Enhancements in WIRE1x version 2.0

Yi-An Chen, Chien-Han Chen, Chien-Chia Chen, Jui-Yi Chen, and Jyh-Cheng Chen
Wireless Internet Research and Engineering Laboratory
National Tsing Hua University
Hsinchu, Taiwan

January 2, 2007

Abstract

WIRE1x is an open source implementation of IEEE 802.1x client (supplicant) developed by Wireless Internet Research and Engineering (WIRE) Laboratory, National Tsing Hua University. The version 0.1 was released on June 18, 2003. On June 29, 2004, we released the first stable version of WIRE1x, which supported Windows XP, 2000, ME, and 98. It also supported EAP-MD5, EAP-TLS, EAP-TTLS, and EAP-PEAP. WIRE1x has received more and more attention in recent years. Almost in every IEEE 802.1x supplicant evaluation literature, WIRE1x is included. We have further enhanced WIRE1x and released the second version on December 11, 2006. This document briefly summarizes the enhancements in WIRE1x version 2.0.

Keywords: Authentication, Extensible Authentication Protocol (EAP), IEEE 802.1x, IEEE 802.11i, Wireless LAN, Security

CONTENTS

List of Figures and Tables

1. Introduction

As the infrastructure of Wireless Local Area Networks (WLAN) are widely constructed, WLAN become much more popular in the last few years. WLAN breaks the restriction of conventional wired communications and provides a convenient way to access network at anywhere and any time. Among many related standards, the IEEE 802.1lb [1] is the most widely-used one currently; however, it has already been proved that the security considerations of IEEE 802.11b are not strong enough in many situations. In order to enhance standard IEEE 802.1lb, the IEEE 802.1li [2] has been released in July 2004.

IEEE 802.11i defines three main security mechanisms, which are key management, encryption, and authentication improvement. In order to manage the key distribution automatically, IEEE 802.11i develops algorithms and protocols based on IEEE 802.11b. It introduces two ways to distribute keys, manual key management and automatic key management. The former requires users to set a pre-shared key manually; the latter incorporates with IEEE 802.1x [3] to support key management services. IEEE 802.11i also develops two advanced cryptographic algorithm to enhance confidentiality. In the mechanism of authentication enhancement, IEEE 802.11i work with IEEE 802.1x, which is a standard defines a port-based network access control, to provide various Extensible Authentication Protocol (EAP) methods.

While end users want to access network supporting IEEE 802.1x, their devices should be IEEE 802.1x-capable first. In order to be authenticated by using the IEEE 802.1x, supplicant (client) side has to install related client-side software unless the IEEE 802.1x functionality has been built in the operating system (OS). WIRE1x is developed for this purpose, and is an open source implementation of IEEE 802.1x client (supplicant) developed by the Wireless Internet Research and Engineering (WIRE) Laboratory [4]. WIRE1x is developed based on OPEN1x [5], which is also an IEEE 802.1x open-source application designed for client-side's authentication ran on UNIX-liked operating systems. Unlike OPEN1x, WIRE1x is specially designed for Microsoft Windows. Both the source code and the executable files of WIRE1x are freely available on our website: http://wire.cs.nthu.edu.tw/wire1x/.

There have been several versions of WIRE1x released up to this year. The latest version released in March, 2006 has already supported four Extensible Authentication Protocol (EAP) methods, EAP Message Digest 5 (EAP-MD5) [6], EAP Transport Layer Security (EAP-TLS) [7], EAP Tunneled TLS (EAP-TTLS) [8], and Protected

Extensible Authentication Protocol (PEAP) [9]. Besides, it also supports several Microsoft Windows versions such as Windows XP (with Service Pack 2), Windows 2000, Windows ME, and Windows 98. However, there are still some drawbacks:

- Original WIRE1x handles different EAP methods separately, that is, users must activate different executable files instead of simply selecting a desired EAP method in a integrated menu.
- Current Graphic User Interface (GUI) is not user-friendly enough. In the previous version, no configuration-handling functions are supported, and thus users have to configure their personal information every time they want to access WLAN.
- WIRE1x uses some open source libraries, such as WinPcap and Libnet to handle packets. However, in some circumstances, these libraries do not work very well with some versions of Microsoft Windows. In addition, the installation steps are also too complicated for most users.

Because of these inconveniences and some system faults, we decide to enhance and revise the version released before. In the whole enhancement plan, we focus on the integration of EAP and the implementation of a friendlier GUI, and fix numerous fetal errors in the previous version

The rest of this report is organized as follows. Section 2 gives an overview of IEEE 802.11i, 802.lx and EAP types supported by WIRE1x. Section 3 provides a roughly introduction of WIRE1x, our enhancement plan and the method to implement. System descriptions will be given in section 4 and we will conclude the report in the last section.

2. The Next Generation Wireless Standards

2.1 IEEE 802.11i

IEEE 802.11b is the most widely-used standard in the broadband wireless networks. It defines two basic mechanisms to enhance the security of wireless communication. The first mechanism is entity authentication and association. In an IEEE 802.11b environment, a supplicant has to connect to an AP before accessing the network. Then it has to complete an authentication work before going to the association step. There are two ways to do the authentication: open system and shared key. In an open system authentication, a supplicant sends messages to an AP, and then the AP will send this identity to the authentication server. In a whole communicating process, the WEP key is not required. In a shared key system, an AP will request the supplicant to provide an encrypted packet using the WEP key. The AP sends messages to the authentication server after the result is verified corrected by AP. If the authentication in the server side is also correct, the association is then connected. The other mechanism is WEP (Wired Equivalent Privacy). The WEP is relayed between a MN and an AP. It enhances the confidentiality of wireless communication. A supplicant encrypts data and sends it with the related information to AP, and then AP decrypt the message using shared WEP key.

However, the security consideration in wireless network is much more complicated than the one in wired network. The two mechanisms mentioned above have been proved to be extremely vulnerable. For example, a WEP protected environment could be easily cracked once the man-in-the-middle collects enough packets, typically, in half an hour. As a result, IEEE 802.11i is released as the next generation wireless standard to enhance security concerned [10]. This standard defines two classes of security framework of IEEE 802.11 WLAN: RSN, and pre-RSN. A station is called RSN security framework if it can generate RSNAs; similarly, it is called pre-RSN if it can generate pre-RSNAs. A pre-RSNA system is similar to the one defined in IEEE 802.11. RSN enhances the security mechanism of original pre-RSN network.

IEEE 802.11i defines some key management procedures as well, the authentication and encryption method. There are two methods to support key distribution, manual key management and automatic key management. Manual key management requires users to configure a pre-shared key. Automatic key management incorporates with IEEE 802.1x to support key management services. However, this mechanism works only in RSNA.

In order to enhance encryption mechanisms, IEEE 802.11i proposes two algorithms: (1) Temporal Key Integrity Protocol (TKIP) and (2) Counter mode with CBC-MAC Protocol(CCMP). Both TKIP and CCMP could provide more confidential data protections than the original WEP.

