CARD-CENTRIC FRAMEWORK - PROVIDING I/O RESOURCES FOR SMART CARDS

Pak-Kee Chan, Chiu-Sing Choy, Cheong-Fat Chan, and Kong-Pang Pun
Department of Electronic Engineering, The Chinese University of Hong Kong
{pkchan, cschoy, cfchan, kppun}@ee.cuhk.edu.hk

Abstract: The Intelligent Adjunct (IA) model, proposed by Balacheff et al in [1], is a novel paradigm for smart card applications, in which off-card resources are provided for smart cards to run application logics. This paper presents the Card-Centric Framework as an evolution of the conceptual IA model to provide a more rigorous solution for smart cards to access off-card I/O resources. It consists of a system model and a communication protocol. With the Card-Centric Framework, smart cards can run any applications that involve user interactions. A system prototype constructed with the current smart card technologies showed only reasonable performance. To cater for performance issues, another demo system that made use of enhanced smart card technologies was implemented. It not only shows a significant improvement in performance, but also proves the feasibility of the framework in the future.

Key words: Card-Centric Framework, Console, I/O, middleware, protocol, smart card, system, user interaction

1. INTRODUCTION

Smart card has a tamper-proof property that makes it an ideal authentication device of the connected terminal. However, it is often equipped with limited processing power that is just enough for security usage. This has been, and will be, limiting smart card's potential of use in the conventional client-server security model, in which smart card is just a slave of the connected terminal. With the continual advancement in microelectronics technologies, the processing power of smart card is being enhanced, and a sufficient condition is available for a change in smart card application model.

In [1], Balacheff et al proposed the Intelligent Adjunct (IA) model to replace the traditional client-server topology by a peer-to-peer one. The model involves two entities: smart card and the connected terminal. The smart card acts as an independent processor, whereas the terminal provides essential resources for use by smart card. They cooperate with each other to run applications, which may involve the use of off-card I/O resources for user interaction.

However, the details of implementations were not clearly defined in [1],

Therefore, we move from concept to implementation, and propose the Card-Centric Framework as an evolution of the IA model. To allow for smart cards to handle off-card resources, we introduce an entity called Console to provide smart card with connected I/O devices. With the framework, any applications that require basic user interactions could be run on smart cards, thus the limitation of smart card as a security device could be relieved.

The rest of this paper is organized as follows. The next section provides a brief image of the Card-Centric Framework. The framework consists of a system model and a communication protocol, which will be described in sections 3 and 4 respectively. Section 5 demonstrates a system prototype based on the framework. Due to limitations of existing smart card infrastructures, it showed only reasonable results. To cater for performance issues, we employ the technologies predicted by RESET roadmap [2] and implement an improved system, which will be described in section 6. Section 7 discusses security issues in employing the framework. The last section sums up with a brief conclusion.

2. CARD-CENTRIC FRAMEWORK

The Card-Centric Framework adopts some of the principles from the Intelligent Adjunct model [1], including:

a) Migration of application logic from the card-connected terminal to the smart card;
b) Adoption of active role by smart card, and;
c) Provision of terminal and network resources for smart card.

It is designed for smart cards to handle off card I/O resources. It describes a more rigid guideline for implementation, including a system model and a communication protocol.

The system model, which conforms to the criteria of middleware for system integration in RESET roadmap [3], consists of two entities: Smart Card and Console. Smart Card, which is interpreted as IA in [1], runs all the application logic. Console is the essential portion of the card-connected

terminal that provides Smart Card with off-card I/O and network resource entities.

The resource entities are objects to be handled by the smart card. The smart card not only sends out manipulation commands for them, but also awaits interrupts from them when events arise. To achieve this, an object-oriented protocol, titled Card-Centric Protocol (CCP), is proposed for the communication between Smart Card and Console.

Details of the system model will be described in section 3, whereas the protocol in section 4.

