Worldwide Smart Card Services

André Gamache[1], Pierre Paradinas[2&3], and Jean-Jacques Vandewalle[1&2]

[1] Université Laval, Département d'informatique, Québec, Canada, G1K 7P4,
tel. : (1) 418 656 2580, fax : (1) 418 656 2324,
emails : gamache@ift.ulaval.ca - jeanjac@iad.ift.ulaval.ca

[2] Rd2p, Recherche et Développement Dossier Portable, CHRU Calmette, rue J. Leclerc,
59 037 Lille Cédex, France, tel. : (33) 20 44 60 44, fax : (33) 20 44 60 45,
emails : pierre@rd2p.lifl.fr - jeanjac@rd2p.lifl.fr

[3] Gemplus Card International, Plaine de Jouques, B.P. 100, 13 881 Gémenos, France.
tel. : (33) 42 32 50 00, fax : (33) 42 32 50 90, email : pierre@gemplus.fr

Abstract. Smart card is looked as a device for *pocket intelligence*. The future of smart cards is viewed as non-predefined and multi-purpose tools offering many services which are issued by many providers and available to fulfil bearer's needs. A way to get flexible smart cards is to store encapsulated functions code outside the card, whereas their associated data are stored inside the card. The card *operating system* becomes a secured execution environment with high level security controls onto the functions code regarding integrity and authentication. In this paper, we mainly discuss conceptual means to implement such distribution of services between *smart cards and information systems*. For that, we propose an object-oriented approach and a metaphor for smart card services based on agents. Problems addressed are *tools for internal and external smart card secured services production schemes* for distribution of services and their loading into the card. Other problems related to the distribution of objects (or agents) such as naming, object migration, or replication are discussed at the end of this paper.

Keywords. Smart Card Services, Object-Oriented Technologies, Smart Card Operating System.

1 Introduction

Service providers and end users increasingly request for new functions to be performed by Smart cards. Loading several and non-related applications into Smart cards is the next challenge. The smart card should become a mobile computer belonging to the bearer. The bearer will be able to purchase services from many providers, and load specific provider data and functions into his smart card. Data should be issued from many applications and should be used in different ways depending on services. Data could have different structures, shared by several providers, and processed by specific provider functions. An smart card application should be a set of data and functions. Functions provide executable code to create and manage data and to define security architecture linked to data structures. As the cards are widely distributed, the card data have to be protected from an user to write or update data in place of another user. That can be done by cryptographic features performing user authentication and signature functions [GQU92] [Riv90].

The main evolution concerns new supplied features that imply new card life cycle. The figure 1 resumes the card life cycle evolution and the new supplied features appear in italic.

Intervening Parties	**Current Smart cards**	**Future Smart cards**
MANUFACTURER (once)	- Define smart card operating system : HAL + data structures and users management commands	- Define smart card operating system : HAL + users management commands + *execution support environment*
CARD ISSUER (once)	- Add application specific functions - Create data structures - Create user certificates and access rights	- *Create service provider certificates*
SERVICE PROVIDERS (many times)	- Not defined	- *Are authenticated by the card* - *Create structure and data stored into the card* - *Create functions performing on those data and distribute them to card users with signatures* - *Provide keys to the card to control of the function signatures* - *Create user certificates and access rights*
CARD USERS (many times)	- Are authenticated by the card - Request the execution of commands according to their access rights - Get a response	- Are authenticated by the card - Request the execution of a service provider functions according to their access rights - *Provide the function code and its signature to the card* - *The card control the signature and execute the provided function* - Get a response

Figure 1 : Card life cycle evolution

These new trends point out requirements concerning operating system features. Smart card hardware based on 8-bits microcontrollers has rapidly evolved during the past decade. New generations of smart card microcontrollers are being developed including a RISC-Based circuit [Pey94]. It will enable the design of more powerful operating system supporting a large variety of smart card applications. Nevertheless, current smart cards (8 bits ones) can be used to implement such sophisticated operating system that will be more efficient with the next generation of smart card microcontrollers.

