Design and Implementation of WIRE1x EAP-SIM Module

Siao-Jie Cai, Chih-Hsuan Lee, Han-Hsing Chiu, Chih-Hsiang Hsueh,
Jui-Yi Chen, Chien-Chia Chen, and Jyh-Cheng Chen
Wireless Internet Research and Engineering Laboratory
National Tsing Hua University
Hsinchu, Taiwan

January 2, 2007

Abstract

This document presents the design and implementation of WIRE1x EAP-SIM module. The WIRE1x is an open-source implementation of IEEE 802.1x client (supplicant) developed by the Wireless Internet Research & Engineering (WIRE) Laboratory. The IEEE 802.1x defines a port-based network access control to authenticate and authorize devices interconnected by various IEEE 802 series LANs. It also works with extensible authentication protocol (EAP) to carry out the authentication. EAP-SIM is a challenge-response mechanism using the GSM Subscriber Identity Module (SIM). The integration of the cellular networks and the WLANs is believed to be a trend of the network development. The motivation for developing WIRE1x EAP-SIM module is to provide a better security and trusty mechanism over the WLANs and to lower the cost of roaming among different WLANs or cellular networks. Moreover, providing SIM-based authentication will be convenient for network operator to manage accounts.

Keywords: Authentication, IEEE 802.1x, Extensible Authentication Protocol (EAP), EAP-SIM, Wireless LANs, Security

Table of Contents

List of Figures

1. Introduction

The wireless local area network (WLAN) is very popular in the metropolis and campus. To make user roaming among different WLANs seamlessly is a popular topic in wireless network research. The National Center for High-Performance Computing (NCHC) [1] in Taiwan has also proposed a plan to prompt the cross-campus WLAN roaming. In addition, coming with second generation mobile networks and third generation mobile networks, it is believed that the integration of the mobile networks and the WLANs will be a trend of the wireless network development. Hence, the authentication, authorizing and accounting (AAA) is the main issue coming with the development of WLANs and mobile networks.

The IEEE 802.11b [2] is a widely adopted wireless standard. The IEEE802.11b standard has defined the following two basic security mechanisms: entities authentication including open system authentication and shared key authentication, and Wired Equivalent Privacy (WEP). But it is has been reported that these two security mechanisms adopted in IEEE 802.11b is vulnerable. Thus, the IEEE 802.11i [3-4] has been proposed to enhance the security of IEEE 802.11b networks. IEEE 802.11i also incorporates IEEE 802.1x [5] as its authentication enhancement.

The IEEE 802.1x standard defines a *port-based network access control* to authenticate and authorize devices interconnected by various IEEE 802 series local area networks (LANs). It is suitable for the authentication in WLAN because it provides better security guarantee. To adopt the IEEE 802.1x authentication, both the access points (AP) and end users need to be capable with IEEE 802.1x. Most of the APs on the market now support IEEE 802.1x. There are also some IEEE 802.1x client software developed by other universities and organizations. National Tsing Hua University (NTHU) has deployed WLAN and adopted IEEE 802.1X and RADIUS [6] to *authenticate* users to achieve a secured WLAN environment. WIRE1x [7] has also been released since July 18, 2003 to provide users a free IEEE 802.1x client software under Microsoft Windows. WIRE1x has supported many EAP (Extensible Authentication Protocol) [8] authentication methods, including EAP Message Digest 5 (EAP-MD5) [9], EAP Transport Layer Security (EAP-TLS) [10], EAP Tunneled TLS (EAP-TTLS) [11] and Protected Extensible Authentication Protocol (EAP-PEAP) [12].

All of the four authentication methods except for EAP-TLS need network access identifier (NAI) and password, while EAP-TLS needs a verified certificate which is usually obtained by downloading from the existing Internet service. For those methods that require ID and passwords bring about the problem of account management, and for EAP-TLS, the certificate is hard to obtain as the user need the

certificate to access the Internet, but the certificate cannot be acquired until there is Internet service, which is obviously a paradox.

Besides the four authentication methods mentioned above, 3GPP [13] proposed another method, EAP-SIM. EAP-SIM is different from other EAP methods that WIRE1x has supported, because it is an EAP mechanism based on the secret key stored on Global System for Mobile Communications (GSM) Subscriber Identity Module (SIM) [14] card. EAP-SIM utilizes the SIM card as the authentication token, allows WLAN to utilize existing GSM authentication infrastructure, which is mature and established, and integrates WLAN and mobile billing as one. It features many advantages that other methods do not possess:

- Removes the user's burden of remembering passwords
- Provides an authentication service that is both strong and easy to use
- Allows rapid deployment due to the high penetration of mobile phones
- Uses open standards and supports interoperability with other systems

The development of WIRE1x EAP-SIM module is using Visual C++ 6.0, and is based on the Open1x [15], an open-source implementation of 802.1x *supplicant* software, which supports Mac OS X, FreeBSD, OpenBSD, and Linux.

This work provides user a more secure and convenient way to perform the authentication when using the WLAN as EAP-SIM supports mutual authentication to protect the user from accessing spurious AP and no more username (NAI) or password is needed. Users can rely on just one SIM card to access different wireless LANs seamlessly if roaming is supported by the different network operators.

The rest of this document is organized as follows. Section 2 is an introduction to IEEE 802.1x and we briefly describe the GSM and EAP-SIM authentication flow in the third section. An overview of smartcard and the ISO 7816 standard will be given in section 4. In section 5, we depict the implementation details of WIRE1x EAP-SIM module, and conclude the report in the last section.

2. Introduction to IEEE 802.1X

The IEEE 802.1x is a standard defining *port-based network access control* to provide a mechanism of *authentication* and *authorization* network devices that attached to IEEE 802 series LANs, and several background knowledge related to IEEE 802.1x will be provided in the following sections.

2.1 IEEE 802.1x Authentication Framework

There are three main components in IEEE 802.1x framework as depicted in Fig.

1: supplicant, authenticator, and authentication server. The port in a system is capable of providing services to other systems or accessing service available via the LAN. In IEEE 802.1x, the supplicant is the port that wishes to access service offered by the authenticator, which is generally a mobile node (MN) in a WLAN; while the authenticator is the port that will enforce authentication before making service available to a supplicant. Authentication server performs authentication function to check the supplicant's credentials and to indicate whether the authentication is successful or not. The authenticator and authentication server can be collocated within one system, but in a WLAN, the access point (AP) usually plays the role of an authenticator, and the Authentication, Authorization, and Accounting (AAA) server such as RADIUS server is an authentication server. There is a port access entity (PAE) in both supplicant and authenticator that operates the algorithm and protocol associated with the authentication mechanisms.

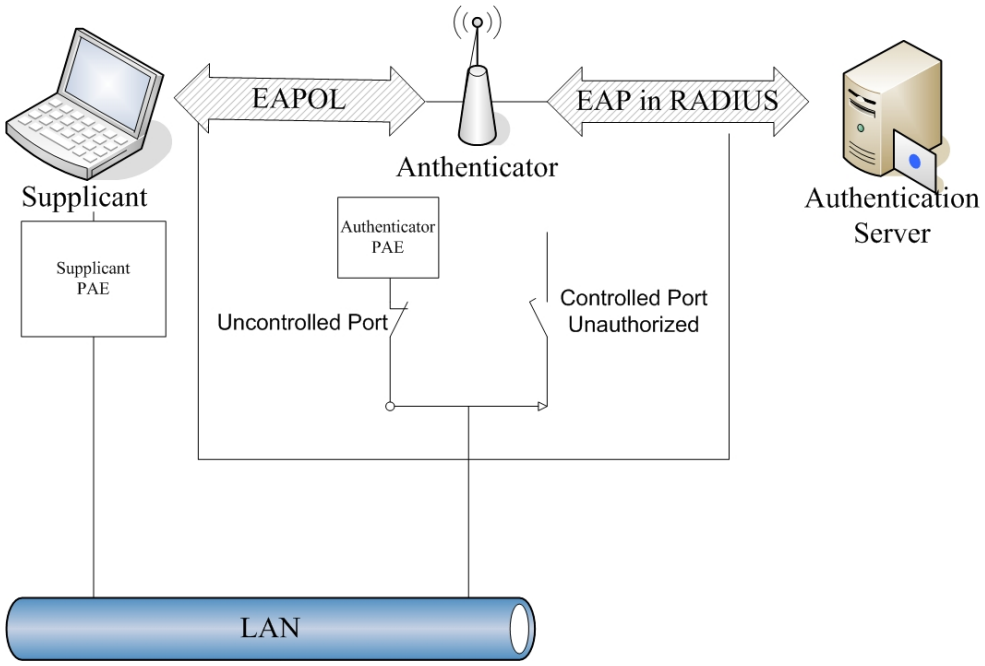

Fig. 1: The IEEE 802.1x Framework.

The authenticator PAE controls the state of the controlled port depending on the result of the authentication process. Before the authentication succeeds, the *controlled port* is in the unauthorized state (switch off) as shown in Fig. 1, and only the *uncontrolled port* is switched on to direct IEEE 802.1x messages to the authentication server. No other traffic is allowed before the authentication is completed. After the authentication is completed successfully, the authenticator PAE will switch on the controlled port to authorized state as Fig. 2. Thus, the supplicant is able to access other services provided by the authenticator's system.

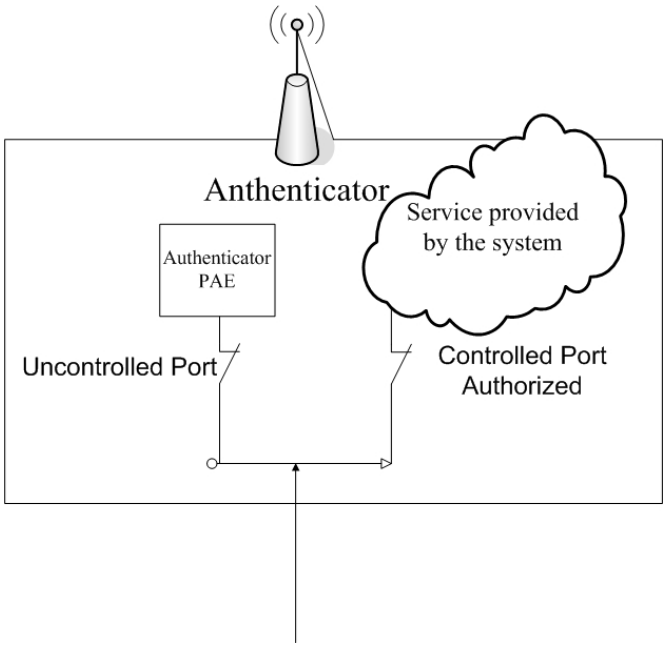

Fig. 2: Controlled port is switched on after authorized.

