A Remote Authentication Model Using Smart Cards

Hakim. L. Fourar

CIS Lab, Prince Sultan University, Riyadh, P.O.BOX 66833 Riyadh 11586, Saudi Arabia

Abstract — **Many companies want to use Internet as a sale channel and to perform e-business transactions. However, many companies doing business on the Web are faced with security issues which must be addressed in order to protect sensitive information. Businesses must find ways to authenticate employees, customers, suppliers and partners while ensuring the security of transactions, and sensitive information. This paper present a solution for facilitating authentication and access control through a web connection. Smart cards are a key feature of this solution. The solution proposed is based on 3 authentication methods and use cryptography techniques.**

Index Terms — **Access Control, Remote authentication, cryptography, PKI, Smart Card, SSL.**

I. Introduction

The lack of user confidence in electronic business transactions is the greatest inhibitor obstructing the growth of e-commerce. Data traveling over public networks such as the Internet can be easily compromised. Businesses must find ways to authenticate employees, customers, remote offices, suppliers and partners while ensuring the security of transactions, sensitive information, applications and online communications. Effective information security policy must have the following six objectives [1]: confidentiality; integrity; availability; legitimate use (identification, authentication, and authorization); auditing or traceability; and non-repudiation. Several solutions aim at securing e-business transactions. A growing number of organizations are building public key infrastructures to solve e-business security issues. A public key infrastructure (PKI) will protect data through encryption and authenticate users through a certificate-based framework of digital signatures. PKI is a system of certification authorities (CA), which establishes security policies and issues digital certificates to users. However, implementing a public key infrastructure is not enough. Once a digital certificate is issued to an individual or organization it must be protected in much the same way you protect a passport or your credit cards. Storing digital certificates on the LAN or on computers makes it fairly easy for others to copy the certificate and then use it impersonates you. By using the smart card solution, storing multiple digital signatures, certificates, private keys, IDs and passwords on a smart card solves a number of the security and portability issues [2]. Due to the cryptographic capacity and portability, smart cards have been widely used in many e-commerce applications [3]. Smart cards are an ideal secure storage device. Sensitive information stored in the card is protected by a PIN (Personal Identification Number). Usually, the user enters the PIN to unlock the card and allows applications the access to the protected information (IDs, Passwords, Keys, Certificates,). Users can freely choose and change their PINs.

II. The Classical Authentication Mechanisms

In order to secure e-business transactions, each organization should authenticate its users (partners, clients, or suppliers). The authentication mechanism grants a user with specific access permissions.

A secure system has to track the identities of the users requesting its services [4]. Authentication is the process of verifying a user's identity. The main reason for authenticating a user is for access control decisions. The major existing systems provide unilateral authentication. The server authenticates the client, but the client doesn't authenticate the server. In this case, the user has no guarantees about the identity of the party at the other side. The client is not sure that is communicating with a trusted party. The user can send confidential information with any guaranty that this information will go to the desired party. For that, it is very important that the user, authenticate the server before any transaction.

Another problem is that most the authentication systems use identification through username and password [5]-[8]. The user is asked to enter a username and a password to be authenticated. It is a real problem, if the password is discovered or guessed by an attacker. In a spoofing attack, a spoofing program presents a fake login screen and asks the user to enter his username/password. The unsuspecting user is unaware of the fact that his username/password has been compromised.

Once authenticated to each other, server and user must negotiate a session key to be used for protecting their subsequent communications [9].

III. Authentication Mechanism Model

To avoid the problems mentioned in the previous section, the authentication model suggested use smart card to store confidential information (ids, passwords, keys). Confidential information is retrieved from the smart card by a client plug-in program embedded in the browser. This plug-in program is activated by the HTML tag <EMBED> or <OBJECT> that the client web page contains. When the client visits his partner's web site, the web server returns a web page with the plug-in tag <EMBED> or <OBJECT> causing the plug-in in the client side to be executed. The plug-in asks for card insertion and client PIN presentation. Confidential information stored in the card is protected by a PIN. The card is blocked after three unsuccessful tries. The model proposed is based on three authentication modes. The authentication can be done with IDs/Passwords, with symmetric keys or with asymmetric keys. After authentication, SSL secure communication session is established between the client and the server.

