published as: X. Zhang, A. Dulaunoy, Christoph Meinel: Enhance opensst Protocol´s Security with Smart Card;
Proceedings of the IADIS International Conference WWW/Internet (WWW/Internet 2003);
WWW/Internet 2003 Algarve (Portugal), 2003, pp. 565-571.

ENHANCE OPENSST PROTOCOL'S SECURITY WITH SMART CARD[1]

Xinhua Zhang
Luxembourg International Advanced Studies in Information Technologies
162a, avenue de la FaiencerieL-1511 LUXEMBOURG
zhang@ti.uni-trier.de

Alexandre Dulaunoy
Conostix S.A
66.rue de Luxembourg L-4221 Esch-Sur-Alzette
adulau@conostix.com

Christoph Meinel
computer department, university of Trier
D54286 Trier Germany
meinel@uni-trier.de

ABSTRACT

As an open source project, Open Simple Secure Transaction (OpenSST) protocol aims to be a secure and transaction-oriented protocol for the unsecured network. At present a prototypes has been implemented. In this paper, we give firstly an overview of OpenSST, and then we describe the key store and authentication mechanism of the prototype, as well as make an analysis on its security. Then we introduce a security-enhanced model based on this prototype, which integrates Smart Card into the OpenSST protocol. This model demonstrates how to use Smart Card to improve the public key authentication process in a secure network protocol.

KEYWORDS

OpenSST, Smart Card, Key store, Authentication, Public Key, Private Key

1. INTRODUCTION

The rapid increase of interest in Internet, the Web, and Electronic Commerce raises a number of security issues. The main issues are usually listed as confidentiality, integrity, authentication and non-repudiation [4]. Here confidentiality ensures that information exchanging between two parts is kept secret from third parts, even if that information exchanges through an insecure medium. Integrity provides a way for the recipient to determine whether any modifications are made when the information is transmitting. Authentication means to verify sender and receiver in order to prove the person is who he claims to be. Non-repudiation prevents both sender and receiver from denying a specific transaction, which is also a proof of the integrity and origin of data exchanged in the transaction.

In order to cope with these issues, many solutions are put forward in a broad variety of security levels. In [1] authors present a new secure protocol called Open Simple Secure Transaction. This protocol tries to realize these four basic security requirements and aims to be a secure and transaction oriented protocol for the unsecured network. However, OpenSST is immature now, some aspects need an ulterior improvement, especially for the mechanism of its key store and user authentication.

In the proceeding of IADIS international conference WWW/Internet 2003, Algarve Portugal, November 5th~8th 2003.

On the other hand, Smart Cards are capable of providing a secure storage and computation environment for a wide range of user credentials such as Keys, certificates and passwords. For this, Smart Cards have been widely recognized as an efficient way to greatly improve the security of a user authentication process [6]. Therefore, integrating with Smart Card should improve OpenSST security remarkably.

In the following, we will firstly give a short overview of Smart Card technology, then we will introduce OpenSST protocol infrastructure and illustrate an OpenSST prototype. A security-enhanced model, which combines with a Smart Card, is introduced in the fourth section. In the last section, we summarize and add an outlook to further development of OpenSST protocol.

2. OVERVIEW OF SMART CARD

A Smart Card is a device that feels and looks like a credit card, but contains a small computer that combines dedicated hardware and software with more standard components such as various types of memory and small operating systems. Depending on the type of embedded chip, Smart Cards fall into three major categories:

- Stored value cards are the most basic form of Smart Card. These cards are not much more than memory on a card, they are best thought of as an electronic version of a magnetic stripe card.
- Microprocessor Memory Cards are much like a mini computer and include RAM, ROM, and EEPROM. These cards typically have more complex processors and additional tamper resistance features.
- Cryptographic Cards are high-end microprocessor memory cards with additional support for cryptographic operations (digital signatures and encryption). These cards are designed to allow secure storage of private keys (or other secret keys), and perform the actual cryptographic functions on the Smart Card itself. In this way, the private key need never leave the Smart Card. Since the EEPROM of these cards is designed to be tamper-resistant, unauthorized individuals are unable to hack the card secrets-it's virtually "hacker-resistant" . As a result, Cryptographic Cards play an essential part of many public/private key systems.

