PAPER *Special Section on Architectures, Protocols, and Applications for the Future Internet*

A Multi-Domain Access Control Infrastructure Based on Diameter and EAP

Souheil BEN AYED[†a)], ***Student Member and*** **Fumio TERAOKA**[††b)], ***Member***

SUMMARY The evolution of Internet, the growth of Internet users and the new enabled technological capabilities place new requirements to form the Future Internet. Many features improvements and challenges were imposed to build a better Internet, including securing roaming of data and services over multiple administrative domains. In this research, we propose a multi-domain access control infrastructure to authenticate and authorize roaming users through the use of the Diameter protocol and EAP. The Diameter Protocol is a AAA protocol that solves the problems of previous AAA protocols such as RADIUS. The Diameter EAP Application is one of Diameter applications that extends the Diameter Base Protocol to support authentication using EAP. The contributions in this paper are: 1) first implementation of Diameter EAP Application, called *DiamEAP*, capable of practical authentication and authorization services in a multi-domain environment, 2) extensibility design capable of adding any new EAP methods, as loadable plugins, without modifying the main part, and 3) provision of EAP-TLS plugin as one of the most secure EAP methods. *DiamEAP* Server basic performances were evaluated and tested in a real multi-domain environment where 200 users attempted to access network using the EAP-TLS method during an event of 4 days. As evaluation results, the processing time of *DiamEAP* using the EAP-TLS plugin for authentication of 10 *requests* is about 20 *ms* while that for 400 *requests/second* is about 1.9 *second*. Evaluation and operation results show that *DiamEAP* is scalable and stable with the ability to handle more than 6 hundreds of authentication requests per second without any crashes. *DiamEAP* is supported by the AAA working group of the WIDE Project.

key words: *authentication and authorization infrastructure, diameter EAP application, multi-domain access control, roaming service*

1. Introduction

The current Internet is the most important communication network for information exchange connecting people all around the world. Nevertheless, with the continuous increase of Internet users, the emerging of services and the new technological advance several critical shortcomings discovered in terms of security, flexibility, mobility, and others leading to start many researches to develop a Future Internet. Additional challenges for the Future Internet have been identified including security/privacy/trust, mobility and availability. In addition, requirements for roaming access control and security are addressed particularly over multi-domains infrastructure.

The access control is a set of mechanisms and systems that enables an authority to control the access to resources and protects valuable data from prohibited access. An access control system is based on some processes essentially providing the functionalities of Authentication, Authorization and Accounting (AAA) in an environment for securing access to resources. Authentication is the process of identifying the requestor of the resource and validating whether the identity is what he/she claimed to be. Authorization process is granting the right that determines what the requestor can do on which resource or which service he/she can receive. Information of the user consumption of resources and services, such as the delivered service, the begin time and the end time of the service, are collected by the Accounting process. Popular AAA protocols, TACACS and RADIUS [1], are the most frequently and widely used. Theses two protocols had some problems and complications which have been improved in the new IETF AAA protocol called Diameter.

Diameter Base Protocol [2] was proposed to overcome the problems that raised with the previous AAA protocols and offering improvements in scalability, flexibility, security and reliability. In addition, Diameter is designed to provide access control in a multi-domain environment. Moreover, for supporting and be compatible with additional protocols and standards, it can be enhanced with additional functionalities through adding new Diameter applications. Recently, many standard bodies including 3GPP [3], 3GPP2 [4], MSF (the MultiService Forum) [5] and WimaxForum [6] have adopted the Diameter protocol as the AAA protocol.

In this paper, we propose *DiamEAP* [7], [8], a part of a project of the AAA working group of the WIDE project [9] to build a "Universal AAA Infrastructure" and design a AAA architecture for next-generation services in the Future Internet. DiamEAP provides easily applicable AAA functions to new services requiring AAA, aiming to become a foundation of AAA services in the Future Internet. *DiamEAP* provides authentication and authorization services for network access based on the Diameter EAP Application [10] over *freeDiameter* [11], an implementation of Diameter Base Protocol. As DiamEAP is the first implementation of the Diameter EAP Application, we design a new state machine interacting with the three state machines: Diameter Base Protocol, EAP, and an EAP method. In addition, DiamEAP architecture is designed to support pluggable components, EAP method plugins, that can be implemented separately.

The main contributions of this paper are: 1) design-

Manuscript received May 16, 2011.
Manuscript revised September 20, 2011.

[†]The author is with the Graduate School of Science and Technology, Keio University, Yokohama-shi, 223–8522 Japan.
[††]The author is with the Faculty of Science and Technology, Keio University, Yokohama-shi, 223–8522 Japan.
a) E-mail: souheil@tera.ics.keio.ac.jp
b) E-mail: tera@ics.keio.ac.jp

DOI: 10.1587/transinf.E95.D.503

ing and implementing the first open source Diameter EAP Application, DiamEAP, capable of practical authentication and authorization services in a multi-domain environment, 2) providing an extensible design able to support any EAP method by enabling loadable plugins called *EAP method plugins* without modifying the main DiamEAP Server, and 3) including EAP-TLS plugin as one of the most secured EAP methods. An evaluation of the basic performance of DiamEAP server is conducted. Further, we tested the scalability and stability of DiamEAP server in a real multi-domain environment.

The remainder of this paper is structured as follows. In Sect. 2, we give a brief overview of protocols and standards involved in the authentication process within a Diameter multi-domain environment. In Sect. 3, we discuss related work and projects. Then, we present the architecture of *DiamEAP* and the EAP plugins in Sect. 4. In Sect. 5, we describe DiamEAP implementation and the authorization service. In Sect. 6, we present the basic performance evaluation results, followed by the scalability and stability test operation conducted in a real multi-domain environment. Finally, in Sect. 7, we conclude the paper.

2. Overview of Diameter Protocol and Authentication Methods

2.1 Diameter Protocol

The Diameter protocol is an Authentication, Authorization and Accounting protocol that comes to overcome limitations and complications revealed with the previous AAA protocols such as RADIUS. It provides a general framework for future AAA architectures. The Diameter protocol is constituted by Diameter Base Protocol defined in [2] and additional Diameter applications. Diameter Base Protocol was standardized in IETF and focuses on general message exchanging. It provides the necessary and common functionalities for negotiation and accounting capabilities. However, it does not define authentication and authorization functionalities because these mechanisms vary among applications. Diameter Base Protocol can be extended by Diameter applications that provide additional functionalities. All Diameter applications are defined on top of Diameter Based Protocol and can benefit from its capabilities (Fig. 1). Various Diameter applications were defined and standardized in IETF such as for IP mobility [12], [13], network access [14] and Session Initiation Protocol (SIP) [15].

