Design and Implementation of WIRE1x Connection Manager

Chun-Yu Chen, Han-Hsing Chiu, Chien-Chia Chen, Jui-Yi Chen, Yi-Ken Ho, and Jyh-Cheng Chen
Wireless Internet Research and Engineering Laboratory
National Tsing Hua University
Hsinchu, Taiwan

November 15, 2007

Abstract

This document presents the design and implementation of WIRE1x connection manager. WIRE1x is an open-source implementation of IEEE 802.1x client (supplicant) developed by the Wireless Internet Research & Engineering (WIRE) Laboratory, National Tsing Hua University. In previous versions of WIRE1x, we focus on enhancing security mechanisms in WIRE1x. After WIRE1x version 2.2, we integrate a Connection Manger into WIRE1x. The Connection Manager includes a user-friendly GUI so users can use the software and connect to a wireless LAN easily.

1. Introduction to WLANs

A WLAN is a wireless local area network, which is the linking of two or more computers without using wires. As the infrastructure of WLAN is widely constructed, WLAN becomes much more popular in the last few years. It breaks the restriction of conventional wired communications and provides a convenient way to access network at anywhere and any time. However, WLAN transfers packets by radio instead of physical wires, so the data may be sniffed and cracked easily by third party. In these years, there are more and more methods to enhance security on WLAN. We will briefly introduce them as follow.

1.1 Open system, shared key and WEP

The IEEE 802.11b [1] standard is a widely adopted wireless Internet access protocol. It has defined the following two basic security mechanisms: open system authentication and shared key authentication. According to the standards, open system has to be implemented on all wireless products. However it has less security because the access point (AP) doesn't authenticate the client station in fact. In open system

authentication, the client station sends an authentication request to the access point and then the access point replies the status of authentication. The passing frames don't have any data for authentication except for the client station's MAC address. But MAC address can be easily modified by intended attack actions.

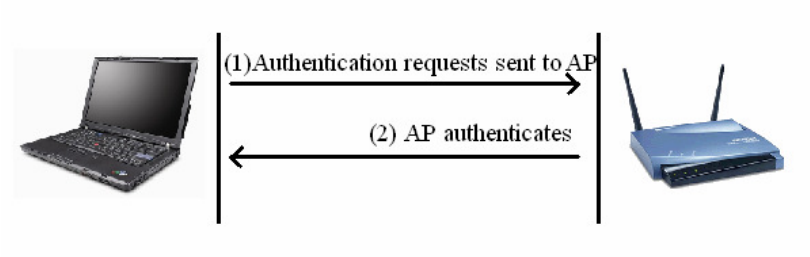

Fig. 1 : Open system authentication

In shared key authentication, WEP is used for authentication. A four-way challenge-response handshake is used:

(1) The client station sends an authentication request to the access point.

(2) The access point sends back a clear-text challenge.

(3) The client has to encrypt the challenge text by using the WEP key and send it back in another authentication request.

(4) The access point decrypts the material and compares it with the clear-text it sent before. Depending on the result of this comparison, the access point sends back a positive or negative response.

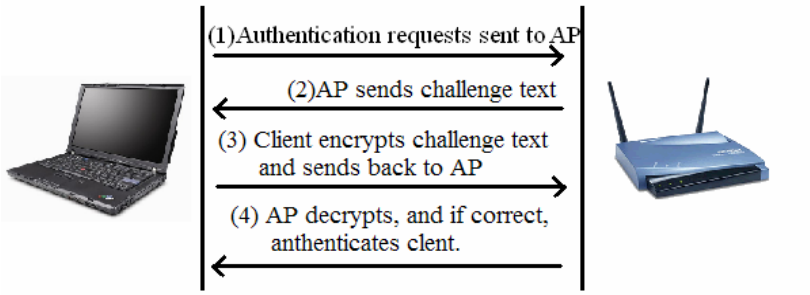

Fig. 2 : Shared key authentication

After the authentication and association, WEP can be also used for encrypting the data frames. WEP (Wired Equivalent Privacy) is part of the IEEE 802.11 standard ratified in September 1999 [2]. WEP uses the stream cipher RC4 for confidentiality and the CRC-32 checksum for integrity. However, it has been reported that WEP adopted in IEEE 802.11b is vulnerable. Key size is not the only security limitation in WEP. Cracking a longer key requires interception of more packets only. In fact, there are many other weaknesses in RC4 [3]. In order to enhance the security on wireless networks, the industry develops some solutions based on 802.1x. Because 802.1x is based on EAP, we have to discuss EAP first at next section.

1.2 EAP

EAP (Extensible Authentication Protocol) [4] is a universal authentication framework frequently used in wireless networks and Point-to-Point connections, not a specific authentication mechanism. Fig.3 depicts the basic framework of EAP system. In the authentication layer, many different authentication methods are developed on the basis of EAP layer. There are currently about 40 different methods defined in IETF RFCs, including EAP-MD5 [5], EAP-TLS [6], EAP-TTLS [7], EAP-PEAP [8] and so on. When a new method is created, it is easy to add to this well-defined protocol.

