WIDE Paper-List in 2011

Network Access Authentication Infrastructure Using EAP-TTLS on Diameter EAP Application

wide-paper-aaa-eapttls-00.pdf

WIDE Project : http://www.wide.ad.jp/

If you have any comments on WIDE documents, please contact to board@wide.ad.jp

Title:	Network Access Authentication Infrastructure Using EAP-TTLS on Diameter EAP Application
Author(s):	Yuki Atsuya, Souheil Ben Ayed, and Fumio Teraoka
Date:	2011-11-09

Network Access Authentication Infrastructure Using EAP-TTLS on Diameter EAP Application

Yuki Atsuya
Graduate School of Science and Technology
Keio University
3-14-1 Hiyoshi, Kohoku-ku,
Yokohama, 223-8522, Japan
atie@tera.ics.keio.ac.jp

Souheil Ben Ayed
Graduate School of Science and Technology
Keio University
3-14-1 Hiyoshi, Kohoku-ku,
Yokohama, 223-8522, Japan
souheil@tera.ics.keio.ac.jp

Fumio Teraoka
Faculty of Science and Technology
Keio University
3-14-1 Hiyoshi, Kohoku-ku,
Yokohama, 223-8522, Japan
tera@ics.keio.ac.jp

ABSTRACT

In our universal AAA (Authentication, Authorization, and Accounting) infrastructure project, we have already developed the implementations of Diameter Base Protocol and Diameter EAP Application. As part of this project, we developed the first open-source of an EAP-TTLS server on Diameter EAP Application for network access control. EAP-TTLS is one of the authentication methods in EAP. EAP-TTLS has two phases. In phase 1, the user authenticates the EAP-TTLS server by the certificate of the EAP-TTLS server. In phase 2, the EAP-TTLS server authenticates the user by user's password transmitted through the secure tunnel established in phase 1. Our implementation supports several authentication methods in phase 2 such as PAP, CHAP, MS-CHAP, and MS-CHAPv2. It was made sure that the EAP-TTLS server worked correctly for several types of user terminals such as Windows, Linux, iPad, and Android. The evaluation results show that the processing time of the EAP-TTLS server is short enough for practical use.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks, Network Protocols

General Terms

Design, Performance

Keywords

AAA, Diameter, EAP, EAP-TTLS

AINTEC'11, November 9–11, 2011, Bangkok, Thailand.

1. INTRODUCTION

AAA (Authentication, Authorization, and Accounting) is an indispensable function for service providers to provide users with services in the Internet. One of our projects aims at building a universal AAA infrastructure for a multi-realm environment on which a service provider can authenticate and authorize a user requesting a service via a *trust chain* between realms even if the realm to which the user belongs and the realm to which the service provider belongs do not have a direct trust relationship.

In the current Internet, RADIUS (Remote Authentication Dial In User Service)[13] is widely used for AAA. Since RADIUS is originally designed for a single realm, it has several problems if it is used in a multi-realm environment. For example, RADIUS uses UDP as the transport layer protocol and does not define a mechanism for secure and reliable message exchange; it does not define a failover mechanism when a server crashes; it does not have server-initiated message exchange. As a successor of RADIUS, Diameter Base Protocol[8] was standardized in the IETF. It is designed for a multi-realm environment and solves the problems RADIUS has. The function of Diameter Base Protocol is to exchange AAA information between a Diameter client and a Diameter server. Application specific functions are defined as Diameter Applications on top of Diameter Base Protocol in separate documents. For example, Diameter EAP Application[10] is defined for authentication and authorization for network access.

In our universal AAA infrastructure project, we have already developed the first open source of Diameter Base Protocol called *freeDiameter*[9] and the first open source of Diameter EAP Application called *DiamEAP*[5]. EAP (Extensible Authentication Protocol)[4] is extensible so that newly defined authentication methods can be added later. The package of DiamEAP includes EAP-TLS[14], which supports mutual authentication between the user and the Diameter server by exchang-

ing the certificates of both sides. Although EAP-TLS is one of most secure authentication methods, it is a little bit difficult to deploy EAP-TLS because it requires that users' certificates be distributed to the users and that the user certificate be installed in user's device.