IEEE 802.11i works with IEEE 802.1x for its key management and authentication services. In authentication mechanism, IEEE 802.11i relies on IEEE 802.1x to provide different types of EAP. It incorporates two components into an original IEEE 802.11 system: IEEE 802.1x port and Authentication Server (AS). In an IEEE 802.1x defined port, the connection will be built only when the authentication is completed, and the authentication is completed only if the supplicant and the server could authenticate the other by the requested EAP methods.

2.2 IEEE 802.1x

The IEEE 802.1x defines a port-based network access control mechanism. It provides an authentication and authorization mechanism for other IEEE 802 LAN devices based upon EAP. In an IEEE 802.1x system, there are three major components: supplicants, authenticators and authentication servers. As depicted in Fig.1, 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). The term ‘port’ used in IEEE 802.1x denotes the association between a supplicant (MN) and an authenticator (AP). There is a Process Access Entity (PAE) in both AP and MN, which is accountable for the authentication-related mechanism, such as algorithms and protocols. As shown in Fig.1, the port controlled by AP is initially in the unauthenticated state. Messages sent to AP will be redirected to the authentication server through Authenticator’s PAE, and then the server will start the authenticating process. If authentication is carried out successfully, the port controlled by AP will transit to the authenticated state and become connected. Eventually, supplicant can access network services through this port.

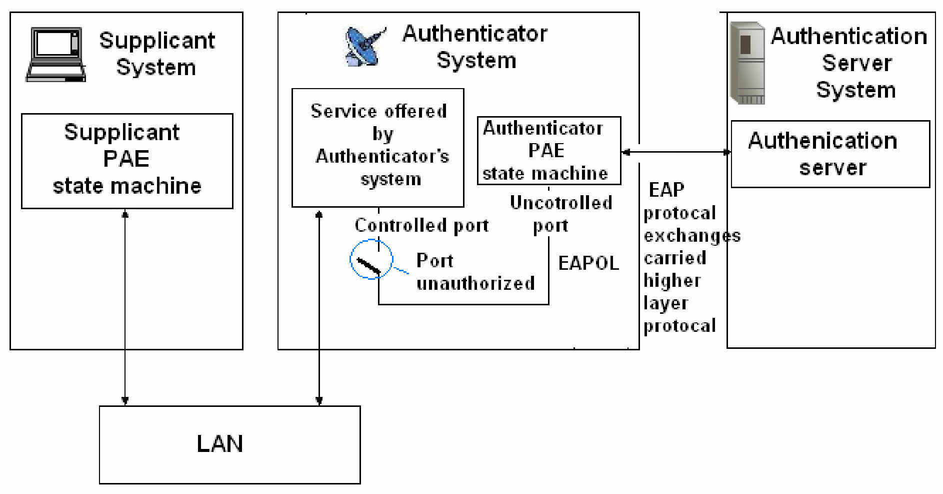

Supplicant System
Supplicant PAE state machine
Authenticator System
Service offered by Authenticator's system
Authenticator PAE state machine
Uncotrolled port
Controlled port
Port unauthorized
EAPOL
EAP protocal exchanges carried higher layer protocal
Authentication Server System
Authenication server
LAN

Fig. 1 : Supplicant, authenticator and authentication server

IEEE 802.1x uses Extensible Authentication Protocol (EAP) to provide several authentication schemes, such as EAP Message Digest 5 (EAP-MD5), EAP Transport

Layer Security (EAP-TLS), EAP Tunneled TLS (EAP-TTLS), and Protected Extensible Authentication Protocol (EAP-PEAP), and so on. 802.1x also defines EAP over LANs (EAPOL) to encapsulate EAP messages between a MN and an authentication server through the PAE of authenticator.

2.3 Extensible Authentication Protocol

Extensible Authentication Protocol (EAP)[11, 12] is the most popular general authentication framework. In addition to the authentication process, EAP also defines a simple mechanism to negotiate a suitable authentication method between servers and clients. Fig.2 depicts the basic framework of EAP system. In the authentication layer, many different authentication methods are developed on the basis of EAP layer. When a new method is created, it is easy to develop based on this well-defined protocol. Among many EAP methods, WIRE1x supports four of the most commonly used methods, EAP-MD5, EAP-TLS, EAP-TTLS, and EAP-PEAP, which will be briefly introduced in the following sections. In the end of this section, a comparison table regarding the differences among these four popular authentication methods was provided in Table 1.

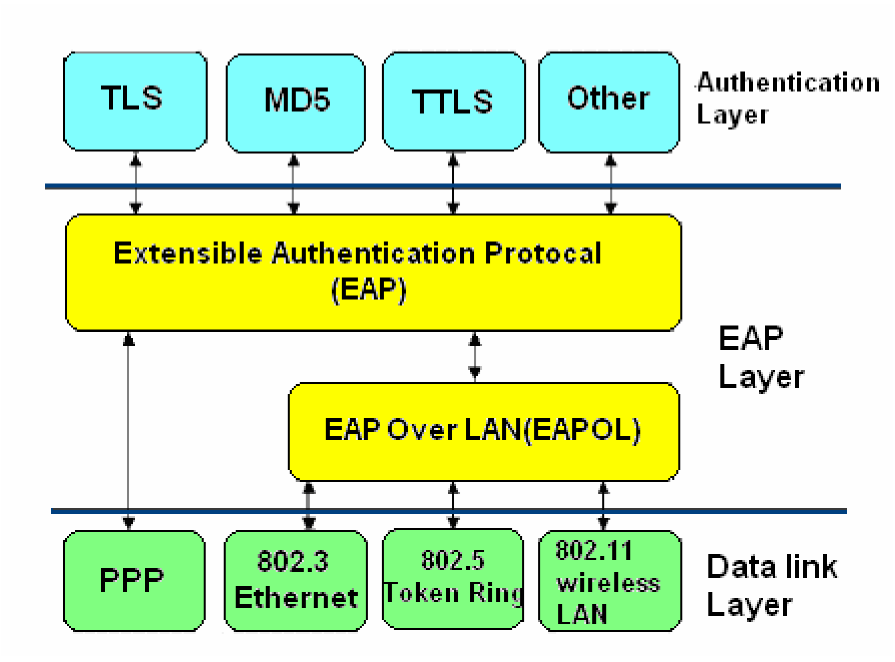


Fig. 2 : EAP and associated layers

The general authenticating procedure in EAP is illustrated in Fig.3. First, supplicant sends an EAPOL-start message to an authenticator. When the authenticator receives this message, it will send EAP-Request/Identity to ask for supplicant’s identity. Then the supplicant replies EAP-Response/Identity, which containing identity information to the authenticator. After receiving this message, the authenticator starts to exchange information with the authentication server. It will send RADIUS-Access-Request and receive RADIUS-Access-Challenge from the server. The information in RADIUS-Access-Challenge will be included in EAP-Request/Auth, which is sent from the authenticator to the supplicant in order to get the authentication data. A response will be sent to the authenticator first and then relayed to the server. After the authenticating procedure, server will send either an “accept” or a “reject” message to the authenticator. If an “accept” message is sent, the authenticator will send EAP-Success message and the supplicant could transit to the AUTHENTICATED state. Otherwise, the supplicant will receive EAP-Failure and transit to the HELD state. This is the general EAP authentication procedure. Some steps or required messages differ from varies authentication methods.

