SNATI
Seminar Nasional
Aplikasi Teknologi Informasi 2004
Yogyakarta, 19 Juni 2004

Aplikasi Berbasis *Smartcard* pada Lingkungan Sistem Operasi Windows

Basuki Winoto
Program Studi Aplikasi Perangkat Lunak, Politeknik Batam
Jl. Yos Sudarso, Batu Ampar, Batam, 29432. Telp. +62-778-458886 ext.202
e-mail: bas@polibatam.ac.id

Abstract

Integrated Circuit Card or commonly named as smartcard is a properiatery device which is based on ISO/IEC 7816 standard. In order to expand smartcard capabilities and interoperabilites, PC/SC Workgroup defines a specification on interoperabilities between PC and smartcard. With this specification, which also has been implemented by Microsoft as Win32 API, one can develop applications which is using smartcard as small amount data storage (identification key, password, or else).

Smartcard-based applications can be developed in Windows operating system environment easily by using Win32 API library, user interface library or Win32 COM library. The application development itself can be done by using Microsoft Visual C++ or Microsoft Visual Basic.

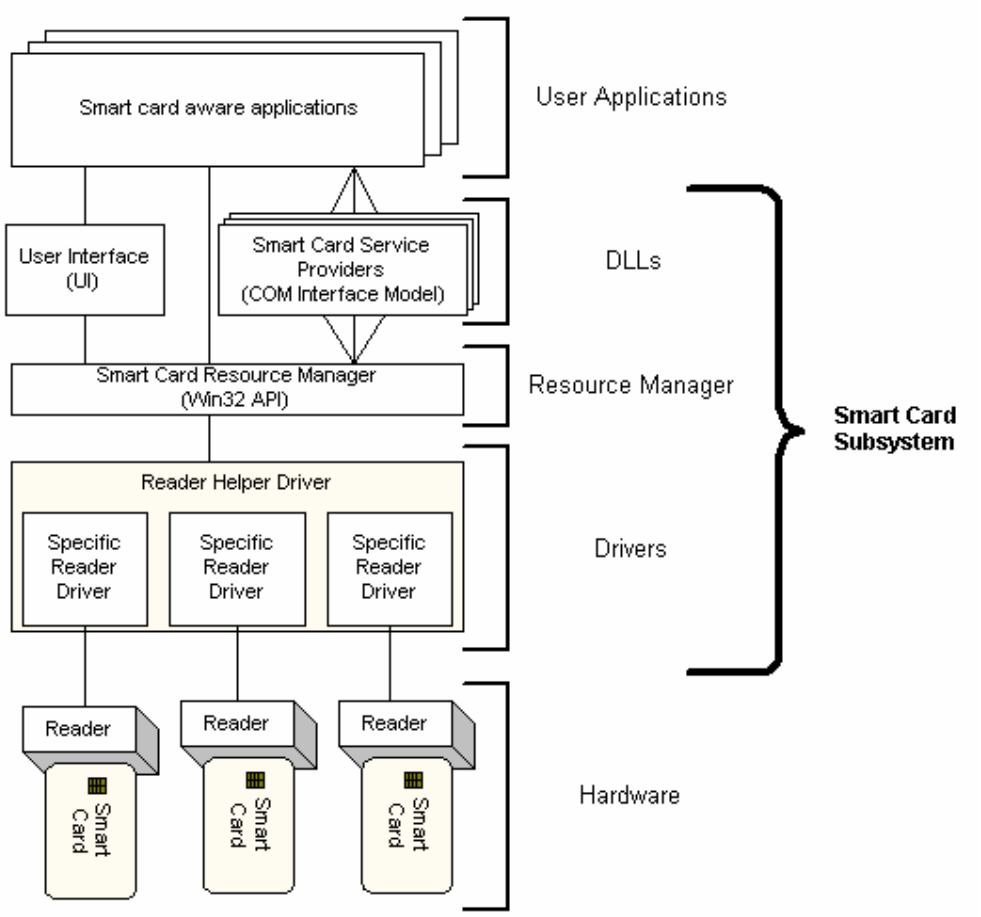
***Keywords**: ICC, smartcard, interoperable, PC/SC workgroup, win32*

1. Pendahuluan

Smartcard (atau ICC – *Integrated Circuit Card*) secara fisik berupa kartu plastik yang di dalamnya terintegrasi dengan IC (*integrated circuit*). Pada *smartcard* terdapat wilayah kontak sebanyak 8 titik yang berfungsi untuk mengalirkan tegangan dan juga sebagai antarmuka masukan dan keluaran (I/O – *input output*). *Smartcard* yang saat ini banyak beredar dan digunakan untuk baik kebutuhan perkantoran, bisnis maupun industri dibuat berdasarkan standar ISO/IEC 7816 [1].