On the other hand, EAP-TTLS[11] also supports mutual authentication but it does not require the user certificate for user authentication. EAP-TTLS has two phases. In phase 1, the EAP-TTLS server sends its certificate to the user and the user authenticates the EAP-TTLS server. In this phase, a secure tunnel is established between the user and the EAP-TTLS server. In phase 2, the user can select one of EAP methods that might not be secure enough such as CHAP and send user's information such as the password to the EAP-TTLS server through the secure tunnel. Since deployment of EAP-TTLS is easier than that of EAP-TLS, several operating systems such as Windows, Linux, and Android support EAP-TTLS as user side network access control. However, as far as we know, there is no open source of an EAP-TTLS server.

This paper describes the design, implementation, and evaluation of the EAP-TTLS server on Diameter EAP Application. As far as we know, this implementation is the first open source of an EAP-TTLS server. It is made sure that our EAP-TTLS server works correctly for several types of user terminals. The performance of our EAP-TTLS server is also shown.

2. RELATED WORK

2.1 Eduroam

Eduraom (education roaming)[1, 6] is an international roaming service for members of different universities, research institutions and educational organizations. In the Eduroam architecture, users from these institutions are able to access the Internet at any of the participated institutions. Eduroam is a hierarchically federated service based on a number of technologies: principally the RADIUS AAA protocol and the IEEE 802.1X technology. The principle of Eduroam control access is that when a user tries to access the Internet, its credentials are checked at the institution to which the user belongs.

Eduroam lacks some critical features such as for controlling user's authorizations to access the Internet. In addition, an institute may provide additional services and resources for visitor users, however, after being authenticated successfully, the home institution and the institution providing Internet access are not exchanging user's attributes which can be useful for access-control at service layers.

2.2 Shibboleth

Shibboleth[3, 12] is Internet2 federated identity management middleware focusing on educational institutions. It is an open-source project based on Security Assertion Markup Language (SAML)[2], an OASIS Security Services Technical Committee XML-based standard for creating and exchanging authentication and authorization information, to protect online resources from unauthorized access. It also provides a federated Single Sign-On (SSO) for web-based applications. The Shibboleth protocol aims to allow federated organizations and different Service Providers (SPs) to manage and exchange information about shared resources. It defines a set of interactions between a service provider and an identity provider to facilitate exchange of attributes. Comparable to many other authentication and authorization environments, the access control in Shibboleth is based on user identity. However, the user identity is not enough and more additional user information should be considered in making decision in order to offer fine-grained access control and grant adequate authorizations for accessing services.

3. OVERVIEWS OF DIAMETER AND EAP

3.1 Diameter Architecture

3.1.1 Diameter Nodes

Diameter defines several kinds of nodes for use in a multi-realm environment. *The Diameter client* is a node that sends AAA requests. An example of the Diameter client is a NAS (Network Access Server). *The Diameter server* is a node that handles AAA requests for a particular realm. *The relay agent* relays the Diameter messages without analyzing their contents. *The proxy agent* relays the Diameter messages and it may modify the Diameter messages based on the policy of the realm. *The redirect agent* does not forward the Diameter messages but notifies the Diameter client, the relay agent, or the proxy agent of the routing information. *The translation agent* performs protocol translation between Diameter and other AAA protocols such as RADIUS.

Figure 1 shows an example of Diameter nodes. There are two realms: Realm-1 and Realm-2. Each realm has its own Diameter server that handles AAA requests for the realm, the relay or proxy agent for message routing to other realms, the redirect agent, and the NAS (Network Access Server) as the Diameter client. In this example, a node `node@realm-2` sends a network access request to the NAS in Realm-1. The Diameter Request and Answer messages are exchanged between the Diameter client in Realm-1 and the Diameter server in Realm-2 via the relay/proxy agents in Realm-1 and Realm-2. As a result, the node is authenticated and allowed to access the Internet.