3. OPEN SIMPLE SECURE TRANSACTION (OPENSST)

OpenSST is an open source project, it is both a protocol for securely exchanging transactions and a message format to encapsulate these transactions in XML form. Cryptography is one of the primary technologies that the OpenSST use to provide secure service. Normally, OpenSST consists of two parts: OpenSST proxy lies in side of client and OpenSST server lies in side of server. OpenSST will firstly establish a secure session for each transaction before this transaction takes place. This secure session is established using public-key authentication with key exchange between the proxy and server to derive a unique session key that can then be used to ensure data integrity and confidentiality throughout the session. In the prototype, the symmetric key algorithm used for session encryption is the very popular DES [10], which is very suitable for encrypting or decrypting big data with an acceptable speed. And the asymmetric algorithm used for authentication and signature is also a typical algorithm RSA [13].

3.1 An OpenSST prototype

Figure 1 shows an OpenSST prototype based on HTTP protocol. Although it is just a rudiment, we still can insight into its main feature and basic work principle. In this prototype, both the OpenSST proxy and Server are running behind, and the browser is the main interface through which user complete his transaction. The Browser and OpenSST proxy are installed on the same client computer. OpenSST server can be either in the same computer with web server and authentication server, or in an individual computer. Here, we discuss mainly its authentication mechanism.

In this prototype, an authentication consists of the following steps:

→ Firstly the browser prompt the user to select a key ID and input a password which is used to protect the private key stored in the key store file, and transfer this password to the proxy (1);

→ After the proxy get user's password, it open the key store file, and read out the private key with the password (2);
→ The proxy then sign the user ID and key ID (if there are many key belonging to the user) with the selected private key identified by key ID, and send this signature with user ID and key ID to the OpenSST server (3);
→ The OpenSST server then transfers this information to the authentication server (4);
→ According as this user ID and key ID, Authentication server chooses the corresponding public key from database to verify the user's signature. At last it respond the verification result to the OpenSST server (5);

Hereto, the whole authentication process is over.

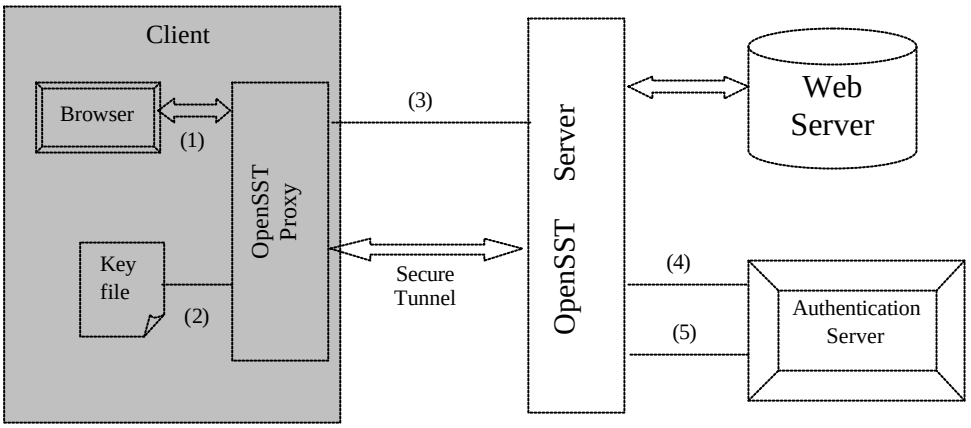

Figure 1 A OpenSST prototype

Once the authentication is successful, the second stage for the OpenSST is to establish a secure tunnel between the OpenSST proxy and the server. This secure tunnel is established as following: firstly the OpenSST server assigns a session ID for this session, and generates, at the same time, a secure key created by a symmetric algorithm to encrypt the transaction. Thereafter the server encrypts the secure key and the session ID with the user's public key. The server also computes a hash value with this cryptograph, and signs the hash value. And then the server sends the signature together with the cryptograph to the proxy. On the client side, the proxy will verify the signature and decrypt this message with the private key to get the secure key. From now no, all the transaction exchanging between the client and the server will be encrypted with this secure key. A secure transaction tunnel is established. OpenSST is designed to allow each transaction to be signed by both sides, which is benefit for non-reputation.