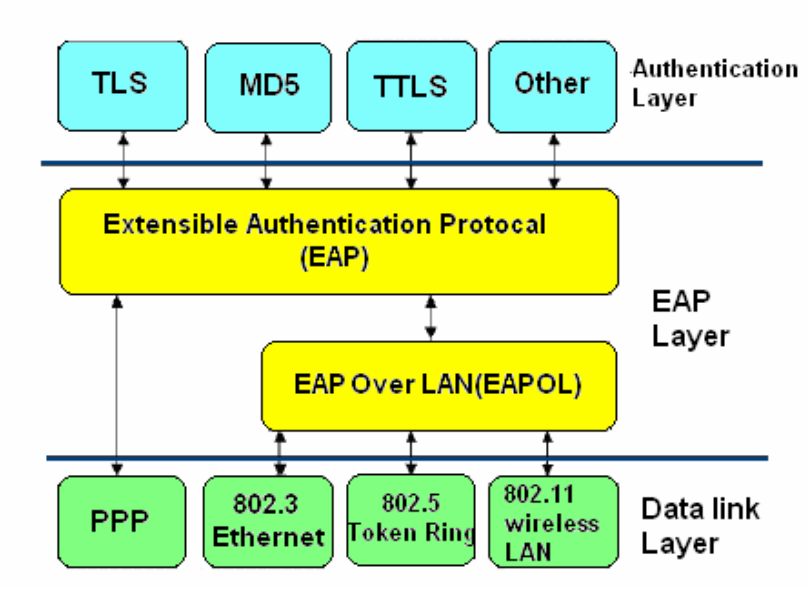

Fig. 3 : EAP and associated layers

The simple process of exchanging EAP messages is illustrated as Fig.4. Actually, the progress needs not be so complex. It simply demonstrates some capabilities of EAP.

(1) The authenticator sends a request packet in order to authenticate the client.
(2) The client gets the identity information from user, and then sends it back for response.
(3) Once the authenticator identifies the user, it sends MD5 challenge to the client.
(4) Because the client uses token card as the default method for authentication, it sends back a NAK and proposes the authenticator use a generic token card [9] to do authentication.
(5) The authenticator sends a challenge using generic token card to the client.
(6) The user keys in the card numbers, and the client sends the response back.
(7) The client's response is correct, and then the authenticator sends the success message.

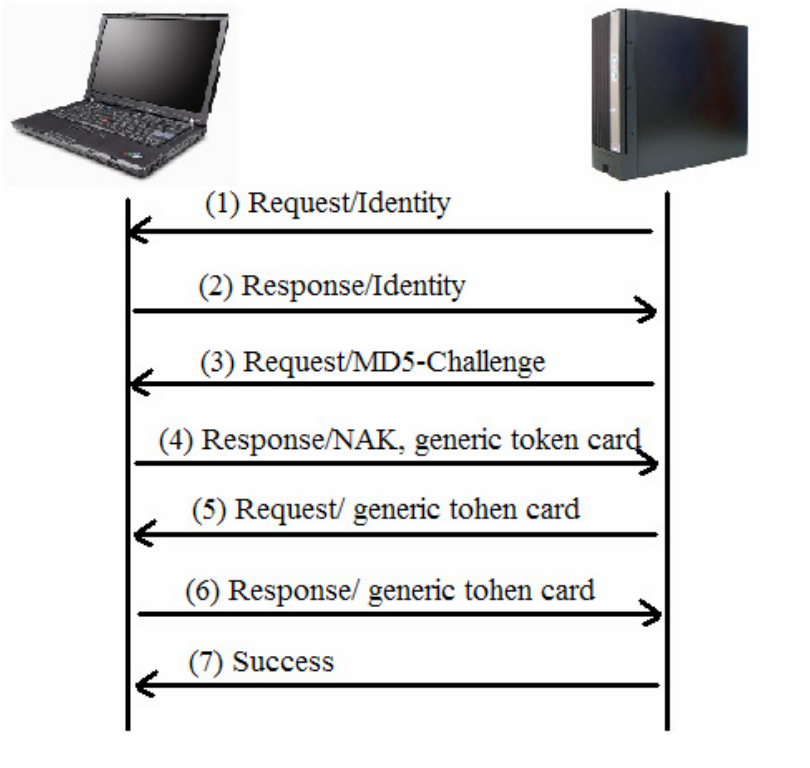

Fig. 4 : A simple process of exchanging EAP messages

EAP merely defines message formats. Each protocol that uses EAP defines a way to encapsulate EAP messages within that protocol's messages. Take 802.1x for example, this encapsulation is called **EAPOL** (EAP over LANs).

1.3 IEEE 802.1x

As we mentioned before, 802.1x is usually used to enhance the security on wireless networks. 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. 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 and end users need to be capable with IEEE 802.1x.

In an IEEE 802.1x system, there are three major components: supplicants, authenticators and authentication servers. A supplicant is usually a mobile node (MN), which is similar to a client. An authenticator handles communications between servers and supplicants, and is usually represented by AP. An authentication server, such as a RADUIS server, provides authentication, authorization, and accounting (AAA).

There is a port access entity (PAE) in both supplicant and authenticator that operates the algorithm and protocol associated with the authentication mechanisms. The authenticator PAE controls the state of the controlled port depending on the result of the authentication process. Before the authentication succeeds, for the sake of

forbidding the user to access other services provided by the authenticator's system, the *controlled port* is in the unauthorized state (switch off). Simply the *uncontrolled port* is switched on to direct nothing but IEEE 802.1x messages to the authentication server. After the authentication is complete successfully, the authenticator PAE will switch on the controlled port to authorized state. Thus the supplicant is able to access services.