A. Authentication with ID/Password

This mode assumes that only the ID and Password are read and retrieved from the smart card by the plug-in program [10]. The user has to present a correct PIN to allow access to the ID/Password stored in the smart card. The plug-in sends the ID and Password back to the server via HTTP POST to an URL specified at plug-in invocation. The server verifies the ID and password in the LDAP database. If there is matching between the ID/Password posted by the client and stored in LDAP directory, the client is authenticated and allowed to perform e-business transactions according his access permissions stored in the LDAP directory.

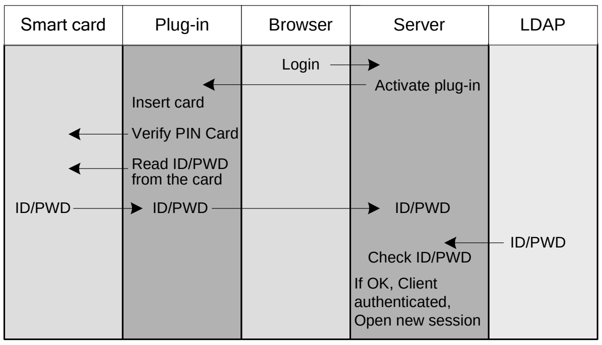


Fig. 1. Authentication flow: Mode 1

B. Authentication with Symmetric Key

In symmetric encryption, both the client and the server use the same methods to encrypt and decrypt messages. These methods generally involve a well-known encryption algorithm and a secret key, known to both client and server. This means keys have to be exchanged, and the safest way to do this is offline, or by using an asymmetric encryption of the secret key.

This mode allows mutual authentication. The client visits the web site and requests an authentication to be able to perform e-business transactions. The server builds a random string R and signs it with the secret key K of the client stored in the LDAP directory. The signature is generated by a triple DES (Data Encryption Standard) encryption algorithm [11]. 3DES requires the use of a key K (Kl: Kr) where Kl is the left part of the key and Kr is the right part (1). The server posts the random string with the signature S1 to the client.

$$\mathbf{S1=3DES(K,R)=DES(DES^{-1}(DES(R,kl),Kr),Kl)} \quad (1)$$

The plug-in client program embedded in the browser signs the random R with the same secret key K stored in the smart card and verifies the computed signature (S2) with the signature S1 received from the server. A PIN protects the secret key stored in the smart card. The user is asked to enter this PIN to allow the client system accessing the secret key. The secret 3DES Key K is shared by both the client and the server. If there is match between S1 and S2, the server is authenticated and the client system sends back the string random R, the signature received (S1) and the signature computed (S2) to the server via an HTTP post. S1, S2 and R are sent to an URL specified in the HTML page sent by the server and retrieved by the client system.

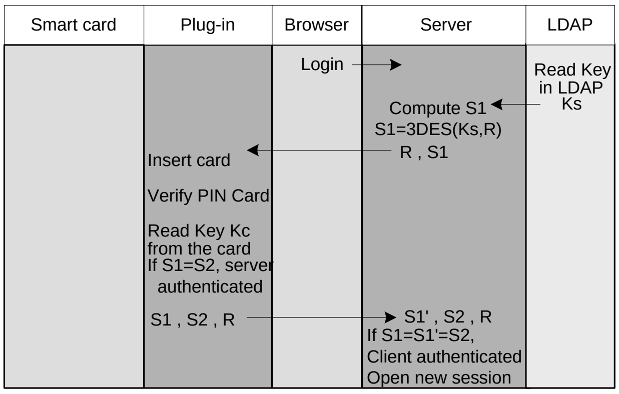


Fig. 2. Authentication flow: Mode 2

Once received, S1 and S2 are checked and compared with the first computed S1. After verification, the web server application authenticates the user and will then return a new URL to the plug-in and the browser is redirected to this new URL location. Fig. 2 illustrates the flow process based on symmetric key.

