bilangan bulat acak yang besar y dan mengirim hasil perhitungan berikut kepada pelanggan:

$$Y = g^y \bmod n$$

c. Pelanggan menghitung

$$K = Y^x \bmod n$$

d. Agen perjalanan *online* menghitung:

$$K = X^y \bmod n$$

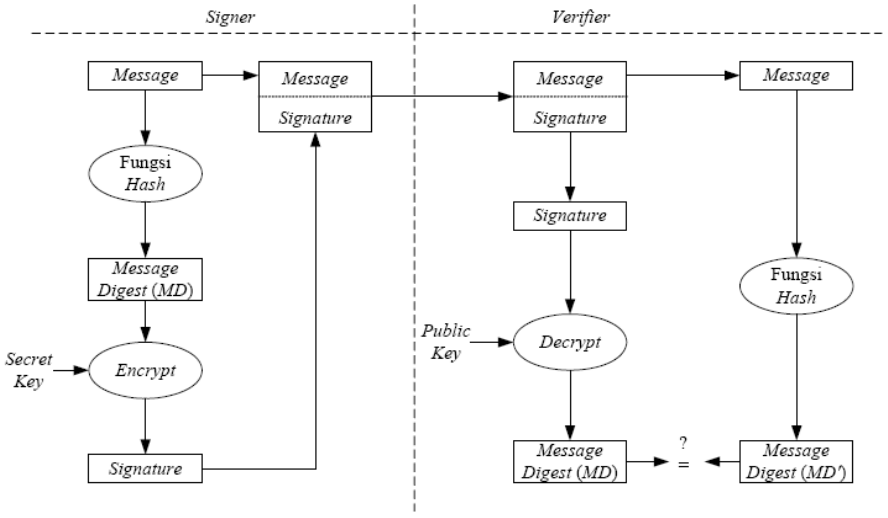
4.3. Tanda Tangan Dijital (*Digital Signature*)

Aplikasi lain dari kriptografi kunci publik, yang dapat memberikan aspek keamanan yang telah dijelaskan sebelumnya adalah tanda tangan dijital. Penandatanganan pesan dapat dilakukan dengan dua cara, yaitu dengan mengenkripsi pesan atau dengan cara menggunakan fungsi *hash* dan kriptografi kunci publik. Penandatanganan pesan dengan cara mengenkripsinya menggunakan kriptografi kunci publik dapat memberikan fungsi kerahasian pesan, otentikasi, dan nirpenyangkalan. Kerahasian pesan tentunya terjamin karena setelah dienkripsi pesan tidak dapat diketahui maknanya. Agar dapat menjamin otentikasi maka pesan dienkripsi dengan kunci privat karena sifatnya rahasia. Dengan demikian, kebenaran pihak-pihak yang berkomunikasi dan kebenaran sumber pesan dapat dijamin.

Agar tanda-tangan dijital dapat memberikan fungsi integritas data maka tanda tangan dijital memanfaatkan fungsi *hash*. Fungsi *hash* adalah fungsi yang menerima masukan *string* dan menghasilkan *string* keluaran yang disebut dengan nilai *hash*. Jika *string* masukan dari fungsi *hash* diubah makan akan dihasilkan nilai *hash* yang berbeda. Dengan demikian fungsi integritas pesan dapat diberikan oleh fungsi *hash*. Skema pemberian tanda tangan dijital dengan fungsi *hash* dapat dilihat pada gambar 1.

Langkah-langkah pemberian tanda tangan dijital adalah sebagai berikut:

a. Pesan yang diubah terlebih dahulu menjadi *message digest MD*.
b. *Message digest MD* dienkripsikan dengan algoritma kunci publik menggunakan kunci rahasia (*SK*) pengirim menjadi tanda tangan dijital *S*.
c. Pesan *M* disambung (*append*) dengan tanda tangan dijital *S*.



Gambar 1 Otentikasi dengan Tanda tangan dijital menggunakan fungsi *hash* [MUN06]

Selanjutnya, langkah-langkah untuk melakukan otentikasi adalah sebagai berikut:

a. Tanda tangan dijital *S* didekripsi dengan menggunakan kunci publik (*PK*) pengirim pesan, menghasilkan *message digest* semula, yaitu *MD*.
b. Pesan *M* diubah menjadi *message digest MD*' menggunakan fungsi *hash* satu arah yang sama dengan fungsi *hash* yang digunakan oleh pengirim.

c. Jika *MD*' = *MD*, berarti pesan yang diterima otentik dan berasal dari pengirim yang benar.

Apabila pesan M yang diterima sudah berubah maka MD' yang dihasilkan dari fungsi *hash* berbeda dengan MD semula. Ini berarti pesan tidak asli lagi. Apabila pesan M tidak berasal dari orang yang sebenarnya maka MD akan berbeda dengan MD' karena kunci publik yang digunakan oleh penerima pesan tidak berkoresponden dengan kunci privat pengirim. Andaikan pengirim pesan M menyangkal telah mengirim pesan, maka sangkalan tersebut dapat dibantah dengan cara berikut: jika ia tidak mengirim pesan, berarti ia tidak mengenkripsi MD dengan kunci privatnya. Faktanya kunci publik yang berkoresponden dengan kunci privat pengirim akan menghasilkan MD=MD' ini berarti MD memang benar dienkripsi oleh pengirim karena hanya pengirimlah yang mengetahui kunci privatnya sendiri.

4.4. Infrastruktur Kunci Publik (*Public Key Infrastructure*)

Pada sistem kriptografi kunci publik terdapat permasalahan dalam distribusi kunci publik yang juga disebut dengan *man-in-the-middle-attack*. Untuk lebih jelasnya, misalkan Alice dan Bob mengirim kunci publiknya masing-masing melalui saluran komunikasi. Orang ditengah misalnya Carol, memutus komunikasi antara Bob dan Alice lalui ia berpura-pura sebagai salah satu pihak (Alice atau Bob). Carol (yang menyamar sebagai Alice) mengirimkan kunci publiknya kepada Bob (Bob percaya itu adalah kunci publik milik Alice), dan Carol (yang menyamar sebagai Bob) mengieimkan kunci publiknya kepada Alice (Alice percaya itu adalah kunci publik milik Bob). Selanjutnya Carol mendekripsi pesan dari Bon dengan kunci privatnya, menyimpan salinannya, lalu mengenkripsi pesan tersebut dengan kunci publik Alice dan mengirim cipherteks tersebut kepada Alice. Alice dan Bob tidak dapat mendeteksi keberadaan Carol.

Serangan yang mirip dengan *man-in-the-middle attack* dan umum terjadi pada kunci publik tanpa identitas adalah penyamaran (*impersonation attack*). Seseorang yang memiliki kunci publik orang lain dapat menyamar seolah-olah dia adalah pemilik kunci tersebut. Contohnya dalam aplikasi e-*commerce* pemesanan tiket dengan dengan sistem pembayaran menggunakan kartu kredit. Pelanggan mengirimkan informasi kartu kredit melalui *website* agen perjalanan *online*. Selama pengiriman, informasi kartu kredit tersebut dilindungi dengan cara mengenkripsinya dengan kunci publik agen perjalanan *online*. Bagaimana pelanggan memastikan *website* tersebut memang benar milik agen perjalanan *online* dan bukan milik pihak lain yang menyamar dengan tujuan untuk mencuri informasi kartu kredit.

Karena terdapat permasalahan tersebut, maka dalam penerapan kriptografi kunci publik dibutuhkan pendukung yang dinamakan infrastruktur kunci publik (PKI). PKI adalah sebuah pengaturan yang menjamin penggunaan kunci publik bagi pihak-pihak yang terlibat dengan sistem penggunaan sistem keamanan. PKI juga mengikat kunci publik dengan identitas pengguna. Dengan PKI, setiap pengguna dapat mengotentikasi satu sama lain. Informasi di dalam sertifikat yang dikeluarkan oleh PKI digunakan untuk enkripsi dan dekripsi pesan antara pihak-pihak yang berkomunikasi. Komponen PKI adalah pengguna (pemohon sertifikat dan pemakai sertifikat), sertifikat dijital, CA (*Certification Authority* yaitu pihak yang mengeluarkan sertifikat dijital) dan direktori untuk menyimpan sertifikat dijital dan CRL (*Certificate Revocation List* berisi nomor seri sertifikat dijital ynag ditarik/ sudah kadaluarsa dan dianggap tidak sah*)*. CA biasanya adalah institusi keuangan (seperti bank) atau institusi terpercaya. PKI menyediakan cara penstrukturan komponen-komponen ini dan mendefenisikan bermacam-macam dokumen dan protokol.

Sertifikat dijital adalah dokumen dijital yang berisi informasi name subjek (perusahaan / individu yang disertifikasi), kunci publik subjek, waktu kadaluarsa sertifikat (*expired time*), algoritma yang digunakan untuk menandatangani sertifikat dan informasi relevan lain seperti nomor seri sertifikat dan lain sebagainya. Standar untuk sertifikat telah disetujui oleh ITU (*International Telecommunication United*) dan dinamakan X.509.

