Layered Approach To Design Of WAP Identity Module For Wireless Internet Security

DaeHun Nyang · YouSung Kang · ShinHyo Kim · ByungHo Chung

WAP Identity Module (WIM) cooperates with wireless transport layer security (WTLS) and wireless application layer security to authenticate participating parties and to provide confidentiality of network traffic. To achieve the purpose, various standards including wireless public key infrastructure (WPKI), X.509, ISO 7816, PKCS#1 and PKCS#15 are supported in the WIM card. We present our experience of developing WAP Identity Module (WIM), which will be a very versatile device for secure electronic commerce. A method to implement PKCS#15 file system is proposed and conversion technique between parameters of WIM primitives and those of APDUs is described. Especially, our WIM implementation is designed in strict layering approach, which separates the transport layer protocol and the application protocol from the overall structure. This layered approach provides easy maintenance, upgrade, repair of the internal WIM modules. The advantage of the proposal is shown in terms of software engineering.

Keywords: Wireless Internet, Security, WIM, Smart card, WTLS

I. INTRODUCTION

WAP (Wireless Application Protocol) Forum has defined WIM (WAP Identity Module) as the secure hardware token to provide wireless Internet with security services such as data privacy, data integrity, user authentication, and non-repudiation. WAP is a result of continuous work to define an industry-wide specification for developing applications that operates over wireless Internet [1]. Prevalence of wireless Internet requires both transaction security and channel security. To provide the security services, WALS (Wireless Application Layer Security) that is designed for end-to-end transaction security service and WTLS (Wireless Transport Layer Security) for wireless channel security have been defined. WALS removes the inherent security hole at WAP gateways by providing end-to-end transaction security in WAP-based wireless Internet. Both WALS and WTLS need handshake procedures for entity authentication and key establishment, where confidential information must be kept securely in clients. A smart card can be a cryptographic token that stores securely confidential information and performs cryptographic operations, and WIM defines how to store confidential information and process cryptographic operations using PKCS#15, ISO 7816 and WTLS.

In this paper, we propose an enhanced WIM card that can accommodate multi-applications and has a layered architecture. WIM is a very versatile tool for electronic commerce and for various secure transactions. The system is separated into application layer, transport layer and physical layer. This layered approach helps developers upgrade and maintain easily the internal software modules, which is a very desirable feature compared with the cross-layer approach. Modules that compose the system include PKCS#15 based file system,

DaeHun Nyang: InHa University
YouSung Kang, ShinHyo Kim, ByungHo Chung: ETRI

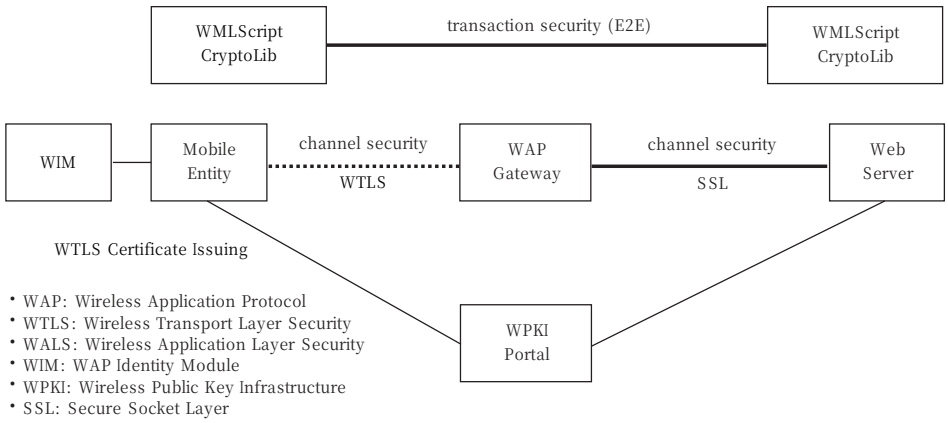


Figure 1. WAP Security System

transport protocol and APDU handling layer. We suggest methods to implement efficiently the file system and easily upgradeable transport protocol layer. Also, to support multi-applications in a smart card, a method is proposed to provide each application with its own working environment related to cryptographic operations. Security Environment (SE) that is defined by WIM standard is utilized to provide a working environment per application without any collision of SEs between applications, where an SE is a kind of template to store a set of cryptographic parameters for easy management of cryptographic operations.