Under the framework, software developers can employ user interactions through I/O resources, and thus enhance user-friendliness. Provided that both the smart card and the card-connected terminal support the Card-Centric Protocol, they can develop any applications to run on the smart card. Consider the application of IA as an example. It is originally proposed by [1] for users to configure user-automated tasks by means of programming. However, smart card owners are often not capable of programming works. With the Card-Centric Framework, the users can get rid of coding and input the tasks through user-friendly interfaces such as display, sound, mouse, etc.

Unlike the IA model, the Card-Centric Framework considers only the case where the smart card makes use of resources on the connected terminal. Therefore, instead of peer-to-peer model, a master-slave one is adopted, but in the inverse direction. In other words, smart card swapped the role with the terminal and becomes the master. Whenever they need to make use of the resources of each others, we suggest adopting the Proactive SIM method as proposed in [1], and running CCP on top of it, such that both Smart Card and Console could become peers.

3. SYSTEM MODEL

As shown in figure 1, the system consists of two major components: the Smart Card and the Console. The Smart Card, similar to a brain, is the core processing unit of the whole system. The Console, like a mere body, provides various off-card resources for use by Smart Card. These resources, including I/O peripherals and Services, are coordinated by a Bridge embedded in the Console. In this section, each of the system components will be introduced.

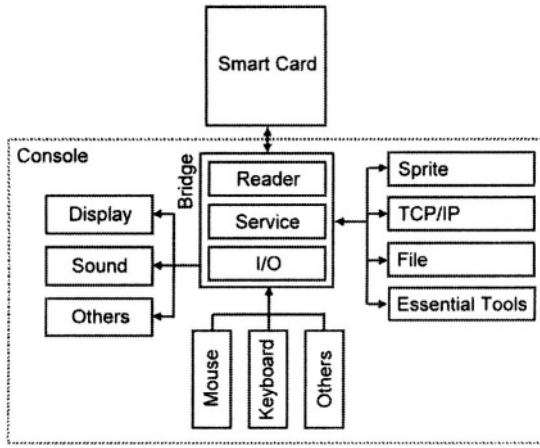

Figure 1. System model of Card-Centric Framework

3.1 Smart Card

The Smart Card not only handles the core applications of the whole system, but also the off-card resources on the Console. It manipulates them by sending commands to the Console. On the other hand, it handles I/O interrupt signals sent from the Console, and then takes the corresponding actions.

In order to achieve a mapping between the on-card resources and the off-card counterparts, interface objects and methods describing the features of Console are available on the card side. When the on-card application calls those methods, the interface objects will send requests to the Console, and then wait for returned data. This mechanism is similar to conventional smart card Remote Method Invocation (RMI) [4][5], but in a reverse manner, such that the card can invoke any remote methods supported by the Console rather than being invoked.

3.2 Reader

The Reader is the serial communication channel between Smart Card and Console. It should be ISO7816 [6] and PC/SC [7] compatible. ISO7816 is a communication protocol designed for smart card, whereas PC/SC is a standard for the integration of smart cards to personal computers. Conventional card readers fix the link speed to only 10kbps. Therefore, it may become a burden of the whole system, especially when the on-card application makes use of resources from fast networks.

3.3 I/O Peripherals

I/O is the major user interface with the on-card application. Each I/O resource is an object. Examples of input are mouse and keyboard, whereas examples of output are display and sound. The output objects could be accessed using the on-card interface objects, whereas the input ones involve the use of interrupts.

On arrival of user event, which may be a mouse move or a key press, the Console will send an interrupt packet to the Smart Card. After receiving the interrupt, the Smart Card will send requests to the Console for more information on the I/O generating the interrupt. The information, such as the coordinates of the mouse cursor or the key being pressed, are abstracted by the Console and then sent back to the Smart Card for further processing.

3.4 Services

In order to save the precious on-card processing power, heavy-weight off-card resources should not be manipulated in detail. For instance if a 24-bit display of 640x480 resolution is manipulated directly by the card dot by dot on a 25fps basis, it will require at least a 180 Mbps serial link for smooth display disregard traffic and processing overheads, and the link that is in Kbps order will then become a bottleneck. Since each Send/Receive operation involves the movement of data into or out of the input/output buffer that are often flash memory devices, larger bandwidth means higher processing power requirements. To cater for this, we make use of abstract instructions, in which the vast amount of operations required to finish a certain task are organized to form one general instruction. Related instructions are further grouped as a Service. Similar to I/O Peripherals, each Service entity is an object. Through Service, the Smart Card can indirectly manipulate I/O resources with lower hardware requirements.