Interoperabilitas *smartcard* dengan perangkat PC (*Personal Computer*) merupakan usaha pengembangan spesifikasi yang dilakukan oleh PC/SC Workgroup [2]. Dan hasilnya, *smartcard* dapat dimanfaatkan sebagai bagian dari suatu sistem aplikasi berbasis *smartcard*. Aplikasi berbasis *smartcard* dalam hal ini berarti perangkat lunak aplikasi yang melibatkan *smartcard* sebagai media penyimpan data terbatas.

Microsoft sebagai salah satu anggota dari PC/SC Workgroup tentu saja berusaha mengimplementasikan spesifikasi yang memungkinkan interoperabilitas antara PC dengan *smartcard* ini ke dalam sistem operasi Windows. Dukungan Windows terhadap *smartcard* bahkan telah ada sejak platform Windows 95 OSR1 dan Windows NT5. Artinya, pada platform tersebut telah memungkinkan pengembangan aplikasi berbasis *smartcard*.



Gambar 1. Arsitektur aplikasi berbasis *smartcard*
(Sumber: MSDN Library Visual Studio 6.0)

Pada gambar 1, terlihat bahwa aplikasi berbasis smartcard dapat berkomunikasi dengan perangkat smartcard melalui infrastruktur pendukung yang disebut sebagai smartcard subsystem. Smartcard subsystem dapat diakses oleh aplikasi berbasis smartcard melalui tiga cara, yaitu:

a. *Resource Manager*, mediator utama yang menghubungkan antara aplikasi dengan perangkat *smartcard*.
b. *User Interface*, pustaka (*library*) antarmuka bagi pengguna untuk melakukan pengaksesan *smartcard* melalui *resource manager*.
c. *Service Provider*, antarmuka bagi aplikasi dalam bentuk COM (*Common Object Model*) sehingga selain mempermudah juga dapat disertai dengan fungsi-fungsi layanan tambahan.

2. Aplikasi Berbasis Smartcard

Sistem aplikasi yang berbasis *smartcard* merupakan kumpulan komponen yang saling berkerja sama dalam membentuk layanan aplikasi bagi pengguna akhir dengan melibatkan *smartcard* sebagai bagian dari sistem. Umumnya, smartcard digunakan sebagai media penyimpan data terbatas seperti nomor identitas, password, digital signature, nomor rekening, dan sebagainya [4].

Sistem seperti demikian dapat dideskripsikan atas tiga bagian utama seperti dalam gambar arsitektur aplikasi berbasis *smartcard* (gambar 1), yaitu:

a. Perangkat keras (*hardware*), termasuk di dalamnya adalah *smartcard* dan *smartcard reader*.
b. *Smartcard subsystem* (*middleware*), tersusun atas lapisan *device driver*, *resource manager* dan pustaka (*user interface* maupun *service provider*).
c. Aplikasi berbasis *smartcard* (*user application*).

Prinsipnya, aplikasi akan dapat memanfaatkan sumber daya *smartcard* melalui *smartcard subsystem*. Setiap proses penulisan dan pembacaan data ke *smartcard* akan dapat dilakukan oleh aplikasi dengan bantuan *resource manager* yang mengendalikan *device driver*. *Resource manager* tersebut menerima permintaan penulisan dan pembacaan melalui *user interface*, *service provider* atau pun secara langsung.

3. Perangkat Keras

Smartcard dan *smartcard reader* merupakan komponen perangkat keras dalam sistem aplikasi berbasis *smartcard* ini. Kedua perangkat tersebut telah didefinisikan dalam standar ISO/IEC 7816, sehingga setiap *smartcard reader* dapat digunakan untuk mengakses *smartcard* dengan cara yang sama [1]. Kompatibilitas antar *smartcard reader* maupun antar *smartcard* dipastikan terjamin sepenuhnya selama *smartcard reader* dan *smartcard* tersebut mengikuti standar ISO/IEC 7816.