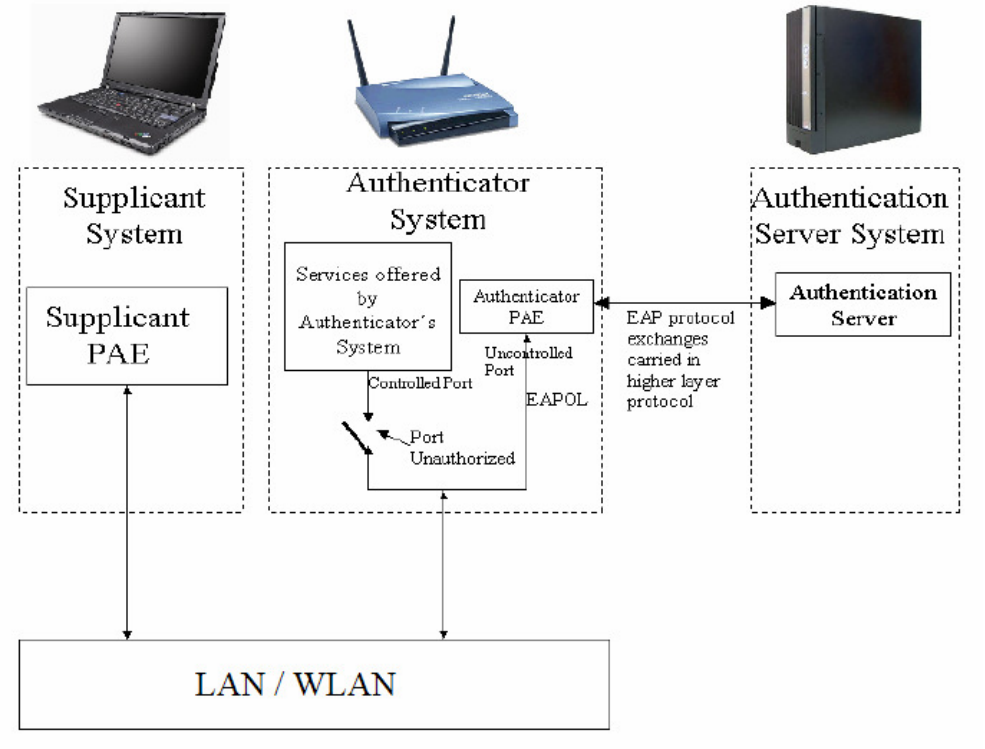

Fig. 5 : 802.1x system

All processes of authentication are finished between the supplicant and the authentication server, so the authenticator is simply a medium. The authenticator uses different types of EAP to send challenge and response packets between them. Between supplicants and authenticators, the protocol used is EAP over LAN (EAPOL) or EAP over wireless (EAPOW). Between authenticators and authentication servers, the protocol used is RADIUS (EAP over RADIUS).

The process of exchanging EAPOL messages is almost the same as the process of exchanging EAP messages. The most significant difference is that the supplicant could actively send EAPOL-Start message to initiate the entire authentication process. A typical process of exchanging EAPOL messages is illustrated as Fig.6.

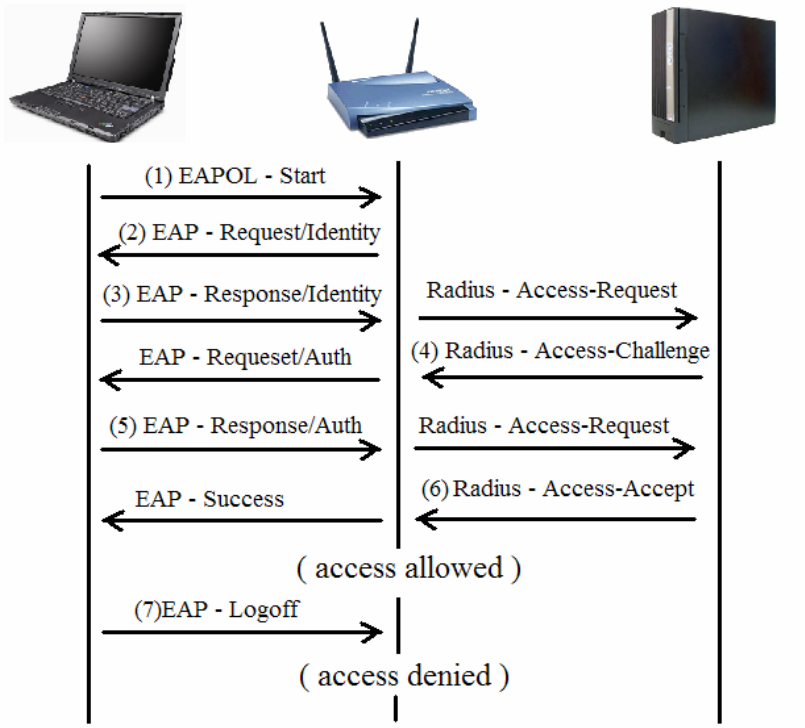

Fig. 6 : The process of exchanging EAPOL messages

1.4 WPA and 802.11i

WPA was created by the Wi-Fi Alliance [10], an industry trade group, which owns the trademark to the Wi-Fi name and certifies devices that carry that name. The implementation of WPA is optional and experimental on Apr. 2003, and it becomes mandatory on Nov.2003. The design of WPA is based on a Draft 3 of the IEEE 802.11i standard, so it implemented a subset of 802.11i. WPA is an intermediate solution for the security issues.

