USB Key as an active device of Security System

Mohammed Nasir Uddin[1], Selina Sharmin[2], Abu Hasnat Shohel Ahmed[3] and Emrul Hasan[4], Shahadot Hossain[5] and Muniruzzaman[6]

Shanto Mariam University of Creative Technology[1, 3] ,Uttara University[4,5,6]

Summary

USB Key represents one of the smallest computing platforms today. With the development of USB Key chips, more and more security features will be supported by USB Key. Network security middleware is a Java and HTTP-based development framework on USB Key. Developers can develop and on line load network security–related programs on this framework. End users can use this USB Key to establish secure network connections. In this design, USB Key became an active secure device on network. Design and implementation of this middleware is given in this paper. At fist, the software architecture and security features of this middleware are discussed in detail, which includes how to implement HTTP server, CGI, security proxy etc. on USB Key. Secondly, application scenarios and security analysis of mutual authentication between Web server and client on this middleware are given. At last, advantages of this middleware and what to do further are given in conclusion and future work.

Key words:

Network Security, CGI Security Proxy, Security Analysis, Crypto Server Provider, USB Key

1. Introduction

With the development of the Internet, more and more people are using the Web for everyday tasks, from shopping, banking and paying bills to consuming media and entertainment. But as the value of what people do online have increased, the Internet itself has become more complex and dangerous. On line identity theft, fraud, security and privacy concerns are on the rise [2].In order to solve this problem USB Key has been used as CSP (Crypto Service Provider) in Web service, which will store keys, certificates and execute crypto operations. In this application scenario, the applications of Web service (e.g. IE, Outlook) on operating system will send commands to USB Key and get the response from it. In other words, USB Key is a passive device and "behind" the Web browser. The problem is that most attacks on the client Web applications (e.g. Trojan horse, spy ware, screen capture) can not be prevented by this scheme because some security–related operations except crypto operations will be done on PC, which is not a secure computing environment. In this paper, a new idea of network security middleware based on USB Key is given. Web server, security proxy, Java and HTTP based development framework will be implemented on USB Key. More security functions are supported on this platform:

(1) Users can visit the USB Key with the Web browser.

(2) Security proxy can be developed on this framework. Therefore USB Key can be a security proxy to finish mutual authentication and establishment of secure channel between remote server and client.

(3) Client Web applications can connect further to other Web sites by this security proxy on Key.

In other words, USB Key become an active device and "before" the Web browser in this scenario. At the same time, all of the security–related operations will be done on USB Key, which is a secure computing environment. This scheme will improve the security of client effectively.

The design and implementation of this middleware is discussed in the following sections. The middleware architecture and differences between this scheme and other similar products are presented in section 2.Application scenarios and security analysis based on this middleware is presented in section 3 and finally the conclusion and future work.

2. Architecture of Network Security Middleware

The goal of this design is to implement Web server, security proxy, Java and HTTP - based development framework on USB Key so that the third party vendors can develop security-related programs on this middleware and distribute it to users' Key on line. In other words, it is a network security middleware on client.

Recent proposals have been made to connect USB key (or smart card) to the Internet [3], [4], [5]. There are some differences between our scheme and those proposals:

(1) The goal of this design is to provide a security middleware platform to support third party's development but those proposals emphasize to provide a Web–enabled

