Implementation of WIRE1x on Windows Mobile System

Chia-Hao Liang, Yu-Cheng Lin, Tsung-Hsun Chang, Ming-Hung Yang, Cheng-Hung Hsieh, Yi-Ken Ho, and Jyh-Cheng Chen
Wireless Internet Research and Engineering Laboratory
National Tsing Hua University
Hsinchu, Taiwan

December 22, 2008

Abstract

The topic of this report is to present the design and implementation of WIRE1x on Windows Mobile system. Basically, this report contains two parts: a new GUI (Graphic User Interface) and the adaptation codes of previous version 2.4 for Windows Mobile 6.0. WIRE1x is an open-source software, and a client software for authentication based on IEEE 802.1x client, developed by the Wireless Internet Research & Engineering (WIRE) Laboratory, National Tsing-Hua University. The IEEE 802.1x standard defines a port-based network access control to authenticate and authorize devices interconnected through various IEEE 802 LANs. IEEE 802.11i also incorporates 802.1x as its authentication solution for 802.11 wireless LANs. In this project, we transplant WIRE1x, which is formerly only adaptive to Windows, into a mobile edition for Windows Mobile system, so that users can safely send messages and surf the Internet through WIRE1x while using hand-held devices.

1. Introduction

WIRE1x is a client software for wireless security, depending on authentication protocols of IEEE 802.11[1]. Compared with other client software which is based on Application Layer, WIRE1x is implemented according to Data Link Layer, and it can therefore offer more security and stability. Besides, WIRE1x provides almost all types of authentication protocols for users to suit alternative environments of APs [2].

WIRE1x is implemented formerly only for PCs and only adaptive to Windows XP and Windows Vista. However, with the development of communication technologies, PDAs are nowadays much more widely used corresponding to well–settled wireless environment, presenting an unprecedented era of wireless communications and convenience. People can now purchase things or read E-mails over the internet almost everywhere. At the same time, the convenience of wireless environment also exposes us to some dangers, such as eavesdropping and various internet attacks, which are nevertheless more inevitable and more difficult to defend. On this condition, security issues can no longer be omitted and should on the contrary be strongly emphasized. We hence decide to adapt PC-based WIRE1x for mobile devices which use Windows Mobile 6.0, the latest version of Microsoft's Window Mobile, which can be considered as evolved from former Window CE versions, as their OS to offer a more dependable wireless environment.

Both security and convenience are our goals. We believe that only when two goals are achieved can users enjoy wireless communication, and that our WIRE1x's mobile version will take good care of both.

2. Background

2.1 IEEE 802.11i and IEEE 802.1x

IEEE 802.11i is an amendment to the original IEEE 802.11 standard specifying security mechanisms for wireless networks [3]. IEEE 802.1x is utilized to implement the part of authentication.

There are several main features about IEEE 802.1x:

1. It provides port-based authentication.
2. It provides an authentication mechanism to devices tending to access a LAN port, establishing a point-to-point connection or preventing access from that port if authentication fails.
3. It is used for most wireless 802.11 access points and is based on the Extensible Authentication Protocol (EAP).

IEEE 802.1x contains three important elements: supplicant, authenticator, and authentication server. The supplicant is software on a client device. Usually, the authenticator is an access point in WLAN, and an authentication server is generally a RADIUS server.

Before authenticating, AP blocks all packets expect 802.1x messages. At this time, Extension Authentication Protocol (EAP) will send authentication data within 802.1x packets to an authentication server. The authentication server then certificates the frames and replies accepting or rejecting packets. If certification successes, the controlled port closed and the user can therefore utilize services offered by authentication's system. If certification fails, the user can decide to authenticate again or not.

