World Academy of Science, Engineering and Technology
International Journal of Information and Communication Engineering
Vol:1, No:6, 2007

Implementation of SSL Using Information Security Component Interface

Jong-Whoi Shin, and Chong-Sun Hwang

***Abstract*—**Various security APIs (Application Programming Interfaces) are being used in a variety of application areas requiring the information security function. However, these standards are not compatible, and the developer must use those APIs selectively depending on the application environment or the programming language. To resolve this problem, we propose the standard draft of the information security component, while SSL (Secure Sockets Layer) using the confidentiality and integrity component interface has been implemented to verify validity of the standard proposal. The implemented SSL uses the lower-level SSL component when establishing the RMI (Remote Method Invocation) communication between components, as if the security algorithm had been implemented by adding one more layer on the TCP/IP.

***Keywords*—**Component Based Design, Application Programming Interface, Secure Socket Layer, Remote Method Invocation.

I. Introduction

VARIOUS APIs are being used in a variety of application areas requiring the information security function. Each standard was developed to fit the application area of each vendor, as integrating security APIs that were designed for different purposes and usages was out of the question. To resolve this problem, we introduced a component design technique based on a more advanced concept than the current information security API, in order to develop the standard proposal for the confidentiality and integrity of the service component interface. To verify the validity and stability of the standard development, the SSL (Secure Sockets Layer) was implemented.

This paper proposes the standard proposal for the confidentiality and integrity service component interface, and shows how to implement the EJB component in the J2EE environment using the SSL v3.0, which was implemented to verify the efficiency of the standard proposal. In section 2, the standard of the information security service component interface has been described, while in section 3the design and implementation of the SSL component have been outlined. The conclusions drawn are detailed in section 4.

Manuscript received May 13, 2005.

Jong-Whoi Shin (phone:+82 2 405 5256; fax: +82 2 405 5219; e-mail: jshin@kisa.or.kr) and Chong-Sun Hwang (e-mail: hwang@disys.korea.ac.kr) are with Department of Computer Science and Engineering, Korea University, 1, 5ka, Anam-Dong, Sungbuk-Ku, Seoul, Korea.

II. Standard of the Information Security Service Component Interface

A. Information Security Component

Various security APIs such as IETF *GCS-API* and *GSS-API*, RSA *Cryptoki*, Microsoft *CryptoAPI*, and Intel *CDSA*, which were all developed separately, have experienced some difficulty in adopting the application in the distributed environment due to a lack of compatibility. Worse still, the proper software could not be provided in time to satisfy the various requirements. The component ensures the enhanced timeliness and productivity of development by preparing the software for the various requirements. It also encourages standardization between various products. The information security component was proposed in order to bring about the standardization of the information security function, and efforts are being made to standardize the interface of the information security components.

The information security component is an independent software component with more than one function to support the information security service at each IT application area. It is designed to provide the core information security functions of the current security API such as confidentiality, integrity, authentication, access control, and non-repudiation [1].

- *Confidentiality service component*

Confidentiality is necessary to conceal important information that is saved or transmitted in online and offline environments from an unauthorized or unidentified party. This component provides encryption and decryption functions based on the conventional encryption algorithm and the public key encryption algorithm.

- *Integrity service component*

Integrity is required to protect information content transmitted via the network from being illegally created, modified, or deleted. This component provides a hash algorithm and a MAC(Message Authentication Code) generation function.

- *Authentication service component*

Authentication is required to clearly identify an entity through information exchange. It is used to verify the qualification of an object that enables access to certain

information, and to verify the object content. This component provides a digital signature algorithm and an integrity service function.

- *Access control service component*

This component provides the authentication service function and prevents unauthorized usage of the resources by controlling access based on access rights, with regards to access of the access subject (user, process, and so on) according to the security policy.

- *Non-repudiation service component*

This component is used to prevent the repudiation of a transmission and its content between the sender and receiver. It provides the authentication service function.

The component should be used with reliability, by ensuring the stability of the information security component itself, when using the information security function based on the component. While the component has scalability and universality by nature, the component itself may contain security vulnerability, which may also be caused at the component composition stage. To secure component security, and to use it as the information security component, the vulnerability possibility should be removed. Currently, many studies are being carried out on this subject. The standard draft proposed by this study stresses the need to minimize this problem when creating the component, by using the interface. To this end, the pre-conditions and post-conditions are clearly presented for the I/O stage of each interface.

B. Framework of the Information Security Component

Since general users are flooded with many security APIs, there is some confusion about the layer location of the information security component and the location of the developed component for the interface. Figure 1 shows the framework of the information security component that was designed with a 3 tier layer (basic distributed environment structure) and the hierarchical structure of the component area. To provide the information security service to every component area in this hierarchical structure, the information security component is included in the general business layer. Various component-based information security products that belong to the upper layer have been implemented with the basic information security components provided by the information security component framework, such as confidentiality, integrity, authentication, non-repudiation, and the access control component. Information security components located at the general business layer in the information security component framework provide the information security service function for the components at another layer.