Manuscript received October 5, 2011
Manuscript revised October 20, 2011

device. The fact is that an open security infrastructure is a trend on the Internet [6].
(2) There is a button on our USB Key. The applets on Key can detect this event when users press this button, which can not be simulated by any software running outside USB Key. Users can actually control operations of network transaction by this button. An example will be given in section 3.
(3) The full TCP/IP and HTTP will be supported in those proposals. The precondition is a high Performance hardware platform, which is 32-bit chip, 24K RAM, 64K EEPROM and USB 2.0 Connection. But the HTTP performance of this device is not very high [4]. A few chips for USB Key can provide those hardware features until now and the price is still very high. The price and Performance is important for client device. Furthermore, it is difficult to be ported onto other common hardware platforms
In this paper, the hardware platform is the 8-bit 8051 architecture chip with 6K RAM, 64K EEPROM and a FLASH memory, which is commonly used in USB Key now. The software platform is a Java Card platform on Key, which supports programming in Java with Java Card 2.2.1 APIs [7].
Two layers software architecture is deployed on this platform:
(1) At first, a communication agent that gets data from TCP/IP connection will stay in the FLASH memory on USB Key. This agent is responsible for establishing TCP/IP connection, encapsulate network data into APDU and transmit to the USB Key by USB Port. It will run in PC operating system.
(2) Secondly, a HTTP Server and security proxy framework are implemented in Java on USB Key, which includes request switch, HTTP server, CGI, security proxy and program load components. All of those components will run in USB Key. This software architecture is depicted in figure 1. The advantages of this scheme are as follows:
(1) The communication agent running in PC operating system will finish TCP/IP transport. This will improve the system performance and lower USB Key hardware requirements because FLASH memory is much cheaper than high performance processors.
(2) All the HTTP and security proxy procedures will be done on USB Key with Java Card platform, which is a secure computing environment [7]. Therefore, system security is guaranteed.
(3) The communication agent in FLASH memory will run automatically (auto run mechanism of USB disk) when the USB Key is inserted on USB port. So it’s convenient to use on mobile devices and users needn’t to install any additional software on operating system.
The communication agent is simple in this system and unnecessary to be described further in this paper. So the design architecture of HTTP Server and security proxy framework on USB Key will be further discussed in the following sections.

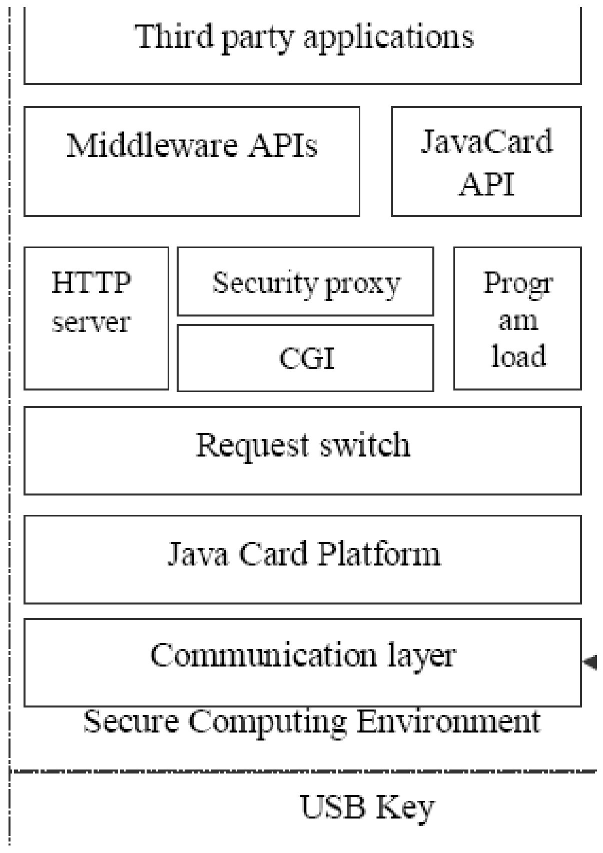

Figure 2.1-Architecture of USB Key middleware

2.1 Request switch component

The first component on USB Key that will process the HTTP data encapsulated in APDU is Request Switch. At first this component will get the HTTP request data from APDU and then parse the command and switch it to the appropriate components, which include HTTP Server, security proxy and program load component. The work flow can be described as follows:

```
Request Switch()
{
Get HTTP request command from APDU buffer;
Parse the request;
switch (request)
{
Case “Request for local static data”:
```

```
Switch request to HTTP server component;
Case "Request for local CGI":
{
Switch request to HTTP server component;
HTTP server call corresponding CGI API;
}
Case "Request for remote CGI":
Switch request to remote CGI component;
Case "Request for remote Web Service":
Switch request to security proxy component;
default:
throw illegal request exception;
}
}
```

At the same time, there is a *request Info* transient object in RAM to maintain information for every request. All of those methods managing HTTP requests are encapsulated in a class – *Request Switch Object* , which can be provided for HTTP–based application development.