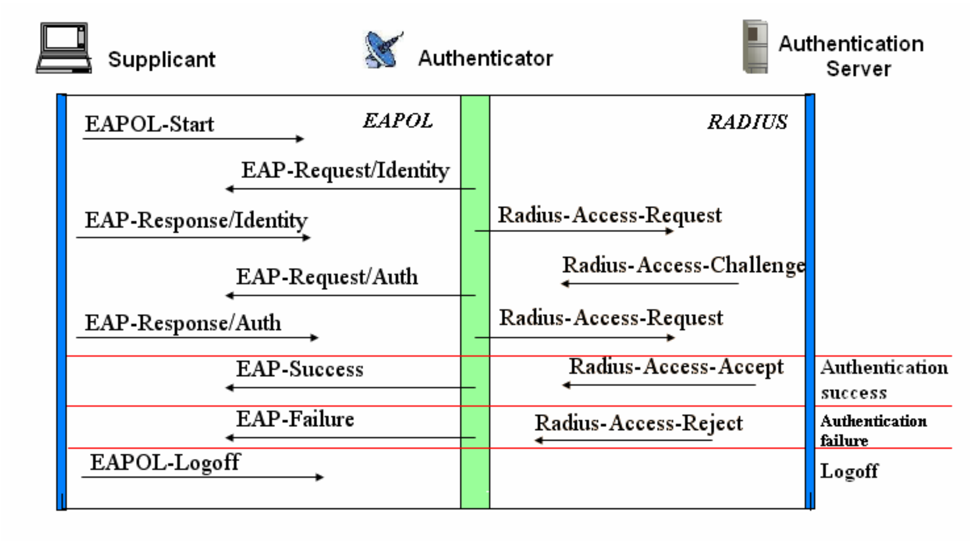


Fig. 3 : A typical 802.1x message flow

2.3.1 EAP-MD5

EAP-MD5 [6] is one of the simplest and most popular authentication methods. Only two data are required: username and password. Users provide their usernames first, and then the supplicants will send the usernames to AP. By the time AP receives these packets, it will reply them via the RADIUS-Access-Request message. After the server receives the data, it will send Challenge/Response packets to the supplicant. Later on, the supplicant then encapsulates the password in EAP-Response/Auth, and sends to the server through AP. Eventually, the authentication is completed after server replies a success message to supplicant. This procedure is quite simple and easy to implement; however, it has been proved more vulnerable than other methods due to its simplicity.

2.3.2 EAP-TLS

The standard of EAP-TLS [7, 12] is considered to be the most secure EAP method and is supported by many manufacturers. EAP-TLS is based on TLS [13] to provide protected cipher-suite negotiation, mutual authentication, and key management. The reason why EAP-TLS can provide a great deal of security is that it will build an encrypted tunnel between a supplicant (MN) and an authenticator (AP) after the negotiation was completed. Then all communications will be carried out in the protected tunnel. In the standard, a server and a supplicant can manually authenticate the certificate of each other, and they can also authenticate the network. A compromised password is not enough to break into EAP-TLS enabled systems because illegal users still should to have the client-side certificate. If the certificate is kept in some hardware devices, for example, a smart card, then it will be difficult to steal the private key in the certificate.

However, there are some chanllenges to manage the EAP-TLS system because valid certificates are both required for supplicants and authentication servers. In the typical EAP authentication procedure, when a server receives a supplicant's identity from EAP-Response/Identity by an AP, it will send a message containing its certificate information and ask for the one of the supplicant. And the supplicants will response the server its certificate information as well. In addition to these basic data, both ends also provide cryptographic material for the session. These session keys can be used to encrypt data.

2.3.3 EAP-TTLS

EAP-TTLS [8] is an EAP protocol that extends from EAP-TLS. In EAP-TLS, a TLS handshake is used to authenticate both client side and server side mutually. When this phase finishes, a secure tunnel will be established. EAP-TTLS extends the authentication negotiation by using a secure connection built before clients and servers exchange additional information. Different from EAP-TLS, EAP-TTLS does not restrict valid certificates in both supplicant and server. A TLS handshake in EAP-TTLS may be two-way or one-way, only the server is authenticated by clients. The server can authenticate the supplicant in the secure tunnel by other authentication methods. Authentication methods used here are not restricted to be EAP methods.

They can be other authentication protocols, such as PPP Authentication Protocols (PAP), PPP Challenge Handshake Authentication Protocol (CHAP), Microsoft PPP CHAP Extensions (MS-CHAP), Microsoft PPP CHAP Extensions Version 2 (MS-CHAP-V2), and so on.

2.3.4 EAP-PEAP

EAP-PEAP [9] is similar to EAP-TTLS, which is also an EAP protocol that extends EAP-TLS. Table 2 briefly summarizes the differences among three TLS-based methods. There are two phases in a PEAP authentication negotiation: (1) handshake phase (2) tunnel phase. In the first phase, a supplicant uses certificate to authenticate an authentication server. A server can also authenticate a client by its certificate, but it is mandatory. After the negotiation finishes, two ends can transfer information in an encrypted tunnel form using the key obtained in the previous phase. The most different concept of PEAP from TTLS is that it restricts protocols used in phase two to be EAP only, such as, MS-CHAP-V2. Briefly speaking, PEAP uses the TLS channel to protect the EAP exchanging later. Authentication must be performed using a protocol that is defined for the use with EAP.

Table 1 : Comparison of authentication mechanism

	EAP-MD5	EAP-TLS	EAP-TTLS	EAP-PEAP
Server Authentication	No	Public Key (Certificate)	Public Key (Certificate)	Public Key (Certificate)
Supplicant Authentication	Password Hash	Public Key (Certificate or Smart Card)	Certificate, EAP, or non-EAP protocols	Certificate or EAP methods
Manual Authentication	No	Yes	Yes	Yes
Dynamic Key Delivery	No	Yes	Yes	Yes

Table 2 : Comparison of TLS-based methods

	EAP-TLS	EAP-TTLS	EAP-PEAP
Basic Protocol Architecture	Establish TLS session and validate certificates for both client and server	(1) Establish TLS between client and TTLS server (2) Exchange attribute-value-pairs between server and client	(1) Establish TLS between client and PEAP server (2) Run EAP exchange between TLS tunnel
Server Certificate	Required	Required	Required
Client Certificate	Required	Optional	Optional
Protection of User Identity	No	Yes, protected by TLS	Yes, protected by TLS

3 WIRE1x

3.1 Introduction to WIRE1x

WIRE1x is an open-source implementation of IEEE 802.1x client (supplicant) 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 [5], which is also a software designed for WLAN authentication, but supports only Linux. Although there are many users use Microsoft Windows, only little free software could provide a comprehensive and convenient authentication solution. For these reasons, WIRE1x is developed for supporting varies versions of Microsoft Windows and providing a number of authentication methods.