II. WIRELESS INTERNET SECURITY SYSTEM

WAP security functionality includes WTLS, WMLScript Crypto Library, WPKI (Wireless Public Key Infrastructure) and WIM. Figure 1 shows the conceptual security system of WAP-based wireless Internet.

WTLS layer operates above a transport protocol layer to provide privacy, data integrity and authentication between two communicating parties. WTLS provides functionality similar to TLS v1.0 (Transport Layer Security) and incorporates new features such as datagram support, optimized handshake procedure and dynamic key refreshing. WTLS is optimized for low-bandwidth bearer networks with relatively long latency [3]. Although WTLS provides transient client authentication for the duration of a WTLS connection, it does not provide persistent authentication for transactions that may occur during that connection. One way to provide such authentication is to associate a digital signature with data generated as the result of a transaction, such as a purchase order or other financial document. To support this requirement, the browser provides crypto.signText among WMLScript functions that asks the user to sign a string of text [4], where WIM helps those cryptographic operations. WPKI is concerned with the infrastructure and procedures required to establish the trust relationships needed for authentication of servers and clients [5].

For maximum security, some parts of the security functionality need to be performed by a tamper-resistant device so that an attacker cannot retrieve sensitive data. Such data is especially permanent private keys used both in WTLS handshake with client authentication and in making application level electronic signatures. In WTLS, master secrets to protect secure sessions are relatively long-lived - which could be several days. This is in order to avoid frequent full handshakes that require heavy computation and large amount of data transfer. Master secrets are used as a source of entropy to calculate MAC (Message Authentication Code) keys and message encryption keys that are used to secure a limited number of messages.

As shown in Figure 1, WAP gateway uses two different security protocols at the transport layer. When the WTLS-to-SSL (Secure Sockets Layer) protocol translation occurs in the WAP gateway, data is unencrypted and temporarily exposed to an attack. To cope with this inherent security hole, WALS is used, which takes advantage of simple security tag after establishing security session at the application layer. WIM can also support WALS by storing and processing secret information such as private keys and master keys.

III. WAP SECURITY TECHNOLOGIES

WIM plays an important role in WAP-based wireless Internet for secure communication. The smart card proposed in this paper is implemented in WIM-based technologies. Thus, it is necessary for the smart card to perform cryptographic operations for WTLS and WALS. In this section, we survey functionalities of WIM. Also, we describe briefly ISO/IEC 7816 and PKCS #15 (Public-Key Cryptography Standards #15), which are references for implementation of our smart card.

1. WIM

WIM specification defines WIM as a tamper-resistant device that is used for WTLS and for application level security functions. A smart card implementation of WIM is based on ISO/IEC 7816. WIM is defined as an independent smart card application, which makes it possible to implement it as a WIM-only card or as a part of multi-application card containing other card applications like GSM SIM (Global System for Mobile communication Subscriber Identity Module). The information structure is based on PKCS #15 that enables a flexible information format on a cryptographic token. PKCS#15 uses the object model that makes it possible to access keys, certificates, authentication objects and proprietary data objects with simple read/write operations. Also, it has an access control feature [1]. In order to perform the cryptographic operations, WIM needs to store data such as information on supported algorithm, key pairs, own certificates for each key pair, trusted CA certificates, pre-master secret, master secret and PINs (Personal Identification Number).