Sertifikat dijital tidak rahasia dan tersedia secara publik dan disimpan oleh CA pada direktori (*certificate repositories*). Salinan sertifikat ini dimiliki juga oleh pemohon sertifikat. Contoh sebuah sertifikat dijital: Sebuah agen perjalanan *online* membawa kunci publiknya dan mendatangi CA untuk meminta sertifikat dijital. CA mengeluarkan sertifikat dijital dan

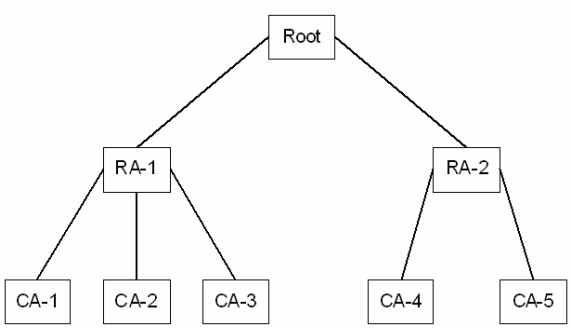
menandatangani sertifikat tersebut dengan cara mengenkripsi nilai *hash* dari kunci publik agen perjalanan *online* (atau nilai *hash* dari sertifikat dijital keseluruhan) dengan menggunakan kunci privat CA. Jadi sertifikat dijital mengikat kunci publik dengan identitas pemilik kunci publik. Supaya sertifikat dijital ini dapat diverifikasi maka kunci publik CA harus diketahui secara luas. Seseorang yang memiliki kunci publik CA dapat menverifikasi bahwa tanda tangan dijital di dalam suatu sertifikat sah.

Contoh penggunaan sertifikat dijital: Agen perjalanan *online* meletakkan salinan sertifikat dijital di *website* miliknya sehingga dapat diakses oleh pengunjung *website* tersebut. Misalkan seorang pelanggan ingin membeli tiket melalui *website* agen perjalanan tersebut, dan komunikasi antara *website* agen perjalanan dengan pelanggannya berhasil diintersepsi pihak ketiga sehingga *request* dari pelanggan masuk ke *website* agen perjalanan palsu yang dibuat oleh pihak ketiga. Pihak ketiga ini meletakkan sertifikat dijitalnya pada *website* palsu, tetapi ketika pelanggan membaca sertifikat dijital tersebut dia langsung paham bahwa dirinya sedang tidak berkomunikasi dengan agen perjalanan asli karena identitas agen perjalanan tidak terdapat pada sertifikat tersebut.

Misalkan pihak ketiga tersebut berhasil mengubah *website* agen perjalanan dan mengganti kunci publik milik agen perjalanan pada sertifikat dijital. Ketika pelanggan meng-*hash* sertifikat dijital tesebut dia memperoleh nilai *hash* yang tidak sama dengan nilai *hash* yang dihasilkan jika tanda tangan dijital diverifikasi dengan kunci publik CA. Pihak ketiga tidak memiliki kunci privat CA sehingga pelanggan dapat meyakini apakah dia memperoleh kunci publik agen perjalanan yang asli atau tidak.

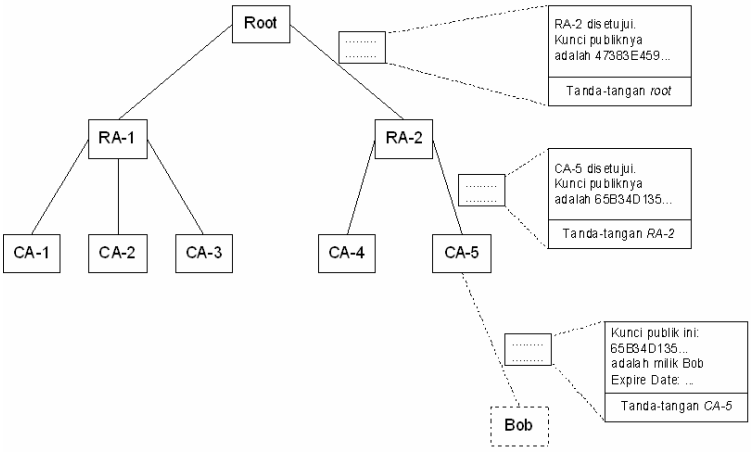
Adanya waktu kadaluarsa pada sertifikat dijital dimaksudkan agar pengguna mengubah kunci publik dan kunci privat pasangannya secara periodik dan jika kunci privat berhasil diketahui pihak lain sebelum waktu kadaluarsa habis maka sertifikat dijital harus ditarik. CA mengeluarkan daftar sertifikat dijital yang ditarik secara periodik dengan mengeluarkan CRL. Protokol OCSP (*Online Certificate Status Protocol*) memungkinkan pemeriksaan sertifikat secara *real time* sehingga lebih efisien dari pada pemeriksaan CRL secara tradisional.

Tentunya tidak mungkin hanya ada satu CA untuk melayani sertifikat dijital dari seluruh dunia. Bentuk PKI sederhana adalah hirarki CA seperti pada gambar 3.



Gambar 2 Hierarki CA dalam PKI

Pada level tertinggi terdapat *root* yang merupakan *root certificate authority*, yaitu *Internet Policy Registration Authority* (IRPA). Root mensertifikasi CA level satu (RA / *Registry Authoriries*) menggunakan kunci privat *root* (*root key*). RA bertindak sebagai *policy creation authority*, yaitu organisasi yang membuat kebijakan untuk memperoleg sertifikat dijital. Sebuah RA mungkin mencakup beberapa area seperti negara bagian, negara atau benua. *RA* menandatangai sertifikat digital untuk *CA* di bawahnya dengan menggunakan kunci privat *RA*. *CA* menandatangani sertifikat digital untuk individu atau organisasi dengan menggunakan kunci privat *CA*. Selain itu, *CA* bertanggung jawab untuk otentikasi sertifikat digital, sehingga



Gambar 3 Contoh Rantai Sertifikat Dijital

CA harus memeriksa informasi secara hati-hati

sebelum mengeluarkan sertifikat digital. Gambar 3 memperlihatkan rantai sertifikat di dalam *PKI*.

Verifikasi sertifikat digital dilakukan dari daun menuju akar (*root*). Misalkan pelanggan memerlukan kunci publik agen perjalanan online untuk melakukan pemesanan tiket, lalu dia mencari dan menemukan sertifikat agen perjalanan tersebut ditandatatangani oleh CA-5. Pelanggan kemudian menandatangani CA-5 dan meminta bukti legitimasi CA-5. CA-5 merespon dengan memperlihatkan sertifikat dijital yang diperoleh dari RA-2, di dalamnya ada kunci publik CA-5 yang ditandatangani oleh RA-2. Dengan menggunakan kunci publik CA-5, pelanggan dapat memverifikasi sertifikat dijital agen perjalanan yang ditandatangani oleh CA-5 dan mendapatkan hasil bahwa sertifikat tersebut sah. Langkah berikutnya, pelanggan mendatangi RA-2 dan meminta nukti legitimasi RA-5. RA-2 merespon dengan memperlihatkan sertifikat dijital yang diperoleh dari *root* , di dalamnya terdapat kunci publik RA-2 yang ditandatangani oleh *root.* Pelanggan memverifikasi sertifikat dijital RA-2 dengan menggunakan kunci publik *root* dan mendapatkan hasil bahwa sertifikat dijital tersebut sah. Pelanggan akhirnya yakin bahwa dia sudah memilliki kunci publik agen perjalanan. Rantai sertifikat yang menuju ke *root* disebut *chain of trust* atau *certification path*

5. Hambatan Penerapan Kriptografi pada *mobile commerce*

Pada bagian sebelumnya dapat dilihat aplikasi kriptografi yang dapat digunakan untuk komunikasi yang aman pada aplikasi *mobile commerce.* Apakah aplikasi kriptografi ini dapat langsung diterapkan?

Penerapan kriptografi pada jaringan nirkabel yang merupakan lingkungan aplikasi *mobile commerce* digunakan, lebih sulit dari pada penerapan kriptografi pada jaringan kabel karena terdapat berbagai batasan. Pada jaringan kabel terdapat beberapa asumsi yaitu *client* memiliki kemampuan komputasi, *bandwith*, *latency* yang tinggi dan tempat pemyimpanan yang cukup. Hal ini bertolak belakang dengan perangkat nir kabel yang memiliki bandwith terbatas, kemampuan komputasi terbatas, memiliki sumber tenaga berupa baterai. Oleh karena itu, banyak aplikasi kriptografi pada jaringan kabel yang harus dioptimasi supaya berjalan dengan efisien pada perangkat nir kabel tetapi harus *compatible* dengan aplikasi pada jaringan kabel.