The Diameter protocol is a peer-to-peer protocol. Any Diameter peer can initiate a request and act either as a client node, a Diameter server or both. It can also act as an intermediate node in the network called the Diameter agent. The role of a Diameter peer depends on its tasks in the AAA network and the supported functionalities. Diameter peers exchange Diameter messages identified by their command codes. The Diameter message consists of a message header and a number of Attribute-Value Pairs (AVPs). The header identifies the type of the message and the Diameter application to which the message is applicable. AAA data as well as security and routing information are carried by AVPs. Diameter introduces agents to support a multi-domain environment that perform added value processing, reduce complexity of authentication networks, and simplify Diameter servers configurations and tasks. A Diameter agent can perform numerous tasks that can be divided into four kinds of agents (Fig. 2): 1) relay agents: used to forward Diameter requests to other Diameter peers based on some information included in the Diameter message such as the Destination Realm AVP. The relay agents may modify routing information of the message but are not allowed to make any change to other parts of the message, 2) proxy agents: are similar to the relay agents, in addition they may modify the Diameter message for providing added value service, enforce policies or perform administrative tasks. 3) redirect agents: are acting as a centralized configuration repository for Diameter nodes such for routing configuration. Redirect agents do not relay message; they only reply to requests with the necessary information and do not modify the message content, and 4) translation agents: are used to provide translation between two protocols such as between RADIUS and Diameter.

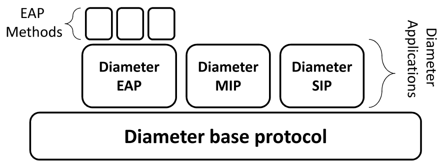


Fig. 1 Relationship between Diameter Base Protocol and Diameter Applications.

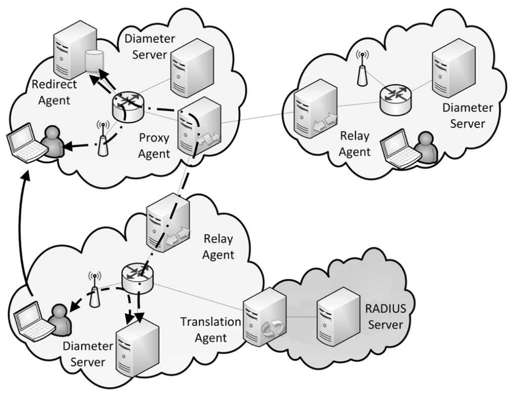


Fig. 2 Diameter Network and Diameter Agents (redirect agent, relay agent, proxy agent and translation agent).

2.2 Diameter EAP Application

Diameter EAP Application [10] is a Diameter application providing authentication capabilities based on Extensible Authentication Protocol (EAP) [16]. The application defines the protocol and Diameter messages for carrying authentication data between the Network Access Server (NAS), acting as the authenticator, and the back-end authentication server.

Diameter EAP Application is based on the Diameter Network Access Server Application [14], which defines a Diameter application providing AAA functionalities for network access service.

The authentication data is exchanged between a requestor and a Diameter authentication server. Basically, the exchanged data includes EAP-Payload and keying materials. In addition, it may accommodate authorization and accounting information. When the NAS receives a user authentication request, it creates and sends a Diameter-EAP-Request message to the Diameter authentication server. If the user is located in his/her home domain, Diameter peers forward the Diameter-EAP-Request to the local Diameter authentication server. Otherwise, Diameter peers, either a relay agent or a proxy agent, forward the message to the user's home domain where it will be handled by the Diameter authenticator server. The client and the Diameter EAP server continue exchanging messages until success or reject of the authentication request.

2.3 Extensible Authentication Protocol

The Extensible Authentication Protocol (EAP) is an authentication framework for execution of an authentication method. It transports authentication information and parameters between the two parties involved in the authentication process, 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 extremities of the authentication. EAP is not an authentication method; it provides necessary functions and defines message formats for supporting authentication method.

The EAP standard defined in [16] was designed to be extensible by adding new EAP methods without any change on the lower layer protocols. It supports various authentication methods called EAP methods such as EAP-MD5 [16], EAP-TLS [17], EAP-TTLS [18], EAP-PSK [19], and many others. Most of them are defined by IETF and some others are vendor methods or proposed methods. Many standards support EAP authentication methods, such as IEEE 802.11, WPA and WPA2 which have adopted EAP as the authentication protocol in addition to a number of authentication methods.

2.3.1 EAP-MD5

EAP-MD5 is a password based authentication method defined in [16]. The user is authenticated by verifying the MD5 hash of his/her password. The method does not support key generation and does not provide mutual authentication, only the user is authenticated to the server. Typically, EAP-MD5 is used on trusted and low risk networks, and it is deprecated on public networks or wireless networks due to its vulnerability to dictionary and man-in-the-middle attacks.

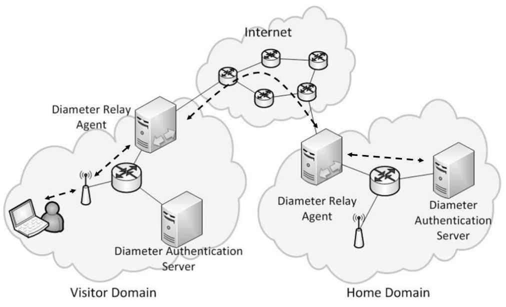


Fig. 3 Access control in diameter multi-domain environment.

2.3.2 EAP-Transport Layer Security

EAP-Transport Layer Security (EAP-TLS) defined in [17] is a PKI (Public Key Infrastructure) based EAP method based on the Transport Layer Security (TLS) [20], a cryptographic protocol defined in IETF, providing a secure communication across a network. EAP-TLS uses the handshake procedure in the TLS protocol to provide mutual authentication. Upon success of the handshake, both the client and the server generate the key material, used for encryption and decryption of messages, to establish a secure connection. EAP-TLS is considered as one of the most secure authentication methods. It is supported mostly by all recent versions of operating systems and all wireless LAN equipments and devices.

2.4 Diameter Protocol in Multi-Domain Environment

Administrative domains are interconnected via the Internet. However, due to lack of security and reliability the previous AAA protocols do not have support for inter-domain AAA applications. The Diameter protocol is designed to handle issues that arose with these previous AAA protocols. With its various features, the Diameter protocol explicitly supports a multi-domain environment.