3.2 Analysis

As figure 1 show, the private key of the user is stored in a file in the client. Although this key itself is protected by a hash based on a password [2]. Thus an attacker can try to guess the password using a dictionary attack if he has chance to access this key store file. Because any password that the user can remember without writing down is in principle subject to some form of dictionary attack [5]. To defense against such an attack, we have to make the password space extremely large, or to slow down the operation of checking a single password. However, neither of these methods is very practical.

Since the key-store file is stored in a specific computer, if the user wants to authentication himself in another computer, he has to copy his key-store file to this computer. It is possible that he don't know whether or not there are some password crackers [8] or keystroke capture programs installed in this computer in advance, so he is possibly at the risk of exposing his private key to a hacker. Besides, if user does not work on his own computer, he has to remember to delete the copy of key store file after he finished the transaction. However it is also possible for people to forget to do this.

In this prototype, when the authentication server generates a public key pair for a user, it simultaneously generates a user ID and a key ID to bind the user and the key, which are used for authentication. No certificate is used in this prototype. This kind of solution is feasible in the scope of a small business or a compact team of people. But beyond that scope, application of OpenSST would be difficult.

4. IMPROVEMENT WITH SMART CARD

From the security analysis above, we can find there are several vulnerabilities in this prototype. In the following we will present a security-enhanced authentication model for the OpenSST by using Smart Card. In this model, authentication server is responsible for not only authenticating the user, but also issuing certificate, to some extent, it play the role of a Certificate Authority (CA). In [7], the author calls the part-function CA as certificate issuer (CI). In fact, it is easy to write a CI program with the free software available. In this model, the user's public key pair is generated by the Certificate Issuer, and the private key is only stored in user's Smart Card. At the same time, a public key certificate signed by this Certificate Issuer is also stored in the Smart Card.

For achieving high-level security, one principle must be satisfied in this enhanced model. That is, although the user's public key pair is created by the CI, the CI should not be aware of all key materials of a users. In other words, it is impossible for the CI to pretend a user to sing a message, or worse, the CI leak the private key to the public domain by accident or by intent. In this new prototype, the user's public key pairs are generated directly on user's Smart Card by a tool provided by the Card manufactory, and only the public key is transmitted to the CI for certification. This guarantees that the user is the unique possessor of private key at any time.

4.1 Work principle

Figure 2 shows how the authentication is performed with a Smart Card. Because we explain mainly the authentication procedure, and the information exchanging between the browser and Web server takes place after successful authentication, we draw both components in broken line, and do not discuss it later.

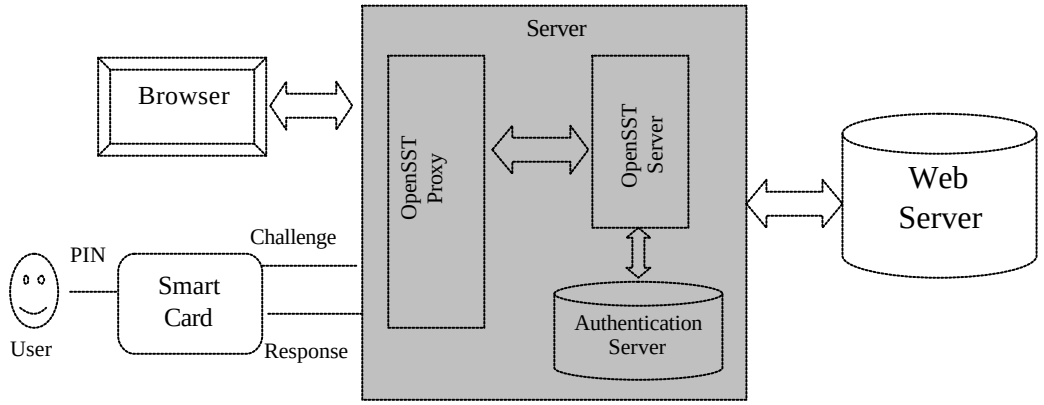

Figure 2 Improvement with smart card

In fact, in this situation it involves two authentications.