Gambar 2. Smartcard reader dan smartcard
(Sumber: http://www.acs.com.hk dan http://www.axalto.com)

Secara fisik *smartcard* berupa kartu plastik dengan 8 titik kontak pada permukaannya. Lima dari delapan titik kontak ini digunakan untuk pemberian tegangan, penulisan dan pembacaan data, serta untuk proses reset. Demikian pula halnya dengan *smartcard reader*, juga memiliki lima titik kontak fungsional. *Smartcard reader* terhubung ke perangkat PC melalui berbagai jenis antarmuka, bergantung pada antarmuka yang tersedia pada *smartcard reader* tersebut. Antarmuka yang digunakan dapat berupa antarmuka serial (RS-232), paralel maupun USB[1].

4. Device Driver

Perangkat *smartcard reader* dapat bekerja pada sistem operasi Windows jika telah dapat dikenali oleh sistem operasi melalui *device driver*. *Device driver* untuk *smartcard reader* umumnya telah disediakan oleh produsen *smartcard reader* dan disertakan dalam paket penjualannya. Dan tak jarang, produsen *smartcard reader* tersebut menyediakan *device driver* yang dapat didownload melalui situs webnya seperti pada gambar 3.

Device driver dari *smartcard reader* berfungsi mengoperasikan perangkat *smartcard reader* untuk melakukan proses komunikasi dengan *smartcard*. Dari hasil komunikasi antara *smartcard reader* dengan *smartcard*, akan diperoleh informasi sebagai berikut [3]:

a. Identitas dan konfigurasi *smartcard* yang diperoleh melalui ATR (*Answer To Reset*).
b. Daftar perintah yang dikenal oleh *smartcard* (*smartcard interface*[1]).
c. Nama pengenal smartcard (*friendly name*) yang umumnya diberikan oleh pengguna.
d. Informasi tambahan berupa *Primary Service Provider* (opsional)

Gambar 3. Layar instalasi driver smartcard reader

Setiap perangkat *smartcard reader* akan memiliki *device driver* tersendiri, dan umumnya masing-masing produsen mengembangkan *device driver* yang hanya berfungsi bagi *smartcard reader* produksinya saja. Bahkan untuk perangkat dengan tipe yang berbeda dari produsen yang sama, sangat mungkin menggunakan *device driver* yang berbeda.

