Journal of King Saud University – Computer and Information Sciences (2013) **25**, 1–5

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

A smart card based framework for securing e-business transactions in distributed systems

Hakim Fourar-Laidi

Department of Information Sciences, College of Computer & Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

Received 31 October 2011; revised 24 April 2012; accepted 8 May 2012
Available online 29 May 2012

KEYWORDS

Smart card;
Distributed environment;
Cryptography;
Secure e-business transaction

Abstract In the recent years, we have seen the emergence and the growing of the e-business via the internet. Many organizations are extending their business transactions by using the Web. This will allow them to reach more customers in a cost effective way and to make their business transactions fast and efficient. Meanwhile, sending sensitive information via the Web must satisfy integrity, privacy, authentication and non-repudiation. Organizations are implementing various infrastructures that allow them to have secure e-business transactions. Many protocols and frameworks have been proposed and implemented to provide secure and trusted exchange between parties involved in the transaction. These frameworks store credentials such as keys in local computers which make them subject to piracy or misuse. In this paper, we propose a framework based on smart card that allows partners to realize secure transactions. The proposed solution use smart cards to store keys and perform cryptographic algorithms.

1. Introduction

The lack of user confidence in electronic business is the greatest inhibitor obstructing the growth of e-commerce. Data traveling over public networks can be easily compromised. Businesses must find ways to authenticate customers, remote offices, suppliers and partners while ensuring the security of transactions and sensitive information. Several solutions aim at securing e-business transactions and protecting sensitive information by implementing different techniques (Chien et al., 2002; Wassim et al., 2007; Mornar et al., 2004). A growing number of organizations are building Public Key Infrastructures (PKI) to solve e-business security issues.

However, implementing a PKI is not enough. Keys, certificates and digital signatures must be protected the same way you protect a passport or your credit cards.

Storing digital certificates on the LAN or on your computer makes it fairly easy for others to copy the certificate and then use it to impersonate you. Smart cards are an ideal secure storage device. This paper presents an e-business framework based on smart cards. Keys, certificates and digital signatures are stored in the card. The card also performs the onboard cryptography operations. Another advantage of this solution is the portability of the smart card that allows the person perform the transaction from any computer on the network. By using the smart card, storing multiple digital signatures, certificates, private keys, IDs and passwords on a smart card solves

E-mail address: fourar@ksu.edu.sa

Peer review under responsibility of King Saud University.

ELSEVIER Production and hosting by Elsevier

1319-1578
http://dx.doi.org/10.1016/j.jksuci.2012.05.002

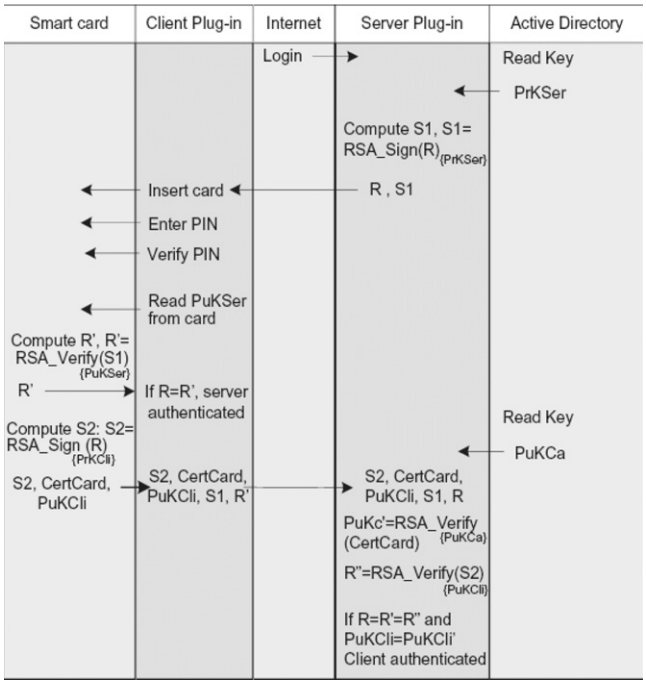

Figure 1 Authentication flow.