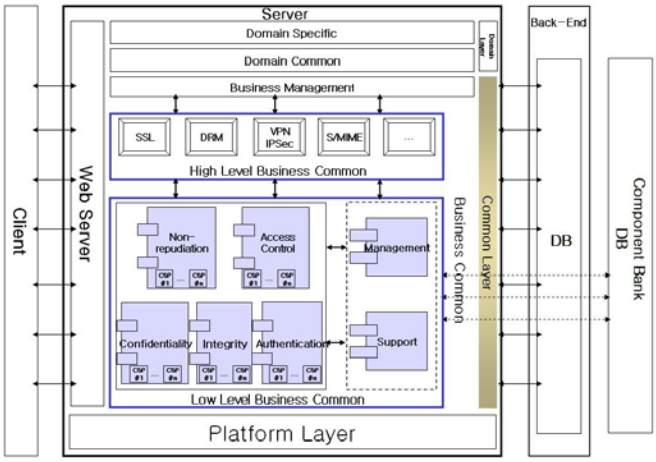

Fig. 1 Information Security Component Framework

C. Interface Specification of the Confidentiality Service Component

The confidentiality service component interface provides a service that encrypts or decrypts the message to provide the confidentiality function. Design is based on the IDL (Interface Definition Language) interface to secure universality on the different platform. Several methods can be used to define the interface. IDL is not needed if *Visibroker Caffein* is used, or the interface can be defined without knowing the IDL if Java RMI or RMI-IIOP is used. However, in order to secure availability, which is one of the requirements with which the information security component should be equipped, the easy to learn standard interface definition language as defined by the OMG (Object Management Group)[2] and IDL was used to specify *IConfidentiality* – the confidentiality service component interface as shown in Figure 2.

```
interface IConfidentiality {
    exception  UnSupport {string why};
    exception  Illegal State {string why};
    exception  Invalid State {string why};
    exception  Internal State {string why};

    void setAlgorithm(in string cipherAlgorithm);
    out string[ ] getAvailableAlgorithms();
    out string getAlgorithm();
    void setNameType(in string IDType);
    out string  getNameType();
    void initialize(in string opMode, in byte[ ] key, in byte[ ] iv);
    out byte[ ] update(in byte[ ] message);
    out byte[ ] finalize(in byte[ ] message);
    out int getLength(in int inputLength);
    void setEncodingType(in string codingType);
    out string getEncodingType();
    }
```

Fig. 2 Confidentiality Service Component Interface (IConfidentiality)

SetAlgorithm operation sets the name or the identifier of the

encryption/description algorithm to be used as the confidentiality function, with the pre-condition that the value of the cipherAlgorithm should be OID, general name or NULL, and that the component should be initialized before calling operation. The post-condition is that proper exception handling should be performed – IllegalState if the component is not initialized, UnSupport if the supporting algorithm is not available, or InternalState if the internal problem occurs inside the component.

II. Designing and Implementing the SSL Component

A. SSL Provided by Current Components

As more components are used, the demand for security concerning component usage increases, and the platform development companies begin to support the security solutions at the server level to accommodate those requirements. Authentication, authorization, and usage of the SSL (Secure Sockets Layer) constitute the security issues in the component platform. The following methods can be used to set the security in the component platform[3].

- *Declarative method*

When the security requirement is set by the deploy tool, the container handles the security requirements properly in the manner of a transaction handling, without affecting the code.

- *Programming method*

Setting the security requirements by programming requires application of the security requirements at the code level. Each time the code is modified, compilation and execution should be performed again. A fine-grained security setting is possible.

Selecting one of two methodologies is a matter of maintaining the balance of convenience and control. SSL is a security communication protocol developed by Netscape to provide confidentiality and integrity through mutual authentication and encryption in the TCP/IP. Generally, SSL is included in the browser and web server, and used for various web communication protocols with the HTTPS format. If the new SSL component is implemented, fine-grained security can be set: discovery and interface in the business component are easy to use, and various encryption mechanisms including the local encryption algorithm can be set according to the security policy of the platform.

B. SSL based on the Information Security Service Component

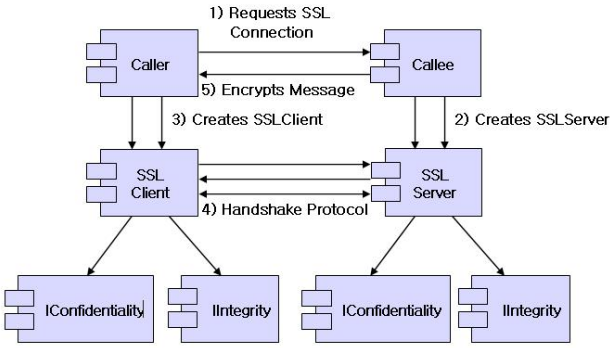

Fig. 3 Transmission of the RMI-based SSL Message

In this structure, the SSL components at the lower level are used when establishing the RMI communication between components, as the security algorithm is implemented in SSL by adding one more step to the TCP/IP. When the Caller component requests the security connection to the Callee component and uses the SSL, the Callee requests the creation of the SSLServer component, and the Caller creates the SSLClient component, and exchanges the encryption algorithm and session key that will be used by the client and server via the SSL handshake protocol. When the handshaking process is complete, RMI parameters and the actual message equal to the result value are encrypted/decrypted using the encryption algorithm that was exchanged, and the security of the exchanged message is provided by creating/verifying the MAC. The execution methods are as follows.