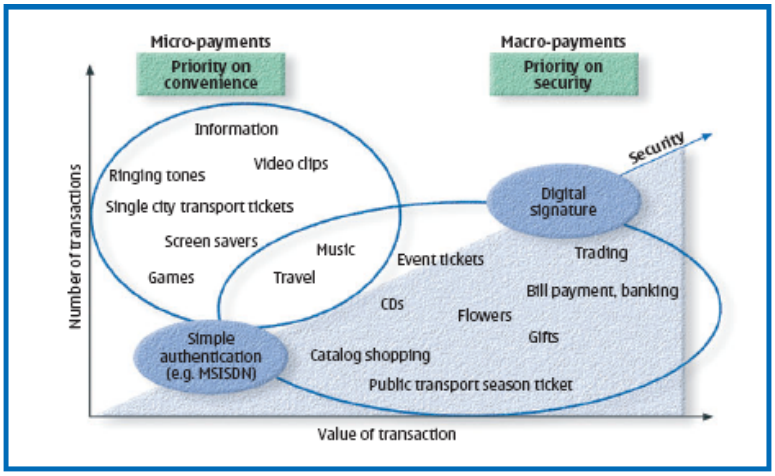
Karena adanya keterbatasan kemampuan komputasi maka algoritma kriptografi yang dapat digunakan pada aplikasi *mobile commerce* menjadi terbatas bergantung dari karakteristik algoritma tersebut. Oleh karena itu, algoritma yang digunakan harus membutuhkan biaya komputasi yang kecil, tetapi tetap dapat memberikan jaminan keamanan yang dibutuhkan.

Karena adanya keterbatasan *latency* yang tinggi pada jaringan nirkabel, maka protokol kriptografi yang diterapkan sebaiknya memiliki jumlah pesan dan pertukaran pesan yang lebih sedikit. Hal ini dapat dilihat pada penggunaan WTLS kelas 3 yang menggunakan handshake sebagai permintaan sertifikat (berbeda dengan SSL).

Keterbatasan *bandwith* juga membatasi ukuran pesan. Hal ini mengakibatkan sertifikat pada aplikasi seperti *mobile commere* sedikit berbeda dan pengiriman rantai sertifikat misalnya url sertifikat tidak dilakukan.

Keterbatasan yang keempat adalah ruang penyimpanan yang terbatas pada perangkat seperti telepon seluler dan PDA. Oleh karena itu, dalam menerapkan kriptografi kode yang dibutuhkan harus diminimumkan misalnya memanfaatkan kode yang dioptimasi untuk *platform* perangkat *mobile*. Solusi lainnya adalah mengurangi ukuran kunci dan menyimpan sertifikat pada *client*.

Karena adanya keterbatasan yang telah dijelaskan di atas maka, tingkat jaminan keamanan yang diimplementasikan hendaknya disesuaikan dengan nilai dari transaksi yang



Gambar 4 Berbagai aplikasi dengan tingkat keamanan yang berbeda [NOK04]

dilakukan karena dalam menjamin keamanan ini sering kali dibutuhkan resource yang relatif besar. Pada gambar 1 dapat dilihat berbagai jenis aplikasi dengan nilai transaksinya serta prioritas tingkat keamanannya.

6. Kriptografi pada Teknologi pendukung *Mobile commerce*

6.1. *SIM Application Toolkit (SAT)*

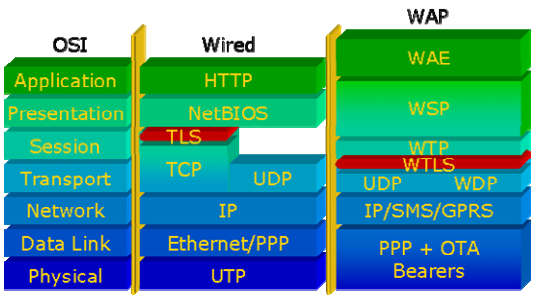
SIM misalnya pada GSM menyimpan data personal pemilik kartu dan dapat diimplementasikan dalam bentuk kartu cerdas (*smart card)* yang sering disebut kartu SIM. SIM *toolkit* merupakan spesifikasi SIM dan fungsionalitas terminal yang memungkinkan SIM mengendalikan perangkat *mobile* untuk beberapa fungsi tertentu. SIM A*pplication Toolkit* (SAT) digunakan untuk membuat aplikasi *mobile commerce* berbasis *Short Message Service* (SMS).

Dalam sistem berbasis SAT komunikasi antara *mobile client* dan penyedia layanan dilakukan melalui SMS. SMS digunakan untuk mengidentifikasi dan mengotorisasi pembayaran. Pengguna diidentifikasi dan diotentikasi oleh layanan otentikasi GSM sehingga operator seluler GSM bertindak sebagai perantara antara *mobile client*, *server* untuk pembayaran, dan penjual layanan.

SAT menyediakan layanan kerahasian, otentikasi, *message replay protection*, integritas, tetapi tidak menjamin nirpenyangkalan. Hal ini adalah yang menjadi faktor kelemahan utama pada aplikasi *mobile commerce* berdasarkan SAT. SAT dapat mendukung standar enkripsi termasuk *triple DES*. Penyedia layanan menempatkan kunci enkripsi pada SIM sebelum diberikan kepada pengguna. Hal ini memastikan kunci rahasia tidak pernah dikirim melalui saluran komunikasi. Otentikasi dapat dilakukan melalui algoritma enkripsi yang dapat dipilih oleh penyedia sistem pembayaran. Integritas data dijamin melalui pesan ringkas (*message digest*) seperti SHA dan MD5.

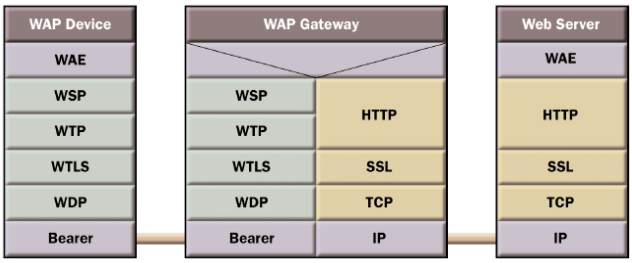
Selain tidak mampu menjamin nirpenyangkalan, SAT juga memiliki kelemahan karena penggunaan kode PIN pada perangkat *mobile client*. Kode PIN ini biasanya berupa angka 4 dijit yang dapat ditebak dengan relatif mudah oleh pencuri perangkat *mobile*.

6.2. WAP



Gambar 6 WAP 1.x vs jaringan kabel

Pada gambar 6 dapat dilihat protokol pada WAP pada setiap *Layer* yang dapat dibandingkan

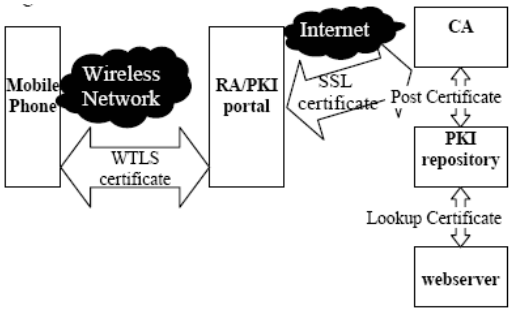


Gambar 5 Arsitektur WAP 1.x

dengan model OSI. Adapun area yang berhubungan dengan keamanan pada WAP 1.x adalah WTLS, *Wireless Identity Module*, WAP PKI, WML Script signText, dan keamanan *end-to-end* pada *transport Layer*.

a. WPKI (*Wireless Public Key Infrastructure*)

WPKI merupakan ekstensi dari PKI, yang dibuat khusus untuk jaringan nirkabel. WPKI memerlukan komponen yang sama dengan PKI yang telah dijelaskan sebelumnya. Namun pada WPKI, *registration authority* (RA) diimplementasikan berbeda dan terdapat entitas baru yaitu PKI Portal. PKI Portal dapat seperti sistem pada dua jaringan seperti halnya WAP *gateway*. PKI Portal berfungsi sebagai RA dan bertangung jawab untuk menterjemahkan pesan dari *client* kepada RA dan berinteraksi dengan CA pada jaringan kabel. RA mevalidasi aplikasi, apakah permintaannya untuk memperoleh sertifikat dijital dikabulkan atau ditolak. Arsitektur WPKI dapat dilihat pada gambar 7.



Gambar 7 Arsitektur WPKI

b. WTLS *(Wireless Transport Layer Security)*

Protokol WTLS merupakan protokol yang mendukung WPKI dan didesain untuk menjamin keamanan komunikasi dan transaksi melalui jaringan nirkabel. Seperti dapat dilihat pada gambar 5, WTLS terdapat pada *transport Layer* antara WAP *client* pada perangkat *mobile* dengan WAP *server* pada WAP *gateway*. WTLS menyediakan fungsionalitas yang mirip dengan fungsi *Layer* keamanan pada internet yaitu TLS/SSL. WTLS dibuat berdasarkan TLS dan dioptimasi untuk komunikasi nirkabel, mendukung *datagram*, memiliki protokol *handshake* yang dioptimasi dan memiliki mekanisme *dynamic key refreshing* [HAM01].

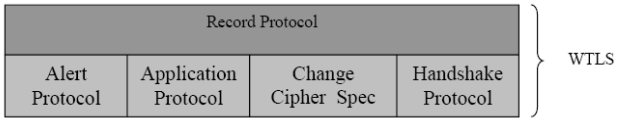
Dynamic key refreshing merupakan sebuah mekanisme yang memungkinkan pengubahan kunci enkripsi dan otentikasi dalam rentang waktu tertentu. Hal ini akan mengurangi kemungkinan *eavesdropper* mendekripsi pesan karena kunci yang berbeda digunakan pada sebuah sesi. Seberapa sering kunci diubah ditentukan saat melakukan *handshake*.