With multi-domain support, the Diameter protocol allows communication and exchange of authentication and authorization information between two different administrative domains. Hence, it provides roaming users with the ability to access network while moving to a visitor domain. For example, when a user wants access to a service provided by a visitor domain, first he/she sends a request to the visitor domain (Fig. 3). The visitor domain starts authenticating the user and forwards the authentication messages to the home domain. The Diameter authentication server at the home domain will handle authentication messages and exchange data with the user. All the authentication messages are exchanged between the two domains through the Diameter Relay agents. After successfully authenticating the user, the Diameter authentication server at the home domain sends user's authentication and authorization attributes to the visitor domain. Based on received attributes, the visitor domain decides to allow or refuse the user to access the service.

3. Related Work

Several access control architectures have been proposed for resource sharing. In these architectures, trust relationships should be established among organizations for authenticating users and granting access to shared resources based on their credentials stated by the organization to which the user belongs. However, adopting authorization decision to access the organization resources is an extremely critical process that should precisely provide the strictly required and enough privileges to users for performing the allowed actions on each of the accessed resources. DiamEAP offers authentication and authorization services for controlling access in a multi-domain environment. Besides, DiamEAP provides the way to support any EAP authentication method as an EAP method plugins.

3.1 Eduroam

Eduraom (**edu**cation **roam**ing) [21] 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. Some solutions were proposed to extend Eduroam with authorization mechanisms [22], [23].

3.2 Shibboleth

Shibboleth [24] is an Internet2 federated identity management middleware focusing on educational institutions. It is an open-source project based on Security Assertion Markup Language (SAML) [25], 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) and designed to offer authentication and authorization 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.

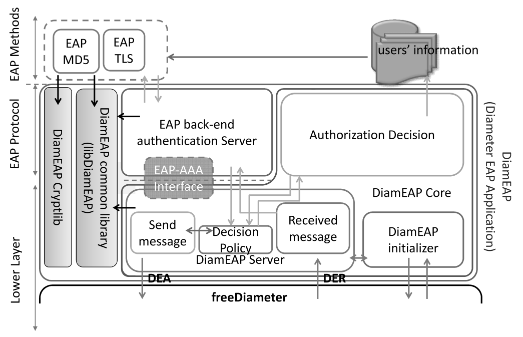


Fig. 4 *DiamEAP* Architecture.

4. Design of DiamEAP and Plugins

4.1 DiamEAP Architecture

As we mentioned above, DiamEAP is a project based on freeDiameter, an open-source implementation of Diameter Base Protocol. DiamEAP is designed as an extension for freeDiameter, implementing the Diameter EAP Application, EAP and authentication methods for authenticating and authorizing a client in a network access environment. Since DiamEAP is interacting and exchanging information, data and signal with the lower and upper layers, Diameter Base Protocol and EAP, in designing DiamEAP we take into account to separate between layers and decompose the DiamEAP into components. The different components and layers communicate and exchange data using some defined interfaces. In addition, two libraries, *libDiamEAP* and *Cryptlib*, are designed to provide common DiamEAP functionalities, cryptography APIs and functionalities that can be used by all the DiamEAP components and EAP methods. Besides, in the way to be extensible and flexible with the variety of EAP methods, *EAP method plugins* are introduced to support any EAP methods.

As shown in Fig. 4, the structure of DiamEAP is principally composed of five components:

- *DiamEAP core* is considered as the main component in DiamEAP. It includes two parts:
  - *The DiamEAP initializer* which is called by *freeDiameter* to initialize the Diameter EAP server, load, configure and start the extension. In addition, it loads and initializes all the configured EAP method plugins to be supported during the authentication process.
  - *The DiamEAP server* is responsible for managing Diameter EAP messages. It is in charge of creating and maintaining authentication sessions, parsing AVPs, checking attributes, delivering EAP information for the authentication process, verifying authorization, making decision based on the authentication and the authorization results, generating answer attributes and sending the Diameter-EAP-Answer.

- *EAP back-end authentication server* composes the authenticator part of the DiamEAP. It includes the EAP process in addition to the management of EAP packets, handling the communication with the EAP method plugins, and the authentication EAP methods.
- *EAP method plugins* are authentication method plugins built on the top of of DiamEAP. Each plugin represents an authentication method that is loaded by DiamEAP and can be invoked by the EAP Back-End Authentication Server during the authentication process. All EAP method plugins are loaded and configured by the DiamEAP initializer. Any loaded EAP method plugin can be initiated for as much sessions as it is required by DiamEAP. Furthermore, it can be invoked simultaneously by two or more separated authentication sessions.
- *Two libraries* are provided with DiamEAP: a common library *libDiamEAP* and *Cryptlib*. Libraries are shared and may be used by any DiamEAP component including plugins.
  - libDiamEAP provides the tools for manipulating EAP packets and related data, accessing user's information database and managing the data structure of user's authentication.
  - Cryptlib includes functionalities for handling and managing the TLS mechanism for EAP-TLS over the EAP layer, besides other cryptographic and hash functions.
- *The user's information Database* stores user's and group's accounts, in addition to policies and rules for the authorization and the authentication processes.

4.2 DiamEAP State Machine

The DiamEAP core represents the central component in the DiamEAP architecture (Fig. 4). It manages Diameter messages and exchanges information among the DiamEAP components. As far as we know, there is no design or implementation of the Diameter EAP Application. Therefore, in order to describe the various behaviors of the system, we propose a new state machine for the DiamEAP server (Fig. 5) with respect to the objectives of the designed DiamEAP architecture. In this design, we considered to satisfy the requirements for the interaction with the other layers state machines: Diameter Base protocol, EAP, and EAP method. This state machine is designed based on the conventions specified in [26], Annex C. In addition, we redesign the state machine of EAP back-end authenticator based on that described in [27]. We updated the EAP state machine to fit the DiamEAP architecture such as to support dynamically loadable EAP method plugins and to take into account of interactions with the EAP method plugins and the Diameter EAP server.

All received Diameter-EAP-Requests are handled by the DiamEAP core. For each new EAP authentication request, a new state machine is created and initialized for managing the authentication session and its attributes. This session is maintained until success, reject or a not achieved request for one of these reasons: the round trip timeout elapsed with no response from the client or the reception of multiple invalid EAP packets. When receiving a new Diameter-EAP-Request, the message is firstly parsed for checking the AVPs contents and retrieving the required attributes for authentication and authorization processes. In addition, before proceeding to the authentication and according to the current state machine, either *INITIALIZE* or *RECEIVED*, the DiamEAP server updates the session parameters with the new message attributes and initiates the DiamEAP–EAP interface with the response authentication information.

Typically, all Diameter-EAP-Requests for Diameter EAP Application are for authenticating the user. In addition, if authorization is requested, Diameter EAP Application can check the user authorization to access the requested service. The DiamEAP server forwards all received requests to the back-end authenticator for authenticating the user, then decides whether to check the user authorization or not. The authorization is checked only if the user is authenticated successfully. In the *AUTHENTICATE* state, one or more *EAP method plugins* may be invoked by the back-end authenticator. Based on the authentication result and the request attributes, the DiamEAP server makes decision on the request as success, reject, continue or waiting for checking the authorization.