A new SSL session is then opened between the client and the new web server application and all the transactions can be done safely. For that, both the server and the client compute a temporary diversified 3DES Key (Session Key: SK) from the 3DES secret key K and the Card Serial Number CSN. The key SK is used to establish an SSL session and to secure e-transactions between the client and the server [12].

$$\mathbf{SK = 3DES(K, CSN)} \quad (2)$$

C. Authentication with Asymmetric Keys

This mode is based on Public Key Infrastructure (PKI). Public keys (PK) are used for authentication purposes by signing and verifying data using private key and public key elements respectively. The RSA algorithm is used for signing and verifying data [13]. A third party is invoked: a Certificate Authority (CA). The role of the CA is to issue certificates for the client and the server that confirm that the keys have been authorized. The public key and user's identity are bound together in a certificate signed by the CA, certifying the accuracy of the binding [14]. The CA issues the certificate by signing the client or server public key (PuK) by using the CA private key (PrKCa).

$$\mathbf{Certificate = RSA\_Sign(PuK)_{\{PrKCa\}}} \quad (3)$$

The client can verify the certificate of the server and vise versa by using the CA public key (PuKCa). Client certificate and key pair are stored in a smart card.

$$\mathbf{PuK = RSA\_Verify(Certificate)_{\{PuKCa\}}} \quad (4)$$

The client visits the server web site and asks to log on. The web server returns a web page with the plug-in tag <EMBED> causing the plug-in to be executed in the client side. The server signs a generated random (R) with its private key (PrKServ). The server sends R, the signature S1 embedded in the client web page and the server public key (PuKServ) to the client.

$$\mathbf{S1 = RSA\_Sign(R)_{\{PrKServ\}}} \quad (5)$$

The client plug-in program verifies the signature S1 using PuKServ. The verify algorithm is computed by the smart card.

$$\mathbf{R' = RSA\_Verify(S1)_{\{PuKServ\}}} \quad (6)$$

If R' and R are equal, the signature is successfully verified and the server is authenticated.

The client plug-in computes a second digital signature (S2) of the random retrieved from the received web page using the client private key (PrKCli).

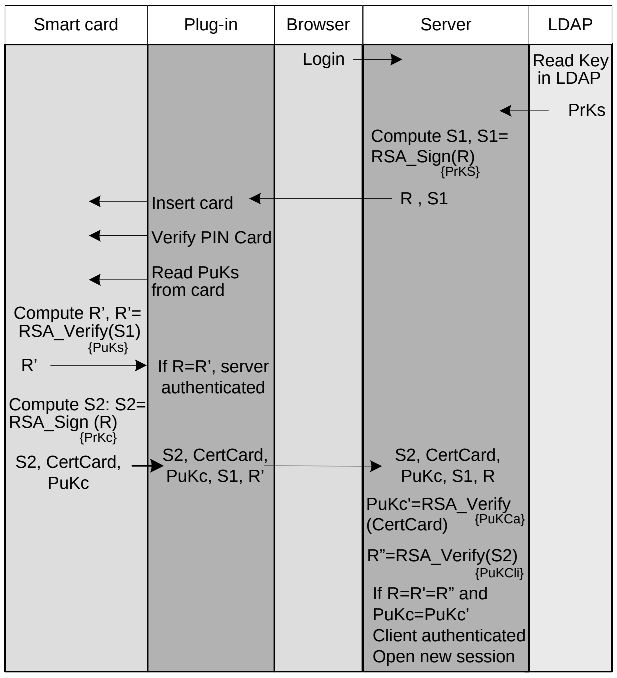
$$\mathbf{S2 = RSA\_Sign\ (R)_{\{PrKCli\}}} \quad (7)$$

For that, the plug-in asks for smart card insertion and client PIN presentation in order to retrieve the key pair and the certificate stored in the card. The client plug-in posts R, S1, S2, CertCard and PuKCli to the server. Once received, the server verifies the certificate and the signature S2.

$$\mathbf{PuKCli' = RSA\_Verify(CertCard)_{\{PuKCa\}}} \quad (8)$$

$$\mathbf{R'' = RSA\_Verify(S2)_{\{PuKCli\}}} \quad (9)$$

If the elements calculated are equal to the elements posted by the client, the server confirms authenticity of the client and new session is opened with the client.