There are four featured groups of services in the Console: Sprite, TCP/IP, Files and Essential Tools. Sprite, an essential technique of 2D game programming, is responsible for basic graphics and text manipulations. With Sprite, the Smart Card can off-load the tedious job of graphics manipulations to Console, and lowers the link usage as well as on-card processing power requirements. TCP/IP adapts Smart Card to the internet, such that the Smart Card can off-load the vast amount of handshaking, packetization and similar jobs necessary for TCP/IP connections to the Console, and handles them by simple pointers to sockets and read/write/listen commands. Files, which are used as temporal storage of data, release the pressure of on-card storage requirements. Essential tools help in the management of resources (e.g. allocation and deallocation) and may sometimes be useful in debugging.

3.5 Bridge

The core of Console is the Bridge that coordinates communications between the reader, I/O and Services. It functions as a message router that runs all traffic over the ISO7816 communication protocol [6]. It parses commands sent from the Smart Card, routes them to the appropriate Service or I/O objects, and then passes back the returned data to the Smart Card. In other words, it is similar to the resource manager of the Intelligent Adjunct model [1]. Moreover, it passes the interrupts generated by resource objects to the Smart Card, and then awaits the subsequent I/O requests. With the Bridge, the Smart Card gains accessibility to any connected off-card resources.

4. CARD-CENTRIC PROTOCOL

4.1 Protocol Layering

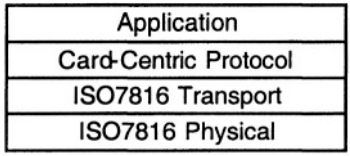

Figure 2. Protocol layer model in Card-Centric Framework

Figure 2 shows the protocol layer model involved in Card-Centric Framework. The CCP layer is located in between the Application and ISO7816 Transport layers, where the former is developer dependent and the latter handles data transfer between Smart Card and Console. In other words, we embed the CCP packets as payloads of ISO7816 messages, leaving the implementations of the underlying layers unaltered. Therefore, it is simple to integrate the framework into existing smart card infrastructures.

4.2 Packet Format

There are two types of CCP packets: Request and Response.

Request packets flow from Smart Card to Console, by which the Smart Card initiates requests and acquires an active role. As mentioned before, each I/O or Service entity is an object. In order to access those off-card objects, the Smart Card must provide the *Pointer* to identity the object, the instruction byte *INS* and the necessary *Parameters* to manipulate the object.

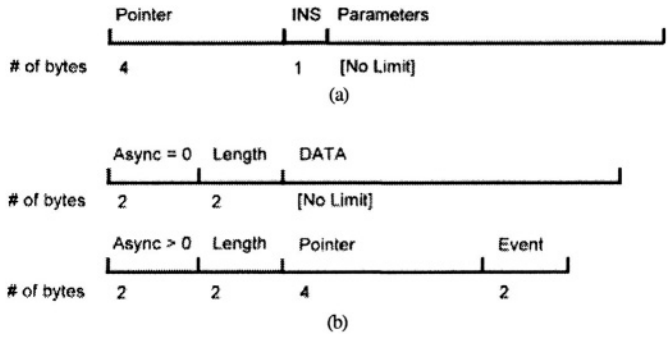

Figure 3. Packet format (a) from Smart Card to Console and (b) from Console to Smart Card

They are arranged in the Request packet sent from Smart Card to Console, as shown in figure 3.

In reply to each Request, there is a Response packet flowing in the reverse way. There are two types of response packets: Synchronous and Interrupt. Synchronous packets (*Async*= 0) are responses to request packets. It consists of the corresponding results *(DATA).* Interrupt packets (*Async* > 0) are issued when there are events from the resource objects. It tells which object is involved *(Pointer)* and how it is involved *(Event).* An alternative to the Interrupt packet is the ENVELOPE command of SIM APDUs, which allows Smart Card to obtain interrupt events from Console [8] [9]. In such case, the Pointer and Events are arranged in the ENVELOPE APDU. In order to support non-SIM environments, the Interrupt packet will be more preferable. The *Async* word not only indicates if a packet is an interrupt, but also the type of the resource generating the interrupt. In case where several objects share the same type, such as various TCP/IP connections sharing the type Socket, the *Pointer* can indicate the exact object involved.