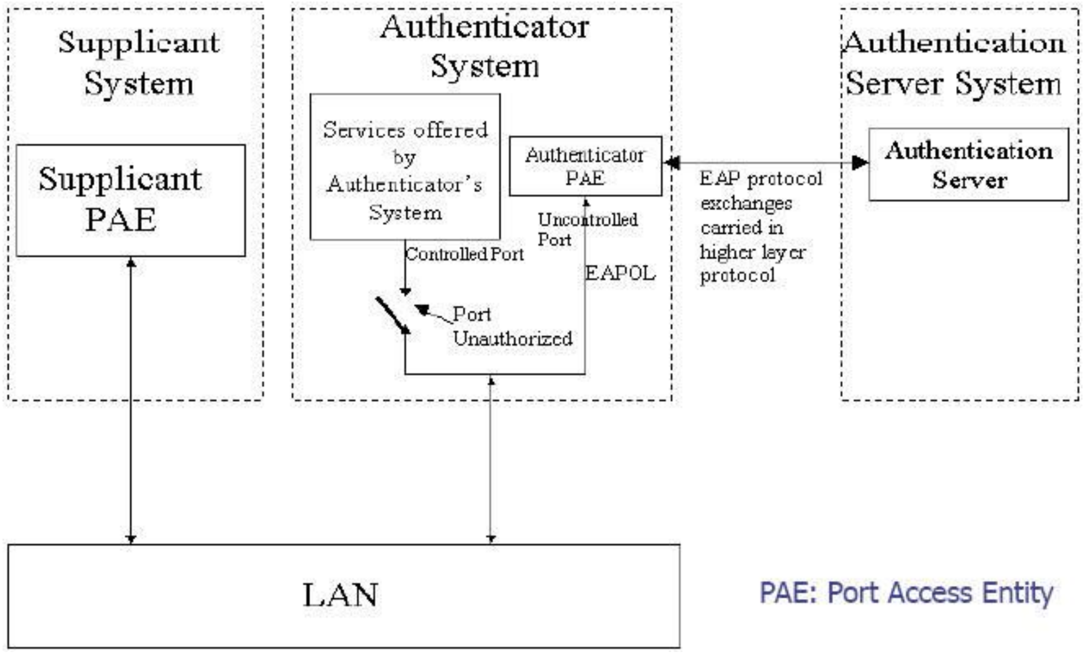


Figure 2.1 Main structure of IEEE 802.1x [4]

WIRE1x supports several EAP methods. However, at present on Windows Mobile system, not all kinds of EAP methods implemented for Windows are translated. Table 2.1 shows the difference between these two versions.

Name	Supported OS	Supported EAP authentication methods	Supported Encryptions
WIRE1x	MS Windows 2000/XP	SIM, AKA, MD5, TLS, TTLS, PEAP, FAST, MS-CHAPv2	WEP, WPA, WPA2
WIRE1x Mobile Version	MS Windows mobile 6.0	MD5, TLS, TTLS, PEAP, MS-CHAPv2	WEP, WPA, WPA2

Table 2.1 EAP methods in different platforms [5]

Here we briefly introduce the five methods supported on Windows Mobile system. EAP-MD5 is implemented by one-way hash function. It utilizes only usernames and passwords to authenticate, so it provides less security and certainly is more vulnerable than other EAP methods. EAP-TLS provides mutual authentication and two end-points can communicate within encrypted TLS tunnel. It also supports Wired Equivalent Privacy (WEP) or Wi-Fi Protected Access (WPA and WPA2); all of them are more reliable. EAP-TTLS first establishes a TLS tunnel and then uses the tunnel to exchange information such as authentication key and accounting information. The tunnel encryptions of EAP-TTLS provide non-EAP protocols: PPP Authentication Protocol (PAP), Microsoft PPP CHAP Extensions, and its second version (MS-CHAPV2). According to TTLS, Mutual authentication is supported, while client-to-server authentication is also applicable. EAP-PEAP is very similar to EAP-TTLS. However, because PEAP is developed by Microsoft, EAP-MSCHAP-v2 is the only choice in second phase.

2.2 Microsoft Windows Mobile 6

Windows Mobile is an operation system combined with a suite of basic applications for mobile devices based on Microsoft Win32 API [6]. It preserves many features from desktop versions of Windows. Third-party software for Windows Mobile can therefore be developed. Windows Mobile has been updated several times. The latest version is 6.1.WIRE1x Mobile Version is developed on Windows Mobile 6.0.

3. Implementation of WIRE1x Mobile

3.1 Graphic User Interface

In the previous version, WIRE1x's GUI is designed for PCs and laptops. Nevertheless, on PDAs, the previous version of GUI cannot work correctly, so a suitable type of GUI is needed. Besides, to make configurations easier for users, we strive to simplify the whole process. Even if users are not familiar with our former version, he or she can easily understand how to set up and connect to the internet.

Furthermore, a new logo is designed for WIRE1x on Windows Mobile system. At present, there are two other logos signifying WIRE1x. But in the future, only one most representative logo will be chosen and reserved.

Figure 3.1 WIRE1x PDA mark

Figure 3.2 WIRE1x Windows XP mark

Figure 3.3 WIRE1x Windows Vista mark

3.2 System Architecture

In this section we will introduce the interaction of GUI and EAP methods' authentication processes, as well as the respective state machine. The following will explain the flow of this program, emphasize the implementation of the whole internal flow, and illustrate what a respective class does in detail. Moreover, profile and registry will be disclosed in section 3.3. The instruction of settings, however, will be explained in section 5. Functions of network interface cards, sending and receiving packets will be introduced in section 4.

■ Starting the Program

Clicking WIRE1x icon will launch the program. Before entering the main dialog, WIRE1x will automatically close the WZC (Wireless Zero Configuration) service. More details of WZC will be mentioned in the next chapter.