The structure of WIRE1x could divide into three components: (1) supplicant PAE state machine, (2) EAP and authentication mechanisms, and (3) Several 3rd party libraries--WinPcap[14], Libnet[15], and OpenSSL[16]. Supplicant PAE state machine follows the specification defined in IEEE 802.1x. The general state diagram of PAE state machine is illustrated in figures 4 and 5. WIRE1x supports various authentication mechanisms in EAP. In the latest version, four EAP authentication mechanisms are supported by WIRE1x, including EAP-MD5, EAP-TLS, EAP-TTLS, and EAP-PEAP.

WinPcap and Libnet are open-source libraries 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 only EAP frames by adjusting parameters in filter. Moreover, WIRE1x use WinPcap to retrieve the list of network interfaces. OpenSSL is also an open-source library that crypts and decrypts messages required by the TLS authentication methods. It is a toolkit that implements the Secure Sockets Layer (SSL) v2/v3 and the TLS v1 protocols. WIRE1x use it to load server certificates, client certificates, and the client private key in different formats. These certificates are required in the authentication process.

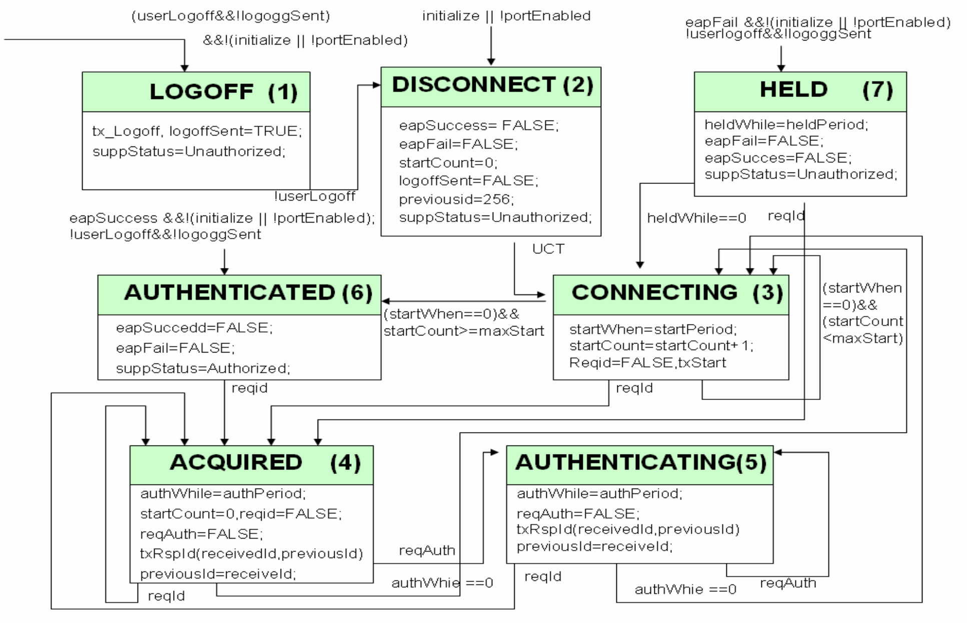


Fig. 4 : Supplicant PAE state machine

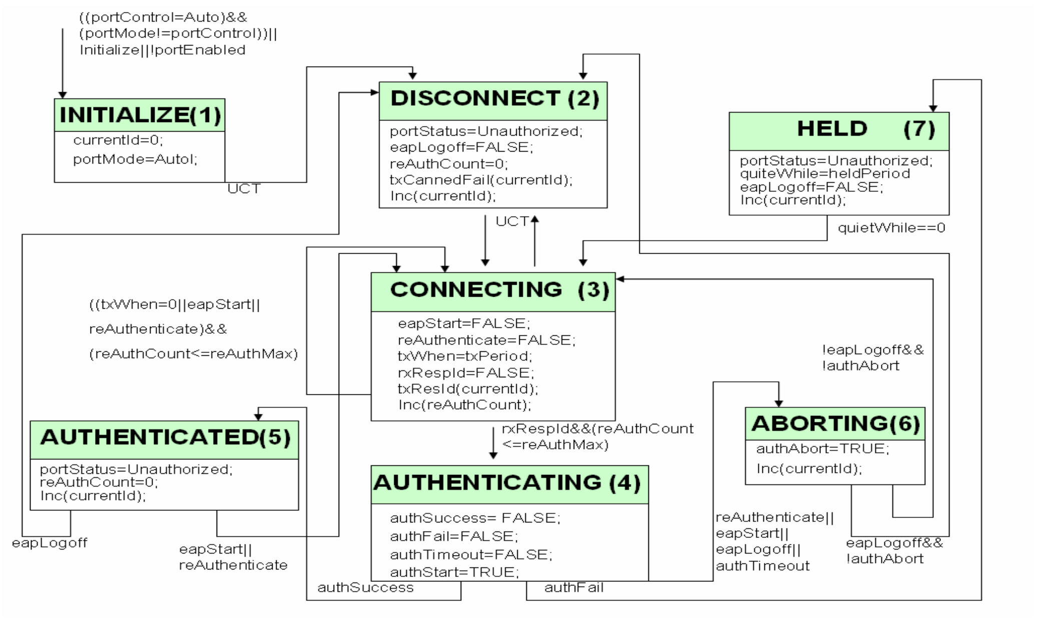


Fig. 5 : Authenticator PAE state machine

3.2 The Enhancement Plan of WIRE1x

Our major targets of the plan are: (1) making WIRE1x more convenient and (2) making the system of WIRE1x more stable. From the previous version, we have found several shortcomings that can be improved, which are as follows:

1. The four authentication methods supported by current WIRE1x are separated into four programs. This may probably make users confused. Besides, users must change from different executables if they want to change authentication methods. How to setup these four programs and other associated files correctly are also suffering for most users.

2. The authentication process of WIRE1x works under the condition that authentication servers and supplicants both held the same methods in the beginning. Since the previous version could not handle the situation that server asks the method that it does not support, once the default method of servers is not supported by the supplicant, the authentication fails. However, according to the specification of EAP standard, the authentication should be successful if supplicants support any one method but not only the default one provided by servers.

3. WIRE1x adopts several open source libraries to capture and send frames, read interfaces, etc. Since WIRE1x must work with these libraries, users have to install them before executing WIRE1x. Unfortunately, according to our testing, some versions of these libraries are not compatible with several versions of Microsoft Windows, and hence might cause WIRE1x crash.

First, we decide to solve the problem about separated GUI. We try to design a new GUI model for WIRE1x and transform our develop environment to Visual Studio 2003. The new GUI should merge four EAP methods into a single window form and display the integrated information. We add a configuration dialog to help users configure these EAP methods. However, we come up with some problems when we port EAP-MD5 from original program to the new one. It seems that WinPcap does not work very well under Visual Studio 2003. The new program always crash once it calls the WinPcap APIs. After several discussions, we decide to abort the development of the new GUI and try to remove the 3rd party libraries from the original version.