Current smart card operating systems are insufficient to provide services required by a non-predefined open multi-purpose smart card. They are always dedicated to a single type of data structures or to a single type of application. They cannot evolve in accordance with new services. Characteristically, operating systems should define an Hardware Abstraction Layer (HAL) to hide the implementation details of the data memory management and the communication protocol with interface devices. Mainly, an smart card operating system should offer a set of rules for supporting the implementation of an application as the relationships among its three components : data structures, functions performing on data and security architecture related to data structures. Moreover, it should provide mechanisms to enable sharing data among applications. It should also enable the evolution of services all along the smart card life cycle. There are the key features of a multi-purpose and open operating system for smart cards.

2 New Directions for Smart Card Operating System

New operating systems are under construction for the next generation of smart cards [Pey94] [PV94]. No longer will they define a set of commands used to send queries to the card and get responses. The smart card interface should provide tools and mechanisms to load programs into the card in order to perform non-predefined services. Required functions could be provided at run time by host systems and executed by the smart card operating system with high level security controls. Functions are designed by many services providers in accordance to the type of data stored into the card. Each provider can tailor his own services without sharing application code into the card. The services can be scattered all over the world in order to make them available wherever and whenever the card is used. Smart cards become portable secured personal data servers with a highly secured execution environment providing services onto the data they stored. In this paper two technologies are discussed for implementing such smart card services.

2.1 Object-Oriented Technology

The first one based on object-oriented technologies is derived from the concepts by Birrell and al [BNOW94]. A *card object* is an object whose methods can be invoked by host programs, in addition to the provider that allocated the object into the card. The program invoking the method is called *client* and the program containing the card object is the *card operating system*. A remote reference in a client program points out to a local object whose methods perform procedure calls to the card's objects. The former are then processed by invoking provider owned methods. In addition to perform procedure calls to the card, the local object provides the corresponding code of the invoked method (see figure 2). The client program does not need to know anything about the method invocation. The local object is called a *surrogate* (also known a *proxy*). The provider uses a *stub generator* to declare the surrogate object *type* which is a subtype of the card object type. It also provides the object card *interface* as an *opaque type* for the client. The object card data fields are stored persistently inside the card, and the object card methods are implemented by the surrogate objects which are also in charge of the transport of methods body and messages between itself and the card operating system.

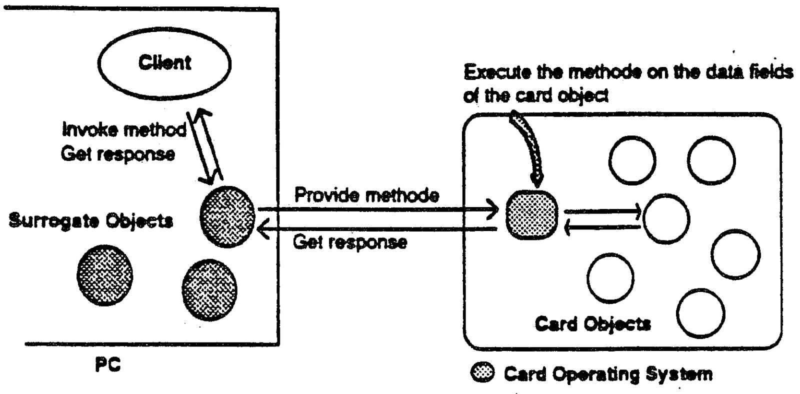

Fig 2. Card object invocation.

2.2 Agent-Based Technology

The second approach, an agent-based one, is compatible with the first one and can be viewed as an abstraction above object-oriented methodologies and languages. An *agent* is a software unit characterized by an ability to negotiate with other agents in order to achieve specific common goals or objectives. In object-oriented implementations, agents negotiate through message passing and method execution. An important benefit is that the agent-passing mechanism enables bundling of messages, requests, and preferences into an intelligent program that travels to a distant computer, gets answers to all the queries, and then returns (see figure 3). Agents can be sent over networks to perform pre-defined operations on distant host systems where they are *interpreted* locally to perform whatever its *originator* puts into it. The sites where agents are interpreted are called *agent acceptors*. Providers, as originators, create locally their agents and can dispatch them over networks in order to be timely available to many agent acceptors. Because smart cards have to protect themselves against misbehaviour of agents, the following security features have to be considered : ascertainment of the authorship of an agent, control of the integrity of an agent, control of access-rights to specific information located in the smart card, prevention of the repudiation of a transaction performed by an agent, and control of an agent life-cycle.