■ Main Dialog (MAIN_DLG.cpp)

1. **Selecting Network Interface Cards.** As "Change Network Interface" button is clicked, the window of Network Interface Selection will be displayed. In this part, the program will fetch all real and virtual network interfaces in a PDA. However, there is usually one network card in such a mobile device. Therefore, this step is ignorable. The important function "wcstok()" is used to separate a string.
2. **Scanning AP.** The program will start scanning available APs after users click "Scan" button and will save all the information in a structure which will be explained later.
3. **Disconnection.** The program will return to the initial state if "Disconnect" button is clicked. At first, it will scan and check if the variable "running" is still on. Secondly, it will check if the program is receiving frame. If yes, close this thread and the main thread. However, if authentication is completed successfully, the

connection status will be set to "logoff".

4. **Showing status.** Here it will demonstrate profile name, Service Set Identifier (SSID) and the current connecting status.

■ AP Selection Dialog (SCAN_DLG.cpp)

1. **Showing AP information.** The function "pcap_GetApInfo()" is used to get the number of APs and other information. When the user selects a certain AP, the column in the middle will show AP information such as SSID, Signal Quality, Max Rate and its MAC, and it will draw out the information of the following variables in the structure "Ap_info": SSID, RSSI, maxrate, and BSSID[n]. Additionally, we use variables (parsed_wpa_ie, parsed_rsn_ie, parsed_wpa_ie_set, and parsed_rsn_ie_set) also in the structure "Ap_info" to identify AP's authentication and encryption mode. The program will rescan if users click "refresh" button.
2. **Selecting EAP methods.** While clicking "Setting" button, it will enter "EAP selection" dialog. If the configuration of a certain AP has been completed formerly, the saved settings will be loaded into the program. If not, default settings will be loaded instead.
3. **Starting Authentication.** After all settings are completed, "Connect" will become clickable. When it starts trying to associate with a certain AP, the program will return to main dialog and show the current status.

   The process above is implemented by following steps:

   a. Call "pcap_ConnectToAPName(Ap_info[i].SSID)" to connect to an AP.
   b. Initialize all settings and read data from profile.
   c. Check the function "get_<EAP method>_enable()" and set username and password of the selected EAP method.
   d. Call "init_eapol()" in EAPOL.cpp and enter initial state.
   e. Return to main dialog.

■ Edit Profile Dialog (EDIT_PROFILE_DLG.cpp)

1. **Initializing settings.** If a user has selected one AP and completed its correspondent settings before, these settings can be found in Windows Registry and at this time can be loaded into the program. Otherwise, default settings are loaded. The user can select the checkbox called "802.1x" to do more settings.
2. **Authentication and encryption settings.** The followings are several encryption modes which can be selected: OPEN-NONE、OPEN-WEP、SHARED-NONE、SHARED-WEP、WPA-TKIP、WPA-PSK、WPA2-CCMP、WPA2-PSK. Nonetheless,

only available modes which the selected AP supports will be shown. (Table 3.1)

Encryption AP supported	Authentication available	Encryption available
WEP、WPA、WPA2	WPA2	CCMP
WEP、WPA	WPA	TKIP
WEP	Open	WEP

Table 3.1 Default setting of Authentication and Encryption

3. **EAP method settings.** As users click "Setting" button, it will turn to EAP method selection dialog.
4. **Writing into registry.** The program will write all information into registry after the users click "Save" button.

■ EAP Methods Selection Dialog (EAP_SELECT_DLG.cpp)

1. **Selecting Authentication Method.** Once again, if a user has completed settings correspond to the chosen AP formerly, those settings will be loaded into the program. If not, a user will have to do those configurations at first. Each method has its own variables to record information, such as <method name>_select, <method name>_set. Importantly, it is allowed to choose more than one EAP methods at a time. Once any blank or empty string in any column is detected, the program will pop out a message box to request the user to complete omitted settings. When everything is ready, "OK" button is clickable and all information will be written into the profile.
2. **Selecting verify server certificate.** All available certificates can be chosen and the chosen one will be saved in registry.

■ EAP Methods Setting Dialog (CONFIG_"METHOD"_DLG.cpp)

1. **Entering username and password.** EAP methods such as EAP-MD5, EAP-TTLS, EAP-PEAP, and MSCHAPV2 need username and password, while EAP-TLS requires supplicant certificate to authenticate. The inputs will be saved in some temporal variables and written into a correspondent profile in the dialog "edit_profile".
2. **Selecting inner-tunnel encryption method.** In EAP methods such as EAP-TTLS, EAP-PEAP, and MSCHAPV2, inner-tunnel encryption methods can be selected. The program will automatically save the information into a correspondent profile

as a string.

■ Exit or Minimize Program

There will be a message box popping out after users click “OK” button on the right upper corner. If “Yes” is chosen, the program will call a function named “pcap_SetWzc(true)” to restart WZC service, and then WIRE1x will be terminated. If “No” is clicked, “ShowWindow(SW_MINIMIZE)” will be called and WIRE1x will be just minimized.

Figure 3.4 is used to depict the trigger relationship between GUI and respective windows.