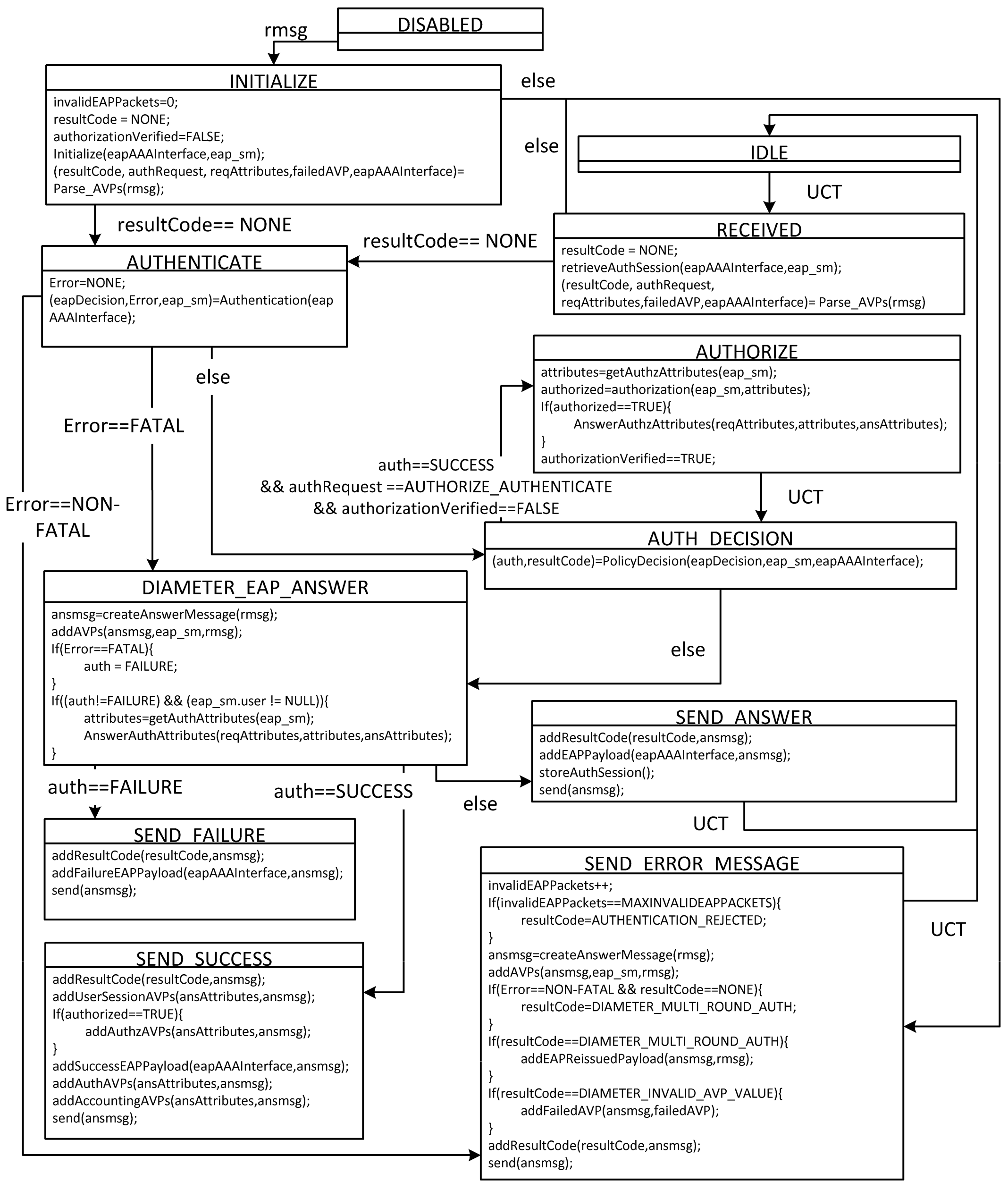
Once decision is made, the Diameter-EAP-Answer is prepared with the appropriate AVPs and attributes. In addition, based on the DiamEAP authentication and authorization decision, Success or Failure AVPs are added for success and rejected authentication, respectively. If the EAP method requires additional exchanges for authenticating the user, an EAP-Payload AVP encapsulating the EAP packet is created and included in the Diameter-EAP-Answer. The state machine passes to *IDLE* waiting for a Diameter-EAP-Request including an EAP response packet.

Upon reception of a response from the client, the DiamEAP server parses the Diameter message, initiates the required attributes and prepares the parameters and interfaces for the authentication process.

4.3 Dynamically Pluggable EAP Method Plugins

In DiamEAP, EAP methods are defined as plugins that can be implemented and integrated for authenticating clients without the need to any modification in DiamEAP. Plugins are dynamically loaded and configured prepared for any authentication session. All *EAP Method Plugins* (Fig. 6) are enabled and unregistered by DiamEAP. When enabled, a plugin is firstly configured and initiated for accepting authentication requests. Each plugin defines its functions and objects that can be invoked externally by the DiamEAP or the back-end authenticator as required for the authentication process and/or the use of the authentication mechanism.