Client Plug-in	Internet	Server Plug-in
$H = MD5(M)$ $S = RSA\_Sign(H)_{\{PrKCli\}}$ $M' = 3DES(M+S)_{\{Sk\}}$ $Sk' = RSA\_Sign(Sk)_{\{PuKSer\}}$ $M', Sk' \rightarrow$	$\rightarrow M', Sk' \rightarrow$	$\rightarrow Sk = RSA\_Verify(Sk')_{\{PuKSer\}}$ $(M, S) = 3DES^{-1}(M')_{\{Sk\}}$ $H1 = RSA\_Verify(S)_{\{PuKCli\}}$ $H2 = MD5(M)$ $H1 =? H2$ If(H1=H2), verification succeed

Figure 2 Integrity and privacy flow.

the security and portability issues. Sensitive information stored in the card is protected by a PIN. Usually, the user enters the PIN to unlock the card and allows applications the access to the protected information (IDs, Passwords, Keys, Certificates,). The user is no more asked to enter his username and password that can compromise the security of the transaction (Chen and Yeh, 2005; Hirano et al., 2007). The communication between the card and the client program is protected by session keys generated by the card. It is almost difficult for an attacker to spy the cryptography mechanism executed by the card. Before initiating the transactions, the user and the server, the parties participating in the e-business transaction, should be authenticated to each other. This paper presents an e-business framework based on smart card. Section 2 presents the e-business transactional model and the necessary phases to provide secure and trusted transactions. In Section 3, we describe the architecture of the proposed e-business framework. We present in Section 4 the implementation of the prototype using the java card technology. In the conclusion we present some planned future work.

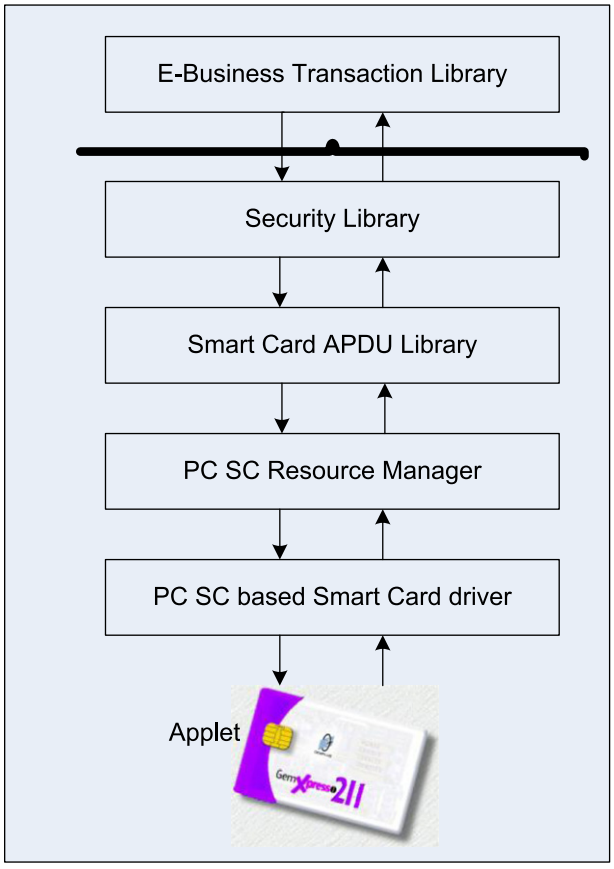

Figure 3 The smart card framework architecture.

2. The transaction e-business model

In (Hirano et al., 2007) an authentication framework was proposed to provide device-oriented authentication and authorization mechanisms for Internet information appliances. The purpose of the framework is to prevent device spoofing, and to restrict unauthorized access to the device in a future ubiquitous network. The framework does not take in charge the entire security of the transition and is limited only to protect the transaction between the device and the user program. Chien et al. (2002) propose an integrated scheme based on hash functions to provide a simpler solution to integrated authentication and access control. However, their scheme is vulnerable to attacks and is limited to authentication and access control scheme based on hash functions. Juang (2004) proposes a multi-server user authentication and key agreement scheme using smart cards. The proposed solution is limited to provide a password authenticated key agreement in multi-server environments. Torres et al. (2007) propose an extended authentication model for the network based on smart cards. The proposed model integrates smart cards in the authentication process applied to a practical payment scenario. This model did not take into consideration security issues related to the transaction.