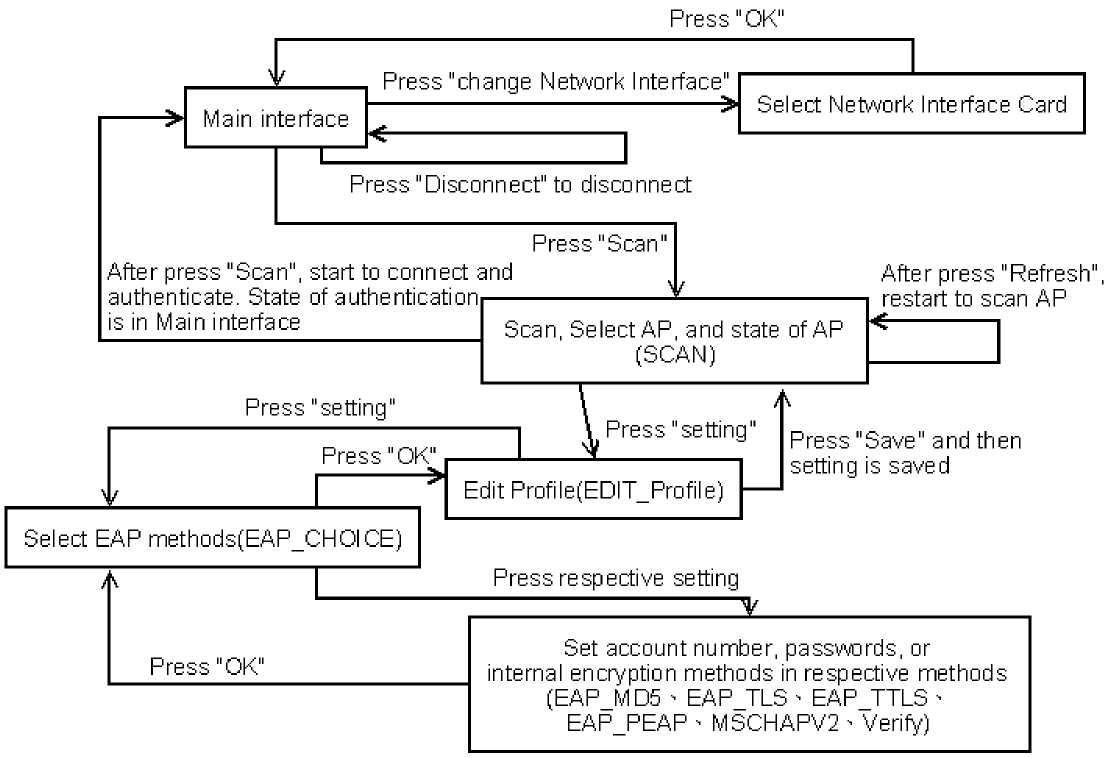

Press "OK"
Press "change Network Interface"
Main interface
Select Network Interface Card
Press "Disconnect" to disconnect
Press "Scan"
After press "Scan", start to connect and authenticate. State of authentication is in Main interface
After press "Refresh", restart to scan AP
Scan, Select AP, and state of AP (SCAN)
Press "setting"
Press "setting"
Press "Save" and then setting is saved
Press "OK"
Edit Profile(EDIT_Profile)
Select EAP methods(EAP_CHOICE)
Press respective setting
Press "OK"
Set account number, passwords, or internal encryption methods in respective methods (EAP_MD5、EAP_TLS、EAP_TTLS、EAP_PEAP、MSCHAPV2、Verify)

Figure 3.4 Graphic user interface and flowchart of internal communication

3.3 Wide Characters and Characters

Basically, it is “char”, a data type of C language, used to store and handle characters and strings. However, when it comes to Chinese words, a “char” is no longer enough because a Chinese word is usually two-byte, indicating that there is one byte more than an English character. Hence, a new data type “wide character (wchar_t)” has been developed to solve this problem. “Wchar_t” is designed as a two- or four-byte data type (always two-byte in Windows Mobile and WIRE1x), but it is not backward compatible. To translate between “char” and “wchar_t”, new functions are consequently needed.

Although Chinese words are not concerned in WIRE1x, the translation between “char” and “wchar_t” is still a crucial mission to deal with. While every character appearing on the screen must be a wide character in Windows Mobile’s GUI, inner functions in WIRE1x such as EAP methods, WEP, WPA, and WPA2, use normal characters to exchange information. To correctly communicate with GUI, we have to use the following two functions to translate between them:

- int MultiByteToWideChar(CP_UTF8,0,Source,-1,Target,Size)
- int WideCharToMultiByte(CP_UTF8,0,Source,-1,Target,Size,NULL,NULL)

Besides, due to more necessary operations of wide characters, such as copying, comparison and so on, many other functions are therefore imported. Table 3-2 shows those functions’ name and the respective functions of normal characters.