2.2 HTTP Server Components

HTTP server component will manage the Web page data stored in USB Key, parse HTTP protocol and construct response data. USB Key becomes a Web server based on this component. Users can visit Key by Web browser and browser the content on Key in Webpage. This component consists of two sub-components:

The first sub-component is a Web data management component. It will receive the Web page data from personalization tools and store it in USB Key persistent memory (e.g. EEPROM, flash). All of the Web–related data will be store in *Web Object*. The relationship between a Web page and *Web Object* is:

Web Page = { x | x is a *Web Object* }

Another component is a HTTP parser. This component will parse HTTP data from request switch component and finish the corresponding operations according to HTTP method. We implement this component in a class – *HTTP parser*. The USB Key can be visited by HTTP protocol through Web browser based on this component. It means that users can browser contents on USB Key and manage USB Key (e.g. download Applets) by Windows IE. Further more, more HTTP–related applets can be built on those classes by application developers.

2.3 Program Load Component

Program load component can receive the programs code of Java applets on line and install or update (if there is an old version of this program) it on USB Key.

At first, the Java Card Applet code will be verified by verifier, which may be on the key or off the key (run in Key operating system or PC operating system), which depends on Java Card platform provided on USB Key. If off the key, the verifier will be stored in FLASH memory and called by communication agent at first.

Secondly, the Applet will be installed in Java Card Runtime Environment on USB Key. Because of security considerations, this component is not open for application development.

2.4 CGI Component

CGI (Common Gateway Interface) is supported on USB Key middleware. Two approaches to implement CGI are used in our design.

The first approach is to implement CGI in terms of APIs (methods) in HTTP server component. In other words, CGI programs run in the same execution context[5] as HTTP server. The advantage of this approach is to eliminate context switch between different applets and improve the running speed. The disadvantage is that HTTP server and CGI programs execute in the same context, which is not secure. Any bug in CGI programs coded by third party can negatively impact the entire server, including URL requests have nothing to do with the bug-containing programs. Therefore we only take this approach to implement some simple, USB Key specific CGI functions, such as verify user PIN, generate SHA-1digest of message etc. We call it as local CGI, which is part of HTTP server.

In the second approach, we implement CGI programs as Java Card applets with SIO (Shareable Interface Object) [8]. The request switch component received the CGI request and then will call this CGI program by SIO.At the same time, execution context will switch to CGI programs and switch back to HTTP server after CGI execution. There is an applet firewall to protect runtime environment between different contexts on Java Card platform [5]. Therefore any bug in CGI programs can not impact entire server because they run in different contexts. We think this approach is more secure than the first approach. We call it as remote CGI, which is a separate component. Remote CGI principles are depicted in Fig. 2.

Any CGI programs developed by third party on our framework in this approach are Java Card Applets with SIO. Those can be loaded into USB Key on line at any time and execute in different context from HTTP server .This CGI scheme is an open, scalable and secure architecture.

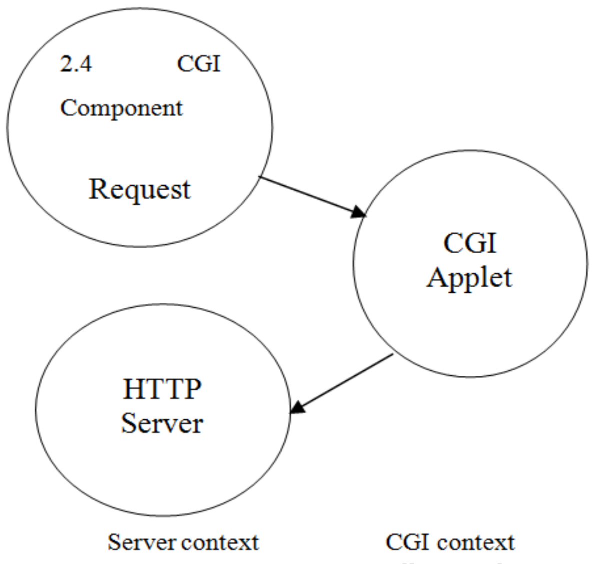

Fig. 2.2- Remote CGI call principles

2.5 Security Proxy Component

It is not difficult to implement proxy component based on components mentioned above. At first, the communication agent is designed be able to establish two connections at the same time, one is to local browser and another is to remote server.