Smart card
Plug-in
Browser
Server
LDAP
Login
Read Key in LDAP
PrKs
Compute S1, S1= RSA_Sign(R) {PrKS}
R , S1
Insert card
Verify PIN Card
Read PuKs from card
Compute R', R'= RSA_Verify(S1) {PuKs}
R'
If R=R', server authenticated
Compute S2: S2= RSA_Sign (R) {PrKc}
S2, CertCard, PuKc
S2, CertCard, PuKc, S1, R'
S2, CertCard, PuKc, S1, R
PuKc'=RSA_Verify (CertCard) {PuKCa}
R"=RSA_Verify(S2) {PuKCli}
If R=R'=R" and PuKc=PuKc'
Client authenticated
Open new session

Fig. 3. Authentication flow: Mode 3

IV. Architecture of The Client System

The architecture of the client system is illustrated by Fig. 4. The client system is implemented as DLLs libraries through four levels. Each library calls functions implemented in the lower library and provides functions to be called by the upper library.

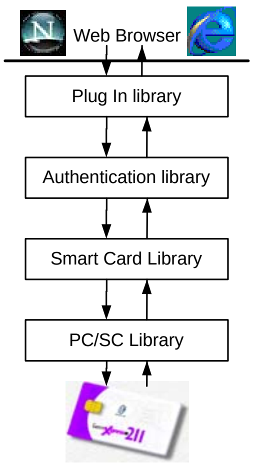


Fig. 4. Architecture of the Client System

A. Plug-in Library

The Plug-in library is the interface between the browser and the smart card authentication library. It is integrated to the browser. The Plug-in library manages the communication and the exchange of data between the authentication library and the authentication server through a secure SSL session. It is also in charge of the data checking and formatting: data posted is base64 encoded when needed, URL encoded and encapsulated in a MIME Object [15].

B. Authentication Library

The authentication library manages the authentication process. It implements the authentication methods depending on the authentication mode requested by the user. The Authentication library also manages the GUI (PIN presentation and error message display). This library is designed to interface with the smart card library.

C. Smart Card Library

The smart card library operates on the low level of the client plug-in system. It implements the functions needed by the authentication process. These functions encapsulate the EMV commands provided by the smart card and defined in the ISO 7816-4 standard. These commands are sent to the card through Application Protocol Data Units (APDU). The smart card (SC) library retrieves information stored in the card (ID/Password, Keys, certificates) and computes signatures (3DES, RSA). The smart card library handles all communication with smart card through PC/SC library. All smart card APDU applicative are available for upper layer as well as some general commands to open and close a session with a card. After executing an EMV command, the smart card returns a status code. The SC library interprets the status code and sends another significant code to the authentication library. The significant code is interpreted by the authentication library and in case of an error; a message is displayed to the user.

D. PC/SC Library

The PC/SC library is the standard model and a device-and-platform independent API for interfacing smart card readers and cards with computers and enabling applications aware smart card. It implements the mechanisms for accessing the services supported by a smart card. The smart card SDK contains the necessary tools and APIs to communicate with the card and to develop smart card applications. The PC/SC library is managed by the PC/SC workgroup and outside the scope of this paper. The PC/SC workgroup set a standard for integrating smart cards and smart card readers into the mainstream-computing environment. They aim to promote a standard specification, to ensure that smart cards, smart card readers and computers made by different manufacturers will work together and help to facilitate the development of smart card applications for PC and other computing platforms.

In order to execute a command implemented in the smart card, the smart card library establishes a connection with the card through the smart card reader, fill the APDU buffer of the command to be send to the card, send the APDU buffer and receive the data returned by the card through the receive buffer. The receive buffer returned by the card contains a status code that indicates successful or failed execution with the error code.

V. Architecture of the Authentication Server

When the validation server starts, it is configured to listen to TCP ports to accept connections from client. When the client connects to the web server, the authentication system receives the request and a new thread dedicated to the client request is started. The authentication system reads data and decrypts it according the SSL standard. It identifies the client and the authentication mode requested. The authentication system retrieve the elements needed for authentication (ID/Password, Keys, certificates) from the LDAP directory and starts the corresponding processes (signature, verification,). If the authentication succeeds, the client browser is redirected to new URL and a new SSL session is created between the web server and the client browser (Fig. 5). If the authentication fails, an error message is sent and the client browser is redirected to another URL (error URL).