3.1.2 Diameter Base protocol and Diameter Applications

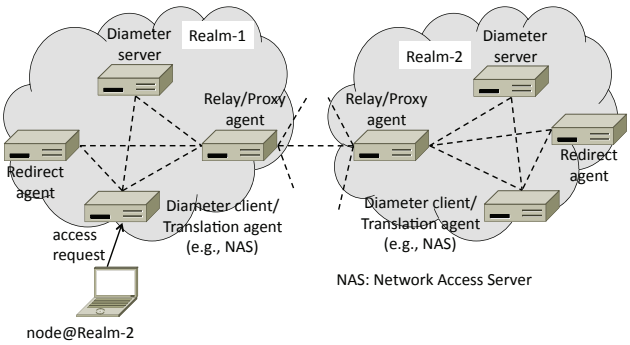

Figure 1: An example of Diameter nodes

Diameter consists of Diameter Base Protocol[8] and a lot of Diameter applications such as Diameter EAP Application[10]. At the Diameter Base Protocol level, all the Diameter nodes are forming an overlay peer-to-peer network with specific security, reliability, and routing properties. On top of this overlay network, Diameter applications are deployed. A Diameter application specifies the role of the different entities involved in a particular service, and the different commands and data that these entities exchange. For example, Diameter EAP Application defines some commands and data for authentication and authorization of a node that tries to connect to a network.

3.2 EAP

3.2.1 EAP Architecture

Extensible Authentication Protocol (EAP) is an authentication framework. It transports authentication information and parameters between the user and the authentication server. In addition, EAP provides functionalities for negotiation and selection of the appropriate authentication method among those proposed by the both sides. EAP is not an authentication method; it provides necessary functions and defines message formats for supporting authentication method.

The EAP standard was designed to be extensible by adding new EAP methods. It supports various authentication methods called *EAP methods* such as EAP-MD5, EAP-TLS[14], EAP-TTLS[11], and EAP-PSK[7]. Many standards support EAP authentication methods, such as IEEE 802.11, WPA and WPA2.

3.2.2 EAP-TLS

EAP-TLS is one of EAP methods that uses the TLS (Transport Layer Security) handshake to authenticate the user and the authentication server. TLS provides mutual authentication securely by exchanging the certificates of the user and the authentication server.

EAP-TLS authentication begins with the EAP-TLS-Start message in the EAP-Request message sent by the

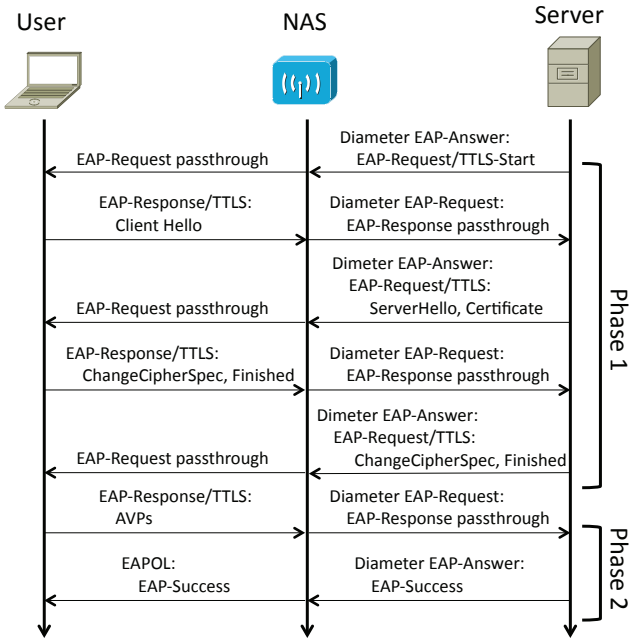

Figure 2: EAP-TTLS sequence on Diameter EAP Application

authentication server. Upon receiving the EAP-TLS-Start message, the user responds with the Client Hello message in the EAP-Response message containing other information such as the TLS version number, the session ID, the random number, and the set of cipher suites supported by the user. Next, the authentication server sends its certificate so that the user can authenticate the authentication server. Next, the user also sends its certificate so that the authentication server can authenticate the user.

The use of certificate makes great advantage for secure authentication. But it also makes disadvantage for the cost of the infrastructure, because distribution of user certificates is required, and users need to bring their own certificates when they use.

3.2.3 EAP-TTLS

EAP-TTLS uses the TLS handshake for server authentication while it uses a password based authentication method for user authentication. It is composed of two phases as shown in Figure 2: phase 1 uses the TLS handshake and phase 2 uses a password based authentication method through a secure tunnel.

In phase 1, the authentication server initiates the EAP-TTLS method with the EAP-TTLS-Start message in the EAP-Request message. Upon receiving this message, the user responds with the Client Hello message. To complete the TLS handshake, exchange of EAP messages continues between the user and the authentication

server as shown in Figure 2. When phase 1 finishes, a TLS tunnel is established between the user and the authentication server.