4.3 Handshaking Sequence

The general handshake sequence between Smart Card and Console is illustrated in figure 4. Each handshake cycle consists of two sequences: request and response. In reply to the Nth request from Smart Card, the Console sends synchronous response #N. Suppose a user event arrives right after the issue of the N+1th request, the Console suspends the process of the current request, and generates an interrupt packet. Receiving the preemptive interrupt, the Smart Card handles the I/O object involved by request #N+2, and then sends a "No Operation" request at the end. The Console detects the request, and then restores the suspended request #N+1 and continues the communication sequence.

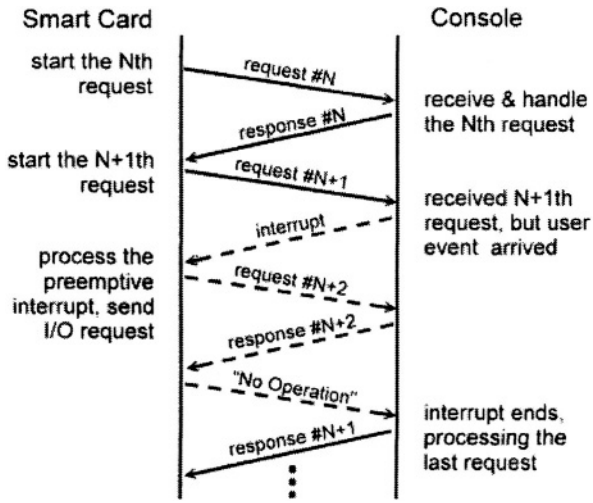

Figure 4. General communication sequence between Smart Card and Console, dashed lines indicates the sequence of interrupt requests.

ISO7816 is originally designed for passive smart card access, in which the card-connected terminal initiates a request whereas the smart card replies with the corresponding results [6]. To initiate smart card-active communication, we employ a slight modification on the smart card's conventional response to the ISO7816 SELECT command at the very beginning of communication, as shown in figure 5. Instead of sending only the conventional "No Error" message (90 00h) and then waiting for incoming requests as in conventional ISO7816 applications, the Smart Card immediately initiates the request-response sequence and takes the active role. In the Intelligent Adjunct model that makes use of SIM APDUs, a similar mechanism was proposed [1], as shown in figure 5(b). However, before starting the request-response sequence, the smart card (or IA) has to wait for the FETCH request sent from the Terminal after the "No Error" message (91 XXh). Therefore, an extra handshake cycle is required for the smart card to take an active role.

In our proposed protocol as shown in figure 5(c), we use neither FETCH nor "No Error" but let the Smart Card issue request immediately after SELECT. Therefore, one request-response cycle is saved.

4.4 Object Allocation

The Smart Card must allocate objects on the Console before manipulating them; otherwise it will receive an error warning that the referred object is invalid. Objects could be either pre-allocated or