To remove these open-source libraries, we begin to trace the source code of SecureW2. SeureW2 is a free IEEE 802.1x supplicant which supports EAP-TTLS for Microsoft Windows, but none of 3rd libraries are involved. We surmise that SecureW2 is based on numerous services provided in Microsoft Windows, which maybe useful for us. After tracing codes and testing the program, we find that the structure of SecureW2 is considerably differs from the one of WIRE1x. SecureW2 does not handle any authentication process but acts as extensions of the Wireless Zero Configuration (WZC) service in Windows. Compared to WIRE1x, SecureW2 releases as a dynamic linked library (dll) file instead of an executable one. Initially, after users install it, SecureW2 adds numerous of registry values into Windows to make itself play a role of a callback routine of the Remote Access Service (RAS), which is a basic service of the WZC service. When the authentication server requests some authentication information, WZC service would receive the requests and invoke callback functions of SecureW2, and SecureW2 would retrieve these information from the user profile and return to the callee, however, which is not a piratical paragon for us, because in order to adopt this model, we should revise the old architecture extremely significantly. For example, the PAE state machine will be no longer necessary, that is, we must destroy the old WIRE kernel and remove all related implementations, which implies that WIRE1x would no longer be WIRE1x. Furthermore, Some EAP methods also have to be removed, for instance, EAP-MD5 could hardly cooperate with Microsoft Windows, since it has been blocked due to several security issues according to the Microsoft announcement. In addition, WIRE1x will depend on these services in this way, that is, every time Microsoft releases new service packs, we may have to revise in order to be compatible with them. Similarly, in most of older Windows versions that do not build in these services, such as Windows 98, WIRE1x would not be executed under these versions due to the lack of the infrastructures. Eventually, by tracing SecureW2, we could scarcely collect any information useful. Hence, we decide to keep the original structure of WIRE1x and continue to develop a new integrated GUI. This time we try to develop it using Visual studio 6.0, the original develop environment of WIRE1x.

To solve other problems, we have the following solutions:

1. We implement a configuration wizard rather than the original configuration dialog. This wizard can also integrate all four EAP methods into one window form. Besides, users could configure their profiles much more easily in this way.

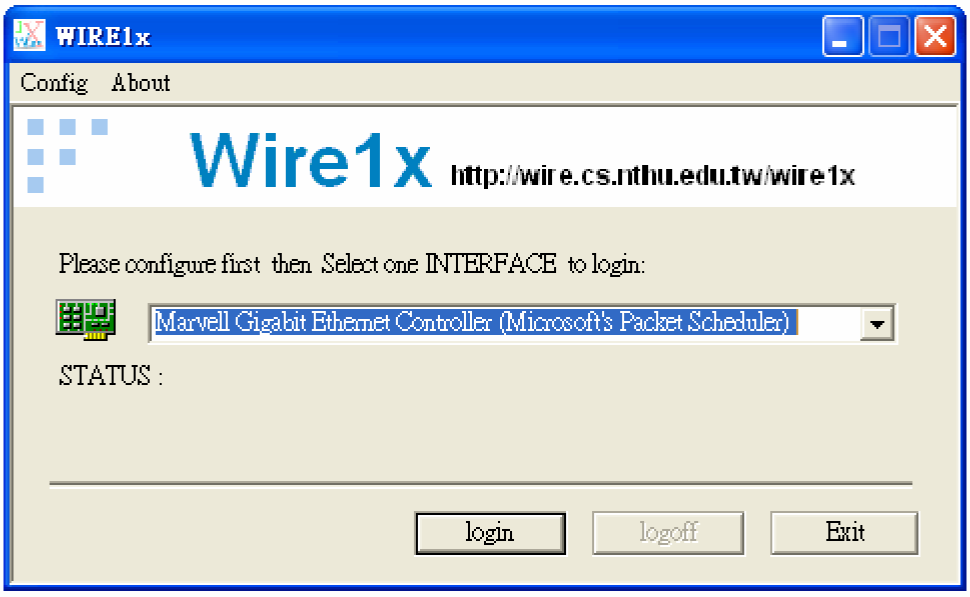
2. We also design an installation wizard to help users install and remove WIRE1x more conveniently. This wizard would install all necessary files by simply clicking a few buttons, and users would suffer no panic about which, where, and how to place the related files.

3. NAK supporting is implemented. When WIRE1x finds the method that the server requests is not available, it will send server a NAK message to indicate which methods are acceptable. If the server supports these methods, it could choose one supported method to proceed the authentication.

More details of the implementation regarding our enhancement implementation will be discussed in next section.

4. System Description

This section describes the implementation details. The new architecture and features will be discussed in the separated sub sections.



Please configure first then Select one INTERFACE to login:

Marvell Gigabit Ethernet Controller (Microsoft's Packet Scheduler)

STATUS :

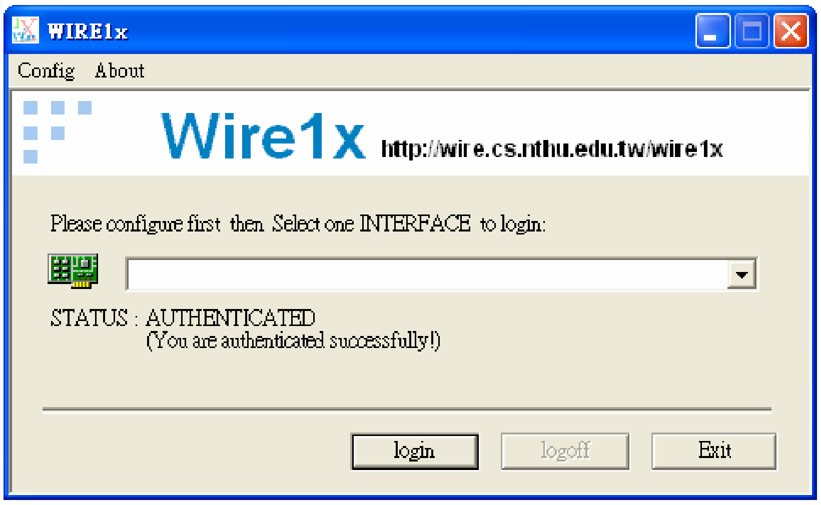
Fig. 6 : New GUI for WIRE1x

4.1 System Architectures

In this sub section, we will give an illustration of the partially new architecture. After the enhancement, the user interface and some functions of WIRE1x have been revised. The major three original components of WIRE1x, supplicant PAE state machine, EAP authentication mechanisms, WinPcap, and OpenSSL, are kept.

First, there is an installation wizard to help user install WIRE1x. Besides, there are three parts in the new interface, which are a toolbar on the top, an interface list in the middle, and a control button on the bottom. The whole interface is shown in Fig.6. The configuration wizard is invoked by clicking the “Config” icon in toolbar. Users have to configure their profiles in the beginning; otherwise, they are not allowed to go to the next steps. When users finish configuring profiles, they could click the “login” button, which would launch the authentication. Real-time state transitions are shown in the textbox titled “STATUS:” in the middle of the window form, so users could clearly know which state WIRE1x is in.