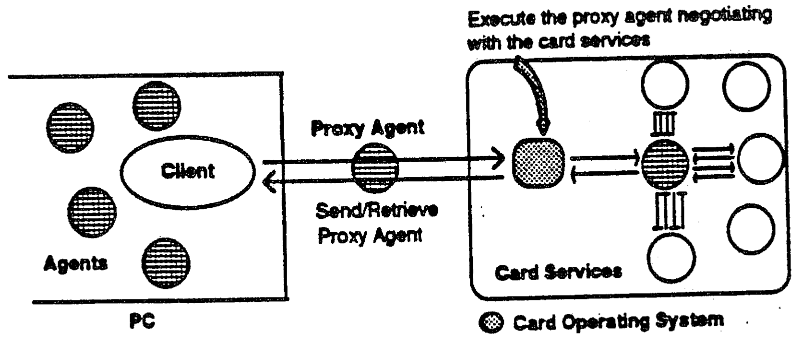
Execute the proxy agent negotiating with the card services
Proxy Agent
Client
Send/Retrieve Proxy Agent
Agents
PC
Card Services
Card Operating System

Fig 3. Agents enable requests and answers to not travel over the network.

2.3 Security Considerations

Any authorized service provider (SP) will have to develop a set of primitives methods to define their own objects stored on each individual card. For instance, such a set should include methods as credit and withdraw for an electronic purse object. Every object's behavior is therefore bound by their method interfaces to a set of primitive whose basic logic implemented in their body can change upon time. An object in a card is thus defined with a structure and a set of method's interfaces. Each object is logically bound to each of its method by a private and unalterable certificate which can be deciphered only by the card operating system. Following a request from a card bearer, the SP creates onto the card an object (structural and behavior parts) instantiated from its class and initializes it with appropriate initial values. It makes also its method's body available to every authorized user site. This program code should be transferable to the card with authentication of source and code integrity measures in such a way that there is no doubt about the method's integrity and its ID supplier. Also, the SP should provide a trusted user site with a certificate for each method whose secret content allows any card to validate the user's right to run it. This secret information should be stored in secure place, for example in a smart card. Finally, the SP puts in a card a secret information to validate the certificate of each method. This secret information will allow a card to verify that a user site has the right to use a method whose source's authentication and method's integrity can be verified.

When a method has to be run, the object requests a specific method from the user site which responds by sending back its code body. On receiving the method's certificate and its body, the card is able not only to authenticate where it is coming from but also to validate its integrity. In case of a positive checking, the method is run in protected mode by the card which may transfer a response back to the user. Once imported, a method is kept inside the card available for other object requests during the same session.

3 Smart Cards Integration into Information Systems

The two above solutions are strongly based on a cooperation between smart cards and information systems. Smart cards and their related applications have to be integrated into large scale networked and wireless communication architectures. No longer will they develop and use specific ways of working to create, design, distribute, access and process their services. Rather they could take many advantages of the works about distributed environments such as *CORBA* [OMG94] which specifies an *Object Request Broker* (ORB) providing mechanisms by which objects transparently make requests and receive responses. The ORB provides interoperability between applications on different machines in heterogeneous distributed environments. Smart cards are specific machines to be integrated in such architecture. For instance, data tools for internal and external smart card services production could be inspired by an *Interface Definition Language* (IDL) which is the CORBA language used to define distributed objects. IDL or a similar language could be used to describe interfaces that client objects call and which are implemented by card objects. Proxies could be implemented as *Basic Object Adapters* (BOA) to access card objects and ORB functions. A proxy object could be used to generate and interpret card object references, authenticate the caller, activate and deactivate card objects, invoke methods through skeletons and provide method bodies to the card by invoking a provider's objects server. Moreover, CORBA could be used to implement security protocols to control the distribution of the provider's objects, and to manage them like other network objects by means of services such as :