2. ISO/IEC 7816

All the smart cards with contacts must conform to ISO/IEC 7816. In the field of information technology, ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) have established a joint technical committee, ISO/IEC JTC1 and subcommittee SC17 manages the part of identification cards and related devices. ISO/IEC 7816 consists of 10 parts with each specific title under the general title of 'Identification cards - Integrated circuit(s) card with contacts'. Part 1 named as 'Physical characteristics' specifies the physical characteristics of integrated circuit cards with contacts and part 2 of ISO/IEC 7816 specifies the dimensions, locations and assignment for each of the contacts on integrated circuit cards of an ID-1 card type. Part 3, ISO/IEC 7816-3 specifies the power and signal structures and information exchange between an integrated circuit card and an interface device such as a terminal. ISO/IEC 7816-4 named as 'Interindustry command for interchange' specifies the content of the message, commands and responses, transmitted by the interface device to the card, and vice versa. This also specifies the structure of files and data, and access methods and a security architecture defining access rights to files and data in the card. Contents that are specified in ISO/IEC 7816-4 are mainly to communicate each other, interface device to the card or conversely [6]. In particular, ISO/IEC 7816-8 has the name of 'Security related interindustry commands' and specifies security protocols, data elements for security support, the use of security algorithm, the use of certificates and security related commands [2].

3. PKCS#15

Many cryptographic tokens such as smart cards are intrinsically secure computing platforms ideally suited to providing security functionality to applications. Unfortunately, use of these tokens for authentication and authorization purposes is hampered by lack of interoperability at several levels. The industry lacks standards on common format for storing digital credentials on the tokens, and mechanisms to allow multiple applications to effectively share digital credentials have not yet reached maturity [7].

In order to cope with these flaws, PKCS #15 defines cryptographic token information format. The format specifies how keys, certificates and application-specific data may be stored on an ISO/IEC 7816 compliant smart card or other media. The format specified in the PKCS #15 allows multiple applications to reside on the card, even multiple PKCS #15 applications [7].

IV. PROPOSED WIM ARCHITECTURE

Our WIM implementation is designed in strict layering approach, which separates the transport layer protocol and the application protocol from the overall structure. This layered approach helps developers upgrade easily the internal software modules of both WIM and mobile equipment. Mobile equipments treat WIM as a