One major improvement in WPA is the Temporal Key Integrity Protocol (TKIP), which dynamically changes keys as the system uses. When combined with much larger length of key, this defeats the well-known key recovery attacks on WEP. Keys can be managed using two different mechanisms. One is *Pre-shared key mode* (PSK, also known as *personal mode*). It is designed for home and small office networks that cannot afford the cost and complexity of an 802.1x authentication server. Each user must enter a passphrase to access the network. The passphrase may be from 8 to 63 printable ASCII characters or 64 hexadecimal digits (256 bits). The other which has more security is *Enterprise mode*. It is designed to use with an authentication server and EAP just like IEEE 802.1x.

Eventually, IEEE 802.11i standard was ratified on 24 June 2004. It is an amendment to the 802.11 standard specifying securities. 802.11i makes use of the Advanced Encryption Standard (AES) block cipher, whereas WEP and WPA use the

RC4 stream cipher. It supports for more robust encryption algorithm CCMP based on AES to replace TKIP. Wi-Fi Alliance uses the final IEEE 802.11i as a new version of WPA, called *WPA2* or *RSN* (Robust Security Network).

2. WIRE1x

2.1 Introduction to WIRE1x

WIRE1x is an open-source implementation of IEEE 802.1x developed by the Wireless Internet Research and Engineering (WIRE) Laboratory, National Tsing Hua University, Hsinchu, Taiwan. This software provides users an easy way to access the WLAN by using EAP and the authentication mechanisms defined by IEEE 802.1x.The implementation of WIRE1x is based on Open1x [11], which is also a software designed for WLAN authentication, but supports only Linux. Although there are many users use Microsoft Windows, little free software could provide a comprehensive and convenient authentication solution. For these reasons, WIRE1x is developed for supporting various versions of Microsoft Windows and providing a number of authentication methods.

WIRE1x uses the third party library –WinPcap [12] for capturing and writing frames to the data link layer. WinPcap is used to capture packets sent from AP. It also helps WIRE1x to receive nothing but EAP frames by adjusting parameters in filter. Moreover, WIRE1x uses WinPcap to retrieve the list of network interfaces.

On initiating WIRE1x, there is a GUI as Fig.7. We must choose a network interface to deliver and accept packets at first. After connecting to one AP (if in wireless environment), we can set the configuration of what methods of EAP we want to use. In last version, four EAP authentication mechanisms are supported by WIRE1x, including EAP-MD5, EAP-TLS, EAP-TTLS, and EAP-PEAP. Finally, we press the “login” button to start authentication. The status of PAE machine will be shown in the textbox titled “STATUS:” in the middle of the window form.