Type	Comparison	Concatenation	Copying	Split	Length
Wide Char	Wcscmp	wcscat	wcscpy	wcstok	wcslen
Char	Strcmp	strcat	strcpy	strok	strlen

Table 3.2 Wide Character’s functions and their respective functions of normal characters

3.3 Profile and Registry

For user convenience, profiles and Windows Registry are utilized to save user configurations so that users, after one-time setting up, can skip most details when WIRE1x is launched again. Users will have to reconfigure only when a new AP station is chosen, in that each profile is mapped into an identical AP respectively. Basically, a value saved in the registry works the same as a profile does, but the two play different roles: a profile acts as a temporary file to contain user configurations when WIRE1x is running; a value saved in the registry holds configurations permanently once “Save” button is clicked. (That is, unless user manually deletes these data in the registry, they are perpetually saved in the system.) Furthermore, the following shows the interaction between profiles and the registry.

1. Launch WIRE1x. 2. Select AP. 3. Scan whether there is the registry value related to the chosen AP. 4. If yes, extract the value and save it in the profile; If no,

initialize a profile. 5. After clicking “Save” button, save the profile data back in the registry.

The structure of profile data will be introduced as follows:

WCHAR ssid[32]	AP’s identification
WCHAR psk[100]	not used now
uint32_t ssid_len	length of ssid
uint32_t type	According to the combination of authentication and encryption
uint32_t capability	to indicate the capability of CCMP or TKIP
uint32_t psk_len	not used now
uint32_t check_1x;	to indicate whether the checkbox of 802.1x is checked
uint32_t check_dhcp;	not used now

Table 3.3 Structure of profile data

Additionally, to memorize user identifications, passwords, certificates, and tunneled encryptions, the other sort of registry values respective to AP stations is used as well.

4. Key Functions in EAP over LANs

4.1 Introduction to WinPcap and NDIS

WinPcap is the industry-standard tool for link-layer network access in Windows environments: it allows applications to capture and transmit network packets bypassing the protocol stack, and has additional useful features, including kernel-level packet filtering, a network statistics engine and support for remote packet capture.

WinPcap consists of a driver, which extends the operating system to provide low-level network access, and a library that is used to easily access the low-level network layers. This library also contains the Windows version of the well known libpcap Unix API.

Thanks to its set of features, WinPcap is the packet capture and filtering engine of many open source and commercial network tools, including protocol analyzers, network monitors, network intrusion detection systems, sniffers, traffic generators and network testers. Some of these tools, like Wireshark, Nmap, Snort, ntop are known and used throughout the networking community.

Winpcap.org is also the home of WinDump, the Windows version of the popular tcpdump tool. WinDump can be used to watch, diagnose and save to disk network traffic according to various complex rules [7].

NDIS is short for the "Network Driver Interface Specification". The primary purpose of NDIS is to define a standard API for "Network Interface Cards" (NIC's). The details of a NIC's hardware implementation is wrapped by a "Media Access Controller" (MAC) device driver in such a way that all NIC's for the same media (e.g., Ethernet) can be accessed using a common programming interface.

NDIS also provides a library of functions (sometimes called a "wrapper") that can be used by MAC drivers as well as higher level protocol drivers (such as TCP/IP). The wrapper functions serve to make development of both MAC and protocol drivers easier as well as to hide (to some extent) platform dependencies.

Early versions of NDIS were jointly developed by Microsoft and the 3Com Corporation. Current NDIS versions used by Windows for Workgroups (WFW), Windows 9X Windows NT, Windows 2000, Windows XP and Windows Server 2003 are Microsoft proprietary specifications [8].

4.2 Explanation of Key Functions

According to the fact that the released version of WinPcap must be run in the platforms of Windows NT4/2000/XP/2003/Vista/Server 2008, the version of Windows Mobile system is now under experiment. Therefore, we decide to rewrite the key functions. NDIS library is used in WinPcap and the library is encapsulated to the DLL (Dynamic Link Library) for platforms respectively. Thus, we take the library provided by NDIS to implement controlling NIC’s, transmitting packets, or connecting to AP, disconnecting to AP and so on. And then we illustrate the use of the key functions. The parts of illustration are referred to the other report [9], [10].

■ Initialize when WIRE1x Starts

1. pcap_OpenDriver(). Open a driver of NDIS user-mode I/O to request, set and inquire after.
2. pcap_OpenDevice(). (1)Allows an application to have NDIS unbound and then bind an adapter to one or all NDIS protocol drivers. (2)Associates the handling of a

file obtained through a “CreateFile” call to a network adapter. (3)Specifies the types of net packets for which a protocol receives indications from a NIC driver. (4)Set Ethernet Type (0x888e) for eapol packets.

3. pcap_GetAdapters(). Get names of clients’ network adapters. It is implemented by the concept of “List”.

■ Scan AP Information Available

1. pcap_GetApInfo(). Get AP information like name, signal quality, transmission rate, MAC and security mode supported and so on.
2. pcap_RefreshBSSIDs(). The miniport driver directs the 802.11 NIC’s to request a survey of BSSs.
3. pcap_GetBSSIDs(). It is called in pcap_GetApInfo() but after pcap_RefreshBSSIDs(). It will return all information about AP including respective special properties.