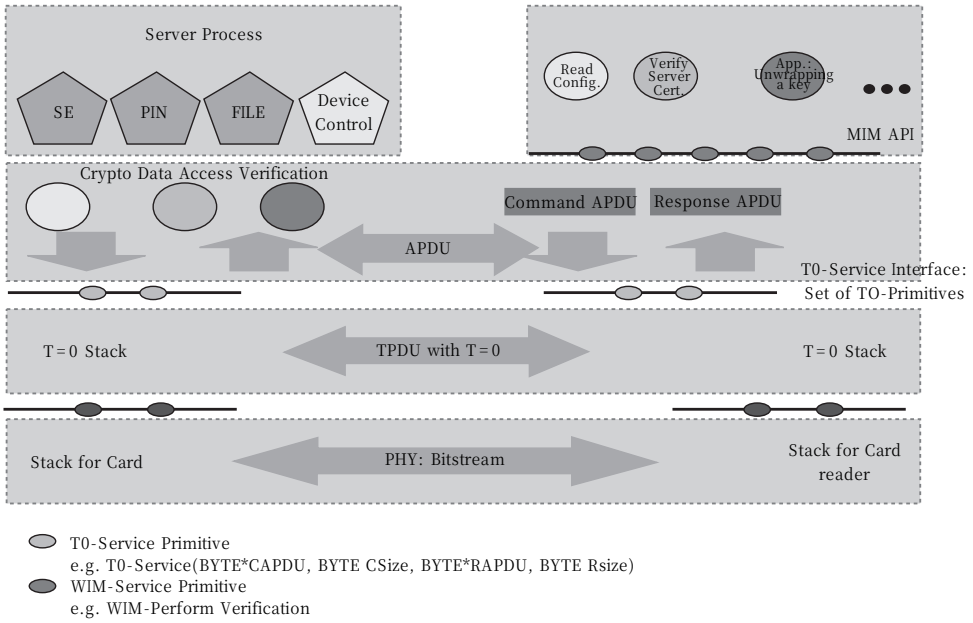


Figure 2. WIM protocol stack

special device that helps some cryptographic operations and as a secure storage device via the device driver that generates proper WIM commands. WIM application in a smart card waits for a command request from a mobile equipment, and responds in a predefined behavior through WIM API. Our WIM architecture separates each protocol layer and defines interfaces between layers. Figure 2 shows protocol layers in the architecture. WIM protocol stack is mainly composed of physical layer, transport layer, APDU handling layer and applications. In the figure, the left side of protocol stack represents smart card application, and the other side represents terminal application. For a mobile terminal, WIM API that includes transport layer and APDU handling layer is implemented in the form of device driver. Usually, the layered design of a protocol stack results in some overhead costs including redundant processing, layer management overhead and additional memory requirement for stack needed for function call. These overhead costs are not required in the cross-layer approach. In the proposed architecture, however, the overhead costs do not affect the overall system performance because of the simplicity of our WIM API, which is just one function call with a depth of one per WIM primitive.

In this section, we suggest a WIM architecture that can accommodate multi-applications and also describe components of our system.

1. WIM API

WIM API is defined using primitives in [1] and it provides application programmers with standardized interface for development of applications using WIM. Owing to the WIM API, WIM card is regarded as a logical unit that can perform secretly cryptographic operations and plays a role of secure storage to an application programmer. That is, a series of process related with APDU/TPDU and process performed inside cards to control WIM is hidden from application programmers.

WIM API is designed on the basis of WIM service primitives in [1]. To implement a set of service primitive in the form of API requires mapping between them. WIM service primitives are relatively simple compared to other communication protocols, so we simply define mapping by defining functions named WIM service primitives. Procedures inside WIM API are composed of a sequence of several card commands. Input/output parameters in the API are defined using parameter fields and data fields in command APDU and response APDU. Also, additional return code is defined for proper handling of the result of function execution. Return code is defined basically by encoding status words of response APDU.

For example, WIM_ReadBinary API is derived from

Case 4. Lc, Data, Le
1. SELECT FILE
2. PSO-DECIPHER, Application Level
3. PSO-COMPUTE DIGITAL SIGNATURE
4. PSO-COMPUTE CRYPTOGRAPHIC CHECKSUM

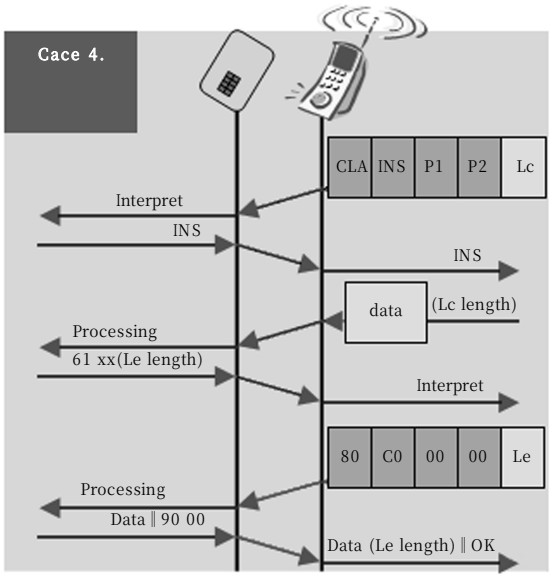


Figure 3. T=0 Transport Protocol

READ BINARY card command that includes high byte of the offset, low byte of the offset and number of bytes to be read and whose response APDU contains data read:

WORD WIM_ReadBinary(WORD
Offset, WORD Length, BYTE * UserData);

Status words (SW1 and SW2) of response APDU is encoded as a return code. The Offset parameter is obtained by encoding high byte of the offset (P1) and low byte of the offset (P2) of command APDU. Also, Length is set to the Number of bytes to be read (Le) of command APDU.