Pada tabel 1 dapat dilihat beberapa teknik kriptografi yang terdapat dalam WTLS untuk memungkinkan koneksi yang aman.

Tabel 1 teknik kriptografi pada WTLS

Otentikasi	SHA-1,MD5
Pertukaran kunci	RSA, Diffie-Hellman, DFEC (Diffie-Hellman Elliptic Curve)
Enkripsi	RC5, DES, 3DEA, IDEA

Arsitektur WTLS dibagi menjadi lima bagian. Terdapat *record protocol* dan empat *client* protokol yang digunakan berhubungan dengan *record protocol* seperti yang pada dilihat pada gambar 8.



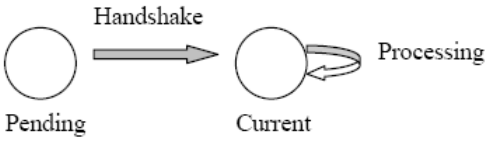
Gambar 8 Arsitektur WTLS

i. *Record Protocol*

Record Protocol dibagi menjadi empat *client* protokol yaitu *alert*, *application*, *change cipher* dan *handshake protocol*. Protokol ini menerima data yang akan ditransfer. Data ini akan dikompresi, kemudian di-*hash* dengan MAC (fungsi *hash* satu arah yang menggunakan kunci rahasia dalam pembangkitan nilai *hash*). Selanjutnya data yang telah ditambah dengan nilai *hash* dienkripsi dan ditransmisikan.

Ketika data diterima pada *record protocol* di tempat tujuan, data tersebut akan didekripsi, diverifikasi, didekompresi kemudian diteruskan ke *layer* diatasnya. Teknik yang digunakan untuk kompresi, otentikasi dan enkripsi opsional dan ditentukan pada saat *handshake*. Tidak seperti TLS tidak ada fragmentasi pada protokol ini, karena fragmentasi dan asembli dilakukan pada *layer transport*.

Ketika terjadi koneksi ke *client* atau *server* maka *record protocol* berada pada *connection state.* Selama *state* ini algoritma untuk kompresi, MAC, dan enkripsi ditentukan. Terdapat dua status koneksi seperti yang dapat dilihat pada gambar 9. Selama berada pada *current state* semua pemrosesan di atas dilakukan. *Pending state* menandakan bahwa *record protocol* sedang menunggu sesuatu. Sebelum proses *handshake* selesai, status koneksi adalah menunggu (*pending*), dan selanjutnya berubah menjadi *current* ketika selesai.



Gambar 9 *connection state* pada WTLS

ii. *Alert Protocol*

Protokol ini digunakan untuk mengirim *alert* yang berbeda antara *client* dan *server*. Sebuah *alert* dapat berupa pesan untuk menutup koneksi antara *client* dan *server* atau sebuah pesan

kesalahan. Pesan kesalahan dapat berisi informasi seberapa parah kesalahan tersebut dan deskripsi permasalahannnya. Terdapat tiga jenis kesalahan yaitu *warning*, *critical* dan *fatal*.

Jika pesan fatal dikirimkan maka kedua belah pihak yang berkomunikasi akan menghentikan koneksi. Koneksi lain yang menggunakan *session* yang sama dapat terus berlangsung, tetapi *session id* akan diset sehingga tidak ada koneksi yang dibangun menggunakan sesi ini. Jika pesan *critical* dikirimkan, maka koneksi akan dihentikan dan koneksi masih dapat dibangun menggunakan sesi yang sama. Jenis pesan yang ketiga, *warning*, bukan merupakan sebuah *error*, tetapi berarti MAC tidak divalidasi dengan benar. Tidak ada koneksi yang ditutup, tetapi paket yang berisi MAC yang *corrupt* akan diabaikan.

iii. *Change Cipher Spec Protocol*

Cipher suite berisi algoritma *bulk encryption* dan MAC yang digunakan. Jika sebuah *client* atau *server* ingin mengubah *cipher suite* maka *change cipher spec* akan dikirimkan. Setelah mengirimkannya, pengirim akan masuk ke dalam status *pending* . Pada saat ini, penerima harus merespon dan mengirim pesan untuk memulai koneksi menggunakan *cipher suite* yang baru dan ketika pengirim menerima konfirmasi maka baik pengirim dan penerima akan memasuki *current state* dan pemrosesan dilanjutkan.

iv. *Handshake Protocol*

Terdapat tiga jenis *handshake* yaitu *full handshake, abbreviated handshake* dan *optimised handshake*. Selain itu, terdapat *Service Data Unit* (SDU) yang digunakan ketika komunikasi terjadi menggunakan *datagram.* SDU terdiri dari beberapa pesan *handshake* yang digabungkan karena memiliki tujuan yang sama.

Selama proses *handshake* dilakukan maka ditentukan informasi mengenai versi protokol, algoritma kriptografi dan informasi apakah otentikasi digunakan dan kunci publik. Proses *handshake* melibatkan langkah-langkah berikut [HAM01]:

a. Pertukaran pesan hello untuk menyepakati algoritma yang digunakan dan pertukaran nilai random.
b. Pertukaran parameter kriptografi yang diperlukan sehingga memungkinkan *client* dan *server* menyepakati *pre-master secret*.
c. Pertukaran sertifikat dan informasi yang memungkian *client* dan *server* mengotentikasi dirinya masing-masing.
d. Pembangkitan *master secret* dari *pre-master secret* dan pertukaran nilai random
e. Menyediakan parameter keamanan untuk *record layer*
f. *Client* dan *server* melakukan verifikasi bahwa pihak yang berkomunikasi dengannya telah menghitung parameter keamanan yang sama dan memastikan *handshake* berlangsung tanpa terjadi pengubahan pesan oleh penyerang.

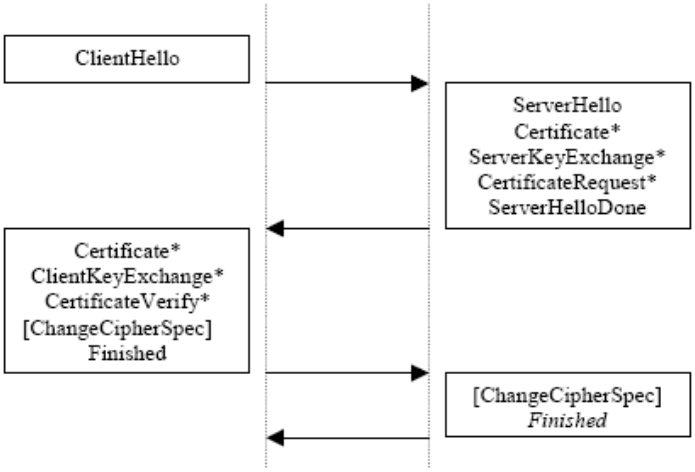
Adapun daftar parameter keamanan untuk komunikasi yang aman dapat dilihat pada tabel 2.

Tabel 2 *security parameters*

Parameter	Deskripsi
Connection End	Menyatakan apakah sebuah entitas adalah sebuah *server* atau *client*
Algoritma *bulk encryption*	Algoritma yang digunakan untuk *bulk encryption.*
Algoritma MAC	Algoritma yang digunakan untuk menjamin integritas pesan dan otentikasi
Algoritma kompresi	Algoritma yang digunkan untuk mengkompresi data sebelum dienkripsi. Semua informasi yang dibutuhkan untuk melakukan kompresi
Master Secret	Data berukuran 20 *byte* yang bersifat rahasia antara dua buah pihak dalam koneksi yang aman
Client Random	Data berkuran 16 *byte* yang disediakan oleh *client*
Server Random	Data berkuran 16 *byte* yang disediakan oleh *server*
Key Refresh	Interval waktu yang menyatakan seberapa sering beberapa parameter *connection state* diupdate (kunci enkripsi, MAC *secret* dan IV/ *Initialization Vector*)

Mode angka urutan	Menyatakan skema yang digunakan untuk menghasilkan angka urutan dalam koneksi yang aman.

a. ***Full hanshake***



Gambar 10 *Full handshake*

Dalam pesan *Client*Hello, *client* mengirimkan informasi metode pertukaran kunci, *cipher suite* dan metode kompresi yang didukung. Selan itu terdapat juga informasi sertifikat mana yang dipercayai oleh pelanggan. Semua informasi ini berada dalam bentuk daftar dan urutan menyatakan *preference client*. *Client* membangkitkan angka acak untuk digunakan kemudian dan jika tersedia *session id* maka informasi ini juga dimasukkan ke dalam pesan *Client*Hello. Untuk memastikan *server* memperoleh semua informasi yang dibutuhkan dalam pesan *Client*Hello juga terdapat informasi versi WTLS dan mode sequence number yang digunakan serta informasi *key refresh*. Setelah mengirimkan Pesan *Client*Hello kepada *server* maka *client* akan menerima data sampai *Server*HelloDone diterima.

Pesan *Server*Hello mengandung atribut yang sama dengan *Client*Hello, tetapi informasi sertifikat yang dipercaya dihilangkan. *Server* memeriksa data yang diterima kemudian merumuskan parameter yang digunakan untuk saluran komunikasi yang aman. Sebelum mengirim pesan, *server* membangkitkan angka acak yang yang akan dikirm bersamaan dengan pesan sama halnya seperti yang dilakukan oleh *client*. Setelah mengirim *Server*Hello, *server* dapat memilih apakah mengirim sertifikat, *Server*KeyExchange dan atau CertificateRequest.