2.2 EAP Authentication Methods

The IEEE 802.1x works with the EAP to do the actual authentication. Since EAP supports many authentication methods, new authentication schemes can be easily deployed. This makes IEEE 802.1x an ideal option to provide a more convenient and securer network environment. Commonly used EAP methods include EAP-MD5, EAP-TLS, EAP-TTLS, and PEAP, EAP-SIM and extensible authentication protocol method for the 3 rd generation authentication and key agreement (EAP-AKA) [16].

The IEEE 802.1x also defines EAP over LANs (EAPOL), an encapsulation technique to carry EAP messages between a supplicant and an authenticator. The authenticator just relays the messages between the supplicant and authentication server. Fig. 3 shows the IEEE 802.1x protocol stack.

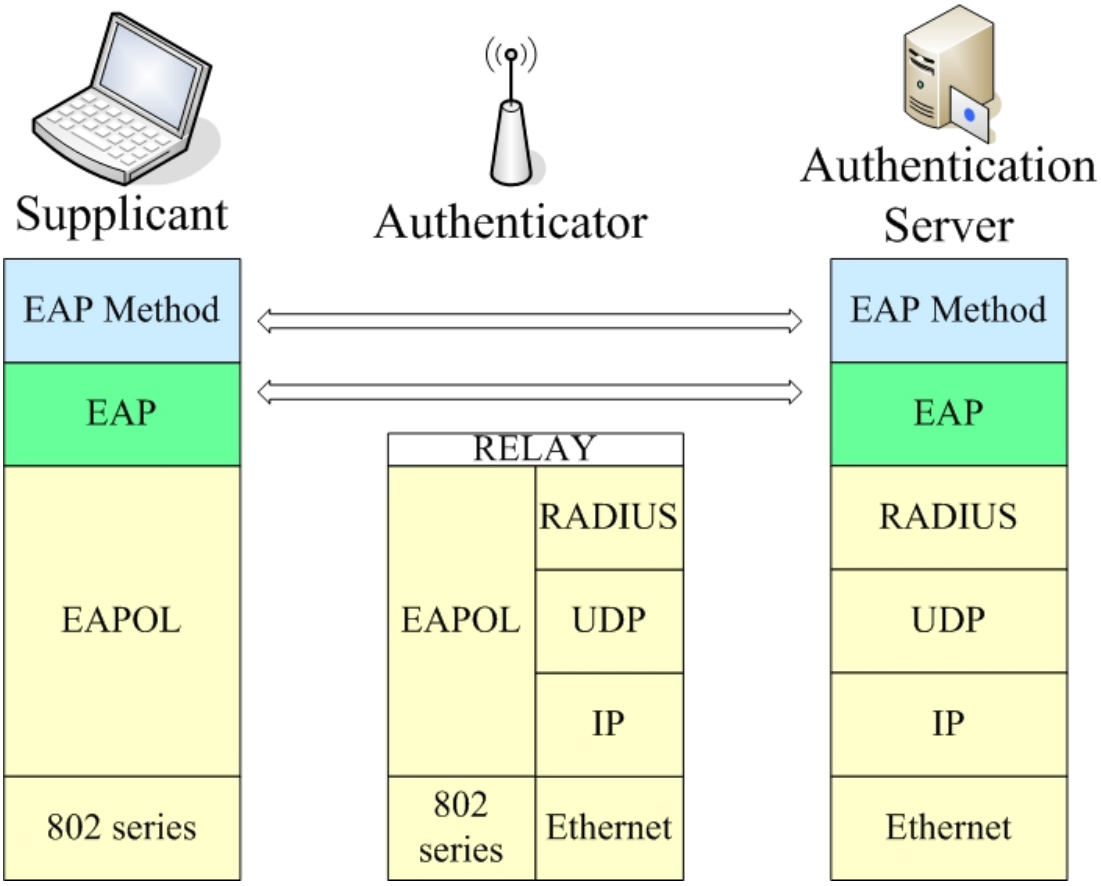

Fig. 3: The IEEE 802.1x protocol stack.

3. EAP-SIM

3.1 GSM Authentication

The EAP-SIM protocol was deployed by 3GPP to specify an EAP mechanism for authentication and session key distribution using the Global System for Mobile Communications (GSM) Subscriber Identity Module (SIM).

The GSM security framework includes the following three algorithms:

- A3 Algorithm is used for authentication.
- A5 Algorithm is a stream cipher algorithm used to encrypt the user traffic.
- A8 Algorithm is used to derive a cipher key that will be used in the A5 algorithm.

The GSM authentication is based on the challenge-response mechanism as depicted in Fig. 4. The authentication relies on a 128-bit long secret key $K_{i,}$ which is stored in the end user's SIM card and the key database of the network operator, a reliable 3rd party, or an organization owned by the government. It's important that this shared key K_i should keep secret and should never be transmitted over the network.

The network also issues a 128-bit random number (RAND) as a challenge to the user. The user and the network both use their own K_i and the RAND as the input to the A3 algorithms to generate a 32-bit signed response (SRES). The user then replies the SRES to the network. The network can authenticate user's identity by comparing the SRES received with the output of its own A3 algorithm.

The A8 algorithms takes the same input as the A3 algorithm but will generate a 64-bit cipher key (K_c). The A5 algorithm then takes K_c and a 22-bit counter value (COUNT) to encrypt the user traffic. The change in COUNT is used to prevent replay attack. Since the network should generate the same K_c as the user, the network could decrypt the encrypted data.

The lack of *mutual authentication* is a major weakness in GSM authentication; furthermore, the 64-bit long cipher key (Kc) is not strong enough for data networks as well [17-18].

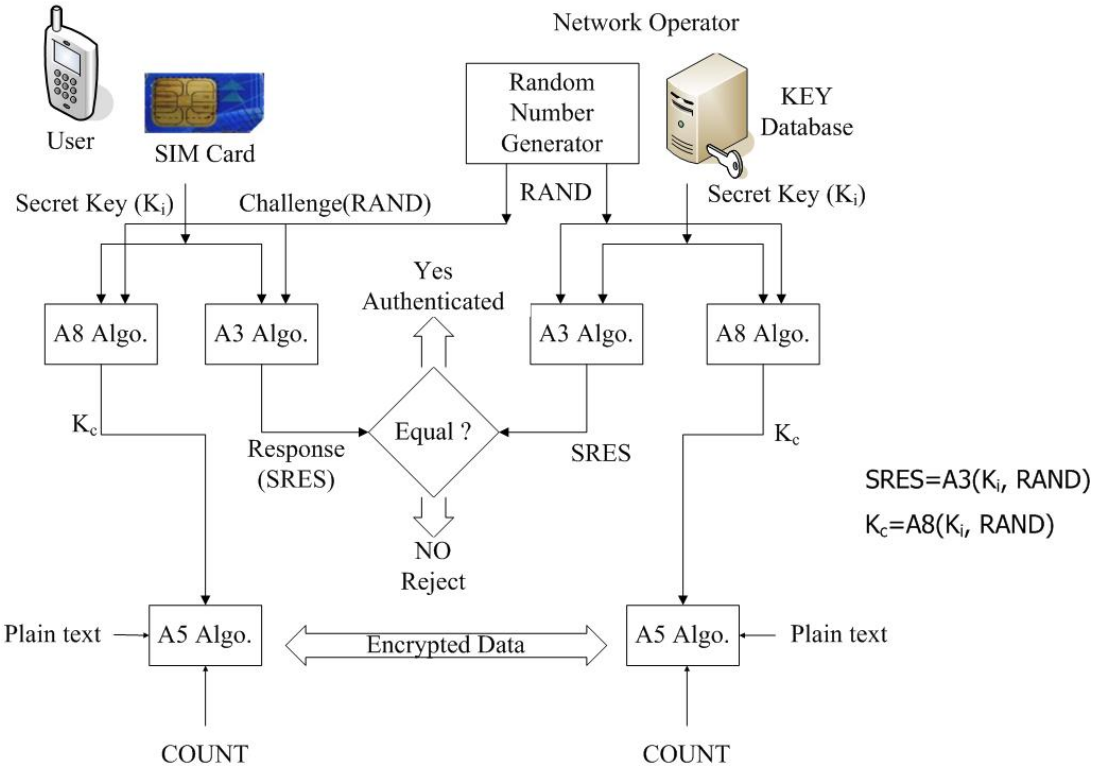

Fig. 4: The GSM Security Framework.

3.2 EAP-SIM Authentication

The security of EAP-SIM relies on underlying GSM mechanisms. But to meet the security requirement of a data network, the EAP-SIM uses several RANDs to generate several 64-bit K_c keys, which are combined to constitute a stronger keying

material. Also, in EAP-SIM, the client will issue a random number NONCE_MT as a challenge to the network to achieve mutual authentication and to contribute to key derivation. Fig. 5 is the architecture of EAP-SIM.