In phase 2, through this TLS tunnel established during phase 1, authentication message exchange will take place with another EAP method such as PAP, CHAP, MS-CHAP, and MS-CHAPv2.

4. FREEDIAMETER AND DIAMEAP

4.1 freeDiameter

freeDiameter[9] is an open source of Diameter Base Protocol[8] developed in NICT (National Institute of Information and Communication Technology), Japan. It is written in C and fully conforms to the specification.

As described in Sec. 3.1.2, Diameter Base Protocol is responsible for reliable and secure exchange of Diameter messages between the Diameter client and the Diameter server. Application specific functions are defined as Diameter Applications such as Diameter EAP Application. In freeDiameter, a Diameter Application can be added as an extension of freeDiameter.

4.2 DiamEAP

DiamEAP[5] is an implementation of Diameter EAP Application developed in our laboratory. It is implemented as an extension of freeDiameter. It provides functions in a Diameter server for authentication and authorization of a user.

As described in Sec. 3.2.1, EAP is just a framework of authentication and authorization and practical authentication mechanisms are defined as EAP methods. DiamEAP is designed to make it possible to dynamically add EAP methods such as EAP-TLS and EAP-TTLS as *plug-ins*.

DiamEAP controls the EAP method plug-ins using callback functions. By registering the functions in an EAP method plug-in with DiamEAP as the callback functions, DiamEAP will call the appropriate callback function at each time. Figure 3 shows the entries of the callback functions. The five main callback functions are described bellow.

Config Function

When DiamEAP starts up, this function is called to read the configuration file of the EAP method plug-in. This function can specify how to parse the configuration file, which variable, e.g., certificate, loads its value from the configuration file.

Check Function

When DiamEAP receives an EAP packet that should be processed by the EAP method plug-in, this function is called. This function checks the format of the received EAP packet.

```
struct register_plugin
{
    char *configure;
    char *init;
    char *initPickUp;
    char *buildReq;
    char *isDone;
    char *process;
    char *check;
    char *getTimeout;
    char *getKey;
    char *unregister;
    char *datafree;
    char *authavp;
};
```

Figure 3: Callback function entries in DiamEAP

Process Function

This is the main function of the EAP method plug-in. It is called after the check function. This function performs the authentication process and sets the result to the EAP state machine structure.

BuildReq Function

This function generates the EAP Request message according to the value in the EAP state machine.

getKey Function

Once the client is authenticated successfully, this function will be called. This function makes the master session key (MSK), which will be shared between the network access server such as a WiFi access point and the client.

5. DESIGN AND IMPLEMENTATION OF EAP-TTLS PLUG-IN

Our implementation of EAP-TTLS is designed as an EAP method plug-in (*EAP-TTLS Plug-in*) in DiamEAP as shown in Figure 4. The protocol stack composed of freeDiameter, DiamEAP, EAP-TTLS Plug-in, and EAP-TLS Plug-in realizes a Diameter EAP server that supports EAP-TTLS and EAP-TLS as EAP methods. Hereafter, the Diameter EAP server that contains EAP-TTLS Plug-in is called *the EAP-TTLS server*.

EAP-TTLS Plug-in is designed according to RFC 5281[11]. The supported authentication methods in phase 2 are PAP, CHAP, MS-CHAP, and MS-CHAPv2. The user can choose an authentication method used in phase 2 from this list during the authentication process. EAP-TTLS Plug-in does not need to configure which authentication method is used in phase 2 in advance.