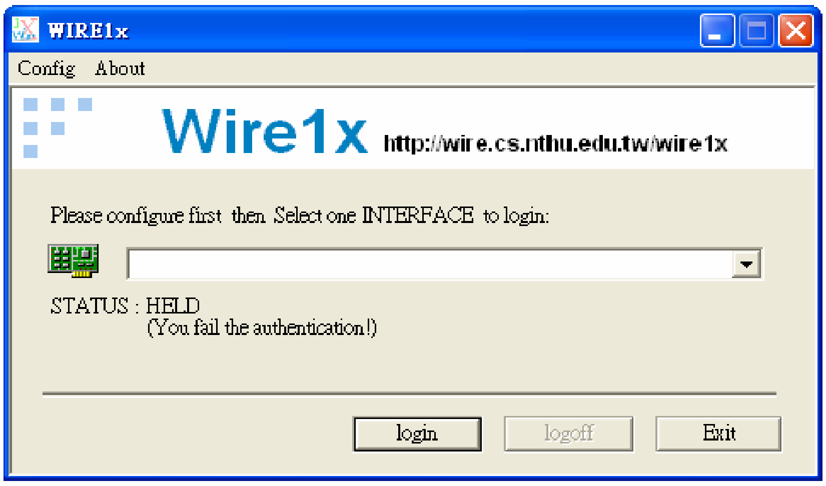
When the authentication process begins, configurations of each EAP method will not be set until the program realize which EAP method server requests by decoding the packets sent from the server. If the method that the server requests is not enabled, a NAK message will be sent. If users authenticate successfully, WIRE1x will transit to AUTHENTICATED state. Otherwise, it will transit to HELD state. In the authentication process, all functionalities of configuration will be disabled temporarily. After the program finishes the authentication, users can click the “logoff” button to send a logoff packet to the server in order to logout. To click the “Exit” button will simply terminate the program without sending any message. The execution screenshots in the different states are shown in figures 7 and 8.



Please configure first then Select one INTERFACE to login:

STATUS : AUTHENTICATED
(You are authenticated successfully!)

Fig. 7 : AUTHENTICATED state (Success to authentication)



Please configure first then Select one INTERFACE to login:

STATUS : HELD
(You fail the authentication!)

Fig. 8 : HELD state (fail to authentication)

The whole process and structure of the system is illustrated in Fig.9. Briefly specking, users install the program by the installation wizard, and then they could execute the program. They have to use the configuration wizard in the beginning to provide necessary information of available EAP methods and then begin to authentication. If the authentication successes, user will see the state shown in the status area transit to the AUTHENTICATED state. If authentication fails, they may have to correct their profiles and login again. After they login successfully, they could click the logout button to send logoff message to the server or just exit when they want to disconnect.

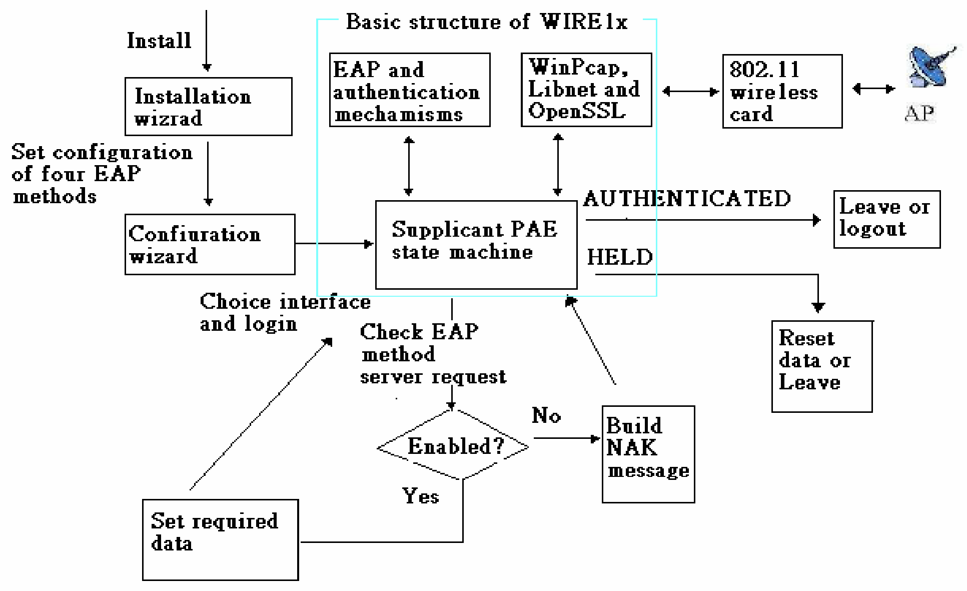

Install
Installation wizrad
Set configuration of four EAP methods
Confiuration wizard
Basic structure of WIRE1x
EAP and authentication mechamisms
WinPcap, Libnet and OpenSSL
802.11 wireless card
AP
Supplicant PAE state machine
AUTHENTICATED
Leave or logout
HELD
Reset data or Leave
Choice interface and login
Check EAP method server request
Enabled?
No
Yes
Build NAK message
Set required data

Fig. 9 : System block diagram

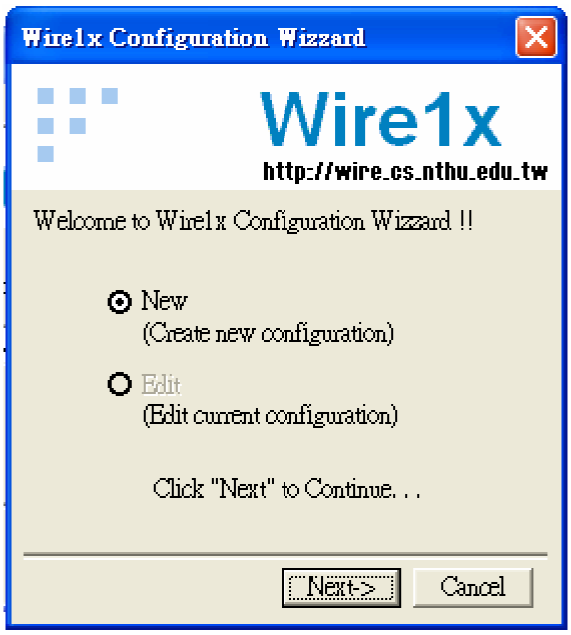
4.2 Supporting NAK

NAK [11] is a special message sent from a supplicant in response of the request of the EAP information from an authentication server. It is used to reply to a server that the authentication method it requests is not supported by the client. In a typical EAP procedure, the NAK message will be sent after a supplicant receives EAP-Request/Auth from the authenticator. In a normal situation, if a supplicant supports the EAP method that server desired, it returns an EAP-Response/Auth to the AP. However, if this method is not available, the authentication fails without the NAK mechanism. In a typical NAK response, the supplicant will indicate one or more supported EAP types. When an AP receives the NAK message, it will send RADIUS-Access-Request to the server again and contain information received in NAK. If the authentication server supports these new authentication protocols, it will send a new RADIUS-Access-Challenge to the AP, and request information of another available method. The authentication fails only if servers accept no method that clients supported.