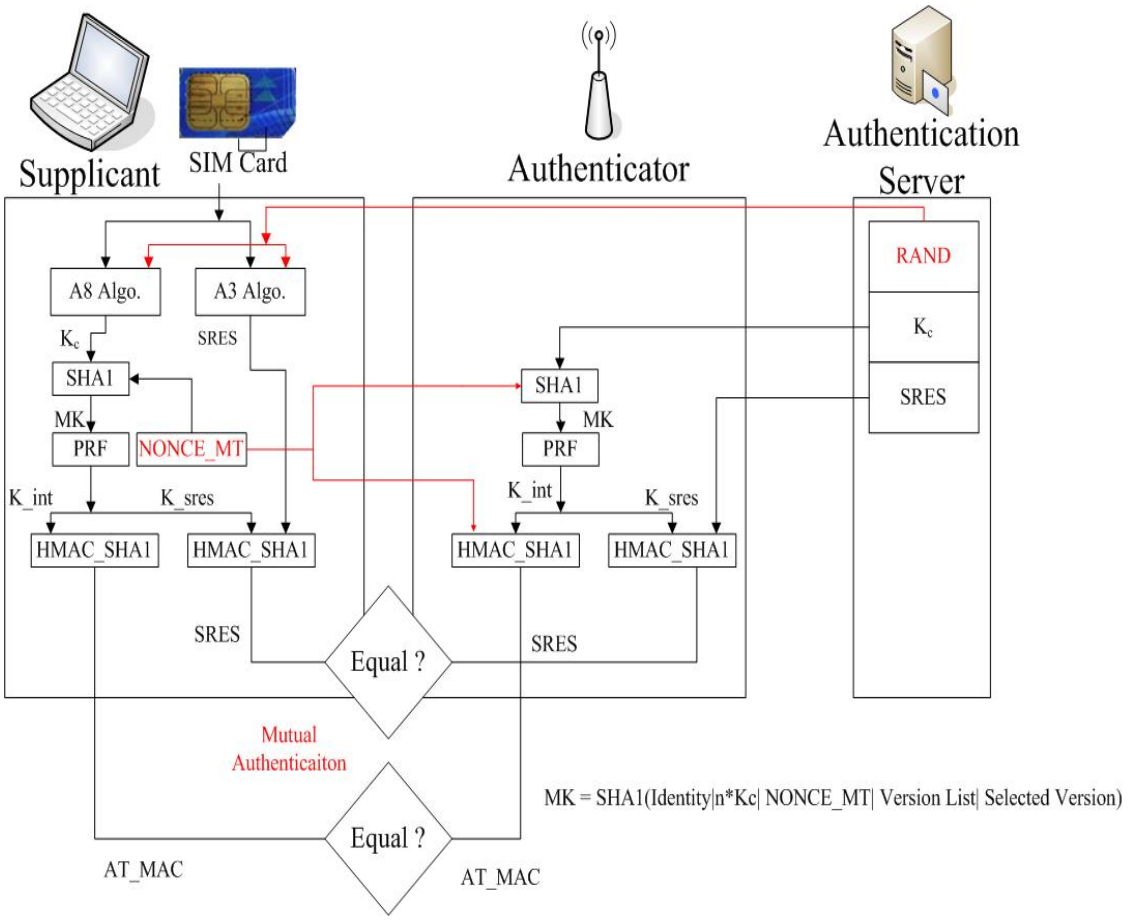

Fig. 5: The EAP-SIM Architecture.

Next, we introduce the full authentication flow of EAP-SIM. For the simplicity of explanation, we only mention the authentication flow of a general case. Fig. 6 is a typical EAP-SIM full authentication procedure.

In the beginning, the supplicant issues an EAPOL-Start packet to inform the authenticator to launch the authentication process. After receiving the EAPOL-Start packet, the authenticator sends an EAP-Request/Identity packet to the supplicant. In EAP-SIM full authentication, the supplicant then sends a response packet with the user's International Mobile Subscriber Identity (IMSI) stored in the SIM card to the authenticator. After receiving the peer's identity, the authenticator sends an EAP-Request of the type SIM and the subtype Start packet with a list of EAP-SIM versions supported by the EAP server stored in the AT_VERSION_LIST attribute. The supplicant then responds to the request with an EAP_RESPONSE/SIM/Start packet which carries a random number NONCE_MT and the AT_SELECTED_VERSION attribute that contains the selected version number to the

authenticator. The authenticator then connects to the authentication server to obtain n (n=2 or 3) GSM triplet to derive the keying material. A GSM triplet is formed by the three GSM authentication values: RAND, K_c, and SRES mentioned before. The next EAP request issued by the authenticator is of the type SIM and the subtype Challenge, which contains the RAND challenges and a message authentication code attribute AT_MAC to cover the challenges. On receipt of the EAP-Request/SIM/Challenge packet, the peer then runs the algorithms depicted in Fig. 5 to derive a copy of the message authentication code and the SRES. If the MAC does match, the supplicant responds with the EAP-Response/SIM/Challenge, with the AT_MAC attribute that covers the peer's SRES response values to the authenticator. If the received SRES matches the one calculated by the authenticator, the authenticator will send an EAP-Success packet to indicate that the authentication was successful.

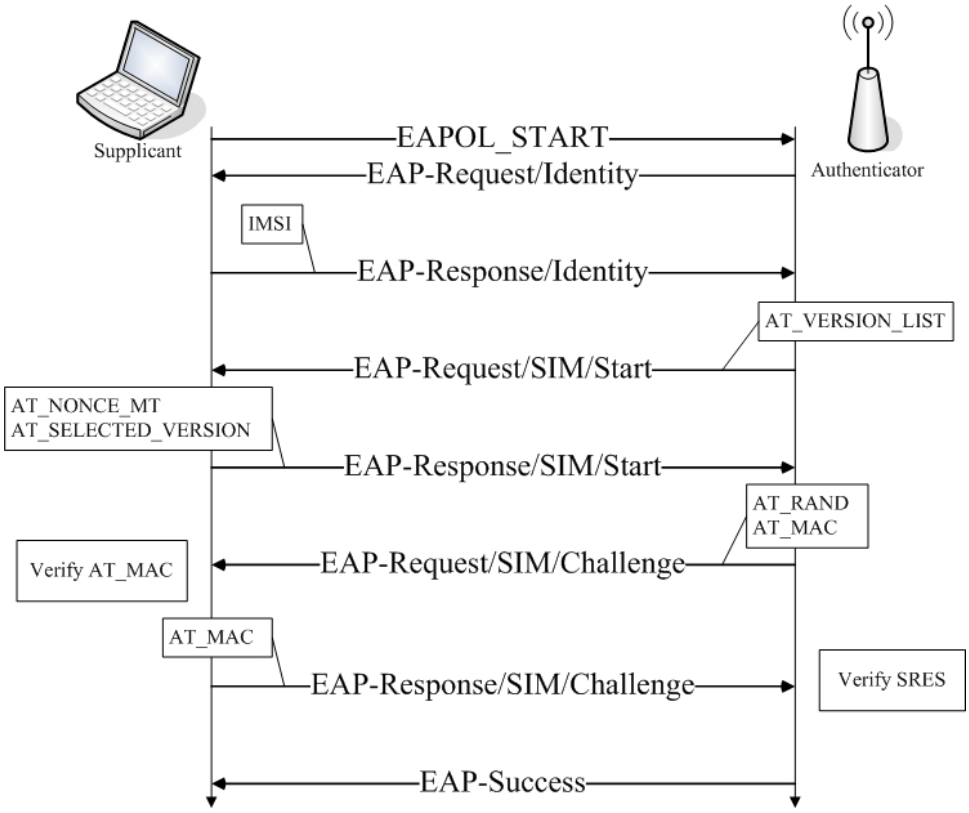

Fig. 6: EAP-SIM Full Authentication Message Flow.

4. Smart Card and the ISO 7816 Standard

4.1SmartCard

A smartcard, chip card, or integrated circuit card is designed as pocket-sized card with embedded integrated circuits. It is exactly the shape and the size of credit card (or smaller, e.g. the GSM SIM card) that stores a lot of sensitive information, carries

out local processing on the data stored, and performs complex calculations.

Let's discuss some of the key features and characteristics of smartcards. The first is its high reliability and low cost. Vendors provide smartcards that mostly meet ISO specification and have passed different crucial tests. Besides, a pocket-sized smartcard can store eight thousand to one hundred and twenty-eight thousand bits information, so the powerful storage capacity is the second feature. Smartcards are user-friendly as they provide simple interface with intended application, and ease of use is also a main characteristic. Last but not least is its high security, information stored on the chip is difficult to duplicate or disrupt unlike the outside storage used on magnetic stripe cards which can be easily copied.

There are two fundamental types of smartcard software. One is "host software", which refers to software runs on a computer connected to a smartcard. Host software is also referred to as reader-side software. The other is "card software", which refers to software that runs on the smartcard itself. As a counterpart of reader-side software, card software is also referred to as card-side software. Host software connects the smartcard and the user retrieves the smartcard contents via this connection.

4.2 The ISO 7816 Standard

The PC/SC [19] specification builds upon existing industry smartcard standards--ISO 7816 [20-21] and EMV [22]. ISO 7816 defines low-level device interfaces and device-independent application APIs as well as resource management to allow multiple applications to share smartcard devices attached to a system.

ISO 7816 includes at least six approved parts and has several additional parts under review. The following are overviews of different parts, but we only focus on the part 4 for exposition.

- Part 1 — Physical characteristics
- Part 2 — Dimensions and location of the contacts
- Part 3 — Electronic signals and transmission protocols
- Part 4 — Inter-industry commands for interchange
- Part5 — Number system and registration procedure for application identifiers
- Part 6 — Inter-industry data elements

A smartcard is attached to system through a card reader. The reader sends command to the card, and operations are carried out on the card. The questions are: What does the command look like? Is there any protocol or standard that sends commands? To support the application protocol APIs, a protocol message format is defined in ISO/IEC 7816-4, through which the function calls, associated parameters,

and status response parameters are exchanged between the reader-side application and the card-side application. This message format is characterized by application protocol data units (APDUs), which are conveyed between the reader-side and the card-side application by the link-level protocol (generally either a T=0 or a T=1 protocol defined in ISO 7816-3). Fig. 7 below shows the communications architecture between reader-side and card-side application.

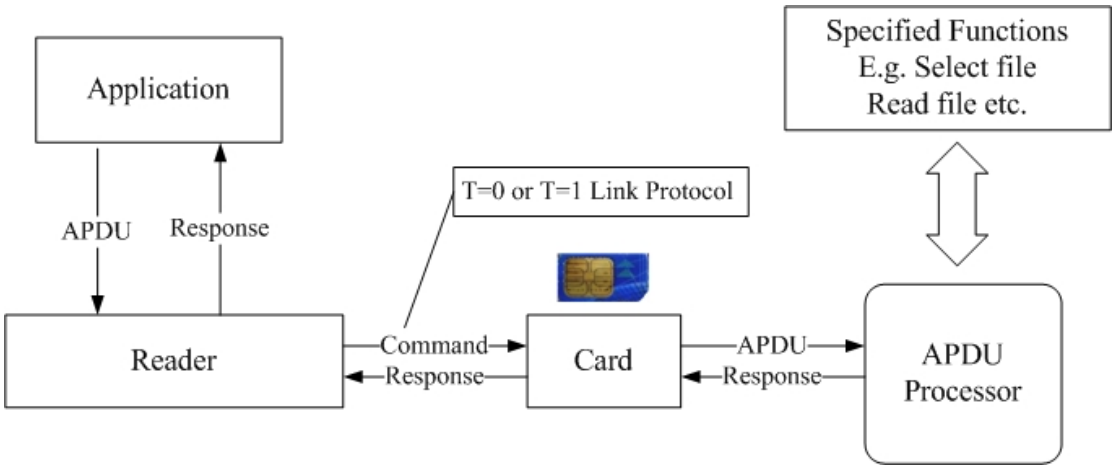

Fig. 7: Application Communications Architecture.