After negotiating EAP methods with the client, the



```
INITIALIZE
invalidEAPPackets=0;
resultCode = NONE;
authorizationVerified=FALSE;
Initialize(eapAAAInterface,eap_sm);
(resultCode, authRequest, reqAttributes,failedAVP,eapAAAInterface)=
Parse_AVPs(rmsg);
```

```
RECEIVED
resultCode = NONE;
retrieveAuthSession(eapAAAInterface,eap_sm);
(resultCode, authRequest,
reqAttributes,failedAVP,eapAAAInterface)= Parse_AVPs(rmsg)
```

```
AUTHENTICATE
Error=NONE;
(eapDecision,Error,eap_sm)=Authentication(eap
AAAInterface);
```

```
AUTHORIZE
attributes=getAuthzAttributes(eap_sm);
authorized=authorization(eap_sm,attributes);
If(authorized==TRUE){
        AnswerAuthzAttributes(reqAttributes,attributes,ansAttributes);
}
authorizationVerified==TRUE;
```

```
AUTH_DECISION
(auth,resultCode)=PolicyDecision(eapDecision,eap_sm,eapAAAInterface);
```

```
DIAMETER_EAP_ANSWER
ansmsg=createAnswerMessage(rmsg);
addAVPs(ansmsg,eap_sm,rmsg);
If(Error==FATAL){
        auth = FAILURE;
}
If((auth!=FAILURE) && (eap_sm.user != NULL)){
        attributes=getAuthAttributes(eap_sm);
        AnswerAuthAttributes(reqAttributes,attributes,ansAttributes);
}
```

```
SEND_ANSWER
addResultCode(resultCode,ansmsg);
addEAPPayload(eapAAAInterface,ansmsg);
storeAuthSession();
send(ansmsg);
```

```
SEND_FAILURE
addResultCode(resultCode,ansmsg);
addFailureEAPPayload(eapAAAInterface,ansmsg);
send(ansmsg);
```

```
SEND_ERROR_MESSAGE
invalidEAPPackets++;
If(invalidEAPPackets==MAXINVALIDEAPPACKETS){
        resultCode=AUTHENTICATION_REJECTED;
}
ansmsg=createAnswerMessage(rmsg);
addAVPs(ansmsg,eap_sm,rmsg);
If(Error==NON-FATAL && resultCode==NONE){
        resultCode=DIAMETER_MULTI_ROUND_AUTH;
}
If(resultCode==DIAMETER_MULTI_ROUND_AUTH){
        addEAPReissuedPayload(ansmsg,rmsg);
}
If(resultCode==DIAMETER_INVALID_AVP_VALUE){
        addFailedAVP(ansmsg,failedAVP);
}
addResultCode(resultCode,ansmsg);
send(ansmsg);
```

```
SEND_SUCCESS
addResultCode(resultCode,ansmsg);
addUserSessionAVPs(ansAttributes,ansmsg);
If(authorized==TRUE){
        addAuthzAVPs(ansAttributes,ansmsg);
}
addSuccessEAPPayload(eapAAAInterface,ansmsg);
addAuthAVPs(ansAttributes,ansmsg);
addAccountingAVPs(ansAttributes,ansmsg);
send(ansmsg);
```

Fig. 5 DiamEAP state machine.

back-end authenticator selects the plugin associated to the approved authentication method. Once selected, the plugin initiates a new session for this authentication request and builds the first request to start exchanging authentication information. All plugins should provide the necessary functions and objects for initiating an authentication session, build request, checking answer and performing the EAP method algorithms. Moreover, additional optional functions and objects are available for the EAP methods that require them such as for generating the keying materials. The plugin session is closed after a success or a rejected authentication.

4.4 Examples of EAP Method Plugins

It is primordial to provide at least one EAP method to DiamEAP in order to test and validate the authentication part and the functionalities of the different components, as well as to provide a full solution for assessing DiamEAP and its authentication and authorization functionalities. Hence, we opted to design and implement two EAP method plugins: EAP-MD5 and EAP-TLS. EAP-MD5 is a password based EAP method that offers a minimal security and only one way authentication. It was important to start with designing a basic authentication plugin such as EAP-MD5 to validate the dynamically pluggable mechanism for loading and invoking

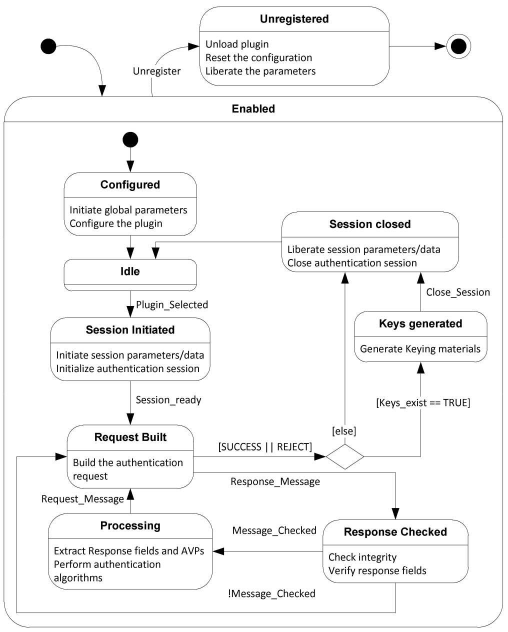


Fig. 6 DiamEAP plugin states.

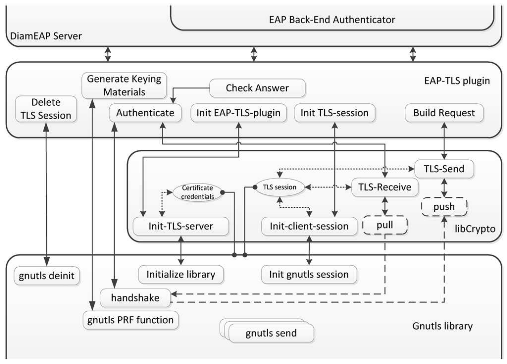


Fig. 7 EAP-TLS plugin.

the authentication mechanism. Then, we decided to provide the EAP-TLS authentication method which is considered as one of the most secure authentication methods. In addition, it is supported by mainly most of the operating systems and network authentication devices. Further EAP method plugins are under design and/or validation such as EAP-TTLS (PAP/CHAP/MS-CHAP and /MS-CHAP2).

The structure of the EAP-TLS plugin and exchanges with external components are designed as shown in Fig. 7. Basically, the EAP-TLS authentication method is based on the handshake phase of TLS for authenticating users. After a success authentication, it generates the keying materials for establishing the secure channel. The plugin provides all functions and objects required by a plugin for loading, configuring and making use of the associated EAP method. As EAP-TLS method may be required by other plugins, we separated the TLS component part from the EAP-TLS plugin and included it to the CryptLib library accessible by all DiamEAP components and plugins. The EAP-TLS plugin focuses particularly on the EAP-TLS layer including checking the message content, building the answer messages and making authentication decision based on the TLS data. In addition, it generates the keying materials and communicates the required information to the EAP layer. The TLS component, making use of the GNUTLS library, establishes the TLS session and manages the TLS exchanged data during the handshake.

5. DiamEAP Implementation and Authorization Policies

5.1 DiamEAP and the EAP Plugins

DiamEAP is implemented as an extension for freeDiameter. It is written in C programming language. The DiamEAP extension is loaded and initialized by the *freeDiameter daemon*. During initialization, DiamEAP advertises the support of Diameter EAP Application and registers a callback for handling Diameter-EAP messages. Besides, it specifies, using the mechanism of "*dynamic linking*", the registered functions and objects for invoking plugins. All messages are parsed by DiamEAP for checking AVPs' validity and storing attributes required by the authentication or the authorization processes. These attributes are maintained in a DiamEAP session until either success, reject or expiration of the authentication.

In order to communicate and exchange data between the different components including DiamEAP, EAP and EAP plugins, DiamEAP server defines data structure interfaces. These interfaces vary from component to other and may contain *EAP method plugin* data for maintaining authentication information.