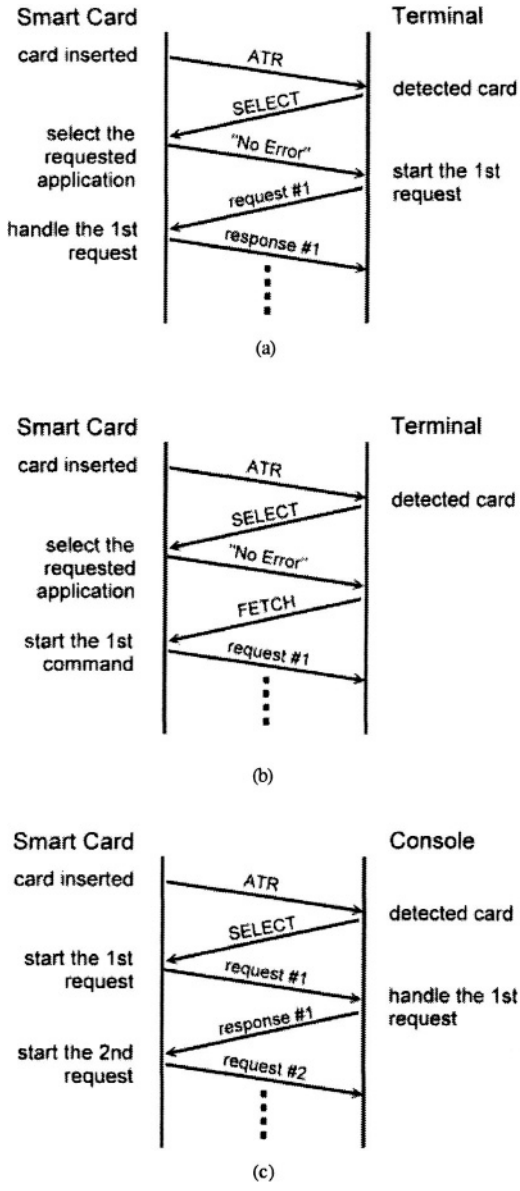

Figure 5. Initialization phase of (a) conventional ISO7816 application, (b) Intelligent Adjunct model and (c) modified communication sequence. ATR is the Answer-To-Reset string sent from smart card right after power-up.

dynamically allocated. Pre-allocation, which is performed automatically since the power-up of Console, results in an intrinsic NULL Pointer. The only object that requires pre-allocation is the Essential Tools Service. Dynamic allocation results in a non-null *Pointer* to be referred by the Smart Card. It is handled by the Object-allocation Instruction (INS = 00h) of the Essential Tools Service object.

The framework supports a maximum of 65536 types of resource objects. The first 1024 types (0-1023) are reserved for standardization, and their manipulation instructions should be the same across different Console implementations. Application specific objects should be implemented with type 1024 or above.

Depending on the need from applications, more than one object could be allocated for each type of resource. For example, several sockets may be required in some applications, whereas only one mouse is needed in most cases. After allocation, the properties of those objects could then be altered by the instructions supported.

4.5 Accessing On-card Resources by Console

The Card-Centric Protocol is designed for on-card applications to make use of resources on the Console. However, there are circumstances where the Console needs to make use of resources on the Smart Card. A similar case is Proactive SIM: it not only initiates requests to the connected handset (or Console), but also accepts requests from the network operator for administrative features [9], where the requests are routed trough the handset. In such case, the on-card application has to issue requests to as well as accept requests from the Console.

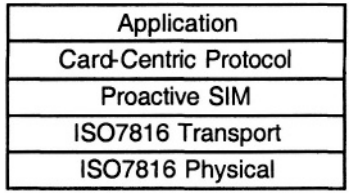

Figure 6. Protocol layer model with Proactive SIM

To cater for this, we suggest running the Card-Centric Protocol over the Proactive SIM protocol [8] [9], as illustrated in figure 6.The CCP Request and Response packets should be encapsulated in the Proactive commands and responses, whereas Interrupt packets in ENVELOPE commands, as shown in figure 7.

Note that adapting CCP to the Proactive SIM Protocol introduces three extra messages for each request-response pairs: status word 91 XX, FETCH

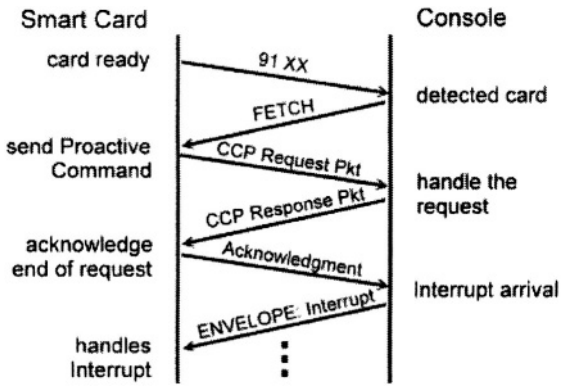

Figure 7. Communication sequence of CCP over Proactive commands

and Acknowledgment. Despite the Acknowledgment message that may consist of the status word 91 XX, this solution results in a 50% overhead. Each message transmitted not only occupies bandwidth, but also precious processing power, especially for smart cards that are equipped with slow flash memories. Therefore, it should be applied only when necessary. In case where only Smart Card issues requests to Console, we suggest eliminating the Proactive SIM layer, and adopting the simplified model of Figure 2.