Secondly, the security proxy component is implemented as a remote CGI for security considerations. The proxy CGI applet with SIO is defined as follows:

```
public interface security Proxy interface extends Shareable
{
void security Process (byte[] buffer);
}
public class secrity Proxy Class extends Applet
implements securityProxyInterface
{
public Shareable
getShareableInterfaceObject(AID clientAID,
byte parameter) { return this; }
public static void install(byte[] bArray, short
bOffset, byte bLength) {…}
public void securityProcess (byte[] buffer){…}
protected final void startSecurityConnection()
{…}
protected void setDestinationURL( byte []
buffer) {…}
protected void sendDataToURL() { …}
protected void receiveDataFromURL() { …}
}
```

Developers can extend *securityProxyClass* and override the *securityProcess()* method according to the specific security scheme, for example, mutual authentication, key exchange, SSL etc.

Another important method is:*protected final void startSecurityConnection()*

We implemented this method as follows:

(1) Display a Web page in Web browser to prompt the user to press button on USB Key.

(2) Wait for the user to press button (detect the interrupt signal of pressing button on USB Key).If not, goto 2.

(3) Clear the signal and then end this method.

It is proposed that *startSecurityConnection()* is called at first in method *securityProcess()* so that only the user can actually control the secure connection but not the Trojan horse or spyware do it hostilely.When developers install a security proxy applet for a Web site the *install()* method will bind this URL to the applet in proxy register table. While the end user visit this URL, the security proxy component will look for this URL in register table and call the corresponding applet's *securityProcess()* to establish security connection.

3. Application Scenarios and Security Analysis

In this section, we will describe an application scenario for simple mutual authentication between client and server with USB Key middleware. After the authentication, the secure channel will be established and user can login with ID and password.

We defined:

Km - the key for mutual authentication (stored in USB Key)

Rc – random number on client (generated in USB Key)

Rs – random number in server

Kc –random key material on client (generated in USB Key)

Ks –random key material in server

E(x,k) – encrypt x with key k

D(x,k) – decrypt x with key k

X(x,y) – x xor y

x||y – x concatenate y

The work flow of this scenario as follows:

(1) The user inserts USB Key and input PIN for Key usage.

(2) The user visits specific Web site requiring secure connection.

(3) When see the prompt information in browser the user press the button on USB Key.

(4) If the USB Key detected the interrupt signal of pressing button, goto 5. Else go to 4.

(5) The Key sends Rc to server.

(6) The server sends E(Rc||Rs||Ks, Km) to the Key.

(7) The USB Key D(Rc||Rs||Ks, Km) and compare Rc. If failed, end this session.

(8) The USB Key sends E(Rc||Rs||Kc, Km) to the server.
(9) The server D(Rc||Rs||Kc, Km) and compare Rs. If failed, end this session.
(10) The server calculate X(Kc, Ks) as session key and request USB Key to send user ID and password.
(11) The USB Key calculate X(Kc, Ks) as session key and sends E(ID||password,X(Kc, Ks)) to login server.

In this scenario, users login Web site with PIN and USB Key (hardware token). This is a multi-factor identity authentication [9].
It is supposed that there is an eavesdropper on the user's PC, which can monitor the data communication on USB. The threat from it will be analyzed as follows:
(1) It can get PIN for USB Key usage but can not start security connection because the signal of pressing button can not be simulated outside USB Key.
(2) It can not get Km , ID and password for server, which are stored in Key and transferred in cipher(ID and password).
(3) It can not attack authentication procedures, which is done in USB Key.
In summary, users' login with our USB Key achieved level 3 security defined in [9]. Furthermore all authentication procedures are done in USB Key and users can actually control when to do those procedures by the input device on USB Key. This scheme will effectively prevent hostile software from doing transactions without users' permission and improve the security on client.