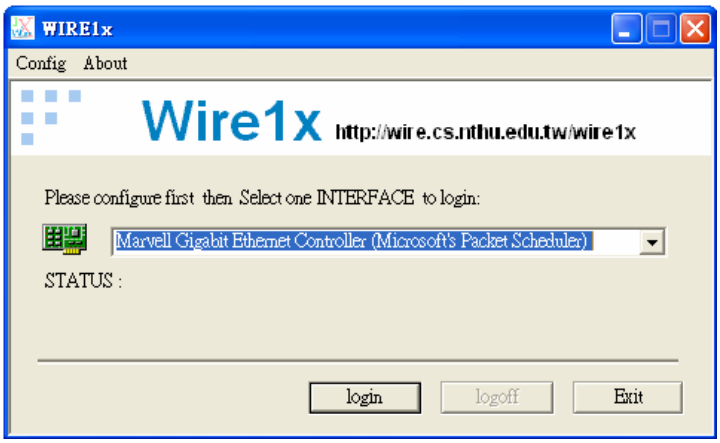

Fig. 7 : Last version of WIRE1x - choose interface

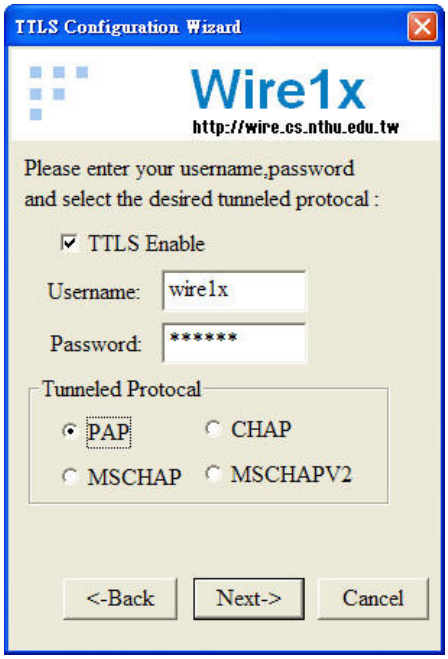

Fig. 8 : Last version of WIRE1x - set EAP method

Fig. 9 : Last version of WIRE1x - status of PAE machine

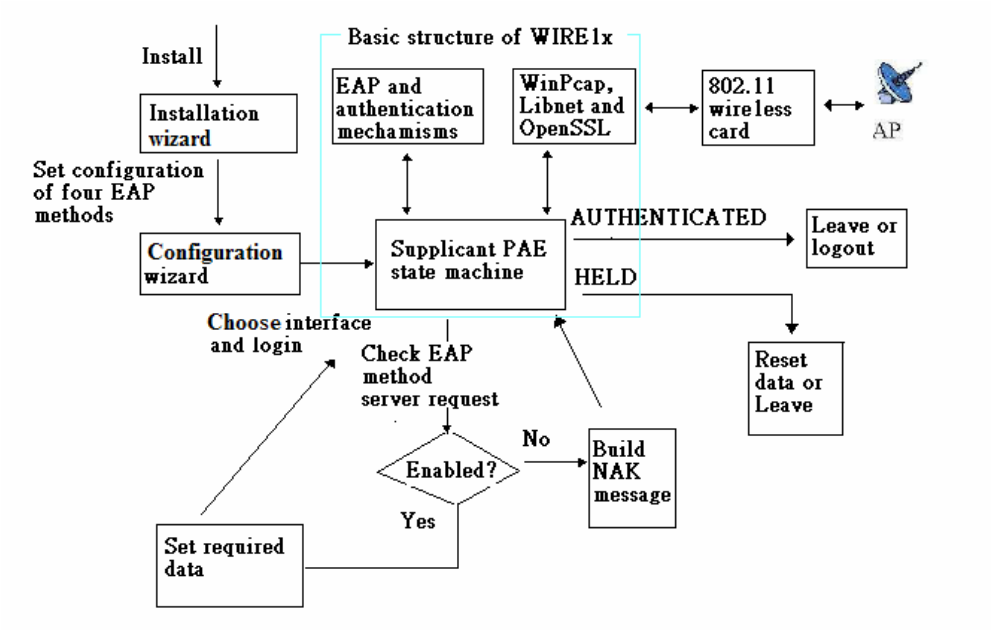

Fig. 10 : Last version of WIRE1x - architecture

But before starting authentication, we will use Wireless Zero Configuration, a connection manager under Windows XP, to connect to one AP first. WIRE1x doesn't have its connection manager. It leads to some problems as follows:

1. If the user's operating environment is not under Windows XP when running WIRE1x, it is essential to find some other software to manage connection with AP. Even though the user is under Windows XP and uses WZC, it is not so convenient for the user to switch two different programs for connection and authentication.
2. If there is more than one AP in the surrounding, it is possible for each AP to use different method for authentication. But when we save the setting of configuration in WIRE1x, the setting we saved doesn't include any information about the AP that we associated with. In other words, if we want to connect to other AP, we should set up the configuration again according to which method of authentication the AP uses.

2.2 The Enhancement Plan of WIRE1x

In order to solve the problems mentioned in the last section, we will create a connection manager and merge it into WIRE1x. Our major targets of the plan are:

1. The connection manager will find out all available APs by scanning within an appropriate range, and shows some information about the APs we find. For example, it will show the SSID, BSSID, signal strength, maximum rates, frequency and the type of authentication and encryption which the AP supports.

2. The connection manager allows the user to select what type of authentication and encryption to use. For example, the user can select open system, shared key, WPA or WPA2 for authentication. Additionally, the user can use 802.1x if necessary. After authentication, the user can decide whether to encrypt the transferring data or not. The user can choose WPA, TKIP or CCMP for encryption.
3. When the connection manager finishes associating with AP, it starts to send EAPOL packets if the user selected 802.1x for authentication. The following progress is the same as what WIRE1x does originally. At last it should show the status of PAE machine on connection manager, like “authenticated” or “held”.

In a word, we have to develop a connection manager based on existed WIRE1x’s architecture. We could change the GUI but couldn’t modify the main architecture too much, or it is hard to integrate with other groups in the future.

3. System Description

3.1 The developing environment

(1) WinPcap

WinPcap is the industry-standard tool for link-layer network access in Windows environments; it allows applications to capture and transmit network packets. So, in our connection manager, we use some functions from this library. We use pcap_findalldev () to list all NIC (network interface card) on the user’s computer. After the user selects NIC, we create a global variable called pcap_descr to handle this NIC. In this way, we can call this to do many operations, like sending the packets or associating to some APs.

(2) Windows DDK

The Microsoft Windows Driver Development Kit (DDK) is a consolidated driver development kit that provides a build environment, tools, driver samples, and documentation to support driver development for the Windows family of operating systems. In order to control most NICs under windows, however, it is impossible to write different codes corresponding to different NIC suppliers. Therefore, we develop our program under Windows DDK.

We can use PacketRequest () --a function provided by WinPcap, and OIDs to set some parameters or query some information with NIC. The IEEE 802.11 WLAN object identifiers (OIDs) are supported by versions 5.1 and later of the Network Driver Interface Specification (NDIS). Miniport drivers for IEEE 802.11

network interface cards (NICs) must support all mandatory WLAN OIDs. For some OIDs, support is recommended but not mandatory.

The WLAN OIDs are listed in the Table.1. In this table, an X in the respective column indicates that the OID supports query (Q), set (S), or indication (I) operations. The table also indicates mandatory (M), recommended (R), or optional (O) support requirements for different operating systems and for WPA and WPA2. For example, when we use OID_802_11_AUTHENTICATION_MODE, we could know the authentication mode which the AP is using if we do query operation or change to other modes if we do set operation.