Functions in WIM API are largely categorized into device control functions, personal identification functions, data access functions, cryptography functions and exception handling functions. Especially, DER (Distinguished Encoding Rule) of ASN.1 is involved in data access functions.

2. APDU Handling Layer

APDU Handling layer is responsible for making command APDU using input parameters of API functions to send the APDU via TPDU handling layer. Also, using response APDU obtained from TPDU handling layer it fills output parameters of API functions. During the conversion between input/output parameters and parameters of APDU, additional data structures are sometimes required. For example, WIM-ComputeDigitalSignature has Private KeyID as an input parameter, whereas PSO-COMPUTE DIGITAL SIGNATURE card command requires a file path and a private key reference. Thus, the conversion requires mapping between a PrivateKeyID and (file path, private key reference) pair. We construct a mapping table that maps PrivateKeyID into (file path, private key reference) during card initialization phase. The file path is represented in PKCS#15 file name format. Besides WIM-ComputeDigitalSignature, WIM-VerifySignature, WIM-KeyTransport and WIM-PHash primitives require proper manipulation of input/output parameters to build command and response APDUs.

3. T=0 TPDU Handling Layer

T=0 Transport protocol is designed for transportation of small amount of data. It delivers information between a

FileID	Name	EEPROM ADDR	KeyRef	Offset	KeyID	Permission	CF_Flag	Length
0×9000	CF(PrkCF)	0×3000	PrkRef1	0×0000	kID1	111000000	TRUE	0×100
			PrkRef2	0×0100	kID2			0×100
			PrkRef3	0×0200	kID3			0×100
0×9010	CF(CCF1)	0×3100	CRef1	0×0000	kID1	111000000	TRUE	0×100
			CRef2	0×0100	kID2			0×100
			CRef3	0×0200	kID3			0×100
0×6033	EF(CDF1)	0×2600					FALSE	0×100
0×6031	EF(AODF)	0×2500					FALSE	0×100
0×6032	EF(PrkDF)	0×2400					FALSE	0×100
0×5032	EF(TokenInfo)	0×2300		N/A			FALSE	0×100
0×2F00	EF(DIR)	0×2200					FALSE	0×100
0×5031	EF(ODF)	0×2000					FALSE	0×100
0×5F00	DF(PKCS#15)	0×1100					FALSE	0×100
0×3F00	MF	0×1000					FALSE	0×100
				…				

Figure 4. File Description Table for WIM

smart card and a terminal by embedding information in APDU. According to the type of information transported, APDU is split into TPDUs and one of four types of handshake methods is applied. The four types of handshakes are defined according to existence of Lc and Le as followings:

- Case 1: Lc=empty, Data=empty, Le=empty
- Case 2: Lc=empty, Data=empty, Le=non-empty
- Case 3: Lc=non-empty, Data, Le=empty
- Case 4: Lc=non-empty, Data, Le=non-empty

Figure 3 shows an example for case 4.

T=0 Transport layer has a single API for terminal, which is in the form of

WORD ME_T0_Service (BYTE* commandAPDU, WORD CAPDU_LEN, BYTE* responseAPDU, WORD* RAPDU_LEN)

Users of T=0 protocol calls the above API giving a command APDU that includes information to be transported to a card, and waits until ME_T0_Service() returns a response APDU that includes information from a card. ME_T0_Service() decides where to branch among the four cases according to class and instruction fields in command APDUs.

4. File System

In this section, we describe how to implement PKCS#15 file structures in EEPROM and Flash memory which are commonly used memory type in smart cards. PKCS#15 specifies only interfaces to handle cryptographic tokens such as private keys, public keys, certificates, PIN, etc. Thus, it is our responsibility to define internal file system to store cryptographic tokens in smart cards. To handle PKCS#15 file efficiently, we define a file description table that acts as a system-wide mapping function between PKCS#15 fileID and addresses in EEPROM. The file description table also includes access control bits, a file length, a bit that indicate whether the file is PKCS#15-compliant or not, and references to miscellaneous values such as KeyReference, PINReference, etc. This table also is stored in EEPROM area during card initialization phase. Figure 4 shows an example of the file description table.