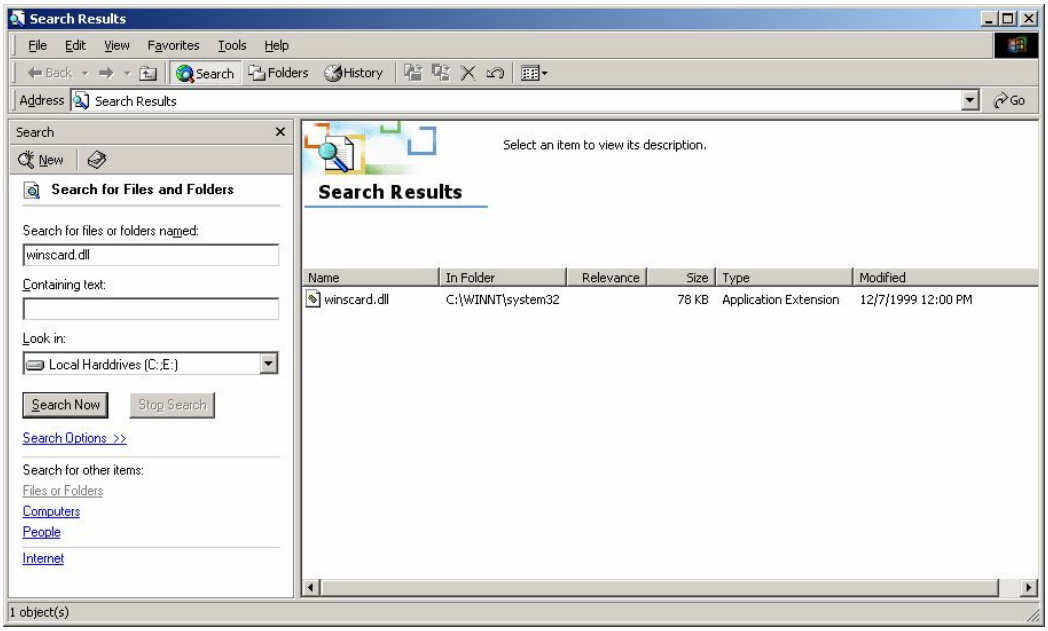
5. Resource Manager

Sebagai tulang punggung dari interkoneksi antara aplikasi dengan *smartcard* adalah bagian *resource manager* [2]. *Resource manager* bertugas melakukan pengelolaan terhadap setiap perangkat *smartcard reader* yang terhubung ke PC. *Resource manager* dapat mengendalikan *smartcard reader* melalui *device driver*. Dengan adanya *resource manager*,

[1] *Smartcard interface* dalam hal ini adalah serangkaian perintah yang dikenali oleh *smartcard* atau yang umum disebut sebagai perintah APDU (*Application Protocol Data Unit*). Spesifikasi APDU dapat dirujuk ke dokumen standar ISO/IEC 7816.

maka setiap aplikasi dapat melakukan penulisan dan pembacaan ke *smartcard* hanya dengan mengakses *resource manager*.

Pada lingkungan sistem operasi Windows, *resource manager* diimplementasikan sebagai Win32 API (*Application Programming Interface*) dalam bentuk file *dynamic link library* (DLL) dengan nama "WinSCard.dll". File tersebut dapat ditemukan di bawah lokasi direktori "%windir%\system32" seperti terlihat pada gambar 4 di bawah. Penggunaan struktur Win32 API memungkinkan pengembangan aplikasi secara langsung dengan melibatkan fungsi-fungsi yang telah terdefinisi dalam pustaka tersebut.



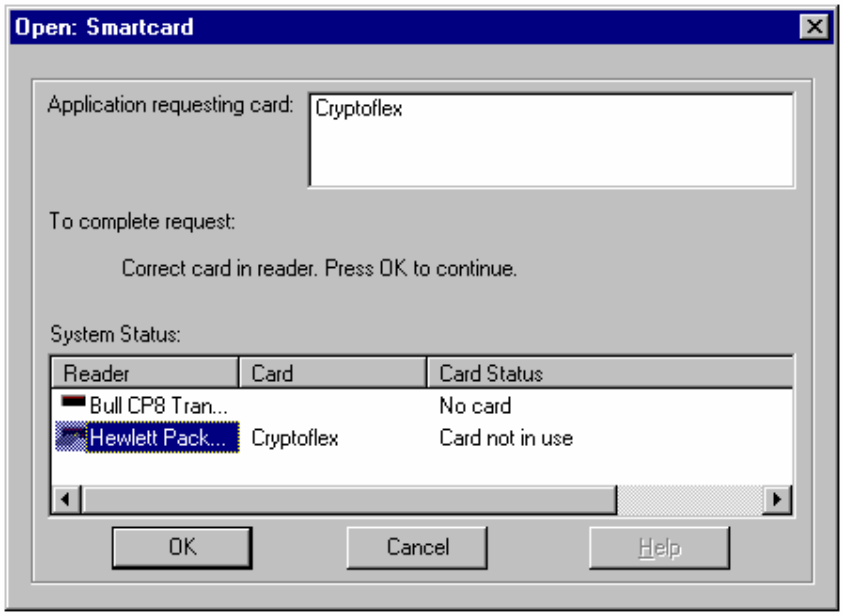
Gambar 4. WinSCard.dll.

Resource manager, yang dalam implementasinya berupa pustaka Win32 API tersebut, menyediakan fungsi-fungsi dasar untuk pengaksesan ke *smartcard reader*. Aplikasi berbasis *smartcard* dapat dikembangkan dengan teknik pengaksesan secara langsung ke *resource manager*, yaitu dengan mengakses ke fungsi yang telah terdefinisi di dalam API. Algoritma pengaksesan *smartcard* melalui fungsi dalam API WinSCard adalah sebagai berikut:

a. Membuat *context handle* yang berasosiasi dengan *smartcard reader*. Pembuatan *context handle* dilakukan melalui fungsi SCardEstablishContext. *Context handle* ini nantinya akan digunakan sebagai parameter pengenal *smartcard reader*.
b. Membuat *card handle* yang berasosiasi dengan *smartcard* dalam *context handle* (*smartcard reader*). Pembuatan *card handle* dilakukan melalui fungsi SCardConnect.
c. Fungsi SCardStatus digunakan untuk mendapatkan status *smartcard* (*card handle*). Fungsi SCardTransmit digunakan untuk mengirimkan perintah APDU (*Application Protocol Data Unit*) ke *smartcard* (*card handle*).
d. *Card handle* dibebaskan dengan pemanggilan fungsi SCardDisconnect. *Card handle* yang telah dibebaskan artinya tidak lagi berasosiasi dengan *smartcard* dalam *smartcard reader*.
e. Terakhir, untuk membebaskan *context handle* yang berasosiasi dengan *smartcard reader* dilakukan dengan pemanggilan fungsi SCardReleaseContext

6. User Interface

User Interface merupakan pustaka yang menyediakan sarana pengaksesan *smartcard* dan *smartcard reader* dalam bentuk antarmuka grafis bagi pengguna. Pustaka ini diimplementasikan sebagai sebuah kotak dialog (gambar 5) yang meminta pengguna melakukan pemilihan perangkat *smartcard reader* yang akan diakses.



Gambar 5. Pustaka *user interface*
(Sumber: MSDN Library Visual Studio 6.0)

Pustaka *user interface* dapat digunakan sebagai bagian dari aplikasi yang berjalan di lingkungan grafis Windows. Algoritma untuk menambahkan pustaka *user interface* ke dalam aplikasi adalah sebagai berikut:

a. Buat deklarasi variabel dengan tipe OPENCARDNAME
b. Beri nilai properti lpstrGroupNames, lpstrCardNames dan rgguidInterfaces (direncanakan untuk versi implementasi berikutnya[3]) untuk melakukan pencarian *smartcard* dengan konfigurasi yang lebih spesifik. Nilai properti dwShareMode dan dwProtocols juga dapat diberikan untuk menentukan spesifikasi yang diinginkan.
c. Lakukan pemanggilan terhadap fungsi GetOpenCardName untuk membangkitkan antarmuka dialog.
d. Melalui antarmuka dialog tersebut, pengguna dapat memilih *smartcard reader* yang ingin diakses kemudian klik pada tombol OK.

Penggunaan pustaka *user interface* memudahkan pengguna aplikasi dalam menentukan *smartcard reader* berikut *smartcard* yang ingin diakses. Kemudahan tersebut diperoleh karena pengguna cukup memilih salah satu di antara daftar *smartcard reader* yang ditampilkan oleh kotak dialog. Dalam kotak dialog tersebut, pengguna selain mendapatkan informasi mengenai *smartcard reader* yang tersedia, juga terdapat informasi mengenai status *smartcard reader* dan status *smartcard* di dalamnya.

Pada lingkungan sistem operasi Windows, pustaka *user interface* diimplementasikan bersama-sama dengan *resource manager* pada file "WinSCard.dll".

7. Service Provider

Selain pengaksesan langsung ke fungsi WinSCard API pada *resource manager* dan penggunaan *user interface*, aplikasi berbasis *smartcard* juga dapat dikembangkan dengan memanfaatkan blok *service provider*. *Service provider* umumnya diimplementasikan dalam bentuk Win32 COM (*Component Object Model*). *Smartcard service provider* memberikan fungsi yang lebih banyak dan bahkan lebih kompleks dari *smartcard* API dengan basis *smartcard* API itu sendiri. Jadi *service provider* dapat dikatakan sebagai pengembangan dari

smartcard API dengan tambahan fungsi yang bertujuan memberikan kemudahan dan kecepatan pengembangan aplikasi.

Pada lingkungan sistem operasi Windows, *smartcard service provider* diimplementasikan sebagai Smartcard SDK (*Standard Development Kit*) yang dapat digunakan dalam pengembangan aplikasi berbasis *smartcard*. Smartcard SDK menyediakan antarmuka bagi aplikasi berupa Win32 COM yang berisi model objek untuk pengaksesan *smartcard* dan *smartcard reader* serta pengiriman perintah APDU. Objek yang telah tersedia dalam Smartcard SDK, di antaranya:

a. ISCard: *Interface* yang mengelola konektivitas dengan *smartcard* dan *smartcard reader*.
b. ISCardCmd: *Interface* perintah APDU
c. ISCardISO7816: *Interface* yang mengenkapsulasi perintah APDU sesuai dengan spesifikasi ISO7816-4.
d. ISCardLocate: *Interface* pencari *smartcard*.
e. ISCardDatabase: *Interface* pengelola basisdata *smartcard* dan *smartcard reader* pada *resource manager*.