5. System Description

The WIRE1x is an implementation of IEEE 802.1x client. It is a free as well as open-source. It has been practically used in real-word applications with FreeRADIUS (http://www.freeradius.org/) as to secure WLAN environments. The source code could be downloaded from (http://wire.cs.nthu.edu.tw/wire1x/index.html). By reading the descriptions below, one should easily understand the architecture of WIRE1x and the authentication flow of IEEE 802.1x using EAP-SIM.

The software architecture of the WIRE1x EAP-SIM module can be divided into four components:

1. The WIRE1x core procedure, which involves the supplicant PAE state machine, the processing of EAPOL and the Ethernet headers, the building of response frames, and the decoding of the EAP packets.
2. EAP-SIM related functions, which decode and process the received EAP-SIM packet. And to derive necessary EAP-SIM related information required for building EAP-SIM response packets.
3. SIM handler in sm_handler.cpp, which is responsible for all connections and data exchanges between supplicant and the smartcard reader attached on the client's system.
4. Other open source library such as WinPcap [23] and Libnet [24] for capturing and writing frames, and OpenSSL [25], for handling

cryptographic hash functions.

5.1 WIRE1x Supplicant PAE State Machine

The supplicant PAE state machine is the core of any implementation of IEEE 802.1x supplicant. It specifies the behavior of the supplicant and the interactions with the authenticator. In WIRE1x, roughly speaking, it is implemented in four files: dot1x_globals.cpp, eap.cpp, eapol.cpp, and os_generic.cpp. And the definition of the variables of state machine is in dot1x_globals.h. Additionally, the EAP code field and type field specified in RFC 3748 [8] are defined in eap.h, and EAPOL header and Ethernet header are defined in eapol.h. The eap.cpp is responsible for building the response frames and decoding the EAP packets. The eapol.cpp is responsible for starting EAPOL process, performing necessary PAE state transitions, decoding and transmitting EAPOL frames.

WinPcap and Libnet are used to capture/write frame from/to link layer. In os_generic.cpp, get_frame() employs pcap_dispatch() to capture EAP frames. The send_frame() employs libnet_write_link() to send EAP frames.

5.2 EAP-SIM

After the packet is decoded and determined to be a EAP-SIM packet, following steps of determining the subtype of EAP-SIM packet, processing attributes included in the packet, establishing link with smartcard reader to derive SIM information needed to run GSM algorithm for deriving session keys, and building response EAP messages are carried out via these SIM related functions. They are implemented in 5 files: eapsim.cpp, sim.cpp, simd5.cpp, simd11.cpp, sm_handler.cpp with SIM specific variables, struct eapsim_data, defined in userconf.h.

eapsim.cpp is accountable for initializing necessary SIM specific variables, acquiring IMSI from the smartcard to use as username (NAI), decoding packet and calling functions for different types of EAP requests, and handling included attributes.

sim.cpp handles the processing of each received attributes and the generation of attributes needed to include in response packets and necessary sessions keys. Each function in sim.cpp is fundamentally designed for processing one specific received attribute and generating corresponding attribute. This function then includes this attribute into the response packet, which constructs part of the packet. After all attributes needed are included, the response packet is complete.

simd5.cpp and simd11.cpp are responsible for calculating MAC. The calculation varies according to the highest protocol version supported by both supplicant and

authenticator. MAC mechanism based on cryptographic hash functions, HMAC function of OpenSSL, is used.

Next, we will demonstrate the authentication procedure of WIRE1x EAP-SIM module in Fig. 8, and Fig. 9 illustrates transmitted and received packets.

1. Users launch WIRE1x EAP-SIM module as Fig. 10 and select the device to be authenticated by calling pcap_findalldevs(). MN begins to associate with AP. Both MN and AP will transit to the CONNECTING state.
2. MN sends an EAPOL_Start frame by libnet_write_link() to AP to initialize the authentication process.
3. When AP receives EAPOL-Start frame, it will send an EAP-Request/Identity packet to obtain the MN's identity. When the MN receives the EAP frame by pcap_dispatch(), the EAP frame is parsed by eap_decodepacket() and eapol_decode packet() located in eap.cpp and eapol.cpp, respectively. Moreover, according to the decoding result, the supplicant PAE state machine transits to ACQUIRED state if the request is received successfully. As for EAP-SIM, eap_build_responseId( )in eap.cpp is called which will later call eapsim_get_username() to acquire IMSI from the smartcard as MN’s identity.
4. MN sends back EAP-Response/Identity containing MN's identity to the authenticator. Subsequently, the authenticator and the authentication server will perform necessary message exchanges as depicted in Fig. 11.
5. When MN receives EAP-Request/Challenge which contains RADIUS-Access-Challenge, the supplicant PAE state machine transits to the AUTHENTICATING state. As SIM is determined to be the authentication method, eapsim_decode_packet() will determine whether the packet received is of subtype start or challenge. Then, the included attributes in that packet are processed by calling corresponding functions defined in sim.cpp.
6. First, MN will receive an EAP-Request/SIM/Start packet. Then the sim_build_start() and sim_at_version_list() are called, each constructing part of the response packet. When all attributes are included, the packet is completed and sent.
7. Next, the EAP-Request/SIM/Challenge with AT_MAC is received by the MN. sim_mac(), sim_v1_response(), sim_rand(), are called to derive MAC and session keys. sim_mac() calls do_v0_at_mac or do_v1_at_mac to calculate MAC depending on the protocol versions. fips186_2_prng() is used to generate pseudorandom number for key derivation. However, if the MAC calculated doesn’t match with the one received, MN will issue an EAP-Response/SIM/Client-Error packet and the authentication exchange will be terminated. If all checks out, the peer responds with the

EAP-Response/SIM/Challenge containing the AT_MAC attribute that covers the peer's SRES response values.

8. On the basis of the result of the EAP authentication method, the RADIUS server decides whether to authorize the user or not and a corresponding EAP packet is sent to the supplicant. If the user is authorized as in Fig. 12, the supplicant captures the EAP-Success packet and the supplicant PAE state machine will transit to the AUTHENTICATED state. Otherwise, the supplicant captures the EAP-Failure and transits to the HELD state.

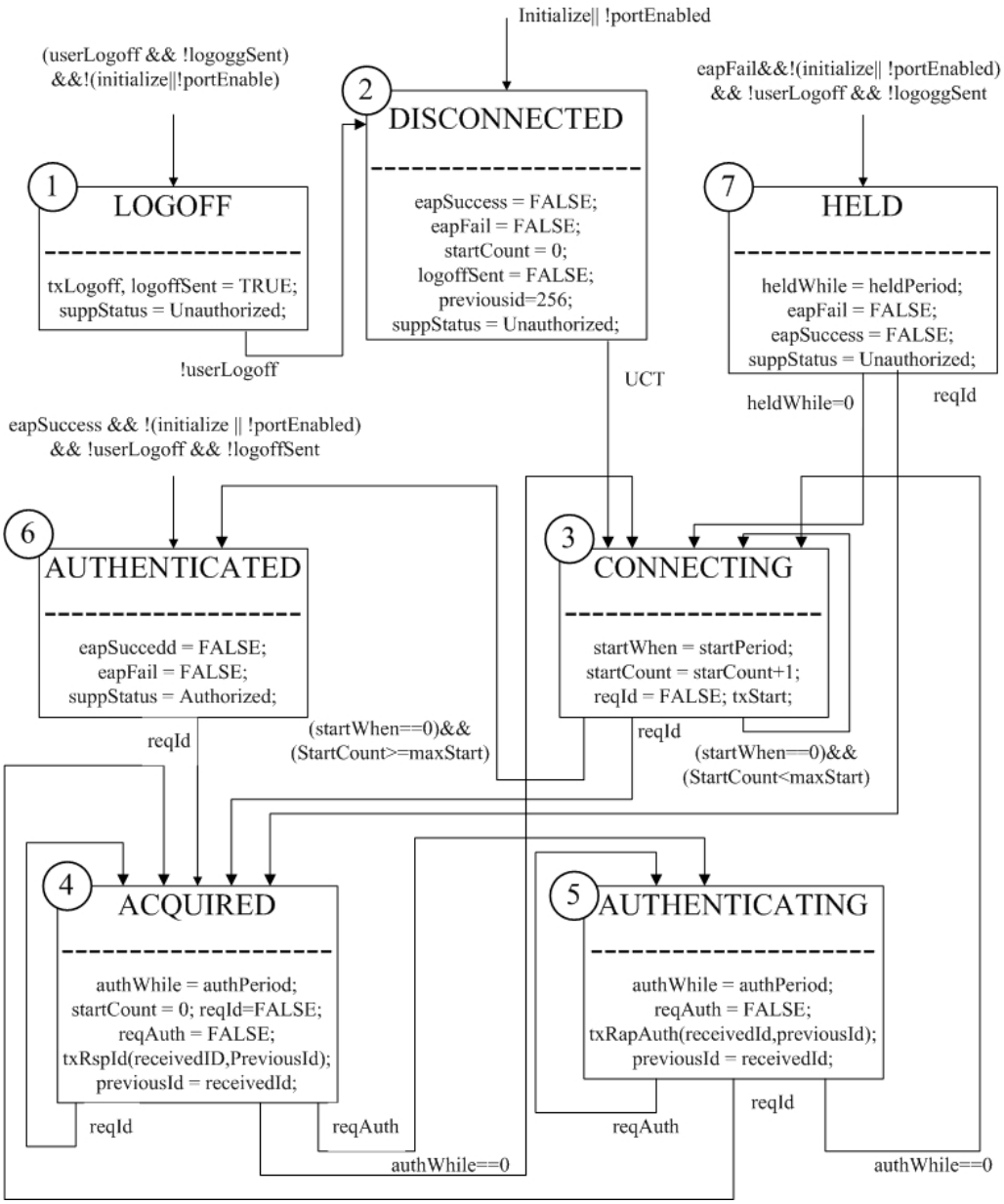

Fig. 8-1: The supplicant PAE state machine.