Our approach proposes an e-business model allowing authentication, integrity and privacy of the transaction by using keys stored in the smart card. This section describes the proposed e-business transaction model. The model describes a secure protocol between two parties involved in the e-business transaction. It allows a client or customer to

perform a secure transaction with a server (supplier). The secure transaction ensures the authenticity, integrity and privacy of the messages exchanged between the client and server during the session. In the following, we designate the term client to identify the party initiating the transaction (customer, client or partner) and the server the party receiving the transaction request (merchant, or supplier). In order to accomplish the transaction, both the client and the server should hold the public key of each other. The e-business model is based on symmetric and asymmetric encryption. The e-business transaction is realized in two phases: the authentication phase and then the integrity and privacy phase. The e-business transaction between the client and the server should start by a mutual authentication between them. Each party (client/server) should authenticate the other party.

2.1. The authentication phase

In order to secure the e-business transaction, each party (client/server) should authenticate the other party. The authentication mechanism grants a user specific access permissions already predefined and stored in the active directory. The authentication process starts by authenticating the client to the server and vice versa. The main reason for authenticating the client is for access control decisions. The majority of existing systems provide unilateral authentication; the server authenticates the client, but the client does not authenticate the server. In this case, the client has no guarantees about the identity of the party at the other side. The client is not sure that he is communicating with a trusted party. He can send confidential information without any guarantee that this information will go to the desired party. For that reason, it is very important that the client authenticates the server before the beginning of any transaction. Once the authentication process succeeds, a secure SSL session is opened between the client and the server.

The authentication mechanism is based on public key cryptography. Public keys are used for authentication purpose by signing and verifying data using private key and public key elements, respectively. The RSA algorithm is used for signing and verifying data (Rivest et al., 1978). 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. The CA issues the certificate by signing the client or server public key (PuK) by using the CA private key (PrKCa). Eqs. (1) and (2) illustrate the signing and verification functions. Both the client and the server can verify the certificate of each other by using the CA public key PuKCa. The client certificate and key pairs are stored in a smart card. The server certificate and key pairs are stored in the active directory.

$$\text{Certificate} = \text{RSA\_Sign}(\text{PuK})\{\text{PrKCa}\} \tag{1}$$

$$\text{PuK} = \text{RSA\_Verify}(\text{Certificate})\{\text{PuKCa}\} \tag{2}$$

The client visits the server web site and asks to log on. The web server returns a web page with the plug-in tag $<\text{EMBED}>$ to get the plug-in executed on the client side. The server signs a generated random number R with its private key PrKSer (Eq. (3)).

$$\text{S1} = \text{RSA\_Sign}(\text{R})\{\text{PrKSer}\} \tag{3}$$

The server sends the random R and the signature S1 to the client. Figure 3 illustrates the authentication mechanism proposed by the e-business model. The client plug-in program verifies the signature S1 using PuKSer stored in the card. The verification algorithm, defined by Eq. (4), is computed by the smart card.

$$\text{R}' = \text{RSA\_Verify}(\text{S1})\{\text{PuKSer}\} \tag{4}$$

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

As illustrated by Eq. (5), the client plug-in computes a second digital signature S2 of the random number retrieved from the received web page using the client private key PrKCli.

$$\text{S2} = \text{RSA\_Sign}(\text{R})\{\text{PrKCli}\} \tag{5}$$

The plug-in then 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 according to Eqs. (6) and (7).

$$\text{PuKCli}' = \text{RSA\_Verify}(\text{CertCard})\{\text{PuKCa}\} \tag{6}$$

$$\text{R}'' = \text{RSA\_Verify}(\text{S2})\{\text{PuKCli}\} \tag{7}$$

If the terms calculated, PuKCli′ and R″, are equal to the terms posted by the client, PuKCl and R′, the server confirms authenticity of the client and a new session is opened.

2.2. The integrity and privacy phase

For many applications and particularly in the financial world, the preservation of data integrity is the most important security requirement. Here, we are concerned with thwarting any event that results in the unauthorized tampering of the data. This includes not only the modification of data but also the addition or deletion. The integrity of the message can be assured by using Public Key Cryptography. Data is encrypted by the private key of the sender and decrypted by the associated public key of the sender. The secret key is stored in the smart card and protected by PIN. This will prevent any person who stole the card from using or accessing the secret key stored on it. To be able to use it, a person should have the physical card and know the PIN.

The need to preserve the secrecy of data exchanged between the parties involved in the transaction is an important factor that should be satisfied. The privacy relates to both the storage of data and the transmission of data in a secure way. When a cryptographic mechanism is employed, it is necessary to preserve the secret and private keys from compromise. The messages exchanged between the two parties should be kept secure. No unauthorized party should be able to explore the data in the message. This challenge is met by encrypting the communication between the parties by using secret key cryptography. The key is kept in the smart card and protected by PIN. The key cannot be read unless the correct PIN is presented (see Fig. 1).