Jika otentikasi digunakan, *server* harus mengirim sertifikatnya kepada *client*. Pesan sertifikat dapat terdiri dari satu atau lebih sertifikat. Format sertifikat dapat berupa X.509. Selain mengirim sertifikat informasi url sertifikat dapat juga dikirimkan.

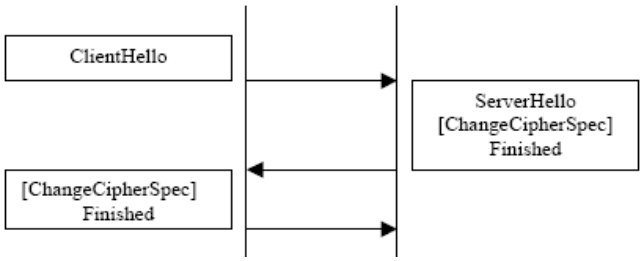
*Server*KeyExchange hanya dikirimkan jika sertifikat yang telah dikirim sebelumnya tidak mengandung informasi yang cukup bagi *client* untuk bertukar informasi *pre-master secret* (nilai inisial yang digunakan untuk menghitung nilai *master secret*). Informasi ini dapat berupa kunci publik RSA yang digunakan untuk mengenkripsi

Setelah menerima pesan *Server*HelloDone, *client* akan mengirimkan data yang dibutuhkan. Hal ini bergantung pada pesan yang sebelumnya dikirim oleh *server*. Jika *client* telah menerima CertificateRequest maka *client* harus mengirimkan sertifikat atau url tempat sertifikat.

Client kemudian menset *pre-master secret* dengan mengirim pesan *Client*KeyExchange. Struktur pesan ini bergantung pada metode pertukaran kunci yang dipilih *server*. Jika *Server* meminta *client* mengotentikasi dirinya, maka *client* harus mengirim pesan *CertificateVerify*. Pesan ini merupakan tanda tangan dari nilai *hash* dari keseluruhan pesan yang sudah dikirim. Setelah pesan ini dikirim, *client* kemudian mengirim pesan ChangeCipherSuite kepada *server*. Selanjutnya mengirim pesan Finisheddengan menggunakan *cipher suite* yang baru. *Cipher suite* ini merupakan hasil negosiasi selama proses *handshake*. *Server* menerima pesan Finished kemudian setelah diverifikasi *server* mengirim pesan Finished dengan menggunan *cipher suite* yang baru dan akan diverifikasi oleh *client*. Hal ini menandakan akhir proses *handshake*. Jika proses *handshake* berjalan dengan lancar maka *client* dan *server* dapat melakukan komunikasi melalui koneksi yang aman.

b. Abbreviated handshake

Jika *client* dan *server* sebelumnya memiliki sebuah saluran komunikasi yang aman maka jika *client* ingin melanjutkan koneksi tersebut, *client*

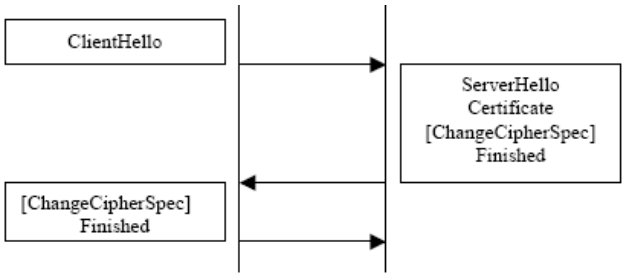


Gambar 11 Abbreviated handshake

dapat memulai *abbreviated handshake*. Proses ini dimulai dengan *client* mengirim pesan *Client*Hello menggunakan *session id* yang akan dilanjutkan. Setelah menerima pesan ini, *server* memeriksa apakah *session id* tersebut terdapat dalam *cache*. Jika *session id* tidak ditemukan maka *server* membangkitkan *session id* yang baru dan protokol *full handshake* akan dijalankan. Jika sebaliknya, *server* mengirim *Server*Hello diikuti dengan pesan ChangeCipherSuite dan Finished. *Client* mengirim pesan ChangeCipherSuite dan Finished. Kemudian, komunikasi yang aman dapat dilanjutkan. Protokol ini memungkinkan pengunaaan banyak saluran komunikasi yang aman dengan menggunakan *session id* yang sama.

c. Optimised handshake

Protokol *handshake* ini digunakan ketika sertifikat *client* dapat ditemukan pada sebuah url atau pada *server* sendiri. Setelah menvalidasi



Gambar 13 *Optimised handshake*

sertifikat tersebut, *server* akan mengirim sertifikat miliknya kepada *client* , diikuti dengan pesan ChangeCipherSpec dan Finished. *Client* kemudian merespon seperti halnya pada protokol yang telah dijelaskan sebelumnya dan salauran komunikas yang aman telah terbangun.

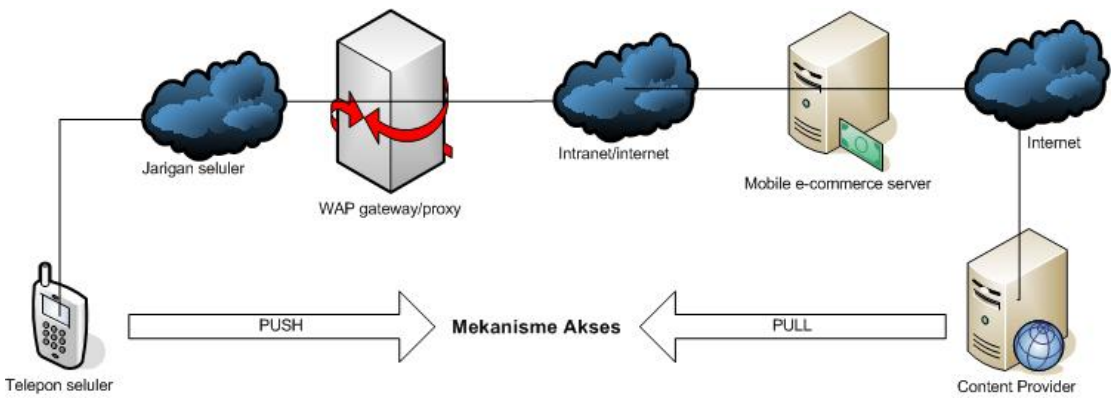
Terdapat tiga kelas pada saluran komunikasi dengan WTLS yang masing-masing memiliki fungsionalitas yang berbeda. Pada kelas 1 tidak diperlukan otentikasi server dan pengguna, pada kelas 2 diperlukan otentikasi server dan pada kelas 3 dibutuhkan otentikasi baik *server* maupun *client*. Untuk lebih jelasnya dapat dilihat pada tabel 3. M berarti wajib/*mandatory* sedangkan O berarti opsional.

Tabel 3Kelas WTLS

Fitur	K-1	K-2	K-3
Pertukaran kunci publik	M	M	M
Sertifikat *Server*	O	M	M
Sertifikat *Client*	O	O	M
Shared secret handshake	O	O	O
Kompresi	-	O	O
Enkripsi	M	M	M
MAC	M	M	M
Interface Kartu cerdas	-	O	O

WAP versi 1.x mengunakan protokol WTLS untuk menjamin keamanan pesan dari perangkat *mobile* ke WAP *gateway* . WAP *gateway* akan mentransformasikan pesan dalam format WAP ke dalam TCP/IP, meneruskan data ke jaringan kabel dan berkomunikasi dengan web *server* yang diakses oleh perangkat *mobile*.

Salah satu contoh arsitektur *mobile commerce* yang memanfaatkan WAP 1.x dapat dilihat pada gambar 14. Ketika terjadi transaksi yang melibatkan data rahasia seperti nomor *credit card*, *password*, alamat, data ditransmisikan lewat jaringan, sehingga pertukaran informasi ini harus dijamin keamanannya. Ada beberapa bagian dimana masalah keamanan memiliki peran penting yaitu bagian yang berhubungan dengan komunikasi lewat jaringan nirkabel,

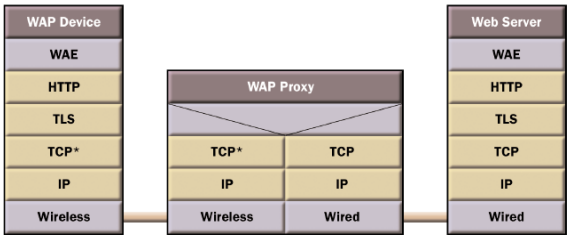


Gambar 12 WAP1.x

keamanan *gateway*, koneksi ke *server*, dan masalah keamanan aplikasi yang berjalan di *server*. Keamanan komunikasi lewat jaringan nirkabel dijamin oleh *Layer* WTLS. *Gateway* umumnya merupakan milik operator seluler sehingga pengembang tidak memiliki kontrol terhadap komponen ini. Peran dari WAP *gateway* adalah menghubungkan jaringan nirkabel dengan jaringan IP. Dengan demikian, data terenkripsi dari jaringan nir kabel akan di dekripsi pada *gateway*, kemudian dienkripsi kembali dengan memanfaatkan SSL untuk dikirim melalui internet. Pada rentang waktu yang singkat terdapat data plainteks pada *gateway*.

c. WAP TLS *Profile*

Karena adanya ancaman keamanan pada WAP *gateway* seperti yang telah dijelaskan di atas, maka pada versi 2.0 WAP menggunakan WAP TLS *profile*. Arsitektur WAP 2.0 bisa dilihat pada gambar 11. Dengan adanya TLS *tunneling* ini maka dimungkinkan keamanan end-to-end antara *client* dan *server*.