Name	Q	S	I	Windows 2000/Me	Windows XP and later	WPA	WPA2
OID_802_11_BSSID_	X	X		M	M	M	M
OID_802_11_SSID_	X	X		M	M	M	M
OID_802_11_RSSI_	X		X	O	M	M	M
OID_802_11_RSSI_TRIGGER_	X	X		O	O	O	O
OID_802_11_INFRASTRUCTURE_MODE_	X	X		R	M	M	M
OID_802_11_NUMBER_OF_ANTENNAS_	X			O	O	O	O
OID_802_11_SUPPORTED_RATES_	X			O	M	M	M
OID_802_11_CONFIGURATION_	X	X		O	M	M	M
OID_802_11_DISASSOCIATE_		X		O	M	M	M
OID_802_11_BSSID_LIST_SCAN_		X		R	M	M	M
OID_802_11_BSSID_LIST_	X			R	M	M	M
OID_802_11_PRIVACY_FILTER_	X	X		O	O	O	O
OID_802_11_AUTHENTICATION_MODE_	X	X		R	M	M	M
OID_802_11_ENCRYPTION_STATUS_	X	X		R	M	M	M
OID_802_11_ADD_WEP_		X		M	M	M	M
OID_802_11_REMOVE_WEP_		X		R	M	M	M
OID_802_11_ADD_KEY_		X		O	O	M	M
OID_802_11_REMOVE_KEY_		X		O	O	M	M

Table. 1 : WLAN OIDs list

3.2 System Architecture

We divide our connection manager into five parts:

1. Open the NIC

After the user selects the NIC, in order to check whether the radio of the NIC is turned on or not, we use OID_802_11_SSID to set a random SSID. Obviously, because the SSID we set is randomly picked up, the NIC doesn't really associate with an existed AP. If we couldn't set SSID optionally, it is showed that the NIC's radio is not switched on actually. Maybe it is a wired network card instead of wireless one. However, the previous version of WIRE1x can be used to do authentication on wired networks. So in this version, we allow users to continue following process even though we detect that the radio couldn't be turned on.

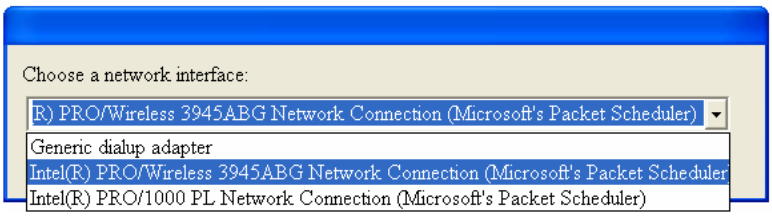

Fig. 11 : The dialogue to select NIC

2. Scan the available networks

After using OID_802_11_BSSID_LIST_SCAN to scan the networks, we could do query operation with OID_802_11_BSSID_LIST to get the data including list of available APs and their information. The data's structure is depicted as Fig.12.

```
typedef struct _NDIS_WLAN_BSSID_EX
{
  ULONG  Length;
  NDIS_802_11_MAC_ADDRESS  MacAddress;
  UCHAR  Reserved[2];
  NDIS_802_11_SSID  Ssid;
  ULONG  Privacy;
  NDIS_802_11_RSSI  Rssi;
  NDIS_802_11_NETWORK_TYPE  NetworkTypeInUse;
  NDIS_802_11_CONFIGURATION  Configuration;
  NDIS_802_11_NETWORK_INFRASTRUCTURE  InfrastructureMode;
  NDIS_802_11_RATES_EX  SupportedRates;
  ULONG  IELength;
  UCHAR  IEs[1];
} NDIS_WLAN_BSSID_EX, *PNDIS_WLAN_BSSID_EX;
```

Fig. 12 : The NDIS_WLAN_BSSID_EX structure

(1) MacAddresss -- AP's MAC address

(2) Ssid -- AP's SSID

(3) Privacy -- a boolean value; it shows that whether the AP is using some method of encryption.

(4) Rssi -- the received signal strength indication (RSSI) in dBm, a relative unit. The power of AP (in watt) can also be conversed from Rssi. [13]

(5) Configuration -- The radio parameter configuration as defined in the NDIS_802_11_CONFIGURATION structure. The frequency of AP can be known in this structure. Because most users are familiar with using channel numbers,

we could use ieee80211_mhz2ieee () -- a function defined under IEEE to do this conversion.

(6) SupportedRates -- the maximum transmission rates can be found in this variable

(7) IEs -- The information elements (IEs) from beacon or probe response messages. The IEs must have the three fixed-size IEs (timestamp, beacon interval, and capability information), and then it is possible to have any variable-length IEs after fixed-size IEs. If there are any variable-length IEs included, we could examine first byte of every IE. If the first byte of variable-length IE is GENERIC_INFO_ELEM, the IE is for WPA authentication; otherwise, if it is RSN_INFO_ELEM, the IE is for WPA2 authentication.