Once the authentication process is completed, the second phase consists of the decryption and integrity verification of the message received. Fig. 2 describes the computation process performed between the client and the server in the integrity and privacy phase. The sender's client program realizes the following computations:

1. Computes a hash digest of the message that will be sent to the server by calling the MD5 hash algorithm.
2. Signs the hash with the sender's private key by calling the RSA cryptogram.
3. Appends the signature to the document.
4. Encrypts the signed document with a symmetric key by using 3DES cryptogram.
5. Encrypts the symmetric key with the receiver's public key using the RSA algorithm, and sends the encrypted message to the server.
6. Once received, the server realizes the inverse computations done by the client. It decrypts the symmetric key with the receiver's private key by using the RSA cryptogram.
7. Decrypt the message by using the 3DES symmetric key decrypted in the previous step.
8. Verifies the signature using the sender's public key using the RSA algorithm. In this step, the verification process will generate the hash digest related to the signature.
9. Computes the expected hash from the received message and compares it with the hash digest calculated in the previous step.

The hash functions as well as the signatures and verifications cryptograms are computed on-board by the smart card. The symmetric key, the private and public keys are stored also in the smart card and protected by PIN. In order to perform the transaction, the client program asks the user to insert a smart card and enter a PIN number. The PIN number grants access and allows the plug-into retrieve keys from the card.

If H1 = H2, privacy and integrity verification succeeds, otherwise the verification fails.

3. Architecture of the proposed framework

The framework proposed in this paper consists of two hierarchical tiers modules. Each tier module is implemented as a class library. The low level tier module handles all communications with smart card through the PC/SC Interface. The PC/SC interface is defined by the international standard protocol ISO 7816. The PC/SC specification ISO 7816 defines a low-level API to allow multiple applications to use the smart card device attached to a system independent of any technology or operating system. This architecture is offering an interesting flexibility and portability, since the application can be developed using any language and use any smart card technology. The APDU library wraps functions related to the smart card APDU applicative commands implemented in the card. The functions implemented at this level are available and used by the second level tier. In Fig. 3, we present the architecture of the e-business transaction framework from the client side view.

3.1. The smart card APDU library

The Smart Card APDU library handles all communication with the smart card through the PC/SC library. All smart card APDU applicative commands proposed by the smart card developed prototype are available for upper layer as well as some general commands to open a session with a card or close a session. Each APDU command will be implemented as a function that prepares the APDU buffer, transmits it to the card, and then gets the message code relative to the execution status of the command. In case the command should produce data, the APDU function should issue another APDU command in order to read the buffer containing data.

3.2. The security library

This library is in charge of combining some functions from the low level library to define them as high level wrapped functions. Each high level function calls functions from the low level library to perform all the needed steps. The security library performs all high level steps related to the authentication, integrity and privacy phases. For example, the "*Login*" function calls all the APDU functions that allocate the "*OpenCard*" structure, establish a session and connect to the card, select the application, ask the user to enter PIN via a graphical user interface and verify it. This architecture makes integration more flexible, since the function "*Login*" can be called many times from different libraries and modules. The function "*signRSA*" loads the keys and the data to be signed, then calls the RSA function implemented in the APDU library and retrieves the signed data.

3.3. The e-business transaction server

The role of the server is to listen on a specific port waiting for transaction requests. Once, the server receives a transaction request, it initiates the first phase to authenticate the client. Once the client is authenticated according the authentication mode requested by the client, the server proceeds to check the privileges assigned to this client stored in the active directory. Based on the transaction requested and the privileges assigned to the user, the server proceeds to execute the transaction in case the client is allowed to do the requested transaction. Otherwise, the server sends a message to the client informing him that he is not authorized to do such a transaction. The information exchange related to the transaction is done securely using an encryption initiated by the client in Section 3.2. In Fig. 4, we show the architecture of the e-business transaction server.