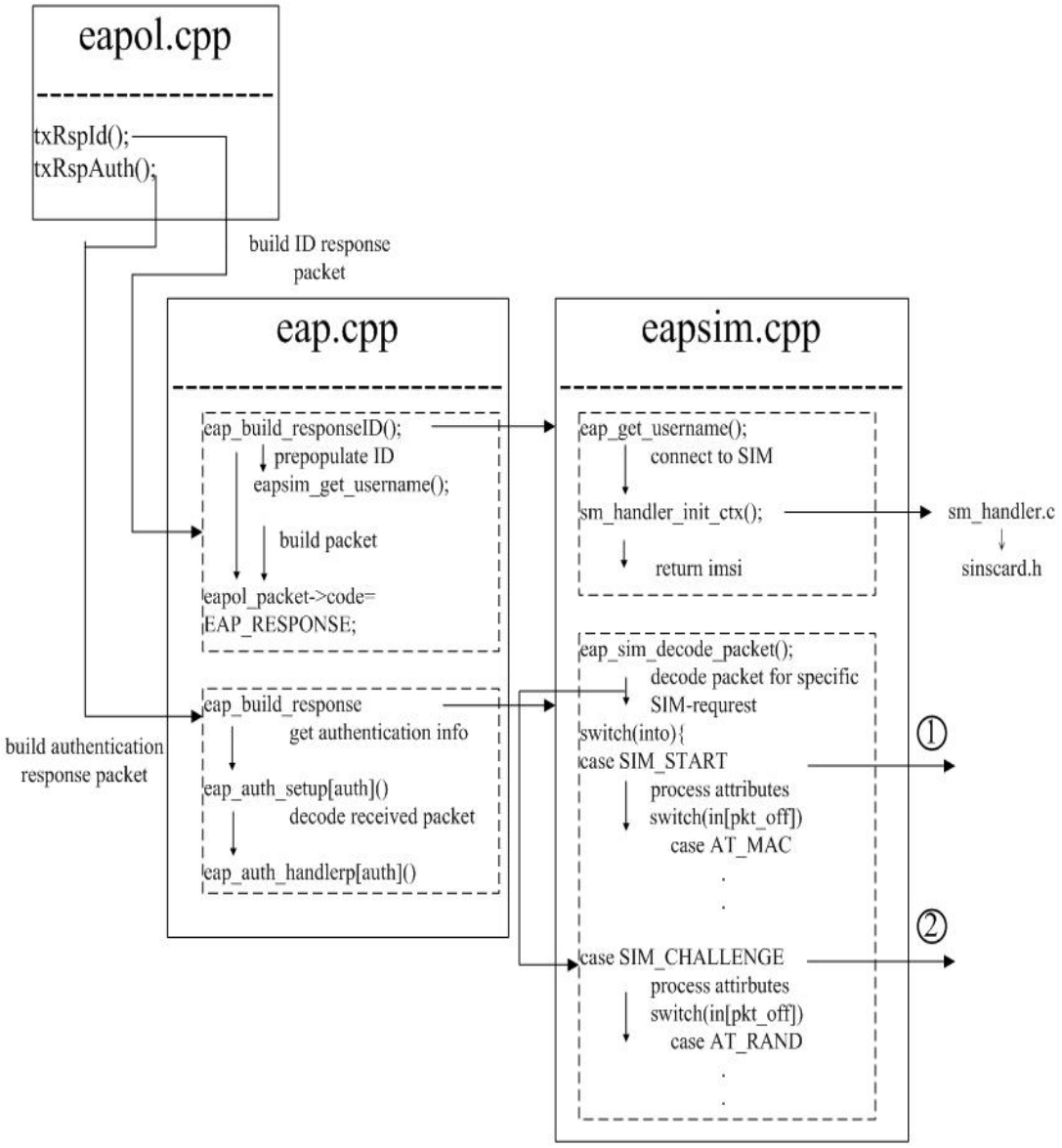

Fig. 8-2: UML diagram for the SIM related function in WIRE1x

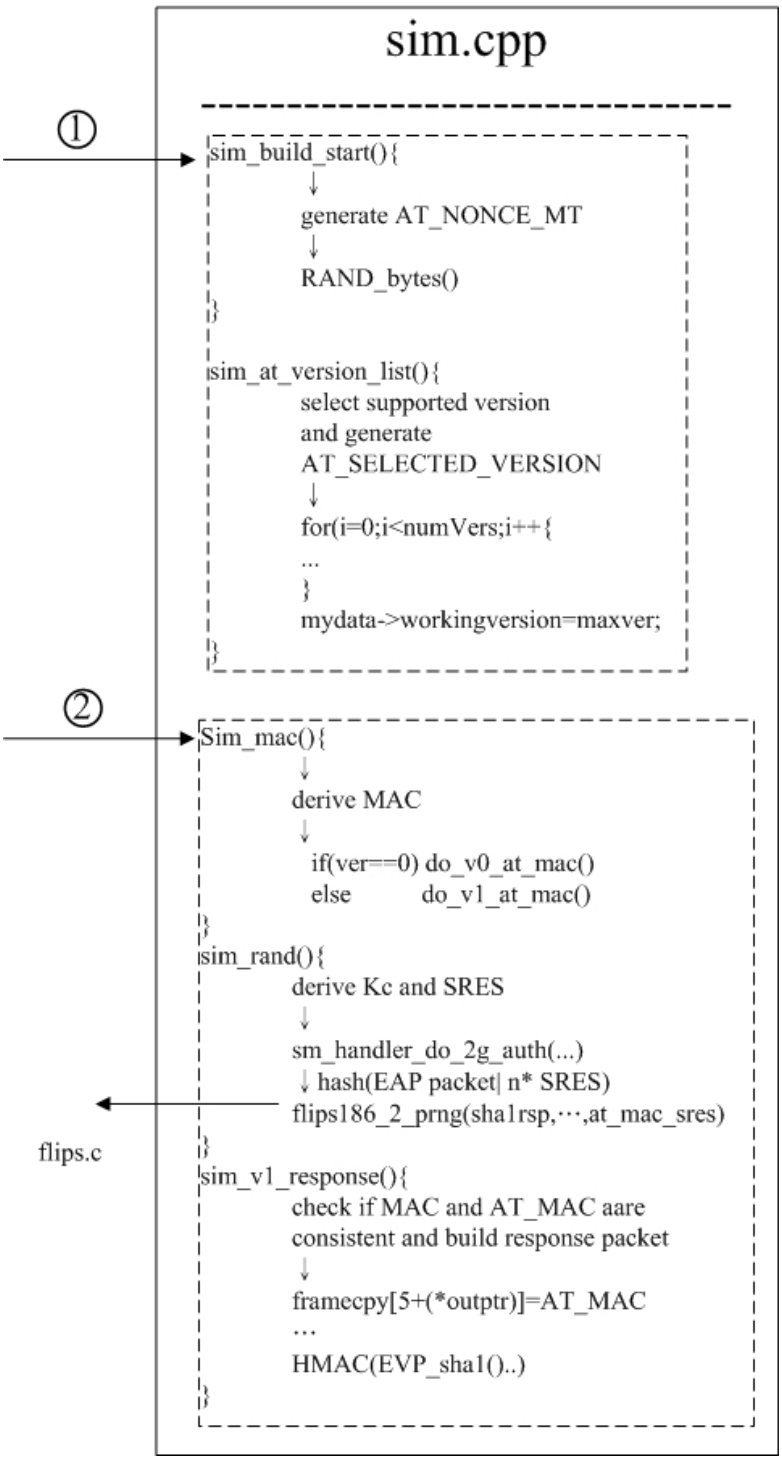

Fig. 8-3: UML diagram for the SIM related function in WIRE1x

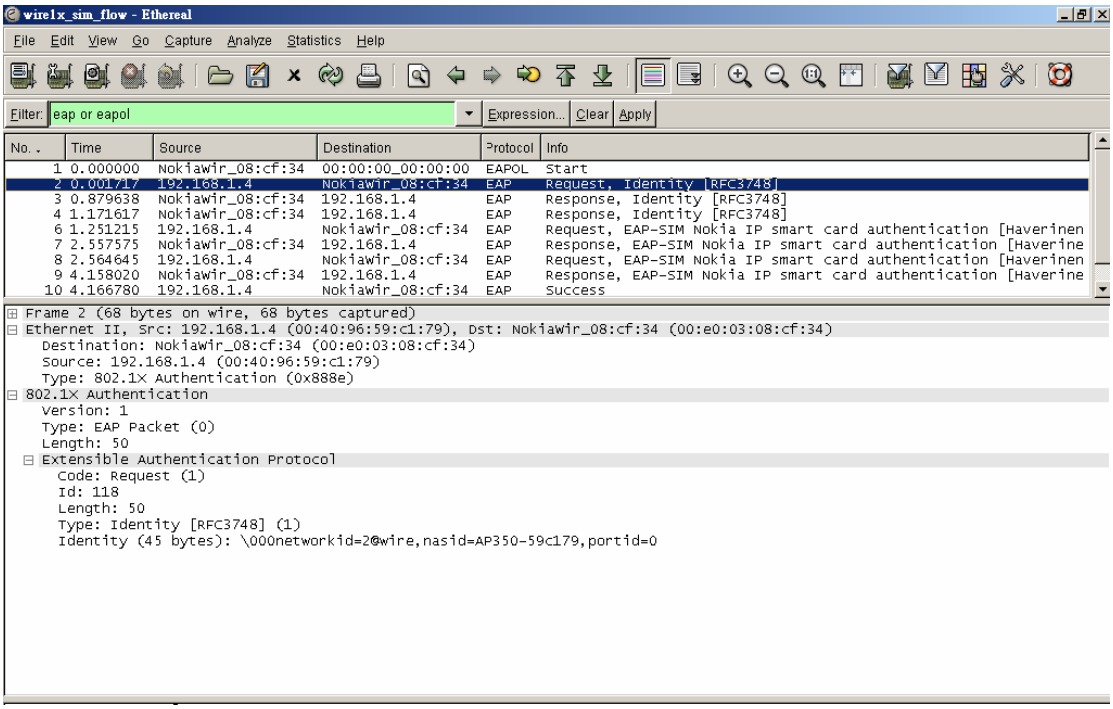

Fig. 9-1: Packet contents of EAP-SIM/Request/Identity

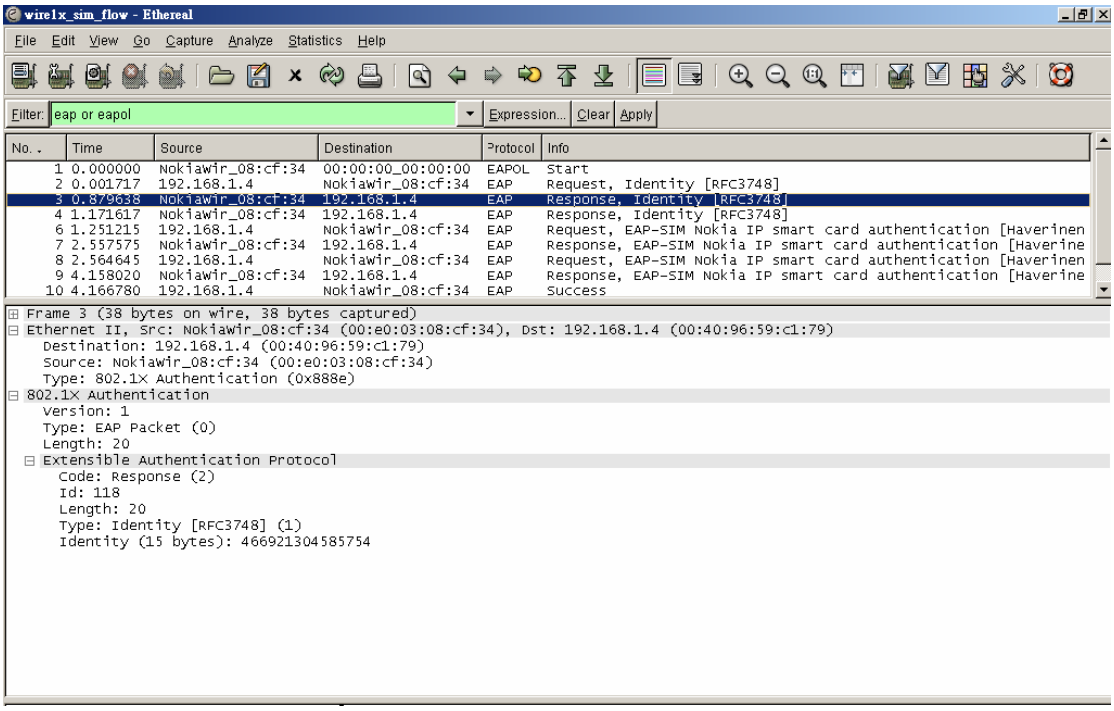

Fig. 9-2: Packet contents of EAP-SIM/Response/Identity

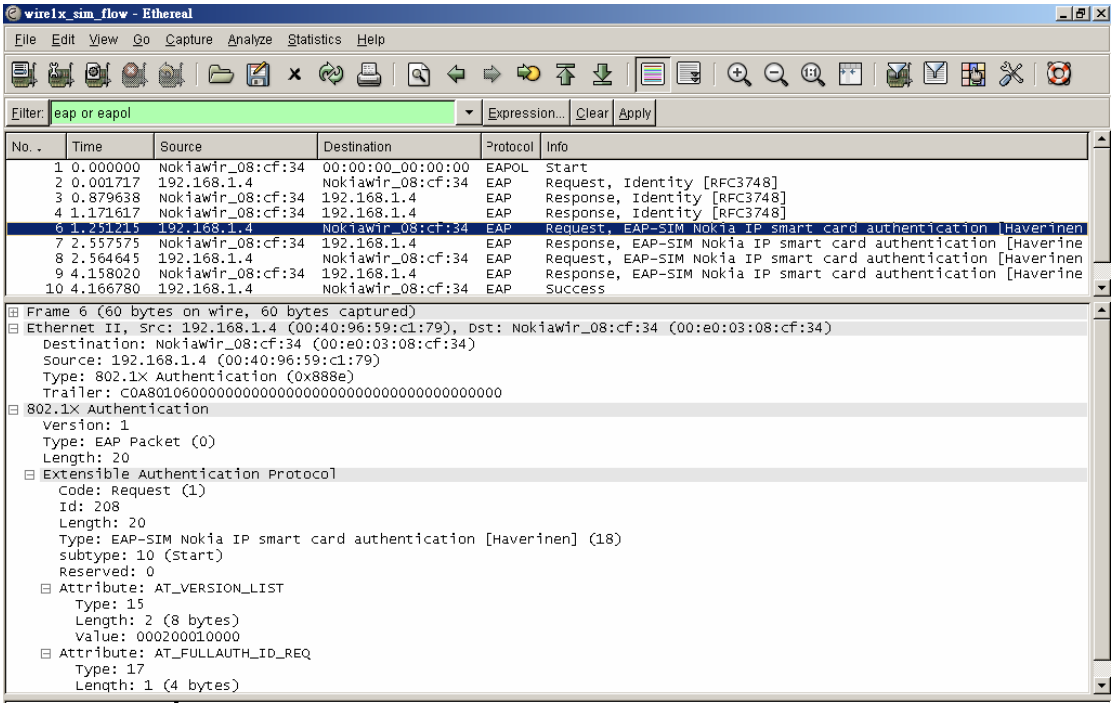

Fig. 9-3: Packet contents of EAP-SIM/Request/Start

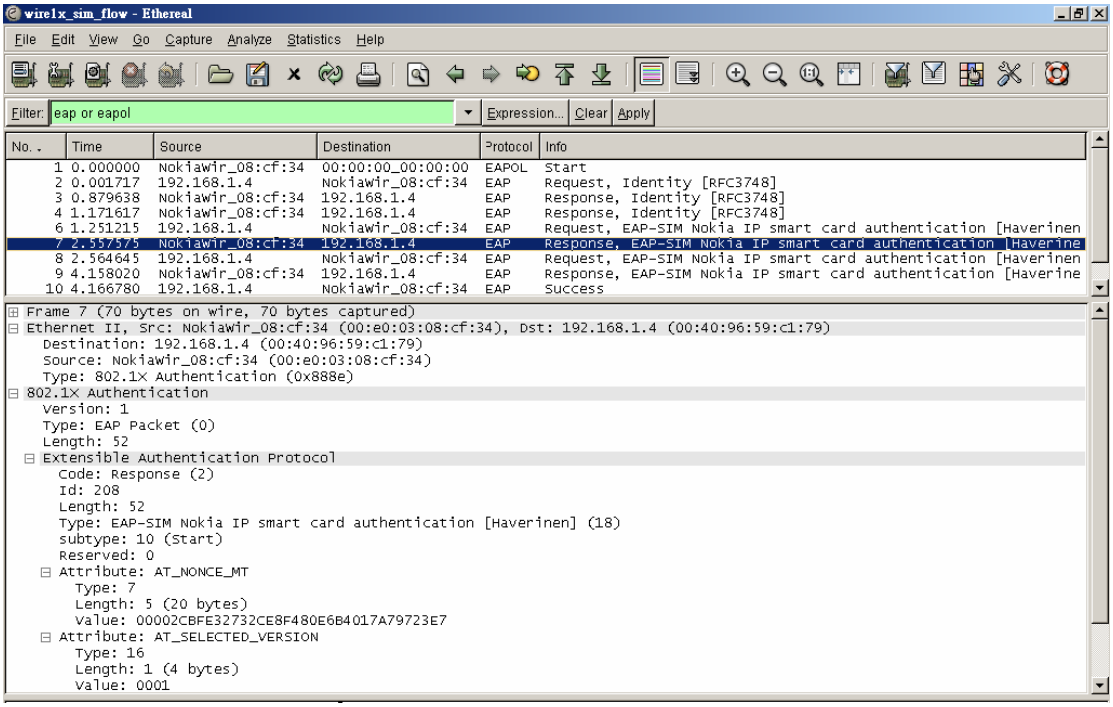

Fig. 9-4: Packet contents of EAP-SIM/Response/Start

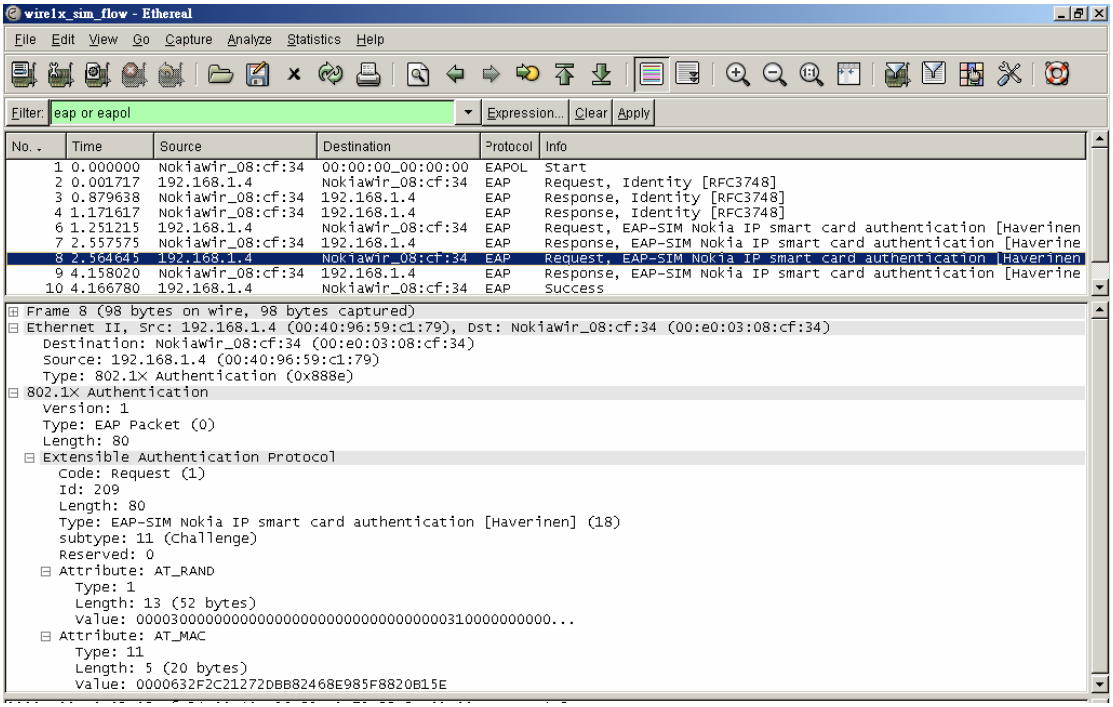

Fig. 9-5: Packet contents of EAP-SIM/Request/Challenge

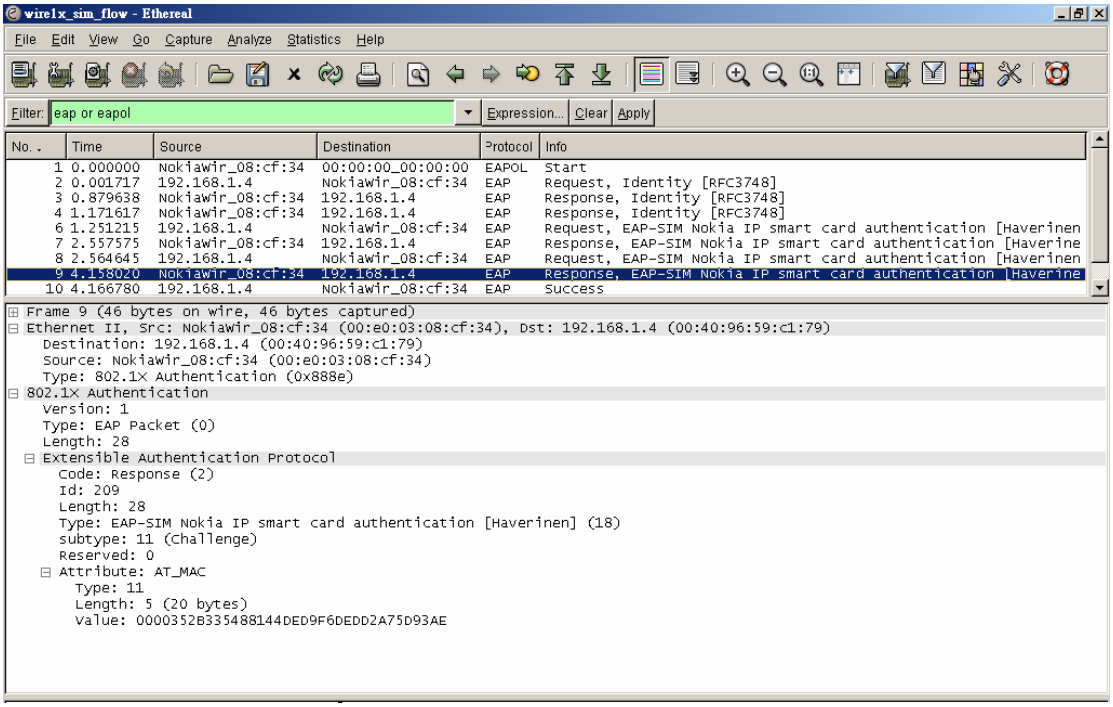

Fig. 9-6: Packet contents of EAP-SIM/Response/Challenge

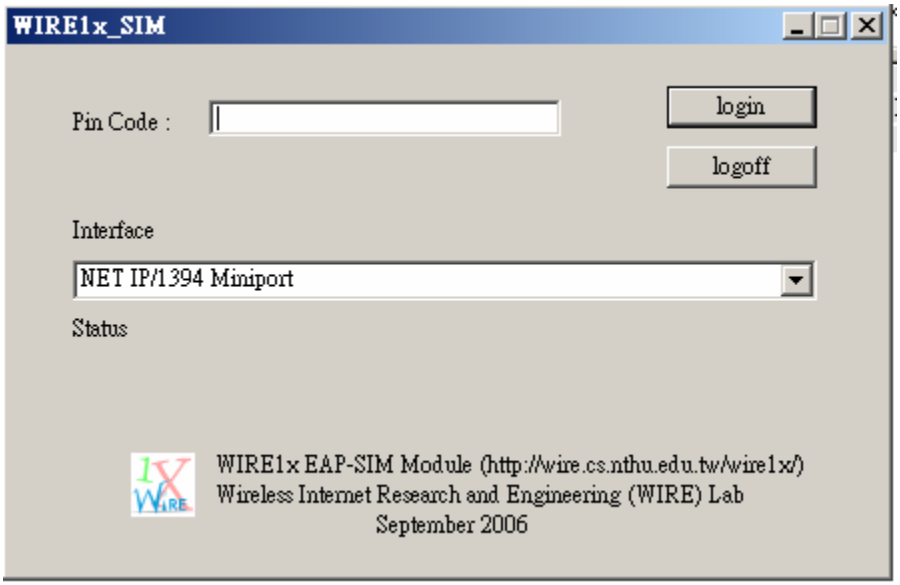

Fig. 10: GUI of WIRE1x EAP-SIM module

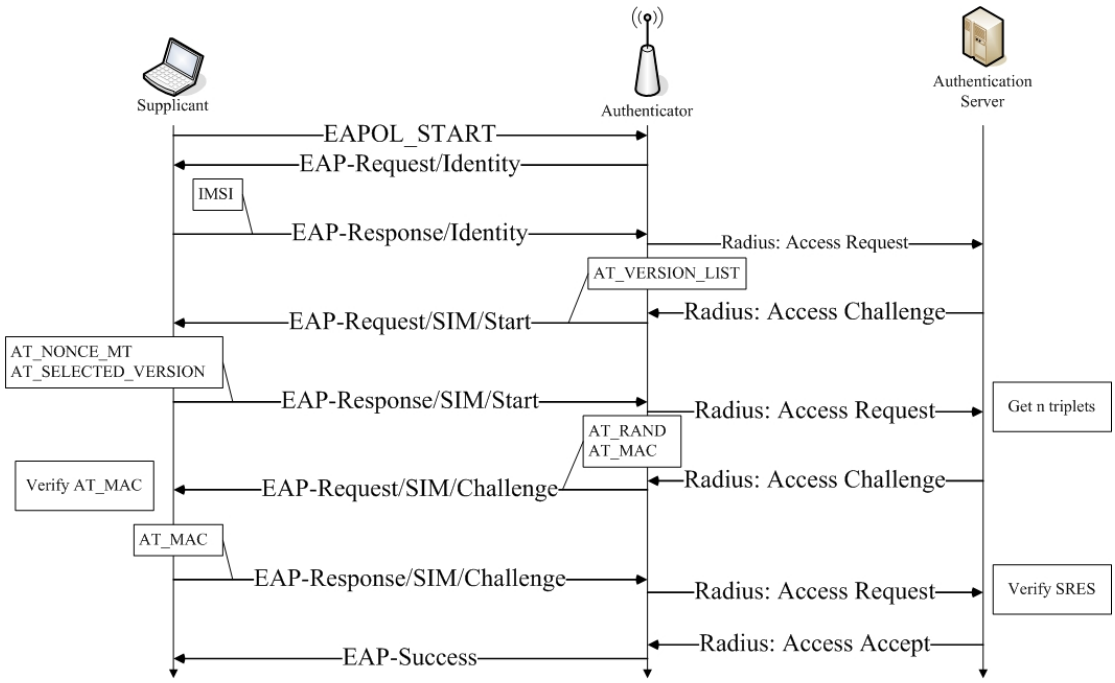