When a terminal asks to read a file through PKCS#15

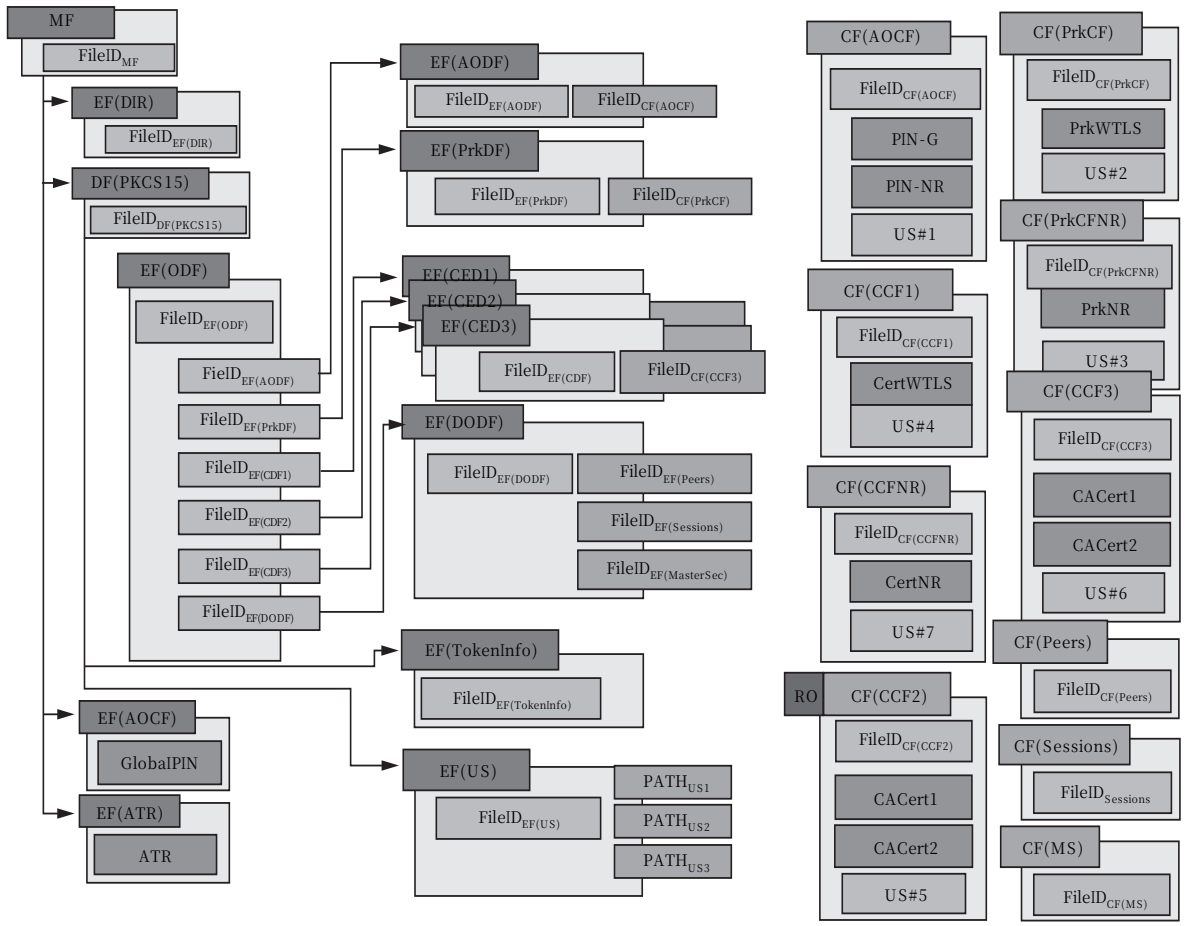


Figure 5. WIM File System Profile

interface, a smart card interprets the PKCS#15 fileID using the file description table. As a result of interpretation, a card gets information about the location of the file, length, and access rights. If the user has access rights for the file, a card reads requested data blocks and returns them to the terminal. Usually, the PKCS#15 files are stored in ASN.1 DER encoded form. Thus, after obtaining a file from a smart card, a terminal must decode using ASN.1 DER decoder to retrieve information. Smart cards do not know the ASN.1 DER but they just store and return the encoded data. Only the terminal is responsible for ASN.1 DER encoding and decoding.