One is that the cardholder authenticates himself to the card; this authentication is an inherent advantage of Smart Card. The typical method used to authenticate the user to the card involves the input of a Personal Identification Number (PIN), which is verified by the Smart Card. This verification is done by comparing the PIN with a reference PIN stored in a secret area of the non-volatile memory. Usually only three attempts are allowed before the Smart Card locks out.

Another is that the user authenticate himself to the server, this authentication involves the user of cryptographic protocols [9]. Since the authentication achieves between the Smart Card and Authentication server primarily, and during the authentication process, the OpenSST proxy and server are mainly responsible for conveying the information, they work as agents. In order to understand easily, as a whole, we regard simply all them as a server.

Authentication can be performed as follows: firstly, the server generates a random challenge and sends to the Smart Card. Then the Smart Card uses the user's private key to generate a digital signature over the challenge. As response, the digital signature together with the certificate associated with the private key in the smart card is sent to the server. At last, the server verifies the certificate and then uses the public key contained in the certificate to verify the signature. Once authentication is successful, a secure tunnel between OpenSST proxy and OpenSST server can be set up as we discussed in section 3.2.

For OpenSST protocol is specified in XML, and there is an element named <timestamp> in each message format of OpenSST. In addition, before a session key is accepted between the proxy and server, every exchanged message will be signed. So even though an eavesdropper records the authentication process, it is still impossible for him to get afterward access to the server by using these recorded packets. OpenSST provides itself the function against anti reply attack.

4.2 Implementation

The OpenSST prototype mentioned above is completely implemented in Java, include the part of smart card. In order to make it possible that the original prototype and the Smart Card application can work separately, the model based development method is adopted. By this way, we only need to modify very little on the original prototype to provide an interface for communication with Smart card, and then both can communicate with each other.

In this model, we choose SARTCOS SPK2.3, a kind of Cryptographic Cards, to implement the functions of key-storage and authentication. This card provides DES, 3DES, and RSA, DSA algorithm. However, there is a small pity that this card does not provide any Java function, which can be invoked to generate a public key pair by an application. So the public key pair has to be generated during initialization of card. Fortunately, the company provides a good tool, through which user can easily create a master file, and include the value of components of key pair. Thinking from another angle, the pity becomes a good thing, because it prevents the IC from being aware of user's private.

OpenCard Framework (OCF) [11] is collaboration between several key players in the smart card industry to form common standards for communication and managing multi-application smart cards. We do the program in terms of OCF standard, and the program focus on implementing some CardService layer functions, such as:

public byte[] sign (PrivateKeyFile privkeyfile,byte[ ] data, String Password, String dest)
public boolean verify(PublicKeyFile pubkeyfile, byte[ ] data, byte [ ] signature)
public java.security.interfaces.RSAPublicKey getPublicKey (CardFilePath path,int KeyID)
……

Because the private do never leave Smart Card, the process of signing and verifying is done on the card. Following figure gives a simple illustration of the implementation.

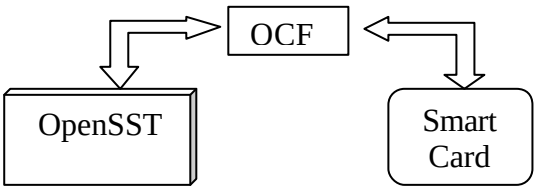

Figure 3 OCF connect OpenSST with Smart Card

4.3 Analysis

When Smart Card is used in OpenSST, it will enhance OpenSST's security from following aspects:

- Secure Storage for private keys is a unique attribute of the Smart Card [14]. In this solution, the user's private key is securely and uniquely stored on Smart Card, and all content in a Smart Card is protected by the card operating system.
- Non-repudiation is another the most unique and compelling advantage of using Smart Card authentication solution in OpenSST. Of course, essentially, the security service of non-repudiation realized by a combination of asymmetric cryptography and certification of key material. However non-repudiation is often violated since the key is compromised. In this enhanced model, because all cryptographic Smart Cards are designed to ensure that a user's private key never leaves the Smart Card. That means the private key cannot be copied, replicated or misused by an invalid individual. As a result, we can be extremely confident the private key used in OpenSST is always in the sole possession of the user, thus we have undeniable evidence that connects a specific user to each transaction.
- Mobility is another value-added quality of using Smart Card in OpenSST. In old prototype, the user is limited to a specific desktop system that contains his key store file. Because Smart Card is such a device that can be easily carried in a user's wallet or packet, and its mobility does not sacrifice its security in the least, user can conveniently and securely authenticate himself from different clients on which OpenSST proxy is installed.

5. CONCLUSION AND FUTURE WORK:

The protocol OpenSST is still rather young and future extensions and approach supplementary researches are important. On the other hand, the Smart Card enhances the public keys authentication process by serving as a secure store for the private-key material and as a cryptographic engine for performing a digital signature or key exchange operation. In this paper, we choose Smart Card as a kind of key store and authentication solution, as a result, it enhances remarkably the security of OpenSST protocol. Since years, the PKI (Public Key Infrastructure) has become a de facto standard for securing net-based communications and transactions [3, 12]. Especially in a closed system, a PKI system satisfies almost all of the marketplace's need, including four basic security requirements. Thus, our next research interest will focus on: how to establish a simple and efficient PKI model based on OpenSST protocol. The improved OpenSST supported by Smart Card and PKI solutions will surely offer numerous advantages in practice application.

ACKNOWLEDGEMENT

The authors would like to thank all the members of the OpenSST project, especially thank Sébastien Stormacq, who designed and programmed the Java-based prototype of OpenSST.

REFERENCES

[1] Alexandre Dulaunoy, Sebastien Stormacq, 2002. *OpenSST: Open Simply Secure Transaction*. URL: http://www.foo.be/current/opensst/

[2] Alexandre Dulaunoy, April 2002. *A XML Schema for the OpenSST message format.* Internal Conostix document

[3] Aaudun Josang, et al, PKI Seeks a Trusting Relationship. *Proceeding of Fifth Australasian Conference on Information Security and Privacy ACISP 2000*, Brisbane Australia

[4] Bruce Schneier, 1996. *Applied Cryptography Protocols, Algorithm and Source Code in C.* second edition, John Wiley & Sons, Inc. ,New York, USA

[5] D.N.Hoover, B.N.Kausik, 1999. Software Smart Cards via Cryptographic Camouflage, *IEEE symposium on Security and Privacy* pp 208-215

[6] E.-M. Hamann, et al, 2001. Securing e-business applications using Smart Cards, *IBM Systems Journal,* Vol 40. No 3, pp 635-647

[7] Kaijun Tan, March 2002. Building Your Appropriate Certificate-based Trust Mechanism For Secure Communications *In Rrainbow Technologies* V1.2

[8] Mark Taber, et al, 1998. *Maximum Security.* second edition, SAMS Publishing, Indianapolis, Indiana, USA

[9] M.Y.Rhee, 1994. *Cryptography and Secure Communication* McGraw-Hill, Highstown, N.J. pp 449-457

[10] National Institute of Standards and Technology. 1994. FIPS 46-3. *The Data Encryption standard*.

[11] OpenCard Consortium (2000), *OpenCard Programmer's Guide*, http://www.opencard.org

[12] Peter Gutmann, 2002. PKI: It's not dead, just resting, *IEEE Computer society*, Vol. 8, pp 41-49

[13] R.Rivest, A.Shamir, and L.Adleman, 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(1978), pp 120-126

[14] RSA Security Inc. The cryptographic Smart Card: a Portable, Integrated Security Platform. CSC WP 0301