5. SYSTEM PROTOTYPE

A system prototype is implemented to demonstrate the feasibility of the Card-Centric Framework. It consists of a typical smart card and a software-emulated Console. The smart card specifications are available in table 1. It cannot store high resolution graphics in just 64KByte of memory. In order to reduce on-card memory usage, the graphics are pre-stored in a web server on the same LAN as the Console, and then fetched and stored as off-card files whenever needed. The Console emulator is written in Visual C++ for x86 computers. It provides mouse, keyboard and display for the smart card to handle user interactions. It also adapts the smart card to the internet by

Table 1. Specification of the Smart Card of system prototype

Model	Gemplus GXP PRO-R3
Processor speed	4 MHz
Memory type	Flash Memory
Memory size	64 KByte
On-card OS	JAVACard 2.1.1
Communication port	10 Kbps serial
Communication protocol	CCP /ISO7816 transport /ISO7816 physical

Table 2. Specification of the Console of system prototype

General Information		
Development Environment:		Microsoft Visual C++ 6.0
Off-card Resources		
Type #	Name	Features Supported
0	Essential Tools	Object allocation and deallocation; Unlimited number of objects; Debugging by message box.
1	Mouse	Interrupt; 3 buttons.
2	Keyboard	Interrupt; 101 keys.
3	Bitmap	Context for raw bitmaps.
4	Font	Context for fonts.
5	Sprite	512x384 resolution; 24-bit color depth.
6	Off-card Files	Max. 4 GB file size; Clear on end of session.
7	Socket	Interrupt; TCP/IP/Ethernet@100 Mbps; Support create, accept, read, write.

Table 3. Performance of system prototype

Max. application time	120 s
Min. application time	100 ms
Transport layer throughput	1.5 Kbps

TCP/IP socket objects. More details of Console could be found in table 2.

The on-card application is a Tic-Tac-Toe game that handles both game arithmetic and user interactions. The game waits for the first move from human player, and then counterattacks using 6-ply minimax procedure and alpha-beta algorithm [10]. User interactions are accessible through the on-card interfaces as described in table 2.

Table 3 shows the performance of the system prototype. The performance parameter involved is the application time measured by the Console between two successive requests disregard the time required for fetching off-card graphics. The results could be affected by two factors: communication overhead of the serial link and the processing power of the smart card. Communication overhead of the 10Kbps serial link, which is in the order of 10^{-4} s, is insignificant when compared with the measured result. Therefore, the result is due to the poor processing power of the smart card, especially the slow write speed of the on-chip flash memory. The minimax procedure involves a vast amount of assignment instructions to the slow flash memory, and thus the 120s required to accomplish one request-response cycle is reasonable. Even the fastest program segment that involves only the copy and transmission of memory contents requires 100ms of application time. These measured values, when compared with the maximum $230\,\mu$s for a 6000MIPS x86 processor (Pentium 4 running at 2.4GHz) to run the same application, are unacceptable for users.

Therefore, it is obvious that the current smart card technologies are not suitable for memory intensive applications.

6. DEMO SYSTEM BASED ON TECHNOLOGIES FROM THE FUTURE

Fortunately, European smart card industry and academic stakeholders noticed that smart card has yet to improve its performance and features for being acknowledged as a new generation element in the consumer domain, and therefore introduced the RESET roadmap to identify and address the major challenges of the smart card industry [11]. According to the roadmap, the smart card should, in the near future, be equipped with faster memory such as floating gate memory, FeRAM or MRAM [2]. The fastest of them could achieve an access time as short as 10ns.