5.1 Callback Functions

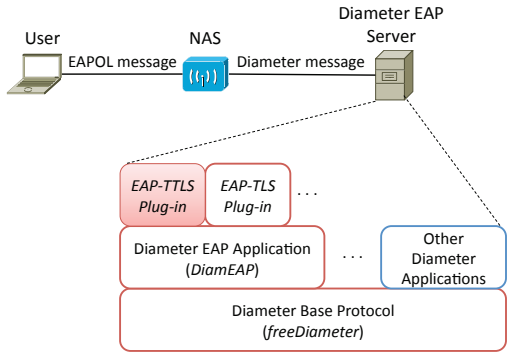

Figure 4: Position of EAP-TTLS Plug-in

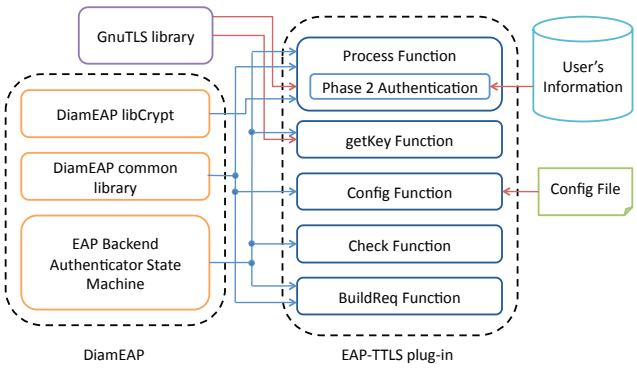

Figure 5: EAP-TTLS Plug-in Architecture

Sec. 4.2 describes the interface between the DiamEAP server and an EAP method plug-in. This section describes the details of each callback function from the viewpoint of EAP-TTLS Plug-in. Figure 5 shows the module diagram of EAP-TTLS Plug-in and Figure 6 shows the definitions of the callback functions.

There is an important data structure, the `eap_state_machine` structure. This structure is defined base on the EAP back end authenticator state machine[15] and used for exchanging various information between DiamEAP and EAP-TTLS Plug-in. It contains the user information registered with the user DB such as the user ID, the password, and the EAP method ID. It also contains a void pointer `methodData` which points to the `ttls_data` structure used in EAP-TTLS Plug-in.

The `ttls_data` structure is used in EAP-TTLS Plug-in and maintains the authentication session, the current phase, the authentication method, and the fragmentation information. The fragmentation information is used for fragmentation and reassembly of an EAP packet.

Config Function

The config function is called when the DiamEAP server starts up. As described before, in EAP-TTLS, the server

```
int eap_ttls_configure(char *configfile)

int eap_ttls_buildReq(struct eap_state_machine *smd,
        u8 id, struct eap_packet *eapPacket)

boolean eap_ttls_check(struct eap_state_machine *smd,
        struct eap_packet eapRespData)

int eap_ttls_process(struct eap_state_machine *smd,
        struct eap_packet eapRespData)

int eap_ttls_getKey(struct eap_state_machine *smd,
        u8 **msk, int *msklen, u8 **emsk,
        int *emsklen)
```

Figure 6: Definitions of callback functions in EAP-TTLS Plug-in

is authenticated with its certificate in phase 1 and the user is authenticated with the password in phase 2. In EAP-TTLS Plugin, the config function opens the EAP-TTLS configuration file and reads several parameters such as server's certificate, server's secret key, and CA's (Certificate Authority) certificate.

Check Function

When freeDiameter receives a Diameter message that contains the Diameter EAP Application AVPs, the Diameter message is passed to DiamEAP. Next, DiamEAP retrieves an EAP packet from the AVPs contained in the Diameter message. Hereafter, this procedure is described as "DiamEAP receives an EAP packet."

When DiamEAP receives an EAP packet, it calls *the check function*. The check function of EAP-TTLS Plug-in checks that the EAP packet is not broken and whether the type field contains the correct value that specifies EAP-TTLS (21).

For example, suppose that the EAP-TTLS server receives an EAP packet that contains three AVPs of MS-CHAPv2: the User Name AVP, the MS-CHAP Challenge AVP, and the MS-CHAPv2 Response AVP. These AVPs are encrypted by the TLS tunnel established in phase 1. The check function extracts just the type field by using a DiamEAP library function `diameap_eap_get_type()`. Then, the extracted type is checked whether the value is correct or not. If it is correct, the check function returns a boolean value `TRUE`.

Process Function

After the check function successfully finishes, *the process function* is called by DiamEAP. The process function of EAP-TTLS Plug-in is composed of three main functions: the reassembly function, the phase 1 function, and the phase 2 function. The reassembly function reassembles the received fragments into the original

EAP packet by checking the fragment flag in the fragment. If an EAP packet is fragmented, the fragment flag is set in each fragment except for the last fragment.

The phase 1 function and the phase 2 function use the *libCrypt* library in DiamEAP and the GnuTLS library. For EAP-TLS, DiamEAP provides a library for EAP-TLS handshake processing. EAP-TTLS Plug-in also uses this library in the phase 1 function. In phase 2, the received EAP packet is encrypted because it is transmitted through the TLS tunnel established in phase 1. The phase 2 function decrypts the EAP packet and performs the phase 2 authentication. When the phase 2 function sends an EAP packet, it encrypts the EAP packet.