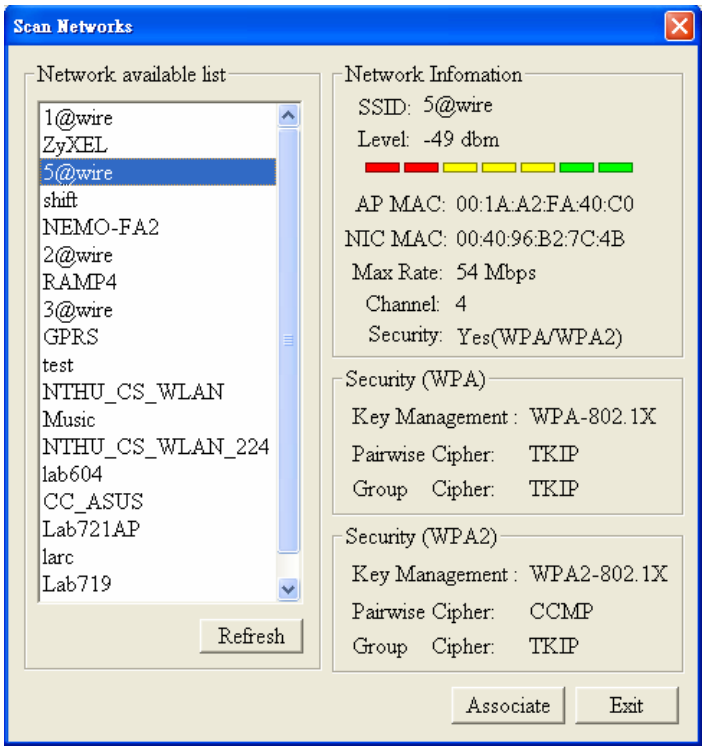

Fig. 13 : The dialogue to scan networks

So, how can we know what method of authentication and encryption does the AP use? If the value of Privacy is false, it is said that this AP doesn't use any method to encrypt data. If the value of Privacy is true, it is said that this AP uses some methods to do authentication and encryption. But how can we distinguish that which method it uses? We can get the answer in IEs. The result is listed in Table.2.

Privacy	IEs	authentication & encryption
false		open system & none
true	doesn't have any variable-length IEs	(open system & WEP) or (shared key & none)

		or (shared key & WEP)
	only WPA IEs	WPA & TKIP
	only RSN IEs	(WPA2 & TKIP) or (WPA2 & CCMP)
	both WPA and RSN	(WPA & TKIP) or (WPA2 & TKIP) or (WPA2 & CCMP)

Table. 2 : Privacy and IEs

3. Set authentication & encryption type

After we get so much information from the scan list, it is time for users to select the AP and set which method of authentication and encryption to use. It is considered that the AP maybe supports other types of authentication backward. For example, if the AP supports WPA, we also allow the user to select open system or shared key to do authentication. Besides two options for users to select which type of authentication and encryption to use, there are two additional options for the user to determine whether to key in password by his own and whether to enable 802.1x as illustrated at Fig.14.

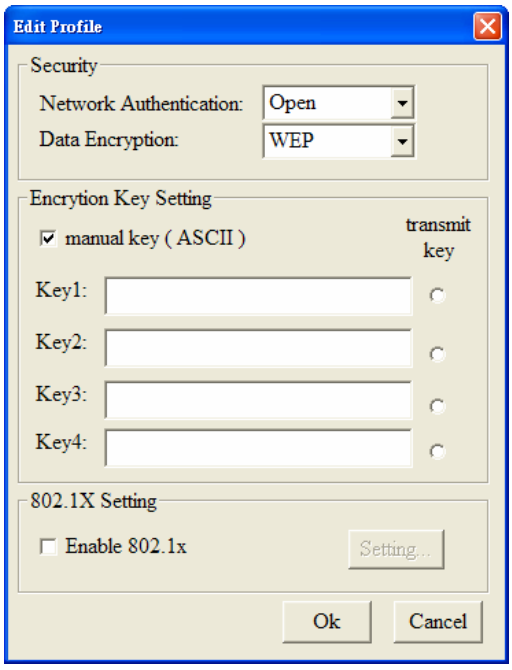

Fig. 14 : The dialogue to set security options

According to the selection of authentication and encryption, there are some rules and limitation for user to choose whether to enable the options of manual key and 802.1x.

authentication	encryption	manual key	802.1x
open system	none	inhibition	could enable it or not

	WEP	enable either of them (If manual key is enabled, static WEP is used. If 802.1x is enabled, dynamic WEP is used.)	
shared key	none	be forced to key in	could enable it or not
	WEP		
WPA	TKIP	not support yet (it is our future work to support PSK)	be forced to enable
WPA2	TKIP		
	CCMP		

Table. 3 : Manual key & 802.1x

When 802.1x is enabled, the user can press the “setting” button to configure it by using the last version of WIRE1x’s GUI as fig.15. In that way, we could integrate our connection manager into WIRE1x.

Fig. 15 : Last version of WIRE1x - 802.1x configuration

4. Start association and EAPOL process

When the user finishes all setting, we start to set some parameters by using OID. We can use OID_802_11_PRIVACY_FILTER, OID_802_11_AUTHENTICATION_MODE and OID_802_11_ENCRYPTION_STATUS to determine which method of authentication and encryption to use. If the user set the manual key before, we could use OID_802_11_ADD_WEP to add WEP key in NIC. In the end, we use OID_802_11_SSID to set SSID name to associate with that AP. After association, if the user wants to enable 802.1x, we will continue to start the EAPOL process as constructed in last version of WIRE1x. If we want to disassociate from the AP, we simply set SSID to a random value.

5. The profile management

For the convenience of using, the user can create several profiles to save the setting according to different APs. In this way, the user doesn’t need to set again next time.