WIM, one of PKCS#15 applications, has many files to store cryptographic tokens and card information. An example of PKCS#15 profile for WIM is shown in figure 5.

5. Security Environment

To support multi-applications in a smart card, the most simple but inefficient solution is to make each application work in an independent channel. If a specific application is asked to be executed, "MANAGE CHANNEL - Open" followed by "SELECT - Application" is sent to a card to serve the request. Though this approach is simple, it has a drawback to have to manage each application separately, which requires additional step to open channel, to select application by application identifier that is composed of Registered Application Provider Identifier and Proprietary Application Identifier Extension and to manage redundant information for each application [8].

More attractive way to support multi-application in a card is to make as many Security Environments (SE) to provide each application with its own working environment. An SE is a logical container of a set of fully specified security mechanisms that are available for reference in security related commands. Each SE specifies references to the cryptographic algorithms to be executed, the modes of operation, the keys to be used and

Comand APDU

CLA	INS	P1	P2	Lc	Data	Le
8X	22	F3	SE Number	Empty	Empty	Empty

Response APDU

Data	SW1-SW2
Empty	Status Bytes

Figure 6. The MSE-RESTORE command - response pair

any additional data needed by a security mechanism [1]. If an application is required to be executed, MSE-RESTORE (Manage Security Environment - RESTORE) command is used to set up a security environment for the application. For the SE, there are several cryptographic service templates. We have used RSA as a public key algorithm and a digital signature algorithm. RSA_DST, a digital signature template, is used for the digital signatures and associated operations, RSA_CT, a confidentiality template, for key transport or deciphering and RSA_CCT, Cryptographic Checksum Template, for deriving master secret and key block calculation.

For example, if we want to design a WIM card including both WTLS and WALS security functions, two SEs are enough to support those applications without performing channel management operation. And also, we can increase the number of SEs up to 127 in proportion to the number of the provided services and the supported algorithms [1]. For example, the templates to be configured into SE are WTLS_RSA, WALS_RSA, WTLS_ECDH, WALS_ECDH, WIM_GENERIC_RSA, WIM_GENERIC_ECC, and so on. Each of SEs has a different SE number. In order to switch between different SEs, the mobile entity uses the card command, MSE-RESTORE, followed by the SE number. This operation is used to restore an SE by replacing the current SE with the SE number mentioned in this command. Figure 6 shows command - response APDU pair of the card command, MSE-RESTORE.

The method that introduces multiple SEs using MSE-RESTORE is a general and standard way conforming to ISO/IEC 7816 series.

V. CONCLUSION

In this paper, we have surveyed technologies to provide wireless internet security service such as WTLS, WIM and WALS. Also, we have presented layered architecture for Wireless application protocol Identity Module. Owing to the strict separation of physical layer, transport layer and application layer, developers can independently treat each layer and easily plug-out and plug-in a layer if required. For example, in the current implementation of WIM, the transport layer supports only T=0 protocol for APDU delivery, but if needed, the transport layer can be replaced with one that supports T=1 protocol without modifying other modules. Since the file system supports PKCS#15, any terminal that supports PKCS#15 is interoperable with our WIM application. Our architecture also supports logical channel and provides application selection methods using PKCS#15 AID (Application Identifier) and WIM AID, so one smart card can accommodate more than one applications. For example, it is possible to design a smart card supporting both WIM and USIM (Universal Subscriber Identity Module) of 3GPP.

Currently, WIM supports only WTLS(Wireless Transport Layer Security), but a study to support TLS (Transport Layer Security) in WIM is on-going. Thus, method to adapt TLS to WIM architecture must be studied.