As we mentioned above, the EAP-TLS plugin is designed to be separated in two parts: the TLS functionalities and the plugin for handling the EAP-TLS packets.

The TLS authentication method functions and data structures are defined in the Cryptlib. These TLS functions make use of the GNUTLS library API, a portable ANSI C based library implementing the TLS protocol and offer a framework for authentication and public key infrastructure [28]. Since TLS data is included in an EAP packet, we implemented the *gnutls transport set push* and *pull* functions to handle the received TLS data and prepare the answer TLS data to be sent.

The EAP-TLS plugin defines the data structures for maintaining the authentication state and parameters. It also provides the functions and objects invokable externally for loading the plugin and performing the authentication mechanisms and algorithms for authenticating users.

5.2 Authorization Service

As mentioned in Sect. 2.2, Diameter EAP Application is mainly focusing on the authentication of the user and providing the user session parameters for accessing the service. Besides, this Diameter application may accommodate

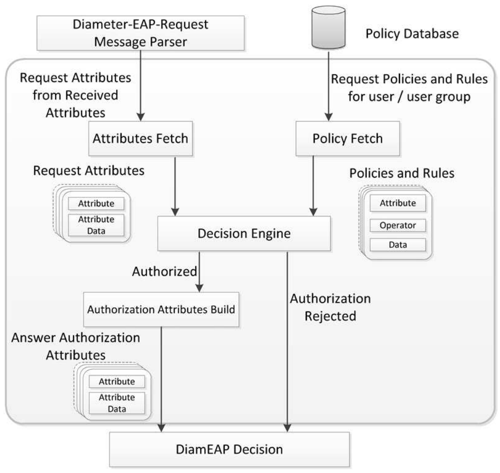


Fig. 8 Authorization Decision Making Process.

a success authentication with the appropriate authorization attributes for the service requested.

The authorization service in DiamEAP consists of a decision-making service to determine whether users have the permitted authority to access services and which privileged are to be granted. The access control authorization is implemented by a simple mechanism able to enforce multiple policies through the introduction of new operators for a flexible authorization. The authorization decision is made based on 1) a set of fine-grained policies, rules and pre-configured parameters defined by the administrator 2) and the user proposed attributes provided with the authorization request (Fig. 8).

The access control policies are based on authorization rules that defines the conditions to meet before being granted access, e.g. "*The user identified by **A** is authorized to Access Network/Service **B** where rule **C** applies.*"

This authorization service provides a simple security model for the formal representation of access control policies and rules. Although it is simple, it allows flexible configuration of rules and conditions.

6. Evaluations and Tests

In this section, we evaluate the basic performances of DiamEAP and its components, then we present the results of the test operation of the DiamEAP server conducted in a real environment. Firstly, we describe our experiment test-bed and the evaluation parameters configured for both the freeDiameter and the DiamEAP server. Then, we present and discuss the results of the experiments. In the second part of this section, we describe and present the result graph of the test operation conducted during the WIDE camp of the WIDE project, where DiamEAP was established and configured as the main authentication server.

6.1 Evaluation of the Basic Performance

In order to evaluate the basic performance of DiamEAP under an environment similar to the real authentication environment, we developed a DiamEAP client: a freeDiameter extension working as the client side for the DiamEAP server in order to generate and send Diameter-EAP-Request of an authentication request. This DiamEAP client also includes a client side of the EAP-TLS plugins for handling EAP-TLS authentication packets and returns the corresponded EAP-TLS answers. Both the DiamEAP client and the EAP-TLS plugin are fully implemented and compatible with EAP and EAP-TLS standards. The DiamEAP client is implemented only for the evaluation purpose.

6.1.1 Evaluation Environment

In this evaluation, the test-bed architecture consists of three Diameter peers: one Diameter EAP server and two Diameter peers acting as NAS servers. All nodes are equipped with an Intel Core 2 Quad processor 2.66 GHz and 4 GB of memory, and running a Linux operating system, Ubuntu-Server, with kernel 2.6.32. We configure the servers with the following parameters:

- SCTP Transport protocol enabled
- 3 streams per SCTP association
- 200 server threads to handle incoming messages at the same time
- DiamEAP extension Enabled (for the two freeDiameter peers)
- NASReq and EAP dictionaries enabled

During the evaluation, the two Diameter peers generate a variable numbers of authentication requests to transmit 10, 50, 100, 150, 200, 250, 300, 350, 400, 450 or 500 requests every one second. Each flow of these authentication requests is conducted separately during a period of 1,000 seconds. Except the maximum CPU results, all other final results are averaged. We fixed the server threads to 200 for the three freeDiameter servers and that for the different flows of requests.

6.1.2 Results

Figure 9 shows the result of the experiment to evaluate the impact of the number of requests per second on the time for processing the authentication by freeDiameter and DiamEAP including EAP component and plugins. We observe that the time for processing the authentications increases as the number of the requests rises. We also note that for the flow 250 *requests/second*, the processing time for freeDiameter exceeds more rapidly than that of DiamEAP. This behavior is due to the server threads parameter of freeDiameter which represents the reserved queues for receiving the requests. However, the DiamEAP processing time was not influenced by this parameter and it increases linearly with the number of requests, reaching an authentication time of DiamEAP server for 500 *requests/second* less than 2.4 *seconds*.

We also examine the average consumption of CPU and

Table 1 The processing time in millisecond of the DiamEAP Server, the EAP back-end authenticator and the EAP-TLS plugin for different authentication requests traffic per second.

Authentication Request/Second	10	50	100	150	200	250	300	350	400	450	500
DiamEAP Server	0.348	0.389	0.512	0.750	0.981	1.230	1.476	1.749	2.459	2.890	3.467
EAP Back-End Authenticator	1.148	3.672	5.291	7.347	9.543	11.832	13.503	15.240	17.127	19.432	21.457
EAP-TLS plugin	18.147	171.057	376.825	631.866	923.544	1223.150	1412.76	1630.265	1894.192	2183.566	2342.134

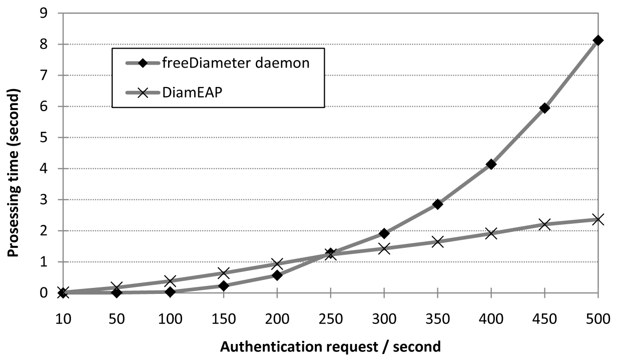


Fig. 9 FreeDiameter and DiamEAP time processing vs. Authentication request traffic.