In WIRE1x, there are several parts handling NAK. The supplicant receives a packet from the AP and decode it by eap_decode_packet(). If it finds that the variable eap_return returns from eap_decode_packet() is zero, the packet is an EAP packet. It will use eap_type_check() to check whether the authentication method that server requests is acceptable in clients. If it is available, related information will be set and EAP-Response/Auth packet will be sent. Otherwise, it will call send_NAK() to send a NAK message. Users could configure multiple EAP methods at the same time, so the NAK message may contain multiple EAP methods and the corresponding information.

4.3 Configuration Wizard

Configuration wizard is also an important achievement for improving the user-friendliness. It not only integrates four EAP methods but also helps the NAK mechanism work successfully. The configuration wizard is in the “Config->setting” button of the toolbar. When users launch it, it will show a serial of dialogs to help them configure their settings for each EAP methods. The first dialog is shown in Fig.10, and it will ask users that they want to create a new configuration or edit the current configuration. The dialog for configuration of EAP methods is illustrated in Fig.11. There will be a little difference among the different EAP methods. There are two main parts in these dialogs: one is used to check whether some settings are enabled and the other one is used to edit necessary information. After users configure available methods completely, the wizard will show a dialog about what configurations have already been set. This dialog is shown in Fig.12, which shows information within the list box in the middle. The “Reset” button would cancel the current settings and the “Finish” button would save and exit



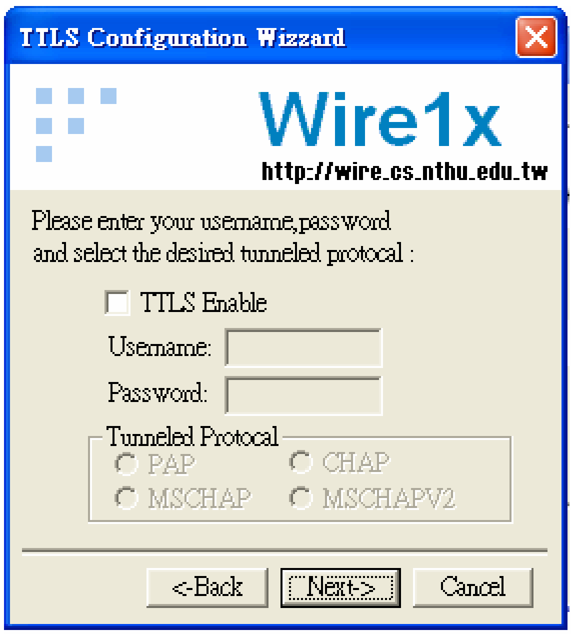
Welcome to Wire1x Configuration Wizzard !!

[✓ New]
(Create new configuration)

Edit
(Edit current configuration)

Click "Next" to Continue. . .

Fig. 10 : Welcome dialog of configuration Wizard



Please enter your username,password
and select the desired tunneled protocal :

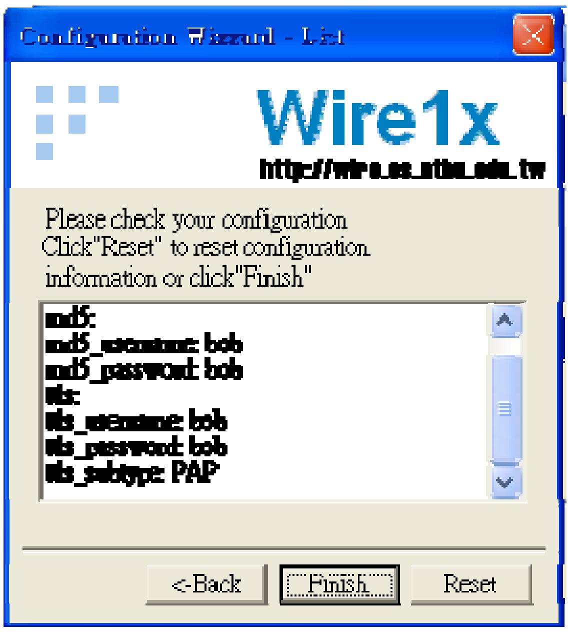
☐ TTLS Enable

Username:

Password:

Tunneled Protocal
PAP CHAP
MSCHAP MSCHAPV2

Fig. 11 : Dialog for setting information of EAP methods

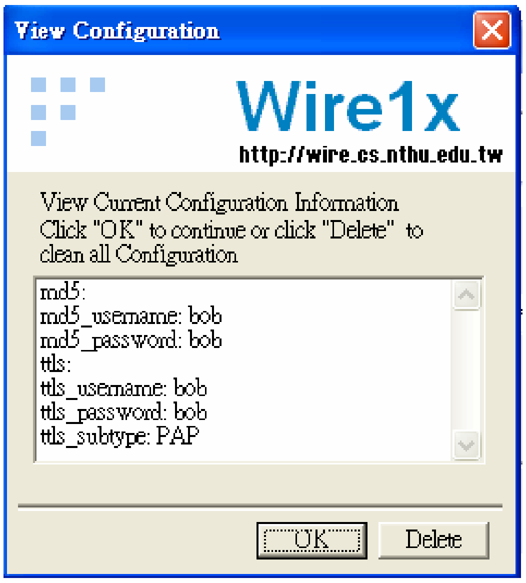


Please check your configuration
Click"Reset" to reset configuration
information or click"Finish"

```
md5:
md5_username: bob
md5_password: bob
ttls:
ttls_username: bob
ttls_password: bob
ttls_subtype: PAP
```

Fig. 12 : Dialog for showing all configurations

In addition to the configuration wizard, there are other functionalities for users to deal with EAP methods’ settings. The “Config->view” button in the toolbar can show current configurations as illustrated in Fig.13. The “Delete” button can clear the configuration already set. When users want to keep their configurations, they can click the “Config->save” button to save the profiles as a file with a extend file name “.1xconf”. They can click “Config->load” button to load their profiles as current configuration any time. All functionalities of configuration, except viewing configuration, will be disabled during the authentication process. Whether the authentication is successful or failed, these buttons will become enabled again when the process is completed.



View Current Configuration Information
Click "OK" to continue or click "Delete" to
clean all Configuration

```
md5:
md5_username: bob
md5_password: bob
ttls:
ttls_username: bob
ttls_password: bob
ttls_subtype: PAP
```

Fig. 13 : Dialog for viewing current configurations

When the configuration wizard is terminated by clicking the “finish” button, profiles will be saved into a “userconf” structure. There will be a “user_setting” pointer points to its memory location. Then all data will be saved into the corresponding fields, such as “user_setting->ttls_username,” “user_setting-> ttls_password,” etc. If users have enabled EAP-MD5, then username of EAP-MD5 is saved in “user_setting->username,” which is the actual location of data used in an authentication process. If EAP-MD5 is not enabled, WIRE1x will search other EAP methods, in the sequence of EAP-TTLS, EAP-TLS, and EAP-PEAP, for available usernames and set them as default. The reason why we have to set usernames in advance is the supplicant should have some data to reply in the EAP-Request/Identity message. The information of the authentication method actually working in a supplicant will not be requested by the server immediately after the authentication procedure proceeds, so we can set other information and then send the correct messages when a server sends an EAP/request message. However, EAP-MD5 differs from other methods because it uses the identity that we reply for authentication, so we must have set correct username of EAP-MD5 as the default setting in the beginning.