[References]

[1] WAP WIM.: Wireless Application Protocol Identity Module Specification, Part: Security, Version 02-Jan-2001, *WAP Forum*, 2001.

[2] ISO/IEC 7816-8.: Identification cards - Integrated

Circuit(s) cards with contacts - Part 8: Security related interindustry commands. First edition, *International Organization for Standardization*, 1999.

[3] WAP WTLS.: Wireless Application Protocol Wireless Transport Layer Security Specification, Draft Version 02-Jan-2001, *WAP Forum*, 2001.

[4] WAP WMLScript Crypto Library.: Wireless Application Protocol WMLScript Crypto Library, Version 20-Jun-2001, *WAP Forum*, 2001.

[5] WAP WPKI.: Wireless Application Protocol Public Key Infrastructure Definition, Version 24-Apr-2001, *WAP Forum*, 2001.

[6] ISO/IEC 7816-4.: Identification cards - Integrated Circuit(s) cards with contacts - Part 4: Interindustry commands for interchange, *International Organization for Standardization*, 1995.

[7] PKCS #15.: PKCS #15 v1.0: Cryptographic Token Information Format Standard, *RSA Laboratories*, 1999.

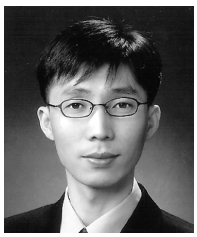
[8] ISO/IEC 7816-5.: Identification cards - Integrated Circuit(s) cards with contacts - Part 5: Numbering system and registration procedure for application identifiers, *International Organization for Standardization*, 1994.

DaeHun Nyang

DaeHun Nyang received his BS degree in electronic engineering from KAIST in 1994, the MS and the PhD degrees in computer science from YONSEI Univ. in 1996 and 2000, respectively. From 2000 to 2003, he had worked for Electronics and Telecommunications Research Institute as a senior researcher. Since 2003, he has been with INHA Univ., Korea, where he is currently assistant professor in Graduate School of Information Technology and Telecommunications. His research interests include cryptography, network security including WPAN security, ad hoc network security
and wireless LAN security.
E-mail: nyang@inha.ac.kr,
Tel:+82-32-860-8424
Fax:+82-32-865-0480

Sinhyo Kim

SinHyo Kim was born in Korea in 1968. She obtained her Master of Science degree in Electrical Engineering and Information & Communication Engineering from the College of Engineering, Chungnam National University, Korea in 2000. In February 1990 she joined ETRI, and
she is now a senior member of engineering staff. Her research interests include the security system and group communication security.
E-mail: shykim@etri.re.kr
Tel:+82-42-860-5485
Fax:+82-42-860-5611



You Sung Kang

You Sung Kang was born in Korea in 1971. He obtained his Bachelor and Master of Engineering degree in Electronics Engineering from the College of Engineering, Chonnam National University, Korea in 1997 and in 1999, respectively. He is now pursuing his Doctor of Philosophy in Electrical and Electronic Engineering from Korea Advanced Institute of Science and Technology (KAIST). In November 1999 he joined Electronics and Telecommunications Research Institute (ETRI), and he is now a senior engineer. Since 2004, he has been the IT international standard expert of Telecommunications Technology Association (TTA). His research interests include the areas of RFID/USN security, wireless LAN security, cryptographic protocol and network security.
E-mail: youskang@etri.re.kr
Tel:+82-42-860-5960
Fax:+82-42-860-5611

Byungho Chung

Byungho Chung received the B.S. degrees in Computer Science from Chunnam National University in 1988 and the M.S. and Ph.D. degrees in Computer Science from Chungnam National Univaersity in 2000 and 2005, respectively. During 1988~2000, he stayed in Agency for Defense Development of south Korea. He now works for Information Security Research Division, ETRI, Korea.
E-mail: cbh@etri.re.kr
Tel:+82-42-860-5489
Fax:+82-42-860-5611