For example, suppose that the process function is called after the previous example in "check function." To decrypt the EAP packet, the process function uses a DiamEAP library function `diameap_tls_record_receive()`. After that, to extract the AVPs from the EAP packt, `diameap_eap_ttls_listAVPs()` is used. Next, the process function decides which phase 2 authentication method should be used according to these AVPs and authenticates the user by the decided authentication method. The result of authentication is set in the `eap_state_machine` structure.

BuildReq Function

The buildReq function generates an EAP packet such as the EAP-TTLS-Start message and the EAP-TTLS-Ack message for a fragmented EAP packet. The data to be sent is generated based on the information in the `eap_state_machine` structure.

getKey Function

After the authentication processing successfully completes, the MSK (master session key) shared between the user and the access point must be generated. According to RFC 5281, the getKey function of EAP-TTLS Plug-in generates the MSK and the EMSK (extended MSK) by using GnuTLS library.

5.2 State Machine of the EAP-TTLS Plug-in

Figure 7 shows the state machine of EAP-TTLS Plug-in. In the initialization phase, EAP-TTLS Plug-in reads the configuration file and initializes the data for authentication session management, etc. After initialization, EAP-TTLS Plug-in moves to the phase 1 processing and waits for the first message from the user. When EAP-TTLS Plug-in receives the first message, it sends the TTLS-Start message to the user. After that, EAP-TTLS Plug-in waits for the EAP-Response message from the user. It continues the phase 1 processing until the Finished message is received. After that, EAP-TTLS Plug-in moves to the phase 2 processing. It sends the request message to the user and waits for the response message from the user. Upon receiving

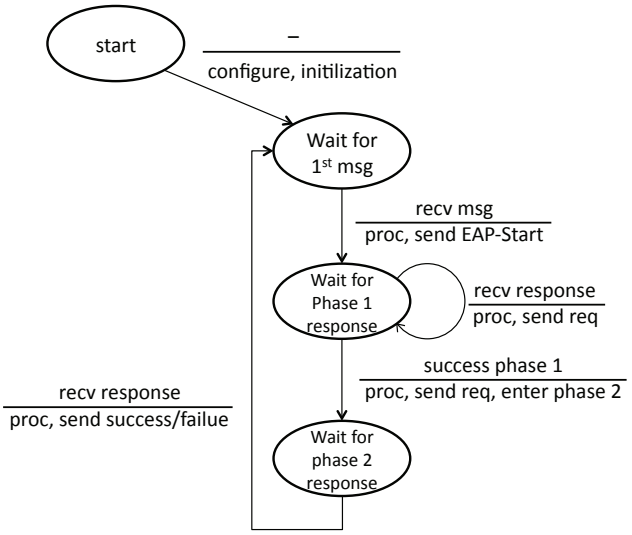

Figure 7: State Machine of EAP-TTLS Plug-in

Table 1: Supported phase 2 authentication methods and their AVPs

authentication method	contained AVPs
PAP	User Name User Password
CHAP	User Name CHAP Challenge CHAP Password
MS-CHAP	User Name MS-CHAP Challenge MS-CHAP Response
MS-CHAPv2	User Name MS-CHAP Challenge MS-CHAPv2 Response

the response message, EAP-TTLS Plug-in performs the phase 2 authentication processing and sends the result (success or failure) message to the user.

5.3 Selection of Phase 2 Authentication Method

RFC 5281 does not specify how the user and the EAP-TTLS server agree with the phase 2 authentication method. In our design, the phase 2 authentication method is selected by checking the AVPs contained in the EAP packets from the user in phase 2. Table 1 is the list of the supported authentication methods and the AVPs contained in each method. For example, if the EAP-TTLS server receives the User Name AVP, the MS-CHAP Challenge AVP, and MS-CHAPv2 Response AVP, EAP-TTLS Plug-in estimates that the user wants to use MS-CAHPv2. If there are lacked or excess AVPs in the received packets, EAP-TTLS Plug-in returns the EAP-Failure message.