■ Pre-setting before Connect to AP

1. pcap_PrepareToConnect(). Four steps: (1)Set the infrastructure mode. (2)Set privacy filter to decide whether receives all packets or the encrypted packets only or not. (3)Set the authentication mode like OPEN, SHARED, WPA, WPA2…and so on. (4)Set the encryption status like WEP, TKIP, CCMP, PSK…and so on.

■ Send and Receive Packets

1. send_frame(). WriteFile() is going to be called to send packets to AP.
2. pcap_ReadFile(), get_frame(). Use pcap_ReadFile() to receive packets continuously and save them in a circular queue. Certain number of packets is fixed. Get_frame() is used to get the packets in the circular queue.

■ Other Functions

1. pcap_setOid(). Allows an application to set an object identifier (OID) of a miniport driver.
2. pcap_getOid(). Allows an application to query an object identifier (OID) of a miniport driver.
3. pcapHackRegForIE(). Revising values in registry is in order to explore IE correctly.
4. pcap_SetWzc(). Open or close WZC service. After exiting the program, it will return to be open status.

4.3 About WZC

Wireless Zero Configuration (WZC) is the system service provided by Microsoft. It is a management interface of wireless network which provides users for setting information about networks. Unfortunately, thanks to the built-in service, it makes WPA and WPA2 work incorrectly. Moreover, it leads to inconvenience for users that lots of information in WZC must be set while authenticating. Therefore, WZC must be closed. About the control of network interface cards and any other setting are going to be managed by our program. It makes user feel convenient and works correctly on each authentication method.

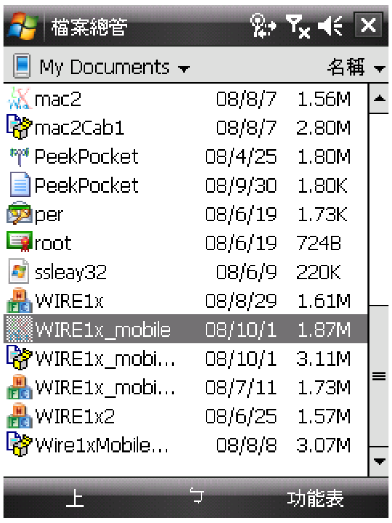
5. How to Use WIRE1x Mobile

5.1 Flow of Operation

This section will demonstrate how to use WIRE1x Mobile step by step.

■ Step 1

After installing WIRE1x, you can find WIRE1x listed in Explorer. Click it.

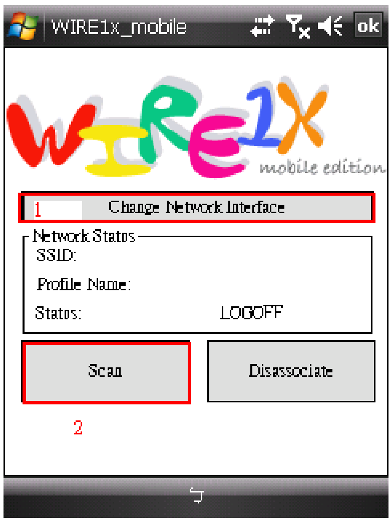


```
mac2              08/8/7   1.56M
mac2Cab1          08/8/7   2.80M
PeekPocket        08/4/25  1.80M
PeekPocket        08/9/30  1.80K
per               08/6/19  1.73K
root              08/6/19  724B
ssleay32          08/6/9   220K
WIRE1x            08/8/29  1.61M
WIRE1x_mobile     08/10/1  1.87M
WIRE1x_mobi...    08/10/1  3.11M
WIRE1x_mobi...    08/7/11  1.73M
WIRE1x2           08/6/25  1.57M
Wire1xMobile...   08/8/8   3.07M
```

Figure 5.1 Step1

■ Step 2

1. Network Interface is selectable. (But usually, there is only one interface.)
2. Click “Scan” button.

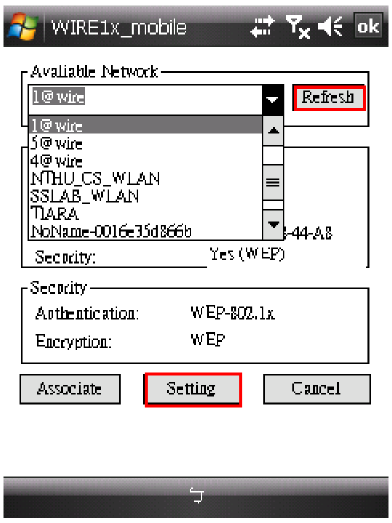


SSID:
Profile Name:
Status: LOGOFF

Figure 5.2 Step2

■ Step 3

1. Click “Refresh” button and then you will see one or more available networks shown on the upper left.
2. Select a network and click “Setting”.