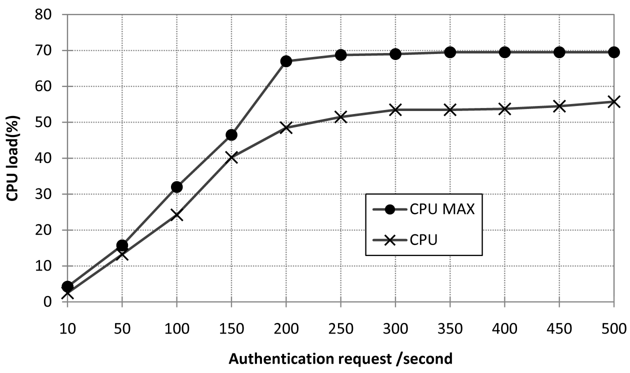


Fig. 10 Average CPU consumption and Maximum CPU consumption vs. Authentication request traffic.

the maximum of CPU reached by the authentication server, including freeDiameter, DiamEAP and plugins while increasing the number of requests. Figure 10 shows the resulting CPU loads. We denote an increase of the average CPU consumption and the maximum CPU reached for flows $\leq$ 200 *requests/second*, followed by a slightly growing average CPU consumption around the 56% and not exceeding the limit of 69%.

Next, we explore the processing time of the three layers of DiamEAP including the DiamEAP server, the EAP protocol component and the EAP-TLS plugin. The results reveal that more than 90% of the processing time of the authentication requests was during the treatment by the EAP-TLS plugin (see Table 1). However, the DiamEAP server and the back-end authenticator components are not exceeding the threshold of 11 *ms* for 200 *requests/second* and the 25 *ms* for 500 *requests/second*. We assume that this difference may due to the complexity of the cryptograph algorithms of the EAP-TLS plugin.

Table 2 Lifetime values of the authentication session for the 4 days of the experiment.

Day	Session Lifetime
The first day	1 hour
The second day	30 minutes
The third day	10 minutes
The fourth day	10 minutes

6.2 Evaluation of Scalability and Stability

We had the opportunity to test the DiamEAP server in a real environment within a AAA architecture during the WIDE camp of the WIDE project. In this environment, we built two network domains and deployed two Diameter EAP servers, one in each domain. Although, despite the significant advantages and improvements of the Diameter protocol compared to its predecessor the RADIUS protocol, until the conduction of this test operation, we were unable to find any access point supporting the Diameter protocol. Therefore, we opted to make use of the Radius Gateway extension of the freeDiameter project to build two Diameter translation agents for translating authentication messages between the Diameter protocol and the RADIUS protocol. The two agents were configured to trust the other Diameter peers and to forward messages to the appropriate authentication server based on the user identification. Thus, the users were able to be authenticated and accessing network even when moving to the visitor domain. We configured the authorization policies for group of users to define authorization attributes such as for session parameters.

The operation was conducted during the 4 days of the event, even during night. The 200 attendees of the WIDE camp were able to access network after being authenticated via the EAP-TLS authentication mechanism using their valid digital certificates without system crash or errors. All users had a valid digital certificate and were able to access the Internet through EAP-TLS authentication mechanism via different client devices.

In order to visualize the stability of DiamEAP and the EAP-TLS plugin, we increase the number of requests received by DiamEAP server and its plugins. Therefore, we varied the session lifetime parameter for all success authentication according to Table 2. As shown by Fig. 11, we noticed that DiamEAP was able to handle authentication requests exceeding 400 *authentication request/second* without observing any errors or malfunction from the DiamEAP

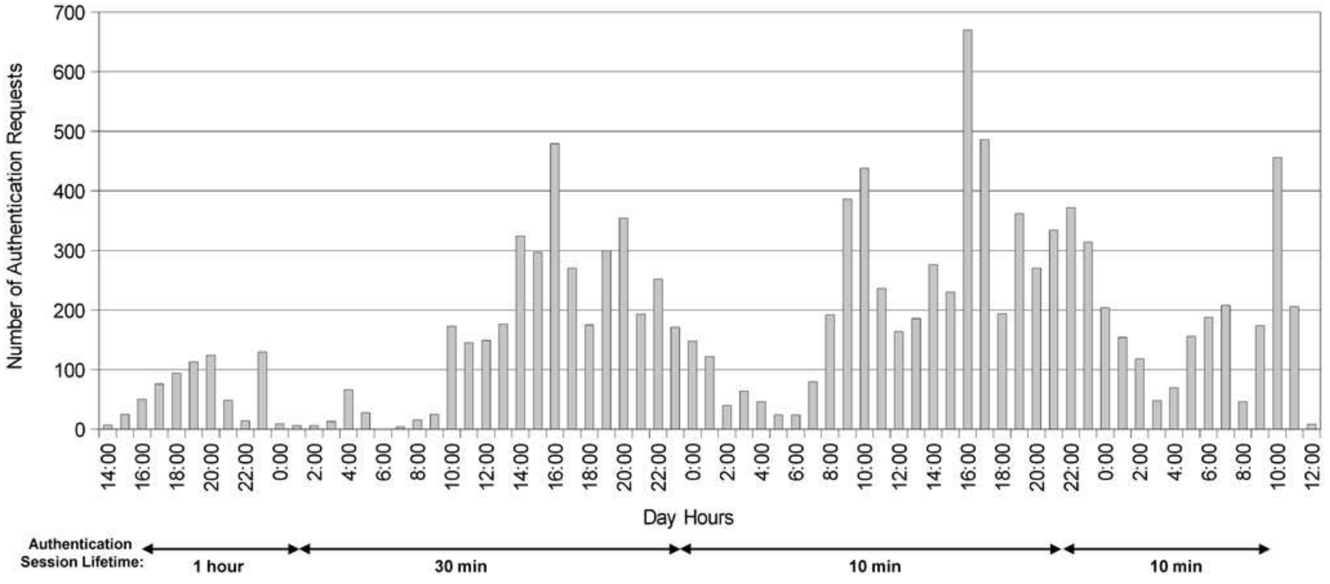


Fig. 11 Test operation of DiamEAP Server: Number of the authentication requests per hour during the 4 days of the experiment. The lifetime of the authentication session varies from 1 hour for the first day, 30 minutes for the second day to 10 minutes for the third and the fourth days.

server or the EAP-TLS plugin. Even with reducing the session timeout, which increases the number of requests for re-authentication, DiamEAP was enough stable that clients did not noticed any discontinuity of network connection.