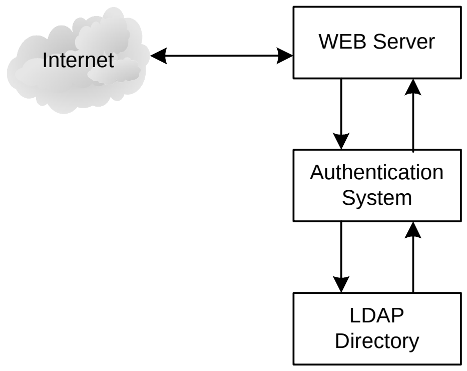


Fig. 5. Architecture of the Authentication Server

VI. Conclusion

The authentication approach proposed in this paper uses smart card to store sensitive information and to compute cryptographic algorithms. The four libraries of the client system have been implemented using C and C++ language using COM and ActiveX technologies. As Netscape does not support ActiveX technology, the plug-in library has been implemented using the Plug-in SDK from Netscape. The PC/SC library communicates with the card via an applet developed and uploaded in the smart card using Java Card technology.

The client can access to the server by using any computer without storing and entering any sensitive information. The card is protected by PIN and cannot be used by another person. The authentication server authenticates clients by computing cryptographic algorithms based on the information posted by the client system. After authentication, the server grants the client accessing the services according his privileges stored in the LDAP directory. The authentication mechanism is not only based on id/password. We have proposed two other authentication modes based on symmetric and asymmetric encryption.

The use of smart cards to secure business transactions assures more protection for sensitive data. Smart cards reduce the threat of unauthorized access by the use of stolen credentials because the hacker must both steal the smart card and obtain the PIN. Smart card authentication reduces also the ability of individuals to deny their actions.

REFERENCES

[1] D. Gollmann, “Computer Security,” John Wiley & Sons 2002.

[2] D. Naccache, and D. M'Raihi, “Cryptographic smart cards,” Micro IEEE, vol. 16, no. 3, pp. 16-24, June 1996.

[3] H. Chien, J. Jan, and Y. Tseng, “An Efficient and Practical Solution to Remote Authentication: Smart Card,” Computers and Security, vol. 21, no. 4, pp. 372-375, 2002.

[4] C. P. Pfleeger, and S. L. Pfleeger, “Security in Computing,” Third Edition. Prentice Hall. 2003.

[5] C. Chang and T. Wu, “Remote Password Authentication with Smart Cards,” IEE Proceeding-Computers and Digital Techniques, vol. 138, no. 3, pp. 165-168, 1991.

[6] C. Chang and S. Hwang, “Using Smart Cards to Authenticate Remote Passwords,” Computers and Mathematics with Application, vol. 26, no. 7, pp. 19-27, 1993.

[7] W. Yang and S. Shieh, “Password Authentication Schemes with Smart Cards,” Computers and Security, vol. 18, no. 8, pp. 727-733, 1999.

[8] K. Tan and H. Zhu, “Remote Password Authentication Scheme with Smart Cards,” Computer Communications, vol. 18, pp. 390-393, 1999.

[9] W. S. Juang, “Efficient multi-server password authenticated key agreement using smart cards,” Consumer Electronics, IEEE Transactions on, vol. 50, issue 1, pp. 251-255, February 2004.

[10] S. Wang and T. Chang, “Smart Card Based Secure Password Authentication Scheme,” Computers and Security, vol. 15, no. 3, pp. 231-237, 1996.

[11] National Bureau of Standard, “Data Encryption Standard,” NBS FIPS PUB 46, January 1977.

[12] T. Dierks, and C. Allen, “The TLS protocol version 1.0,” Internet Technical Report, RFC 2246, January 1999.

[13] R. Rivest et al., “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” Comm. of the ACM, vol. 21, no. 2, pp. 120-126, Feb 1978.

[14] R. Merkle, “Protocols for Public Key Cryptosystems,” Proc. IEEE Symp. on Security & Privacy, pp. 122-133, 1980.

[15] B. Ramdell, “S/MIME version 3 Message specification,” Internet Technical Report, RFC 2633, April 1999.

[16] H. Sun, “An Efficient Remote User Authentication Scheme Using Smart Cards,” IEEE Transactions on Consumer Electronics, vol. 46, no. 4, pp. 958-961, November, 2000.