5.4 Implementation of MS-CHAPv2

As mentioned before, EAP-TTLS Plug-in supports

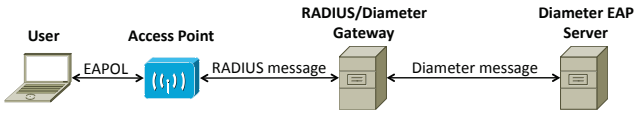

Figure 8: Evaluation environment

PAP, CHAP, MS-CHAP, and MS-CHAPv2. As an example, this section describes the implementation of MS-CHAPv2 in EAP-TTLS Plug-in. MS-CHAPv2 is a challenge handshake authentication protocol defined by Microsoft. The authentication process of MS-CHAPv2 is the same as that of CHAP; it compares the user name, the challenge, and the password hashed by the challenge. Both the user and the server severally generates the challenge. Since the challenge is generated by using the MSK (master session key) and random numbers exchanged in phase 1, they can generate the same challenge.

6. EVALUATION

This section shows the working test and the performance of the EAP-TTLS server. In the measurement, multiple operating systems are used as the user terminals. The result of the measurement is the average of 5 times tests.

6.1 Evaluation Environment

Figure 8 shows the evaluation environment. Several operating systems and devices such as Windows7, Linux, iPad, and Android are employed as the user terminal. These operating systems and devices originally support EAP-TTLS and are not modified for our test. The EAP-TTLS server is installed in a Linux machine. The NAS (Network Access Server) is an off-the-shelf WiFi access point. The user terminal sends the EAP packet to the NAS by EAPOL (EAP over LAN). The NAS retrieves the EAP packet from the EAPOL packet and encapsulates it in the RADIUS packet. Thus, since there is no off-the-shelf WiFi access point that supports Diameter, we employed the RADIUS/Diameter gateway that performs protocol conversion between RADIUS and Diameter. This gateway retrieves the EAP packet from the RADIUS packet and encapsulates it in the Diameter packet. Finally, the EAP packet reaches the EAP-TTLS server. In the reverse path, the EAP packet sent by the EAP-TTLS server is forwarded to the user terminal by the Diameter packet, the RADIUS packet, and the EAPOL packet. The specification of each machine is shown in Table 2.

6.2 Working Test

First, the sequence of the EAP-TTLS server is checked by capturing packets. As a result, it was made sure that the EAP-TTLS server authenticated the user in

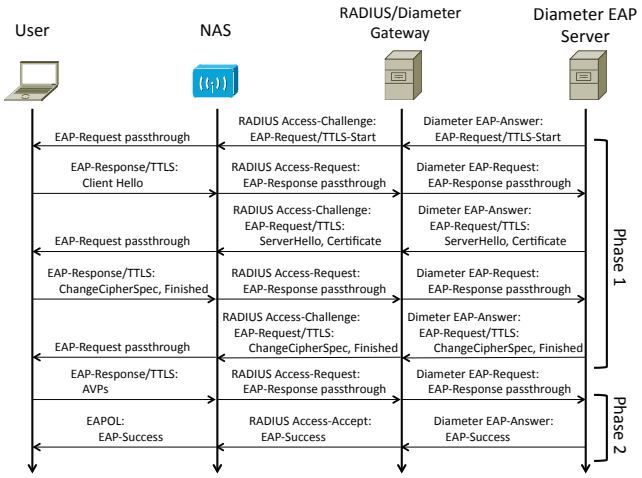

Figure 9: Sequence of evaluataion

Table 2: Machine specification

	machine	CPU	memory
user terminal	ThinkPad X200s	Intel Core2Duo 1.4GHz	4GB
	iPad	Apple A4 1GHz	256MB
NAS	Allide Telesis CentreCom 8724SL V2	–	–
RADIUS/ Diamter gateway	DELL OPTIPLEX 380	Intel Core2Quad 2.66GHz	4GB
EAP-TTLS server	DELL OPTIPLEX 380	Intel Core2Quad 2.66GHz	4GB

the manner of the specification and that each message contains the appropriate data. When the EAP-TTLS server receives an illegal user name or password, it rejects authentication and returns Access-Reject. If the NAS is configured as a DHCP server, the user terminal can obtain an IP address and access the Internet. Table 3 shows the tested operating systems and devices of the user terminal. It was made sure that all phase 2 authentication methods supported by the operating systems or devices worked correctly.