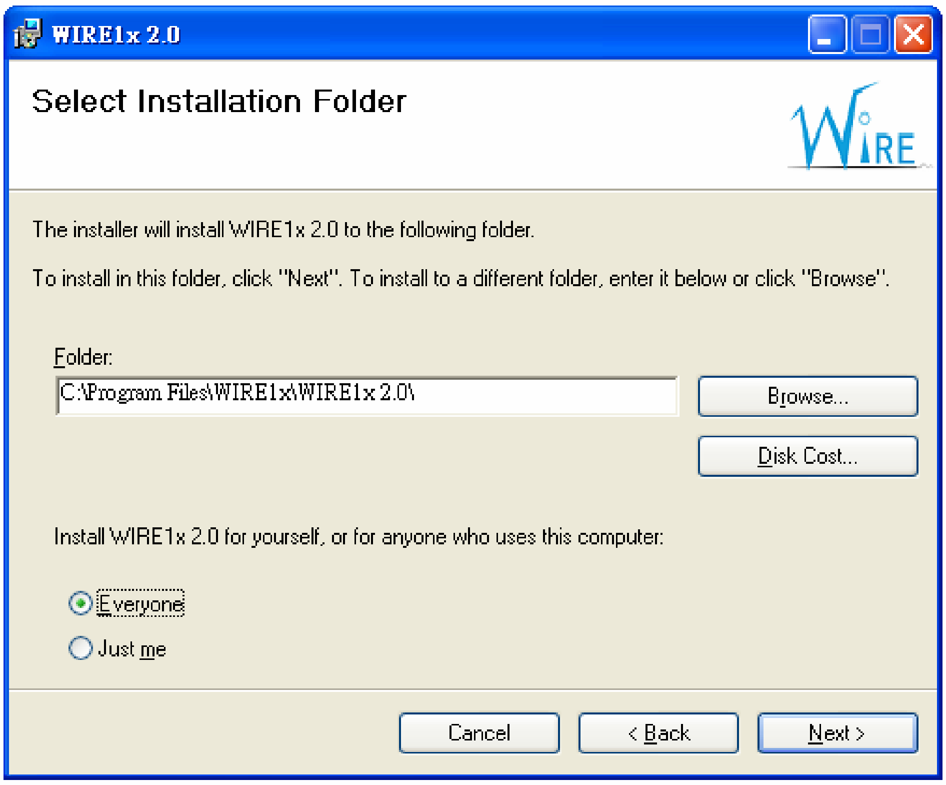
4.4 Installation Wizard

The installation wizard is implemented to help users install and remove the program more conveniently. In the installation of the previous version of WIRE1x, users must download all related files and locate them manually in the correct place and order. In this version, with installation wizard, users could just follow the instructions to install WIRE1x. The wizard also creates a shortcut in the desktop and in the start menu.

The installation wizard is designed by using the functionality of installation project in Visual Studio 2003. It encapsulates all associated files in a package. When users launch it, it shows a serial of dialogs to help users. The first step shows some messages to welcome users. The next step, as illustrated in Fig.14, requests the installation path and the priority to other people who use this computer. When users finish the decision of the path and the priority, the wizard begins to decompress the package to the desired directory, and establish shortcuts and links. There will be another dialog to inform users after the installation is complete.

When users want to remove WIRE1x, they can go to ”Add or Remove Programs” in the “Control Panel.” There is an icon that indicates users where to start the uninstallation process. After users click the remove button the system removes

WIRE1x automatically. Shortcuts will be removed as well.



WIRE1x 2.0

Select Installation Folder

The installer will install WIRE1x 2.0 to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder:
C:\Program Files\WIRE1x\WIRE1x 2.0\

Install WIRE1x 2.0 for yourself, or for anyone who uses this computer:

[✓ Everyone] Just me

Fig. 14 : Installation Wizard

5. Conclusions

Due to the popularity of wireless networks, the importance of the related software is also increased. Compared to the wired networks, security considerations of the wireless networks are much more critical. Hence, to access the wireless networks through a much more secure standard, such as the next generation wireless standard—IEEE 802.11i, is an unavoidable trend.

After our revision, the latest WIRE1x becomes a much more powerful comprehensive solution for the next generation wireless environment. In the previous version, there are so many disadvantages that make users extraordinary panic to install, to configure, and to use, and thus these drawbacks motivate us to have WIRE1x reach a whole new milestone.

In this paper, we show that how our project significant is. First of all, we integrate different EAP methods into a single window form, that is, users no longer have to activate different executables when switching to different methods. This new version also supports multiple EAP methods simultaneously, in other words, users could configure more than one methods and WIRE1x would choose the proper one to proceed automatically. In addition to supporting multiple EAP methods, this version contains the NAK negotiating function as well, which could select another method while the default one of servers is not supported by clients. Furthermore, in order to make WIRE1x more user-friendly, several wizards are involved, which are configuration wizard and installation wizard. The configuration wizard is an extremely facility in managing user profiles. By using this wizard, users scarcely suffer the inconvenience of editing, saving, and loading profiles before enjoying the convenience of the wireless networks. The complicated and verbose installation instructions are also extinct, since the other wizard, installation wizard, dramatically simplifies the manual installation to a few simple mouse clicks.

WIRE1x is expected to have a significant impact in the IEEE 802.1x supplicant market, because it is currently the only comprehensive client solution for the next generation wireless security standard for Windows. However, the finishing of our project is not the end of WIRE1x maintenance but another whole new beginning of it. In the future, by supporting more secure and popular authentication methods, and by supporting more operating systems, even for some handheld devices, we expect that, as the wireless services are more widely provided and used, WIRE1x could become more significant and more powerful.

6. References

[1] 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.

[2] 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.

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

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

[5] “Open1x” http://www.open1x.org/.

[6] R. Rivest, “The MD5 Message-Digest Algorithm,” IETF RFC 1321, Apr. 1992.

[7] B. Aboba, D. Simon, “PPP EAP TLS Authentication Protocol,” IETF RFC 2716, Oct. 1999.

[8] 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.

[9] 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.

[10] J.-C. Chen, M.-C. Jiang, and Y.-W. Liu, “Wireless LAN Security and IEEE 802.11i ,” IEEE Wireless Communications, vol. 12, no. 1, pp. 27-36, Feb. 2005.

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

[12] “Extensible Authentication Protocol,” Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Extensible_Authentication_Protocol

[13] T. Dierks and C. Allen, “The TLS protocol, version 1.0,” IETF RFC 2246, Jan. 1999.

[14] “WinPcap” http://winpcap.polito.it/.

[15] “Libnet” http://libnet.sourceforge.net/.

[16] “OpenSSL” http://www.openssl.org/.