In order to demonstrate the feasibility of Card-Centric Framework over the future technology, the smart card in our demo system is replaced by a system-on-programmable-chip that makes use of comparable memory (SRAM). It is an FPGA-based development board from Altera [12] configured to simulate the behavior of a smart card, and thus called *Smart Card* hereafter. Its specifications are listed in table 4. Similar to smart card, it has two types of on-board memory: SRAM and Flash. Flash memory is used for application storage, whereas SRAM for program execution. For full utilization of the 50MHz SRAM, a processor running at the same speed is implemented on the FPGA. With this configuration, the application time required for those memory intensive procedures in the game could be significantly reduced.

With a great enhancement in processing power, the application time is predicted to be in the order of 10^{-4} s, thus comparable to the communication overhead of the serial link that is in Kbps order. In order to reduce communication overheads for more reasonable measurements, the original serial link is now replaced by l00Mbps Ethernet.

The Console Emulator and on-card application used in the system prototype of section 5 were adopted. To cater for the change in communication link, the communication drivers are modified to run over TCP/IP. To achieve this, both Smart Card and Console are equipped with a TCP/IP stack, and offered a unique IP address. With the assumption that the

Table 4. Specification of the Smart Card in the 2nd system

Model	Altera Nios development board, Stratix Ed.
Processor speed	50 MHz
Memory type	SRAM, Flash
Memory speed	50 MHz
Memory size	1 Mbyte (SRAM) 8 Mbyte (Flash)
On-card OS	N /A
Communication port	100 Mbps Ethernet
Communication protocol	CCP /ISO7816 transport /TCP /IP /Ethernet

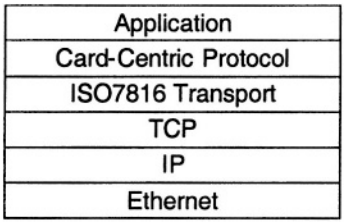

Figure 8. Protocol layer model in 2nd system

Smart Card's address is known, the Console can connect to the Smart Card as usual, and then encapsulate ISO7816 messages in TCP/IP packets. This is illustrated in the protocol layer diagram in figure 8. On top of ISO7816, the Card-Centric Protocol is run without modifications.

Table 5 shows the performance of the 2nd demo system. With an enhancement in smart card hardware technologies, the application speed increases drastically. The link utilization increases 200 times, therefore the choice of high speed Ethernet is appropriate.

The application speed is not as fast as Pentium 4, yet the application delay is unnoticeable by end users. Namely, the user interface is smooth enough. In fact, applications other than games usually require lower processing power. Consider the example application of [1], where the end user configures the smart card to run user-defined tasks through Console,. The on-card application may only need to receive a command string from the keyboard and then store it in the on-card memory. These simple procedures require far lower processing power to fulfill than games. Therefore, the near-future technologies predicted by [2] are enough for the Card-Centric Framework.

Table 5. Performance of 2nd system

Max. application time	1 ms
Min. application time	520 μ s
Transport layer throughput	304 Kbps

7. SECURITY ISSUES

Smart card, originally designed for security, has a tamper proof property that favors the protection of data stored inside. The Card-Centric Framework inherits this benefit, but does not consider the security of data transferred outside the smart card. Therefore, software developers are required to handle the security of such case.

In the framework, the information transferred between smart card and Console are mainly I/O signals, which may sometimes be critical

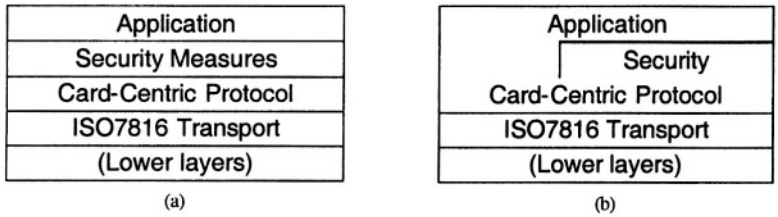

Figure 9. Protocol layer model for security: (a) A separate layer for security, (b) Security measures integrated into the application layer

information displayed on Console or keyed in by the user. It may become insecure when the compromise of Console or the interception of communication link becomes possible.