Aplikasi yang mengakses *smartcard* melalui *service provider* mengikuti alur algoritma sebagai berikut ini:

a. Membuat *instance* dari objek *interface* ISCard, ISCardCmd, ISCardISO7816.
b. Membangun koneksi ke *smartcard* melalui objek interface ISCard.
c. Menyusun perintah APDU sebagai objek *interface* ISCardCmd. Penyusunan perintah APDU dapat dilakukan dengan bantuan metode yang terdefinisi pada ISCardISO7816 untuk perintah-perintah APDU yang sesuai dengan spesifikasi ISO7816-4.
d. Mengirimkan perintah APDU yang telah dienkapsulasi sebagai objek *interface* ISCardCmd dengan menggunakan metode milik objek *interface* ISCard.
e. Memeriksa jawaban atas perintah APDU yang ditampung dalam objek *interface* ISCardCmd.
f. Membebaskan *instance* ISCard, ISCardCmd, ISCardISO7816 setelah selesai digunakan.

Jika dibandingkan dengan pengaksesan langsung melalui *resource manager*, maka penggunaan Smartcard SDK memberikan jaminan pengaksesan *resource manager* dengan cara yang lebih bersih dan konsisten. Dengan struktur Win32 COM, *service provider* lebih siap digunakan pada lingkungan pengembangan aplikasi berorientasi objek. Tentu saja bertambahnya lapisan penghubung antara aplikasi dengan *smartcard* menyebabkan kebutuhan sumber daya yang lebih besar. Namun, dari segi kecepatan pengembangan aplikasi, penggunaan *service provider* dinilai lebih menguntungkan.

8. Kesimpulan

Lingkungan sistem operasi Windows telah mendukung pengembangan aplikasi berbasis *smartcard*. Hal ini dirasa wajar mengingat Microsoft merupakan salah satu anggota dari PC/SC Workgroup yang menyusun spesifikasi interoperabilitas antara PC dengan *smartcard* (*PC/SC Workgroup Specification 1.0*).

Pengembangan aplikasi berbasis *smartcard* pada lingkungan Windows dapat dilakukan dengan syarat sebagai berikut:

a. Perangkat *smartcard* dan *smartcard reader* sesuai dengan standar ISO/IEC7816
b. Perangkat *smartcard reader* disertai dengan antarmuka ke perangkat PC. Antarmuka tersebut dapat berupa serial (RS-232), paralel, maupun USB.
c. Perangkat *smartcard reader* dilengkapi dengan *device driver* untuk sistem operasi Windows.
d. Perangkat PC menggunakan sistem operasi Windows 95 OSR1 atau Windows NT5 atau yang lebih baru lagi.

Aplikasi berbasis *smartcard* dapat dikembangkan dengan mengakses *resource manager* berupa WinSCard API (Win32 API) secara langsung, melalui pemanggilan pustaka *user interface* atau dengan memanfaatkan *service provider* berupa Smartcard SDK (Win32 COM). Penggunaan WinSCard API memungkinkan ukuran aplikasi yang lebih kompak namun dengan kode program yang lebih rumit jika dibandingkan dengan penggunaan Smartcard SDK. Pustaka *user interface* memberikan kemudahan dalam pembuatan antarmuka, namun terbatas dari segi kustomisasi. Dibandingkan dengan teknik lainnya, pustaka Smartcard SDK lebih siap digunakan untuk pengembangan aplikasi dengan paradigma pemrograman berorientasi objek.

Daftar Pustaka

[1] ISO/IEC, ISO/IEC 7816: Integrated Circuit(s) Cards with Contacts

[2] PC/SC Workgroup, PC/SC Workgroup Specification 1.0, http://www.pcscworkgroup.com

[3] Microsoft, MSDN Library Visual Studio 6.0: Platform SDK: Windows Base Service: Smartcard.

[4] Basuki Winoto, *Smartcard: Solusi Pemalsuan Kartu Magnetik*, Harian Sijori Mandiri, September 2002.