1) First, the caller requests SSL connection (secure connection) to callee.
2) Callee creates the SSLServer component object.
3) Caller creates the SSLClient component.
4) Execute the handshaking protocol between SSLClient and SSLServer.
5) Encrypts the RMI transmission message, using the handshaking result.

When execution of the SSL handshaking protocol is complete, the messages are encrypted or transferred as the parameter by calculating the MAC using *IConfidentiality* and *IIntegrity* based on the SSLClient component algorithm, which was decided when calling by the methods of the Callee component and Caller component. Likewise, Caller verifies the MAC using the defined algorithm to ensure integrity, and decrypts the received message using the defined encryption algorithm. The same method is used when sending the processing result of the callee to the caller. This implementation method is not limited by the specific component platform environment, since the lower TCP/IP socket is not directly used. Figure 4 shows the relationship between the classes that are running inside the SSL component.

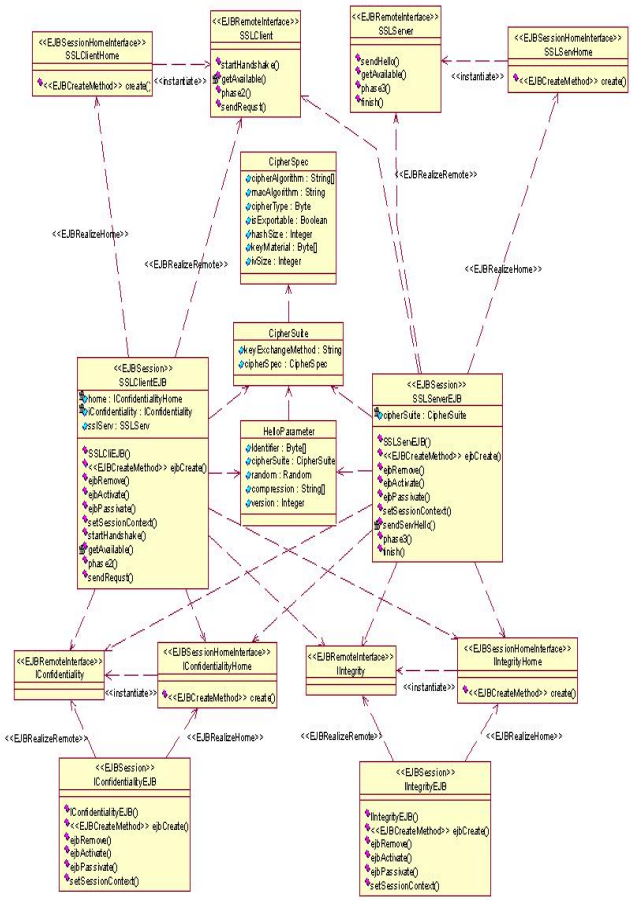

Fig. 4 SSL Component Class Diagram

III. Conclusion

This paper proposed an interface standard draft for a confidentiality and integrity service component. To verify the validity of the proposed standard, the method for implementing SSL v3.0 using the EJB component in the J2EE environment has been described. By introducing a component design technique that is more advanced than the current security API, the quality of the software requiring the security service should be ensured. Furthermore, the standard should be developed for the security service component to enhance system development productivity and compatibility, by maximizing the reusability of the components registered as the standard parts through testing and verification. At present, the information security components are not standardized at all in Korea or in overseas countries. Korea is at the initial stage as can be seen by the definition of the terminology and the guidelines only with regards to the components. Therefore, the groundwork has been laid by the study for the standardization of the information security components, which can contribute to the enhancement of international competitiveness in the information security industry.

References

[1] ISO 7498-2,"Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture", 1989.
[2] OMG Laboratories, "OMG IDL Syntax and Semantics", OMG, July 2002.
[3] SUN, http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Security.html
[4] Y.R.Choi and S.J.Park, "Study on standardization of information security component framework interface", KISA, Dec.2002.
[5] ETRI, "The standard of Open GIS interface for CORBA", TTA, Feb. 2002.

Jong-Whoi Shin received his B.S. degree in Electronic Engineering from Kangwon National University and M.S. degree in Computer Science and Technology from Korea University, South Korea, in 1990 and 2001, respectively. He is currently working towards a PhD degree in the Department of Computer Science and Engineering, Korea University, South Korea. His research interests include intrusion tolerance system and CBD.

Chong-Sun Hwang received the B.S. and M.S. degrees in mathematics from Korea University, South Korea, in 1966 and 1970, respectively, and a PhD degree in computer science and statistics from the University of Georgia, in 1978. From 1978 to 1980, he was an associate professor in University of South Carolina, Lander, USA. He has been a professor in the Department of Computer Science and Engineering, Korea University, South Korea. His research interests include distributed computing and mobile computing systems.