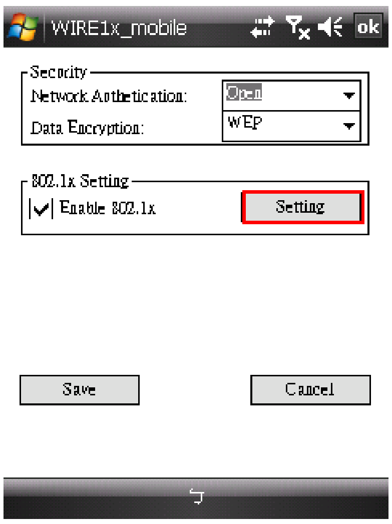


Security: Yes (WEP)
Authentication: WEP-802.1x
Encryption: WEP

Figure 5.3 Step3

■ Step 4

1. Select a suitable authentication method and an encryption method.
2. Enable 802.1x and click “Setting”.


WIRE1x_mobile

Security
Network Authentication: Open
Data Encryption: WEP

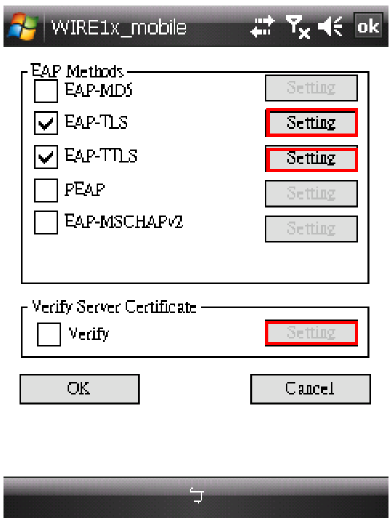
802.1x Setting
[✓] Enable 802.1x — Setting

Save — Cancel

Figure 5.4 Step4

■ Step 5

1. Select one or more suitable EAP methods. Click “setting” button for each method to set username and password.


WIRE1x_mobile

EAP Methods
[ ] EAP-MD5 — Setting
[✓] EAP-TLS — Setting
[✓] EAP-TTLS — Setting
[ ] PEAP — Setting
[ ] EAP-MSCHAPv2 — Setting

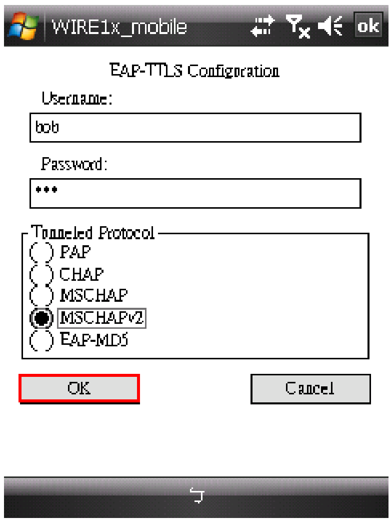
Verify Server Certificate
[ ] Verify — Setting

OK — Cancel

Figure 5.5 Step5

■ Step 6

1. (TTLS for example) set username and password and then select a tunneled protocol.
2. Click “OK” button.


WIRE1x_mobile

EAP-TTLS Configuration

Username: bob

Password: •••

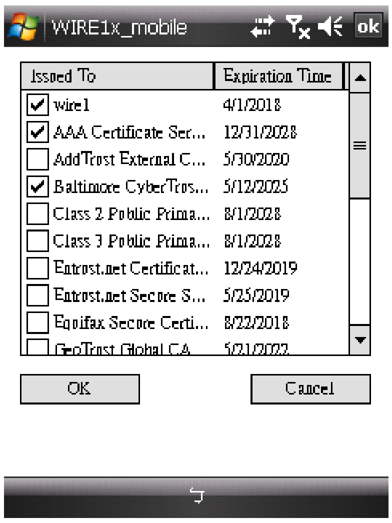
Tunneled Protocol

( ) PAP
( ) CHAP
( ) MSCHAP
(●) MSCHAPv2
( ) EAP-MD5

Figure 5.6 Step6

■ Step 6 (Optional)

1. If “verify” button is checked, select one or more certificates.

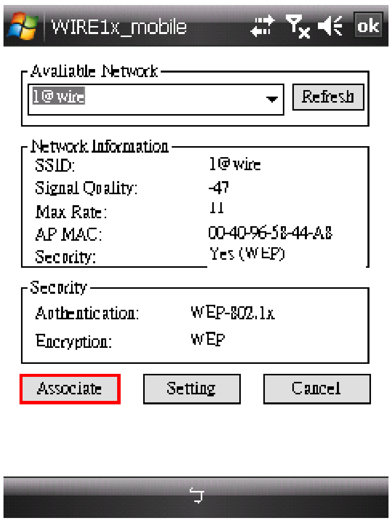

WIRE1x_mobile

	Issued To	Expiration Time
✓	wire1	4/1/2018
✓	AAA Certificate Ser...	12/31/2028
	AddTrust External C...	5/30/2020
✓	Baltimore CyberTrus...	5/12/2025
	Class 2 Public Prima...	8/1/2028
	Class 3 Public Prima...	8/1/2028
	Entrust.net Certificat...	12/24/2019
	Entrust.net Secure S...	5/25/2019
	Equifax Secure Certi...	8/22/2018
	GeoTrust Global CA	5/21/2022