- *Naming* : object names are fundamental to identify and access various object implementations. Names are human-comprehensible values that identify an object. The name service provides a mapping of names to CORBA object references. As the card objects are located into cards, specific card object names should be used. They could be defined by the providers using a mechanism of indexing such as :

  Card.CardSerialNumber.ServiceProviderName.ObjectIDentification

  The keyword *Card* is used to refer to a connected card, the *CardSerialNumber* is given by the card at the connection time (answer to reset), and the *ServiceProviderName* and *OID* are defined by the provider. The name service should enable binding and unbinding card objects names and CORBA object references.
- *Replication* : the replication service provides management of consistency for the explicit replicated copies of provider's proxies in a distributed environment. As providers shall distribute their proxies to many services consumers, the mutual consistency among replicas is a primary requirement. When updating a proxy, the provider has to propagate the changes to all other replicas.
- *Transaction* : the transaction service provides support for ensuring that a computation consisting of operations on card and host objects provides some or all of the ACID properties (Atomicity, Consistency, Isolation and Durability).

Implementing a transaction service requires a transactional infrastructure and facilities which take into account smart card features such as non-power supplied, mobile, and intermittently connected devices.

- *Licensing* : the licensing service controls and manages remuneration of providers for card object services. The licensing scheme may also incorporate the value of the information being accessed and establish related charging schemes between consumers and suppliers of services.

Agents are compatible with objects. The CORBA architecture could be a good transport layer and location manager of distributed agents. Before long, a special interest group could be created within the OMG (Object Management Group) to address issues regarding agent standardization. Agents can be viewed as particular objects, with capabilities to negotiate with other agents and travel over a network. Therefore, they could be easily integrated in the ORB architecture. Telescript, the language used by General Magic to execute agents code within agent acceptors, could become a *de facto* standard. Smart cards, as agent acceptors, could host such an interpreter (at least a reduced version of it) to enable agent execution with a recognized high level of security [GQU92] [Riv90]. A language mapping from IDL to Telescript could be implemented to ease the development of card objects. Security protocols are very important and must be integrated in such an organization. Thus, mechanisms of double authentication, signature verification, message ciphering and deciphering, using public key or zero-knowledge algorithms could be implemented between smart cards and host machines.

4 Conclusion

The approach described in this paper is based on a merge of many technologies such as object, agent, CORBA, Telescript, smart card operating system, and cryptography. This envision underlines a lot of work. Nevertheless, it seems to be well suited to enable implementation of smart cards as open systems which can provide non-predefined and cooperative services making use of the coming electronic data highway. The distribution and use of services all over the world can be eased by the integration of various proposals of standards such as CORBA.

References

[BNOW94] A. Birell, G. Nelson, S. Owicki, and Edward Wobber. *Network Objects*. Digital. Systems Research Center. Research report n° 115. February 1994.

[GQU92] L. C. Guillou, J-J. Quisquater, and M. Ugon. *The Smart Card : A standardised Security Device Dedicated to Public Cryptology*. Contemporary Cryptology, ed. G. Simmons, IEEE-Press, 1992.

[OMG91] Object Management Group. *OMG Common Object Request Broker Architecture: Architecture and Specification (CORBA)*, Revision 1.1, OMG Document Number 91.12.1, December 1991.

[Pey94] P. Peyret. *RISC-Based, Next-Generation Smart Card Microcontroller Chips. in* proceedings of CardTech'94, pp. 9-36, Washington D.C., U.S.A., April 1994.

[PV94] P. Paradinas and Jean-Jacques Vandewalle. *New Directions for Integrated Circuit Cards Operating Systems*, Sixth ACM SIGOPS European Worshop about "Matching Operating Systems to Application Needs", Warden, Germany, September 1994.

[Riv90] R. L. Rivest. *Cryptography, in* Handbook of Theoretical Science, ed. J. van Leeuwen, Elsevier Science Publishers B.V., North-Holland, pp. 719-755, 1990.