Conclusion

USB Key (smart card) represents one of the smallest computing platforms today. With the development of smart card chip, more and more security features will be supported by USB Key. A new idea of network security middleware based on USB Key is given in this paper. Web server, HTTP proxy, Java and HTTP – based development framework were implemented on USB Key.

In this paper, it is given a choice of a design following open infrastructure argument and developers can build an open and scalable security system for Web service on our USB Key middleware. USB Key, which is a passive device in the past, becomes an active device depending on this technology. Furthermore, users can actually control operations of network transaction by input device on Key. This scheme will improve the security of client effectively.

References

[1] Dawei Zhang,"Network Security Middleware Based On USB Key",IEEE spectrum 08
[2] http://www.identityblog.com/stories/2005/07/05/IdentityMetasystem.htm
[3] http://www.gemplus.com/smart/rd/publications/pdf/MD02gdc c.pdf
[4] Henrich C. Pohls&Joachim Posegga,"Smartcard Firewalls Revisited",Tarragona Spain 2006
[5] http://research.sun.com/brazil
[6] http://www.identityblog.com/?page_id=354
[7] Sun Microsystems ," Inc.: Java Card 2.2 Virtual Machine Specification ",Sun Microsystems Santa Clara,2002
[8] Zhiqun Chen, " Java Card Technology for Smart Cards :Architecture and rogrammer's Guide",Addison Wesley Boston,2000
[9] http://www.cio.gov/eauthentication/documents/P800-63V6_3_3.pdf

Mohammed Nasir Uddin received PhD in Computer Science from Moscow Power Engineering Institute (Technical University),Moscow. Russia. Masters of Science in Computer Engineering, and Bachelor of Engineering in Computer Engineering from State University of Lvivska Polytechnic, Lvov, Ukraine. Presently Working as an Assistant Professor, Department of Computer Science and Engineering, Shanto Mariam University of Creative Technology, Dhaka, Bangladesh. His areas of interest include Information & Computer security, Microprocessor system, Bio-Informatics and Digital Systems.

Selina Sharmin is a Lecturer of CSE & CSIT at Shanto-Mariam University of Creative Technology. She completed M.S and B.Sc with concentration in Computer Science & Enginnering, from University of Dhaka, Dhaka, Bangladesh. She has interest in the field of Bio-Informatics, Programming in Critical field and Data Mining.

Abu Hasnat Shohel Ahmed is a Lecturer of CSE & CSIT at Shanto-Mariam University of Creative Technology. He completed M.S and B.Sc with concentration in Applied Physics, Electronics and Communication Engineering from University of Chittagong, Chittagong, Bangladesh. He is the Associate Member of Bangladesh Computer Society. He has interest in the field of Wireless & Microwave communication, Networking and Bio-Informatics

Shahadot Hossain is a Lecturer of CSE at Uttara University. He completed M.Sc and B.Sc(Engg.) in Computer Science and Engineering from Uttara University,Dhaka, Bangladesh. He has interest in the field of Digital System, Virtual Networking and computer and Network security.

Emrul Hasan is a Lecturer of CSE at Uttara University. He completed M.Sc and B.Sc(Engg.) in Computer Science and Engineering from Uttara University,Dhaka, Bangladesh. He has interest in the field of Digital System, Algorithm in critical field of Programming, Virtual Networking and computer data security.

Muniruzzaman is a Lecturer of CSE at Uttara University. He completed M.Sc and B.Sc(Engg.) in Computer Science and Engineering from Uttara University,Dhaka, Bangladesh. He has interest in the field of Digital System, of Bio-Informatics, Programming in Critical field, Computer security and Data Mining.