4. The smart card prototype

In order to validate the proposed framework, we describe in the following the implementation of the developed prototype. Any e-business system can use the libraries proposed in the framework to secure the transactions and protect the exchanged information. The prototype is using the APDU and the security libraries developed using the C++ language. The APDU library implements all the APDU functions defined in the standard ISO 7816. An APDU applet was

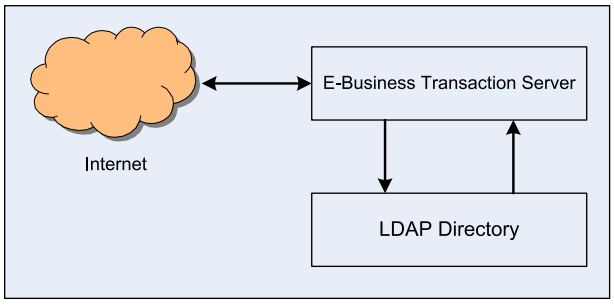

Figure 4 Architecture of the e-business transaction server.

developed for the functions and commands requested by the APDU library. The applet was loaded in a java card open platform, Sun Microsystems (2006). The applet library was developed using the Java language. Once the applet is successfully compiled, we use the Java Card Tool to generate the APDU sequences to be uploaded in the card. The APDU tool is used to exchange the APDU commands with the java card. We can monitor the APDU execution with the java card by examining the returned status code that should be 0x9000 for a successful transaction. In case the transaction fails, the status code returned by the card informs exactly what the error was.

The role of the APDU library is sending an APDU command to the card. In case of an error, the command returns a specific code specific to the error. If the command is executed without error, the card returns the code x9000. For the commands expecting data, the card should extract the data from the reading buffer. The APDU library implements all the APDU commands needed by the security library such as: *allocOpenCard*, *establishContext*, *connectToCard*, *selectFile*, *readFile*, *updateFile*, *verifyPin*, *compute3Des*, *computeRsa*.

The security library wraps APDU commands in more high level functions. The security library contains functions like: *openSession*, *login*, *sign3Des*, *signRSA*. The function *openSession* makes all necessary steps to allocate the *OpenCard* structure, establish context with the resource manager, connect to the card, and select the application. If any APDU command fails, the *openSession* function should receive a specific error code relative to the encountered problem. The *signRSA* function reads the data to be signed, loads the RSA certificate and computes the signature using the RSA function and the private or the public key of the other partner involved in the transaction.

5. Conclusion

In this paper, we presented a smart card framework that allows partners to achieve e-business transactions. The e-business transactions are performed in a secure way by using advanced cryptology computations. From the client side, the computations are performed by a smart card by using credentials (certificates, keys) stored in the card. These credentials are protected by PIN and not accessible to the users. The key pairs stored in the card are also used to create a secure session between the partners involved in the transaction. For that, the smart card should be personalized by loading keys and setting password to be used in e-business transactions. From the server side, after authentication using the credentials stored in the active directory, the server grants the client performing the transactions access according to his privileges stored in the active directory. A smart card system prototype was developed for this purpose. The prototype was implemented based on the proposed framework. Any e-business system can implement the client side using the proposed framework. The developed prototype handles only the client side. We developed a light version of the server in order to use the framework and validate our prototype. In the near future, we are planning to implement the server side by using the active directory to store and use credentials and privileges.

References

Chen, Y.C., Yeh, L.Y., 2005. "An efficient authentication and access control scheme using smart cards". Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05), IEEE Computer Society.

Chien, H., Jan, J., Tseng, Y., 2002. An efficient and practical solution to remote authentication: smart card. Computers and Security 21 (4), 372–375.

Hirano, M., Okuda, T., Yamaguchi, S. 2007. "Application for a simple device authentication framework: device authentication middleware using novel smart card software". Proceedings of the International Symposium on Applications and the Internet Workshops (SAINTW'07). Computer Society.

Juang, W.S., 2004. Efficient multi-server password authenticated key agreement using smart cards. Consumer Electronics, IEEE Transactions 50 (1), 251–255.

Mornar, V., Palavra, D., Kalpic, D. 2004. Application of smart cards in distributed information systems. 2nd International Conference in Information Technology Interfaces IT/2004, Croatia, pp. 187–192.

Rivest, R. et al., 1978. A method for obtaining digital signatures and public-key cryptosystems. Communication of the ACM 21 (2), 120–126.

Sun Microsystems. "Java Card™ Platform, Version 2.2.2". 2006. Development Kit User's Guide. Sun Microsystems.

Torres, J., Izquierdo, A., Sierra, J.S. 2007. Advances in network smart cards authentication. Computer Network 51, 2249-2261.

Wassim R.M., Sheltami, T., Sallout, M. 2007. "A smart card based prepaid electricity system". IEEE Xplore.