6.3 Measurement of Authentication Time

The time of entire EAP-TTLS authentication and the time of phase 2 authentication were measured. The entire time is from the time when the EAP-TTLS server sends EAP-TTLS-Start message to the time when the EAP-TTLS server sends EAP-Success message. In EAP-TTLS, each authentication method was measured. In addition, the time of EAP-TLS authentication was also measured for comparison.

As Table 4 shows, the time of the entire EAP-TTLS

Table 3: Confirmation of user terminals

phase 2 auth method	OS/device of user terminal			
	Windows 7	Linux	Android 2.1	iPad
PAP	OK	OK	not supported by Android	not supported by iPad
CHAP	OK	OK	not supported by Android	not supported by iPad
MS-CHAP	OK	OK	OK	not suppored by iPad
MS-CHAPv2	OK	OK	OK	OK

Table 4: Authentication Time

	Phase 2 Auth method	Total time (msec)	Phase 2 time (msec)
EAP-TTLS	PAP	16.39	0.184
	CHAP	16.06	0.217
	MS-CHAP	16.02	0.442
	MS-CHAPv2	15.83	0.436
EAP-TLS	-	29.08	-

authentication is about a half of that of the EAP-TLS authentication. Basically, authentication processing based on certificate takes much longer time than that based on password. The difference of authentication time of EAP-TTLS and that of EAP-TLS is caused by the number of times of authentication processing based on certificate; EAP-TTLS performs certificate base authentication once while EAP-TLS performs certificate base authentication twice.

The time of the phase 2 authentication depends on the authentication methods. The more complex the authentication method is, the longer the processing time is. However, the differences are negligible from the viewpoint of practical use.

7. CONCLUSION

As part of our universal AAA project, the first open source of EAP-TTLS is developed on Diameter EAP Application for network access control. In EAP-TTLS, the user authenticates the EAP-TTLS server by server's certificate while the EAP-TTLS server authenticates the user by the password transmitted through a TLS tunnel. Currently, several operating systems and devices support EAP-TTLS. Our EAP-TTLS server works correctly for several types of user terminals such as Windows7, Linux, Android, and iPad. The authentication time of the EAP-TTLS server is short enough for practical use.

8. REFERENCES

[1] eduroam home page. http://www.eduroam.org/.

[2] Security Assertion Markup Language (SAML) OASIS Standard. http://saml.xml.org/.

[3] Shibboleth home page. http://shibboleth.internet2.edu/.

[4] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP), June 2004. RFC 3748.

[5] S. B. Ayed and F. Teraoka. DiamEAP: an Open-Source Diameter EAP Application and Its Evaluation. In *Proceedings of the 16th Asia-Pacific Conference on Communications (APCC 2010)*, October-November 2010.

[6] F. Bernal, M. Sánchez, G. López, A. F. Gómez-Skarmeta, and O. Cánovas. Trusted Network Access Control in the eduroam federation. In *Proceedings of 2009 3rd International Conference on Network and System Security (NSS'09)*, pages 170–175, October 2009.

[7] F. Bersani and H. Tschofenig. The EAP-PSK Protocol: A Pre-Shared Key Extensible Authentication Protocol (EAP) Method, January 2007. RFC 4764.

[8] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol, September 2003. RFC 3588.

[9] S. Decugis and F. Teraoka. freeDiameter: An Open Source Framework for an Authentication, Authorization, and Accounting Infrastructure. *JSSST Computer Software*, 2011. (to appear).

[10] P. Eronen, T. Hiller, and G. Zorn. Diameter Extensible Authentication Protocol (EAP) Application, August 2005. RFC 4072.

[11] P. Funk and S. Blake-Wilson. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0), August 2008. RFC 5281.

[12] W. Jie, A. Young, J. Arshad, J. Finch, R. Procter, and A. Turner. A Guanxi Shibboleth based Security Infrastructure for e-Social Science. In *Proceedings of 2008 12th Enterprise Distributed Object Computing Conference Workshops*, pages 151–158, September 2008.

[13] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS), June 2000. RFC 2865.

[14] D. Simon, B. Aboba, and R. Hurst. The EAP-TLS Authentication Protocol, March 2008. RFC 5216.

[15] J. Vollbrecht, P. Eronen, N. Petroni, and Y. Ohba. State Machines for Extensible Authentication Protocol (EAP) Peer and Authenticator, August 2005. RFC 4137.