In order to enhance the security of such case, we suggest two non-intrusive measures to the Card-Centric Framework:

a) A custom layer could be introduced between the application and CCP layer, as shown in figure 9(a). In this layer, any security related measures could be implemented.
b) The security features are fully integrated into the application layer, over which the smart card has full control. This is illustrated in figure 9(b). In other words, the security measures could be accessed by the application when necessary.

For the security features, unilateral or mutual authentication could be implemented for Smart Card and Console to verify if they have the right to access each other [13]. Provided trust is established, they could communicate by CCP in plain text within the session. In case a packet could not be sent in plain text, such as when critical information is transmitted, the on-card application could protect the data by cryptography [14]. Note that cryptographic measures may introduce serious processing overhead to the system and degrade user perception. It should be employed only when necessary.

8. CONCLUSIONS

The Intelligent Adjunct model initiates a new usage model for smart cards, allowing them to run application logics. We move one step further, and proposed the Card-Centric Framework for smart cards to handle off-card I/O resources. Under this framework, any on-card applications that require user interactions could be developed, thus becomes more user-friendly. To enable this, we propose a system model and a set of communication protocol as guidelines for implementation.

The Card-Centric Framework is demonstrated by a smart card-based system. Although it only shows reasonable results, the system based on the enhanced technologies predicted by RESET roadmap [2] shows drastic improvement and proves feasibility of the framework in the future.

The framework is for the purpose of providing I/O devices for smart cards, yet security issues are to be implemented by software developers depending on the application involved. In the future, some more rigorous solutions might evolve, such as more appropriate measures for the security layer, and a compromised mechanism for I/O resource allocation.

REFERENCES

[1] B. Balacheff, B. Van Wilder, and D. Chan, "Smartcards – from security tokens to intelligent adjuncts", in Proceedings of Cardis98, Louvain-la-Neuve, Belgium, pp. 71-84, September 14-16, 1998.

[2] ERCIM and Eurosmart, "Micro-electronics", RESET Final Roadmap, Mar. 2003, pp.52-59.
[http://www.ercim.org/reset/]

[3] ERCIM and Eurosmart, "Systems and Software", RESET Final Roadmap, Mar. 2003, pp.19-20.
[http://www.ercim.org/reset/]

[4] J. –J. Vandewalle and E. Vétillard, "Developing smart card-based applications using Java Card", in Proceedings of Cardis98, Louvain-la-Neuve, Belgium, pp. 105-124, September 14-16, 1998.

[5] Sun Microsystems, Inc., Java Card 2.1.1 API, Rev. 1.0, May. 2000.
[http://java.sun.com/products/javacard/specs.html]

[6] International Organization for Standardization, International Standard ISO/IEC 7816: Integrated circuit(s) cards with contacts, parts 1-4, 1987 - 1998.

[7] PC/SC Workgroup, PC/SC Workgroup Specifications 1.0, parts 1-7, Dec. 1997.

[8] T. M. Jurgensen, and S. B. Guthery, "SIM APDUs", Smart Cards: The Developer's Toolkit, Upper Saddle River, NJ: Prentice Hall PTR, 2002, pp. 267-287.

[9] Proactive SIM, GSM 11.14 specification, July 1997.

[10] P. W. Frey, "An introduction to computer chess", Chess Skill in Man and Machine, 2nd ed., Springer-Verlag: New York, 1983, pp.61-68.

[11] ERCIM and Eurosmart, "Introduction", RESET Final Roadmap, Mar. 2003, pp.8.
[http://www.ercim.org/reset/]

[12] Altera, Nios Development Board – Reference Manual, Stratix Ed., Rev. 1.1, Jul. 2003.
[http://www.altera.com/literature/manual/mnl_nios_board_stratix_1s10.pdf]

[13] W. Rankl, and W. Effing, "Protection: authentication", Smart Card Handbook, 3rd ed., Hoboken, NJ : Wiley, 2003, pp.559.

[14] W. Rankl, and W. Effing, "Protection: secure data transmission", Smart Card Handbook, 3rd ed., Hoboken, NJ : Wiley, 2003, pp.558-559.