Fig. 11: Message exchange between the authenticator and the authentication server.

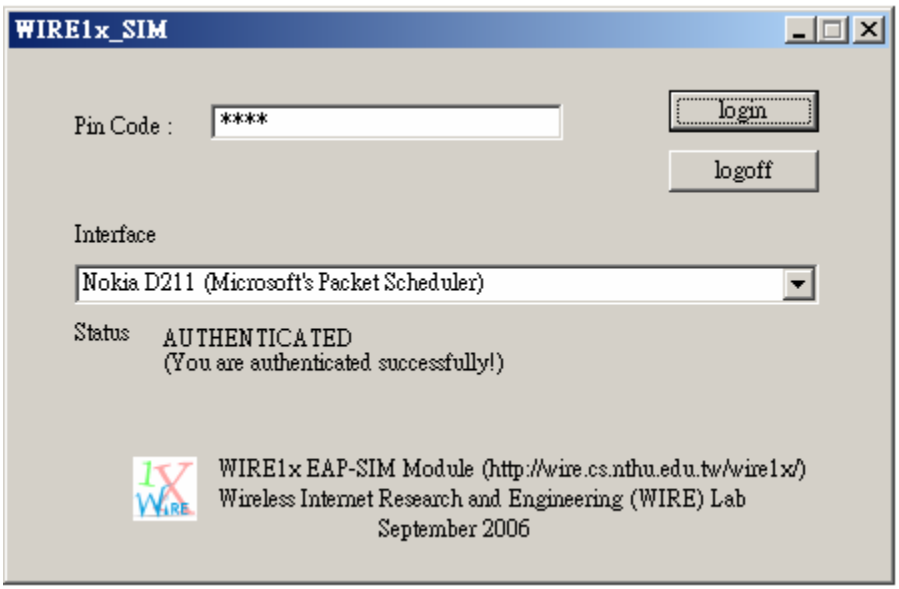

Fig. 12: Supplicant is authenticated successfully.

5.3 SMhandler

During the EAP-SIM authentication process, we must use a SIM card and retrieve attributes from it. Due to the prevalence of smartcard in recent years, it is necessary for standards to be established; fortunately, the existence of PC/SC solves this problem. Nowadays, most of the smartcards and readers are compatible with this standard. We use Microsoft's PC/SC API to develop our application.

The operations between application and smartcard are as follows:

1. Application uses PC/SC API to transmit APDU commands to operating system.
2. Operating system transmits APDU commands to reader through a driver which adopts PC/SC standard.
3. Reader transmits the commands to the smartcard transmission controller using protocol T=0 or T= 1.
4. The transmission controller transmits APDU commands to the smartcard COS, and after the smartcard COS executes APDU commands, it returns the results.

The file winscard.h is a PC/SC API. The development of smhandler is based on winscard.h to provide smartcard manipulation. Fig. 13 is an overview of the operations and the state transitions of smhandler. The operations are generalized as follows:

1. sm_handler_init_ctx() in sm_handler.cpp initializes context and gets ready to authenticate SCardEstablishContext.
2. sm_handler_get_readers() in sm_handler.cpp retrieves available reader.
3. sm_handler_card_connect() in sm_handler.cpp connects to the smartcard
4. sm_handler_wait_card_ready() in sm_handler.cpp keeps waiting for up to 20 seconds for the smartcard to become ready.
5. sm_handler_2g_imsi() in sm_handeler.cpp transmits several APDUs to retrieve IMSI.