Gambar 14 Arsitektur WAP 2.0

Perubahan pada WAP versi 2.0 yang berhubungan dengan keamanan adalah sebagai berikut:

a. Ditambahkan WAP TLS *Profile* yang mendukung beberapa algoritma seperti RSA, RC4/3DES dan SHA-1.
b. WAP *Certification Profile*. Terdapat penambahan spesifikasi untuk menangani *TLS server certificate*
c. Perubahan kecil pada WIM, WTLS dan WML Script Crypto Library
d. WIM (*Wireless Identity Module*)

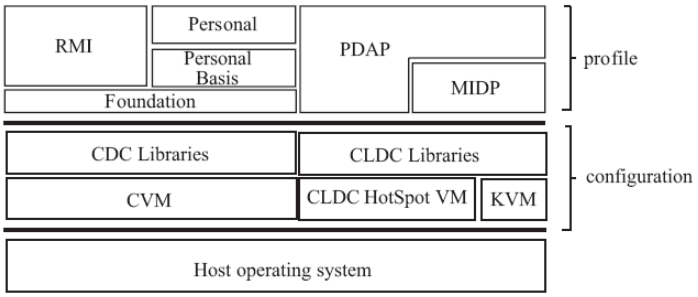
WIM digunkan pada fungsi yang berhubungan dengan WTLS /WAP TLS *profile* dan keamanan pada level aplikasi dengan menyimpan dan memproses informasi seperti kunci rahasia dan sertifikat yang dibutuhkan untuk memberikan layanan otentikasi dan nirpenyangkalan. Untuk mencegah pengubahan, WIM diimplementasikan sebagai perangkat lunak pada kartu cerdas yang berdasarkan *microprocessor*.

e. WMLScript

WMLScript dapat digunakan untuk mengimplementasikan tanda tangan dijital pada konten WML (*Wireless Markup Languange* yang bisa dianalogikan dengan HTML pada jaringan kabel). Fungsi *signtext* memungkinkan pengguna untuk menandatangani sebuah transaksi secara dijital dengan mekanisme seperti yang telah dijelaskan pada bagian aplikasi kriptografi.

6.3. JAVA ME

Ada dua JSR (*Java Spesification Requests*) yang berhubungan dengan kriptografi dan jaminan keamanan pada aplikasi *mobile commerce* yaitu *Mobile Information Device Profile*, MIDP 2.0 dan *Security and Trust Services API*. MIDP 2.0 merupakan spesifikasi *framework* keamanan untuk aplikasi java yang didesain untuk dijalankan pada lingkungan MIDP. Fitur penting lainnya pada Java ME adalah penyimpanan data pada tempat penyimpanan permanen melalui *record store* atau pada *file resource* aplikasi yang dapat saja digunakan untuk menyimpan sertifikat atau kunci.



Gambar 15 Arsitektur JAVA ME

a. *Security and Trust Services API*

Security and Trust Services API (SATSA) merupakan API yang menyediakan kemampuan tambahan keamanan pada platform Java ME CLDC. API ini menspesifikasikan kumpulan API yang menyediakan layanan keamanan dan kepercayaan (*trust*) dengan mengimplementasikan sebuah *Security Element.* SE merupakan sebuah perangkat keras atau perangkat lunak yang merupakan komponen dari sebuah Java ME *device*. SE menyediakan fitur tempat pemyimpanan yang aman untuk menyimpan data sensitif dan operasi kriptografi. Dengan fitur ini, aplikasi Java ME dapat

menyimpan kunci dan juga kemampuan untuk melakukan enkripsi dan dekripsi yang seing kali dibutuhkan pada aplikasi *mobile commerce*. Implementasi SATSA dapat berupa:

(1). Kartu cerdas (*smard card*) pada telepon seluler
(2). Dapat diimplementasikan sendiri oleh perangkat *mobile*, misalnya dalam bentuk *embedded chip* atau fitur khusus dari perangkat keras
(3). Diimplementasikan seluruhnya dalam bentuk perangkat lunak

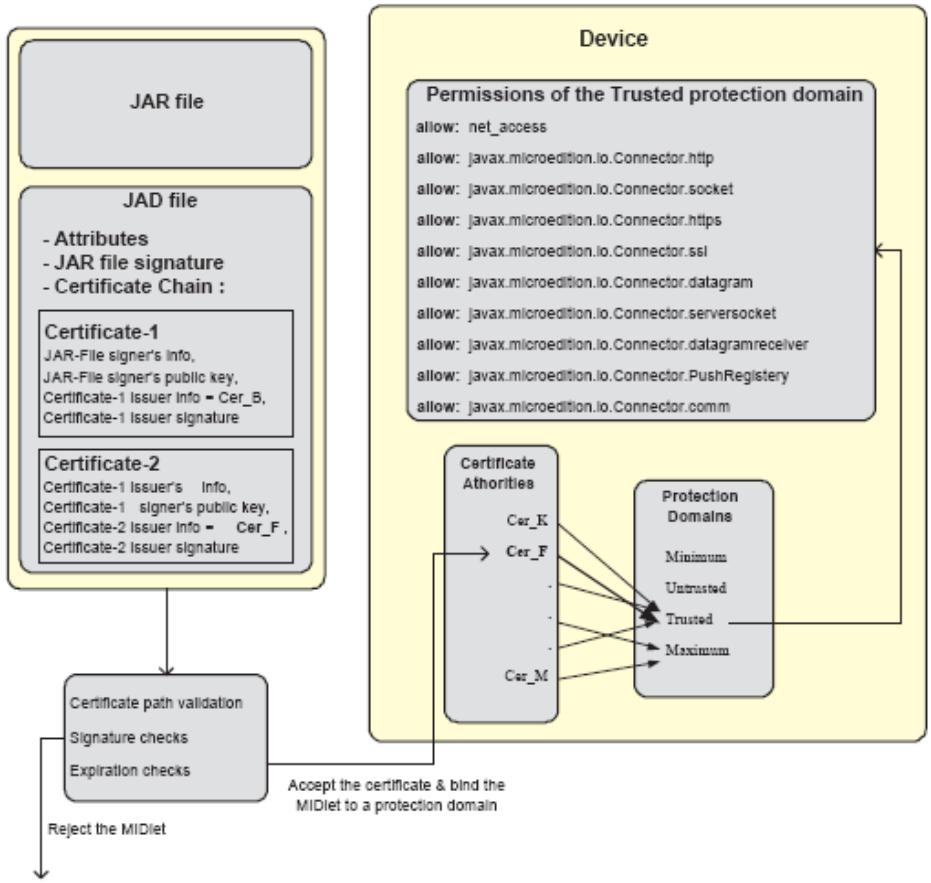
Walaupun beberapa API SATSA dioptimasi untuk implementasi pada kartu cerdas, spesifikasi SATSA API (JSR 177) tidak menutup kemungkinan berbagai implementasi dari SE.

b. Otentikasi Aplikasi

Berbeda dengan WAP yang menyimpan bisnis logik aplikasi di *server*, pada J2ME bisnis logik dapat diimplementasikan pada perangkat *mobile*. Oleh karena itu, pengunduhan aplikasi/*midlet* ke perangkat *mobile* mungkin berbahaya apalagi jika tidak diketahui siapa yang membuat aplikasi tersebut dan dari mana asalnya, misalnya pengguna bisa saja mengunduh aplikasi *mobile commerce* yang telah dimodifikasi oleh pihak ketiga. Aplikasi ini mungkin melakukan koneksi yang tidak terotorisasi ke jaringan untuk mengirimkan informasi yang diinginkan oleh pihak ketiga. Oleh karena itu, pada MIDP terdapat mekanisme *protection domain* yang dapat diset sehingga memberi jaminan aplikasi yang *didownload* pengguna aman untuk dijalankan.

Protection domain yang terdiri dari dua bagian. Bagian pertama adalah ijin untuk operasi-operasi sensitif yang dapat dilakukan aplikasi dan operasi mana yang harus ditanyakan kepada pengguna (apakah operasi tersebut dijalankan atau tidak), sedangkan bagian kedua berupa kriteria cara masuk ke dalam *protection domain*. Apabila *protection domain* ini tidak di set maka secara default aplikasi berada *domain untrusted*. Selain itu, pada perangkat *mobile* biasanya pengguna dapat menset ijin yang diberikan kepada aplikasi.