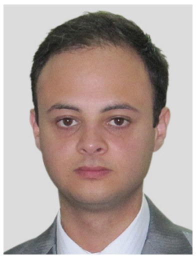
7. Conclusion

With features improvements in terms of mobility, security and flexibility, Future Internet will enable a multitude of new services and applications for a multi-domain environment. For the Future Internet services, AAA functions are important for protecting remote access to resources and services. In this paper, we proposed an open-source Diameter EAP Server for building a multi-domain access control infrastructure within a AAA environment. DiamEAP provides a new state machine for the Diameter EAP application for the interaction with other layers. It provides a pluggable architecture design to support any EAP method plugin. EAP-TLS plugin is also provided for practical operation. DiamEAP provides a fine-grained authorization service in addition to the authentication service. Performance evaluation showed a large variation in the processing time among *freeDiameter*, *DiamEAP* server and the *DiamEAP method plugins*. We had the opportunity to test and validate our implementation of DiamEAP and to experience it in a real multi-domain environment. Results showed that DiamEAP was enough stable and scalable with handling more than 6 hundreds of authentication requests without any crashes or errors. With DiamEAP we intend to build a foundation of AAA services in the Future Internet.

References

[1] C. Rigney, S. Willens, A. Rubens, and W. Simpson, “Remote authentication dial In user service (RADIUS),” RFC 2865, June 2000.
[2] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko, “Diameter base protocol,” RFC 3588, Sept. 2003.
[3] The 3rd Generation Partnership Project. http://www.3gpp.org/
[4] The 3rd Generation Partnership Project 2. http://www.3gpp2.com/
[5] The MultiService Forum. http://www.msforum.org/
[6] WiMAX Forum. http://www.wimaxforum.org/
[7] DiamEAP. http://diameap.yagami.freediameter.net/
[8] S. Ben Ayed and F. Teraoka, “DiamEAP: an open-source diameter EAP application and its evaluation,” Proc. 16th Asia-Pacific Conference on Communication (APCC 2010), pp.515–520, Oct.-Nov. 2010.
[9] The WIDE Project. http://www.wide.ad.jp/
[10] P. Eronen, T. Hiller, and G. Zorn, “Diameter extensible authentication protocol (EAP) Application,” RFC 4072, Aug. 2005.
[11] freeDiameter. http://www.freediameter.net/
[12] P. Calhoun, T. Johansson, C. Perkins, T. Hiller, and P.J. McCann, “Diameter mobile IPv4 application,” RFC 4004, Aug. 2005.
[13] J. Korhonen, J. Bournelle, H. Tschofenig, C. Perkins, and K. Chowdhury, “Diameter mobile IPv6: Support for network access server to diameter server Interaction,” RFC 5447, Feb. 2009.
[14] P. Calhoun, G. Zorn, D. Spence, and D. Mitton, “Diameter network access server application,” RFC 4005, Aug. 2005.
[15] M. Garcia-Martin, M. Belinchon, M. Pallares-Lopez, C. Canales-Valenzuela, and K. Tammi, “Diameter session initiation protocol (SIP) application,” RFC 4740, Nov. 2006.
[16] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and E.H. Levkowetz, “Extensible authentication protocol (EAP),” RFC 3748, June 2004.
[17] D. Simon, B. Aboba, and R. Hurst, “The EAP-TLS authentication protocol,” RFC 5216, March 2008.
[18] P. Funk and S. Blake-Wilson, “Extensible authentication protocol tunneled transport layer security authenticated protocol Version 0,” RFC 5281, Aug. 2008.
[19] F. Bersani and H. Tschofenig, “The EAP-PSK protocol: A pre-shared key extensible authentication protocol (EAP) method,” RFC 4764, Jan. 2007.
[20] T. Dierks and E. Rescorla, “The transport layer security (TLS) protocol version 1.2,” RFC 5246, Aug. 2008.
[21] eduroam (education roaming). http://www.eduroam.org/
[22] G. López, ı. Cánovas, A.F. Gómez-Skarmeta, and M. Sánchez, “A proposal for extending the eduroam infrastructure with authorization

mechanisms," Comput. Stand. Interfaces, vol.30, pp.418–423, Aug. 2008.
[23] M.S. Cuenca, G. López, Ó.C. Reverte, and A.F. Gómez-Skarmeta, "A proposal for extending the eduroam infrastructure with authorization mechanisms," Security in Information Systems, Proc. 5th International Workshop on Security in Information Systems, WOSIS 2007, In conjunction with ICEIS 2007, Funchal, Madeira, Portugal, June 2007, ed. M.I.Y. del Valle and E. Fernández-Medina, pp.23–32, INSTICC Press, 2007.
[24] Shibboleth. http://shibboleth.internet2.edu/
[25] Security Assertion Markup Language (SAML) OASIS Standard. http://saml.xml.org/
[26] "Standard for local and metropolitan area networks: port-based network access control," IEEE 802.1X-2010, Feb. 2010. Institute of Electrical and Electronics Engineers.
[27] J. Vollbrecht, P. Eronen, N. Petroni, and Y. Ohba, "State machines for extensible authentication protocol (EAP) peer and authenticator," RFC 4137, Aug. 2005.
[28] GNUTLS Library. http://www.gnu.org/software/gnutls/



Souheil Ben Ayed received Eng. degree - the Department of Computer Science, INSAT - and MSc degree in Telecommunication from Carthage University of Tunisia, in 2005 and 2008, respectively. From October 2009 to the present, he has been working toward the PhD degree in computer science at Keio University, Japan. His work is focused on the access control on multi-domain distributed environments a part of a WIDE Project-AAA working group to build a "Universal AAA Infrastructure". From 2005 to 2006, he has worked as an engineer at Actielec-ARDIA, at Fujitsu Siemens Tunisia on 2007 then moved to Audaxis in 2008 until 2009. His research interests lie in the areas of computer network, network security, distributed systems and network mobility.

Fumio Teraoka received a master degree in electrical engineering and a Ph.D. in computer science from Keio University in 1984 and 1993, respectively. He joined Canon Inc. in 1984 and then moved to Sony Computer Science Labs., Inc. (Sony CSL) in 1988. Since April 2001, he is a professor of Faculty of Science and Technology, Keio University. He received the Takahashi Award of JSSST (Japan Society for Software Science and Technology) and the Motooka Award in 1991 and 1993, respectively. He also received the Best Paper Award in 2000 from IPSJ (Information Processing Society Japan). His research interest covers computer network, operating system, and distributed system. He contributed to the activity of the Mobile working group of IETF by developing Virtual IP (VIP). He was a board member of the WIDE Project from 1991 to 2010. He was a board member of IPSJ from 2000 to 2002. He was a board member of JSSST from 2005 to 2009. He is a member of ACM, IEEE, JSSST, and IPSJ.