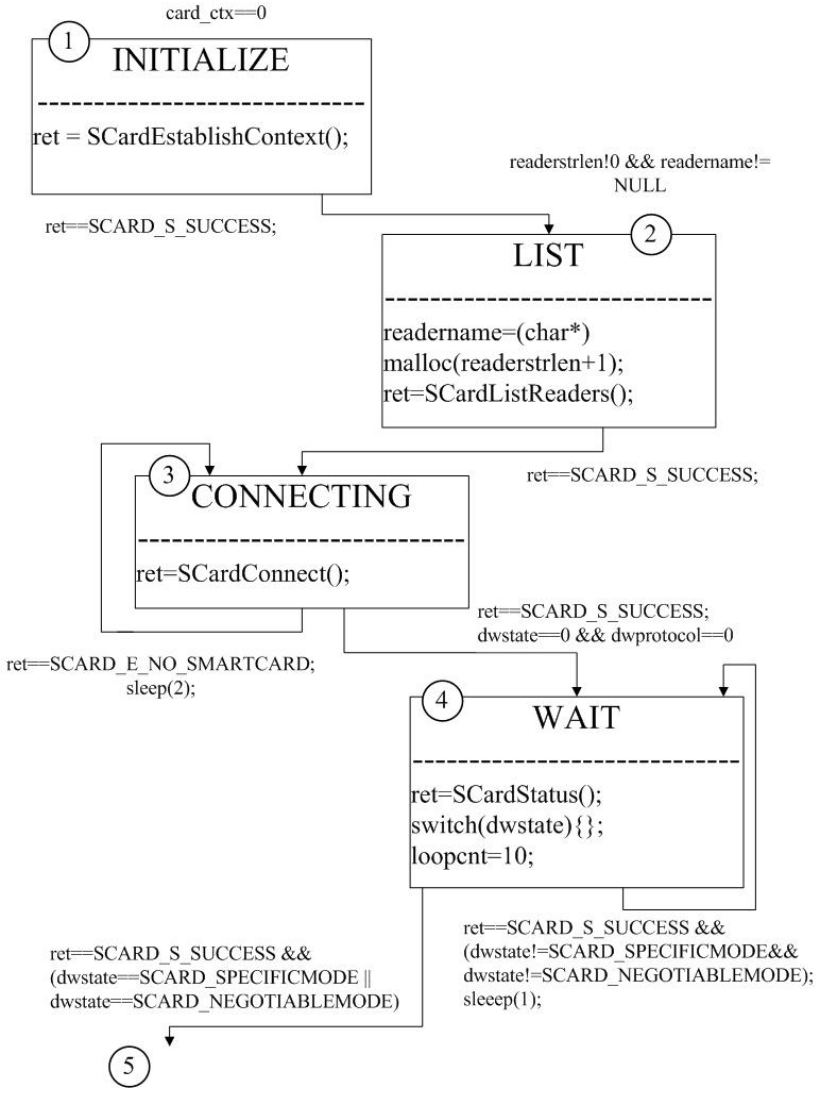

Fig. 13-1: The operations and state transitions of sm_handler.

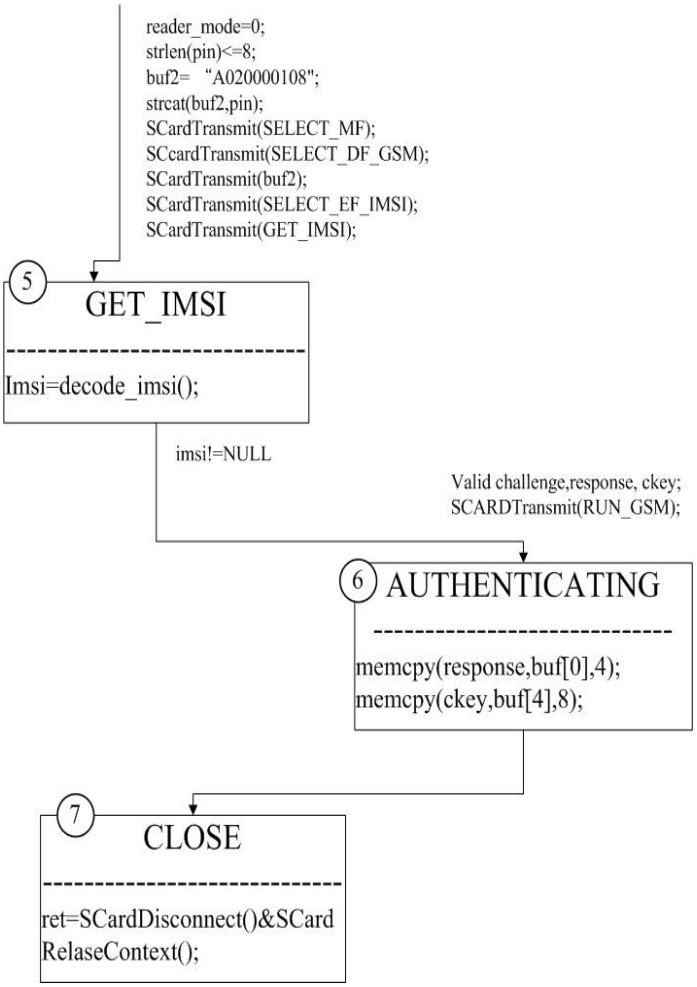

Fig. 13-2: The operations and state transitions of sm_handler.

6. Conclusions and Future Work

We believe that EAP-SIM will be a suitable authentication mechanism used in the integration of WLAN and cellular networks in the future. From users' point of view, the users can just rely on one SIM card to roam among different WLAN and cellular networks and does not have to manage a multitude of passwords. Besides, the authentication process is very simple and intuitive, no complicated settings is required. EAP-SIM will also provide users a much better protection when using the wireless networks because mutual authentication is supported. As for network operators, providing SIM based authentication will lower the cost of account management, and will also to achieve higher protection of the valuable assets of network operators. Most importantly, the threshold for deployment SIM based authentication is quite low because it is build upon the underlying GSM structure. Due to its usage simplicity, its superior security, and its cost efficiency, unsurprisingly, EAP-SIM will be one of the most widely adopted authentication method in the future.

Thus, WIRE1x EAP-SIM module is expected to have a tremendous impact in the promotion of IEEE 802.1x and wireless networks since it provides users using Microsoft Windows a simpler, more user-friendly, and securer authentication mechanism. Most importantly, WIRE1x is a free as well as an open-source software. It is believed that open-source is fundamental for any security-related software because it can be examined by anyone who concern about the implementation detail.

There are some optional functions specified in RFC 4186 that have not been implemented in WIRE1x EAP-SIM module yet. Moreover, the 3GPP has specified an enhanced Authentication and Key Agreement (AKA) architecture for the Universal Mobile Telecommunications System (UMTS), a 3rd generation (3G) AKA mechanism which includes mutual authentication, replay protection, and derivation of longer session keys. EAP-AKA is an EAP method based on 3G AKA, which is a more secure protocol than EAP-SIM. As 3G mobile network becomes increasingly popular, EAP-AKA will also be adopted to provide better secure network environment. Hence, the future work of WIRE1x is to provide more functionality about EAP-SIM and to implement EAP-AKA and more EAP methods to catch up with the trend.

References

[1] "NCHC" http://www.nchc.org.tw/

[2] IEEE Standard 802.11b-1999, "LAN/MAN Specific Requirements – Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band," 1999.

[3] IEEE Standard 802.11i, "Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Amendment 6: medium access control (MAC) security enhancements," July 2004.

[4] J.-C. Chen, M.-C. Jiang, and Y.-W. Liu, "Wireless LAN security and IEEE 802.11i," *IEEE Wireless Communications*, vol. 12, pp. 27.36, Feb. 2005.

[5] IEEE Standard 802.1X-2001, "IEEE standard for local and metropolitan area networks, port-based network access control," Oct. 2001.

[6] C. Rigney, S. Willens, Livingston, A. Rubens, Merit, W. Simpson, and Daydreamer, "Remote Authentication Dial In User Service (RADIUS)," IETF RFC 2865, June 2000.

[7] "WIRE1x," http://wire.cs.nthu.edu.tw/wire1x/index.html

[8] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, "Extensible Authentication Protocol (EAP)," IETF RFC 3748, Jun. 2004.

[9] R. Rivest, "The MD5 Message-Digest Algorithm," IETF RFC 1321, Apr. 1992.

[10] B. Aboba, D. Simon, “PPP EAP TLS Authentication Protocol,” IETF RFC 2716, Oct. 1999.
[11] P. Funk and S. Blake-Wilson, “EAP tunneled TLS authentication protocol version 1 （EAP-TTLSv1）,” IETF Internet Draft, <draft-funk-eap-ttls-v1 -00.txt>, work in progress, Feb. 2005.
[12] A. Palekar, D. Simon, J. Salowey, H. Zhou, G. Zorn, and S. Josefsson, “Protected EAP protocol （PEAP）, version 2,” IETF Internet Draft, <draft-josefsson-pppext-eap-tls-eap-10.txt>, work in progress, Oct. 2004.
[13] “3GPP,” http://www.3gpp.org/
[14] H. Haverinen, and J. Salowey, “Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM),” IETF RFC 4186, January 2006.
[15] “Open1x,” http://open1x.sourceforge.net/
[16] J. Arkko, and H. Haverinen, “Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA),” IETF RFC 4187, January 2006.
[17] Jyh-Cheng Chen, and Tao Zhang, “*IP-Based Next-Generation Wireless Networks: Systems, Architectures, and Protocols*,” Wiley, January 2004.
[18] “GSM Security,” http://www.gsm-security.net/
[19] “PC/SC Workgroup,” http://www.pcscworkgroup.com/
[20] “ISO,” http://www.iso.ch
[21] “EMVCo,” http://www.emvco.com
[22] “CardWeck,” http://www.cardwerk.com/smartcards/smartcard_standard_ISO7-816.aspx
[23] “WinPcap,” http://winpcap.polito.it/
[24] “Libnet,” http://libnet.sourceforge.net/
[25] “OpenSSL,” http://www.openssl.org/