Figure 5.7 Step6 (Optional)

■ Step 7

1. Just click several “OK” or “save” buttons to return to the window as follow.
2. Click “Connect”.



1@wire

Network Information

SSID:	1@wire
Signal Quality:	-47
Max Rate:	11
AP MAC:	00-40-96-58-44-A8
Security:	Yes (WEP)

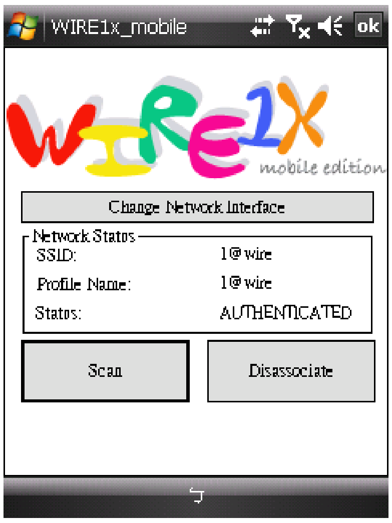
Security

Authentication:	WEP-802.1x
Encryption:	WEP

Figure 5.8 Step7

■ Step 8

1. Finally WIRE1x will display information about the connection.



Network Status

SSID:	1@wire
Profile Name:	1@wire
Status:	AUTHENTICATED

Figure 5.9 Step8

■ Step 9

1. Click “Disconnect” if you want to terminate the connection.
2. Click “Yes” if you want to exit; otherwise, click “NO”.

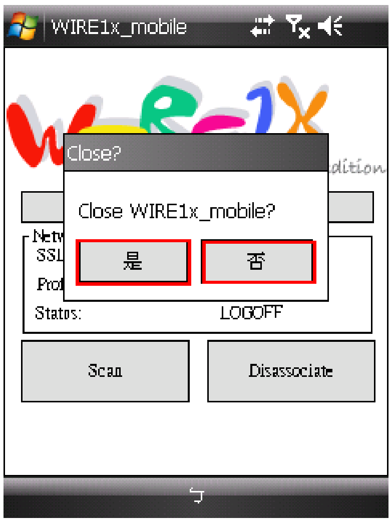


Figure 5.10 Step9

6. Conclusion

Brought out with totally new GUI, easier and more convenient configuration, WIRE1x on Windows Mobile system offers PDA users an instinctive way to take good use of wireless communication.

Since wireless technologies are an unavoidable trend, security issues are nowadays the place to attach importance on. The convenience of wireless components, at worst, may otherwise become an unintentional betrayer to expose crucial user information such as credit card numbers to internet criminals. Thus, besides convenient experience, secure wireless environment requested eagerly today is one of our major goals which we try to present.

The PC versions of WIRE1x have been downloaded accumulatively more than hundred thousands of times from our website. Mobile version is already released as well, and it is totally free and open source. Please visit our website (http://wire.cs.nthu.edu.tw/wire1x/). We expect any suggestion and modification to improve our work.

Reference

[1] ANSI/IEEE Std 802.11, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” 1999.

[2] “Wire1x,” http://wire.cs.nthu.edu.tw/wire1x/

[3] IEEE Std 802.11i/D3.0, Wireless LAN Medium Access (MAC) and physical layer (PHY) specifications: Specification for Enhanced Security, November

2002.

[4] Y.-P. Wang, Y.-W. Liu, and J.-C. Chen, "Design and implementation of WIRE1x ," In Proc. of Taiwan Area Network Conference (TANET '03), (Taipei, Taiwan), October 2003.

[5] Y.-P. Wang, J.-C. Chen, and Y.-W. Liu, "Design and implementation of WIRE1x ," Technical Report: NTHUCCRC-2005-101, Computer and Communication Research Center (CCRC), National Tsing Hua University, Nov. 2004.

[6] “Windows Mobile,” http://www.microsoft.com/taiwan/windowsmobile/default.aspx

[7] “WinPcap,” http://www.winpcap.org/default.htm

[8] “NDIS,” http://www.ndis.com/

[9] J.-C. Chen and Y.-P. Wang, “Extensible Authentication Protocol (EAP) and IEEE 802.1x: Tutorial and Empirical Experience, *IEEE communication Magazine*, vol. 43, no. 12, pp. S26-S32, Dec. 2005.

[10] Y-C Liu, Y-P Zhang, S-J Chen, F-W Chen, T-H Lu, Y-K Ho, and J-C Chen, "Enhancement of WIRE1x Network Adapter Control Module," Technical Report, Wireless Internet Research & Engineering (WIRE) Lab, National Tsing Hua University, Dec. 2008