MIDP 2.0 menspesifikasikan bagaimana sebuah aplikasi Java yang telah ditandatangani dapat diverifikasi sehingga dapat masuk ke dalam *protection domain* yang nantinya dapat dipilih pengguna. Idenya adalah, pengembang aplikasi dapat memperoleh sertifikat dari CA yang dikenal, kemudian tanda tangan dari aplikasi dan sertifikat yang bersesuaian dapat ditempatkan



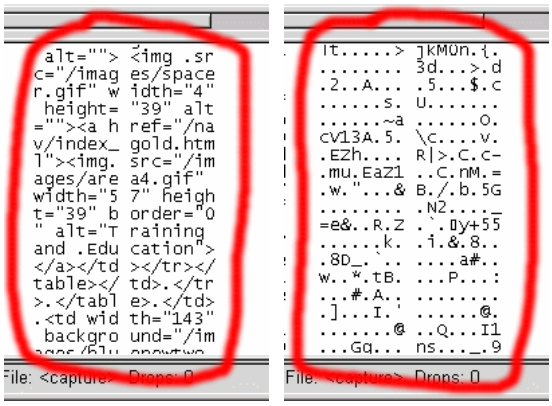
Gambar 16 Contoh Otentikasi dan penentuan *protection domain* [DEB04]

pada *application descriptor* sehingga informasi ini dapat digunakan untuk menjamin integritas aplikasi dan otentikasi sumber aplikasi. Mekanime verifikasi midlet yang sudah ditandatangani sudah terimplementasi pada perangkat *mobile* dalam *Application Management System* (AMS).

c. Koneksi HTTPS (*http secure*)

HTTPS merupakan koneksi http yang menggunakan SSL dan banyak digunakan pada aplikasi *e-commerce*. Pada MIDP terdapat implementasi SSL yang disebut KSSL (*kilobyte* SSL) dan mulai digunakan sejak MIDP versi 1.0.3.

Implementasi HTTPS pada MIDP 2.0 merupakan implementasi http yang diatas salah satu dari protokol TLS 1.0, SSL 3.0, WTLS , WAP TLS *Profile and Tunneling Spesification* [KNU02]. Koneksi yang aman pada MIDP 2.0 diimplementasikan dengan menggunakan kelas antara lain `HTTPSConnection`, `SecurityInfo`, dan `Certificate`. Keamanan dari HTTPS sangat bergantung pada *cipher suite*. Contoh *cipher suite* TLS_RSA_WITH_RC4_128_SHA, berarti digunakan protokol TLS, RSA digunakan untuk pertukaran kunci, RC4 dengan panjang *session key* 128 bit, dan SHA-1 digunakan sebagai fungsi *hash*. Protokol TLS dimulai dengan proses *handshake* yang pada prinsipnya sama dengan penjelasan WTLS di atas. Pada gambar 16 dapat dilihat contoh data dengan HTTPS dan tanpa HTTPS yang berhasil ditangkap oleh *network sniffer*.



Gambar 17 Data dengan HTTPS (kiri) dan tanpa HTTPS (kanan)

d. Kriptografi pada level aplikasi

Pada J2SE, enkripsi dapat digunakan pada aplikasi dengan memanfaatkan *Java Cryptographic Extension* (JCE). JCE menyediakan *framework* dan implementasi enkripsi, *key generation and agreement*, dan algoritma MAC. JCE mendukung algoritma enkripsi kunci simetri, asimetri, *block cipher* dan *stream cipher*. Namun, JCE tidak dimasukkan sebagai bagian dari JAVA ME karena selain ukurannya besar, JCE juga memerlukan memori yang tidak sedikit.

SATSA merupakan JSR yang relatif baru dan dapat diimplementasikan seluruhnya dalam bentuk perangkat lunak. Akan tetapi, menulis sendiri kode enkripsi dan dekripsi pada Java ME pun tidak disarankan karena terdapatnya kesulitan yang kompleks dalam memilih dan mengimplementasikan algoritma kriptografi dengan berbagai batasan yang ada. Namun, pengembang dapat memanfaatkan *toolkit* yang dikembangkan oleh vendor lain.

Pada CDC sebagian kecil dari paket pada JCA (Java Cryptography Architecture) dapat didukung secara opsional. Namun, beberapa API penting tidak tersedia sehingga beberapa algoritma tidak dapat diimplementasikan dan kalaupun bisa, kemungkinan besar kurang efisien.

Oleh karena itu, untuk menggunakan kriptografi pada level aplikasi pada Java ME, pengembang dapat menggunakan *toolkit* yang dikembangkan oleh vendor lain. Contoh *toolkit* yang dapat digunakan adalah Bouncy Castle lightweight API, Phaos Technology Micro Foundation toolkit, NTRU Neo for Java toolkit, dan B3 Security. Ada beberapa hal yang harus diperhatikan terkait dengan *toolkit* yang akan digunakan pada aplikasi *mobile* untuk mengimplementasikan kriptografi yaitu:

a. Membutuhkan waktu yang relatif singkat karena perangkat *mobile* yang harus responsif walapun memiliki CPU yang lambat dan *resource* yang terbatas.
b. Membutuhkan media penyimpanan yang kecil. Kebanyakan API kriptografi berukuran sampai beberapa MB, sementara perangkat *mobile* (khususnya MIDP) sebagian besar memiliki memori terbatas.

c. Mendukung berbagai algoritma penting baik algoritma simetri, algoritma kunci publik dan tanda tangan dijital.
d. Vendor dapat dipercaya dan responsif terhadap kelemahan pada *toolkit* (*bug*).
e. Kemudahan identifikasi dan serialisasi kunci.

7. Analisis

Pada WAP, keamanan dijamin lewat protokol WTLS (pada WAP 1.x) dan WAP TLS *Profile* (pada WAP 2.0). Protokol ini menyediakan layanan keamanan integritas data, kerahasian dan otentikasi. Salah satu persoalan keamanan pada WAP 1.x yang dikenal dengan "WAP *gap*" disebabkan oleh adanya saat dimana pesan berada dalam bentuk plainteks pada Wap *gateway* ketika sedang dikonversi. Namun, hal ini sudah diatasi pada WAP 2.0.

Pada SIM kebutuhan keamanan dapat mengatasi masalah keamanan umum pada *transpot Layer* seperti *peer authentication*, integritas pesan, deteksi balasan dan integritas urutan, bukti penerimaan dan kerahasiaan pesan. Setiap pesan dari aplikasi dibagi ke dalam paket yang masing-masing dijamin keamanannya dengan melindungi isi pesan dan menambahkan *header* untuk keamanan. Akan tetapi, nirpenyangkalan tidak didukung sehingga harus diimplementasikan pada level aplikasi.

Pengirim dan penerima pesan diidentifikasi sehingga penyerang tidak dapat mencoba memalsukan kecuali jika mengkloning kartu SIM. Jadi pesan SMS dapat digunakan untuk otentikasi. Lebih jauh lagi data SMS dienkripsi untuk menjamin kerahasian data. Meskipun demikian, perlindungan ini berakhir pada jaringan, tidak ada keamanan *end-to-end* dan operator seluler dan infrastrukturnya harus dipercaya ketika tidak ada lagi perlindungan terhadap pesan SMS.

Java ME menyediakan beberapa tingkat keamanan seperti *class loader*, *byte code verifier*, dan *security manager*. Tingkat keamanan ini melindungi sistem dari program yang tidak terjamin keamanannya. Kelebihan JAVA ME dibandingkan dengan WAP adalah adanya keamanan *end-to-end* (versi 1.x), penggunaan jaringan yang lebih sedikit dan mendukung enkripsi isi pesan. Pada WAP 1.x, pesan dari perangkat *mobile* dienkripsi pada WTLS dan akan didekripsi pada data TLS. Selama konversi berlangsung pada *gateway*, data berada pada bentuk plainteks. Hal ini tidak diperlukan pada Java ME sehingga memungkinkan keamanan *end-to-end*.

Penggunaan jaringan pada Java ME lebih sedikit dibandingkan dengan WAP karena Java ME memungkinkan pemrosesan data dan pembangkitan GUI secara lokal, tidak seperti WAP yang membutuhkan koneksi ke jaringan untuk setiap pemrosesan data (pada WAP bisnis logik aplikasi diambil dari web *server*). Fitur ini mengurangi kemungkinkan data yang hilang atau dicuri pada Java ME. Selain itu, pada Java ME juga dimungkinkan enkripsi *content-based* karena aplikasi Java ME dapat memproses data sebelum dikirimkan ke jaringan sehingga isi pesan mungkin saja dienkripsi jika dibutuhkan.

HTTPS diperlukan pada MIDP 2.0 dirilis pada tahun 2002. Implementasi dari HTTPS sebaiknya menghindari spesifikasi WTLS untuk menghindari celah keamanan seperti *WAP-gateway*.