Fig. 16 : The dialogue to create a new profile

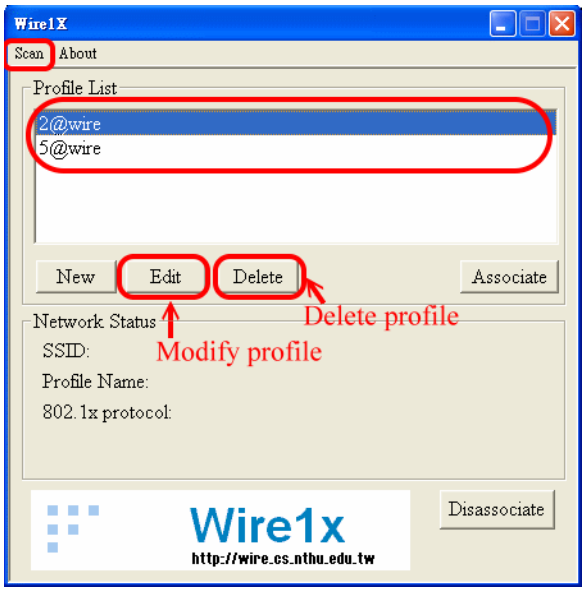

Fig. 17 : The main dialogue

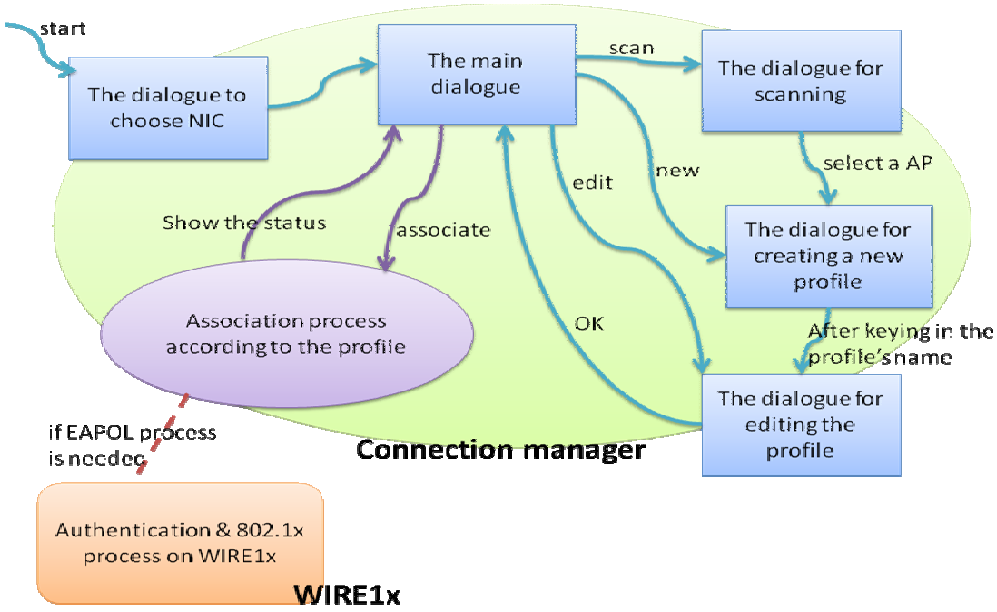

Fig. 18 : The relation between the connection manager and WIRE1x

4. Conclusion and future work

In order to associate with the AP using WPA or WPA2, we have to set some parameters on NIC before we start the process of EAPOL. If we don't do these setting,

the following process of authentication and encryption will not be successful as we expected. However, in last version of WIRE1x without connection manager, it is WZC's work to deal with the problem of association. It leads to the problem that we couldn't control the situation between AP and NIC, and there is no consecution with the following works, including authentication and encryption.

After we finished the connection manager and integrated it into WIRE1x, the most significant improvement is that we could accomplish all works including association, authentication and encryption at one go. It doesn't request users to execute other individual programs anymore. For most users who don't know many details, it becomes easier to configure and manage their wireless networks.

However, we can make our connection manager better and better. First, in this version of connection manager, it can't monitor the status between the AP and the NIC well. We have to depend on the tray icon to detect that the connection with AP is sometimes cut off. Second, there is still room to improve the beautification of GUI. After all, the friendlier GUI makes users feel comfortable when operating WIRE1x.

References

[1] "IEEE 802.11" http://grouper.ieee.org/groups/802/11/

[2] IEEE Standard 802.11-1999, "Part 8.2:The Wired Equivalent Privacy (WEP) algorithm," Sep. 1999

[3] "Weaknesses in the Key Scheduling Algorithm of RC4." Scott R. Fluhrer, Itsik Mantin and Adi Shamir. Selected Areas in Cryptography 2001, pp1 – 24

[4] "Extensible Authentication Protocol (EAP)" RFC 3748 (June 2004)

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

[6] B. Aboba, D. Simon, "PPP EAP TLS Authentication Protocol," IETF RFC 2716, Oct. 1999.

[7] 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.

[8] 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.

[9] "Generic Token Card (GTC)" RFC3748, Section 5.6

[10] "Wi-Fi Alliance's WPA page", http://www.wi-fi.org/knowledge_center/wpa/

[11] "Open1x" http://open1x.sourceforge.net/

[12] "WinPcap" http://www.winpcap.org/

[13] "P(dBm)=10*log[P(mW)]"

http://www.rfcafe.com/references/electrical/mw2dbm.htm