Adakalanya HTTPS tidak dapat memenuhi kebutuhan aplikasi misalnya untuk kebutuhan aplikasi berikut:

a. Content based security:

HTTPS, SSL, dan TLS merupakan protokol keamanan yang tidak berdasarkan *content*. Ide dasarnya adalah mengamankan saluran komunikasi, dengan demikian akan mengamankan semua hal yang melewati semua saluran tersebut. Pendekatan ini memiliki beberapa permasalahan antara lain:

1. Harus dibangun koneksi langsung antara *client* dan *server*
   Jika aplikasi kita memiliki banyak perantara untuk menyediakan *value added service*, maka lebih dari satu koneksi HTTPS harus disambungkan. Hal ini tentunya menimbulkan potensi adanya lubang keamanan, tetapi juga menimbulkan kesulitan dalam manajemen kunci publik sertifikat. Contohnya sebuah aplikasi pada sebuah perusahaan yang memungkinkan pegawai yang bekerja di luar perusahan melakukan pembelian barang secara *online* melalui aplikasi *mobile commerce.* Akan tetapi, untuk dapat melakukan pemesanan, pegawai harus mendapatkan tanda tangan dijital dari manajer keuangan sebagai bukti bahwa permintaan tersebut sudah diverifikasi dan disetujui. Permintaan dari pelanggan yang sudah ditandatangani manajer keuangan akan diverifikasi oleh bank. Kemudian bank akan menandatangani

status pembayaran dari permintaan pegawai. Selanjutnya, permintaan pegawai yang sudah ditandatangani bank akan disampaikan kepada toko *online*. Sesudah memverifikasi tanda tangan dari bank, maka toko online dapat mengirimkan barang yang diminta pegawai.

2. Terdapat kebutuhan semua isi pesan harus dienkripsi.
3. HTTPS tidak fleksibel untuk aplikasi yang memerlukan jaminan keamanan dan performansi khusus. HTTPS kurang fleksibel untuk aplikasi yang membutuhkan *handshake* dan mekanisme pertukaran kunci khusus.

b. Distributed access control:

Aplikasi *mobile* adakalanya berinteraksi dengan lebih dari satu *server*, mengambil informasi yang dibutuhkan dari server-*server* tersebut, kemudian menyusunnya untuk ditampilkan kepada pengguna. Setiap penyedia informasi mungkin memiliki protokol otentikasi dan otorisasi yang berbeda. Tentu saja untuk melakukannya secara manual pada setiap *server* akan merepotkan pengguna.

8. Kesimpulan dan Usulan

Pada makalah ini telah dikaji bagaimana kriptografi diterapkan pada teknologi pengukung aplikasi *mobile commerce* sehingga memungkinkan komunikasi yang aman pada aplikasi *mobile commerce*. Berbagai teknologi tersebut juga sudah dianalisis kelebihan dan kekurangannya. Berikut ini adalah kesimpulan yang diambil dan beberapa usulan yang mungkin dapan digunakan untuk mengatasi kelemahan yang ada:

a. Penerapan kriptografi pada aplikasi *mobile commerce* memanfaatkan aplikasi kriptografi pada jaringan kabel, tetapi diadaptasi dan dioptimasi sehingga dapat berjalan secara efisien pada aplikasi *mobile commerce*. Bentuk adaptasi/optimasi misalnya menggunakan lebih sedikit komunikasi misalnya pada protokol *handshake* WTLS, implementasi kriptografi pada perangkat keras misalnya *smard card*, menggunakan perantara yang mengubungkan (menterjemahkan) pesan dari jaringan nirkabel ke jaringan kabel.

b. Tingkat jaminan keamanan yang diimplementasikan hendaknya disesuaikan dengan nilai dari transaksi yang dilakukan karena dalam menjamin keamanan ini sering kali dibutuhkan resource yang relatif besar. Aplikasi *mobile commerce* yang melibatkan nilai transaksi yang kecil dapat dikembangkan dengan SAT sedangkan pada aplikasi yang nilai transaksinya besar dapat dikembangkan menggunakan Wap 2.0 dan Java ME dengan penggunaan teknik kriptografi yang sesuai.

c. Umumnya pada aplikasi *mobile commerce* masih terdapat kurangnya otentikasi yang lengkap. Perangkat mobile dan server umumnya mengotentikasi dirinya masing-masing, tetapi pengguna biasanya tidak perlu membuktikan dirinya kepada telepon seluler. Oleh karena itu, seharusnya sebelum menggunakan aplikasi pengguna harus diotentikasi, hal ini penting karena perangkat *mobile* sering kali hilang atau dicuri. Dengan demikian, akses yang tidak terotorisasi pada data yang disimpan oleh aplikasi *mobile commerce* pada *device* dapat dicegah.

d. Kemampuan komputasi yang terbatas mengakibatnya terbatasnya jenis algoritma yang dapat dilakukan. Algoritma kunci publik yang sudah banyak digunakan adalah RSA. Namun, pembangkitan kunci dan *signature* dengan algoritma ini membutuhkan biaya komputasi yang besar dan akan semakin besar dengan semakin bertambahnya ukuran kunci. Dengan demikian, RSA sebaiknya digunakan untuk untuk otentikasi *server*.

   Alternatif algoritma yang dapat digunakan adalah *ECC (Elliptic Curve Cryptography)*. Pembangkitan kunci dan *signature* dengan ECC membutuhkan biaya komputasi yang lebih kecil dan memungkinkan pembangkitan kunci dan otentikasi secara lokal. Untuk enkripsi isi pesan algoritma AES dapat digunakan karena relatif cepat dan mendukung kunci yang cukup panjang.

e. Masalah *distributed access control* seperti yang telah dijelaskan pada bagian analisis tidak dapat ditangani oleh HTTPS. Salah satu solusi untuk masalah ini adalah penggunaan sebuah layanan tunggal untuk melakukan otentikasi dan otorisasi terhadap semua *server*. Sebuah server khusus dapat didedikasikan untuk dapat memberikan layanan ini. Server ini akan mengelola profil pengguna dan menyediakan *token* berdasarkan waktu seperti tiket karberos untuk mengotentikasi pengguna. Penyedia aplikasi *mobile commerce* dapat berinteraksi

dengan sebuah *server* ini untuk mengotentikasi pengguna. Selain itu, jika terdapat lebih dari satu server yang didedikasikan, maka server-server tersebut dapat terhubung jika diperlukanseperti halnya pada aplikasi Microsoft .Net Passport.

f. Pada WAP 2.0 dan Java ME terdapat mekanisme *end-to-end security*. Namun, hal ini tidak terjamin pada WAP 1.x karena pesan terdapat dalam bentuk plainteks pada ketika pengubahan format pesan di gateway.

Untuk mengatasi hal ini ada dua cara:

Cara Pertama, proses dekripsi dan enkripsi kembali semuanya dilakukan di memori. Selain itu, kunci dan data plainteks tidak boleh disimpan ke *hardisk*. Selanjutnya, memori yang digunakan untuk proses enkripsi dan dekripsi di-*free* setelah selesai digunakan. Selain itu, jika dimungkinkan oleh sistem operasi memori ini dapat dilindungi sehingga tidak ada aplikasi yang dapat mengaksesnya selama pengubahan format pesan dilakukan. Solusi lainnya adalah menempatkan *WAP gateway* pada lembaga yang dapat dipercaya seperti pada bank yang melayani pembayaran pada aplikasi *mobile commerce*.

Cara kedua adalah enkripsi isi pesan pada level aplikasi. Namun, hal ini akan menambah kerumitan aplikasi. Untuk meminimumkan *resource* yang digunakan, data yang dienkripsi hanalah data yang bersifat sensitif sedangkan data-data lain seperti tag-tag WML tidak perlu dienkripsi.

g. Masalah *content based security* seperti yang telah dijelaskan pada bagian analisis tidak dapat ditangani oleh HTTPS. Oleh karena itu, hal ini harus diatasi dengan kriptografi pada level aplikasi.

h. Untuk menjamin keamanan yang lengkap pada aplikasi *mobile commerce* penerapan aplikasi kriptografi saja tidak cukup. Kelemahan pada implementasi aplikasi juga penting diperhatikan, misalnya *bug* pada aplikasi dapat digmanfaatkan pengguna untuk meningkatkan hak aksesnya menjadi aministrator. Selain itu, faktor lain seperti kesadaran pengguna atas pentingnya keamanan dan keamanan teknologi pendukung lain (perangkat keras seperti keamanan server, BTS dan lain sebagainya) juga harus dijaga.

9. Daftar Pustaka

[DEB04] Debbabi, Mourad et al. 2004. *Security Analysis of Wirelss Java*. Concordia University, 2004.

[HAM01] Hammarstedt, Christian dan Jonas Stewen. 2001. *PKI Solution for Mobile Systems*. Växjö University 2001.

[HOW00] Howell, Ric. *WAP Security*. Concise Group Ltd. http://www.topxml.com/conference/wrox/wireless_2000/howell1.pdf. Diakses Tanggal 20 Oktober 2006

[JOR99] Jormalainen, Sami dan Jouni Laine. 1999. *Securoty in the WTLS*. Helsinki university of Technology, 1999.

[KNU02] Knudsen, Jonathan. 2002. *MIDP Application Security*. *http://developers.sun.com/techtopics/mobility/midp/articles/security/*. Diakses tanggal 1 September 2006

[MUN06] Munir, Rinaldi. 2006. *Diktat Kuliah IF5054 Kriptografi*. STEI, ITB 2006.

[NAM04] Nambiar, Seema et al. 2004. *Analysis of payment Transaction Security in Mobile commerce.* Virginia Polytechnic Institute and State University dan University of the District of Columbia, 2004

[NOK04] Nokia.2004. *Connecting mobile consumers and merchants*. Nokia white papers. http://www.nokia.com/