published as: X. Zhang, Christoph Meinel, A. Dulaunoy: A Security Improved opensst Prototype combining with Smart Card;
In Proceedings of International Conference on Computer Networks and Mobile Computing (ICCNMC 2003);
IEEE ICCNMC 2003 Shanghai (China), 2003, pp. 358-361.

A Security Improved OpenSST Prototype Combining with Smart Card

Xinhua Zhang
LIASIT
162a, avenue de la Faiencerie
L-1511 Luxembourg
zhang@ti.uni-trier.de

Christoph Meinel
University of Trier
D54286 Trier Germany
meinel@uni-trier.de

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

Abstract

Open Simple Secure Transaction (OpenSST) protocol aims to be a secure and transaction-oriented protocol for the unsecured network. OpenSST is a open source project, at present a simple prototype based on HTTP protocol has been implemented. In this paper, we give firstly an overview of openSST, and then we describe the basic architecture of this prototype, as well as make an analysis on its security. Then we introduce a security-enhanced model based on this prototype, which utilize the tamper-resistant nature of Smart Card. We will discuss several possible ways of integrating Smart Card into the OpenSST.

1. Introduction

In the recent past, the breakthrough of the Internet as a worldwide-accepted communication medium, business platform has stimulated many works in the area of high security and reliable transaction protocols. These works aim to provide user with the guarantee on confidentiality, integrity, authentication and non-repudiation [3]. 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 the proceeding of the international conference of Computer Networks and Mobile Computing (ICCNMC 03), Shanghai China, October 20th ~ 23th 2003.

Among almost all secure protocols, cryptography is hard-core used to realize confidentiality, authentication, integrity and non-repudiation. Although cryptography solves security problems in open networks, it also brings forward the requirement of secure key storage. On the other hand, it is agued that 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.

In this paper, we will firstly give a short overview of Smart Card technology, and then we will make a brief introduction of the protocol: Open Secure Simple Transaction (OpenSST) [1, 2]. After this section, we will principally narrate and analysis an OpenSST prototype without Smart Card. A security-enhanced model, which combines with a Smart Card, is presented in the fifth section. In the last section, we summarize and add an outlook to further development of OpenSST protocol.

2. Smart Card

A Smart Card is a device that feels and looks like a credit card and can even be used as one, but it has a major advantage over credit magnetic strips: its own processor. The smart card processor allows a higher level of security than a simple magnetic strip. It gives user the ability to do computations internal to the smart card (primarily signature and decryption). It also allows user to block out certain parts of the card by PIN numbers. Smart cards don¡ft allow direct access to stored information. With the processor controlling access to the memory on the card, commands have to be sent to the processor on the card itself to get at the information on the card. This allows the processor to require the user to authenticate them to the smart card before information is returned. 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;

- *Microprocessor Memory Cards* are much like a mini computer and include RAM, ROM, and EEPROM;
- *Cryptographic Cards* are high-end microprocessor memory cards with additional supports for cryptographic operations (digital signatures and encryption).

3. Open Simple Secure Transaction

OpenSST is an open source project, it aims to be a secure and transaction oriented protocol for the unsecured network. OpenSST is both a protocol for securely exchanging transactions and a message format to encapsulate these transactions. OpenSST is designed to allow each transaction to be signed, which is benefit for non-reputation. OpenSST specifies a simple message format in XML syntax, and it recurs to cryptography 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.

4. A prototype without Smart Card

In order to demonstrate the OpenSST protocol, a prototype, the OpenSST Http Proxy implementation, has been developed. Although this prototype is just one approach of OpenSST, we still can insight into its main features 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. Figure 1 shows the architecture and mechanism of this prototype.

4.1 Architecture and mechanism

In this prototype, an entire transaction consists of two steps: authentication and establishment of secure tunnel.

Authentication includes the following steps: (1) the user gives a key ID and corresponding password from the browser, which is used to protect the private key stored in the key store file; (2) the proxy opens the key store file, and reads out the private key with the password; (3) The proxy then sign the user ID and key ID (if the user has more than one key) with the selected private key identified by key ID, and send this signature with user ID and key ID to the OpenSST server; (4) The OpenSST server then transfers this information to the authentication server; (5) Authentication server verify the user¡fs signature with the public key corresponding to user ID and key ID, and responds the verification result to the OpenSST server. Hereto, the whole authentication process is over.

Once the authentication is successful, OpenSST is to establish a secure tunnel between the OpenSST proxy and the server as following: firstly the OpenSST server assigns a session ID for this session, and generates a secure key with a symmetric algorithm; Thereafter the server encrypts the secure key and the session ID with the user¡fs public key; A hash value is also computed with this cryptograph, then the server signs the hash and sends it together with the cryptograph to the proxy. On the client side, the proxy verifies the signature and decrypts this message with the private key to get the secure key. After both the proxy and server have the secure key, each transaction between them will be encrypted with this secure key. A secure transaction tunnel is established.

4.2. Security analysis

As mentioned above, the private key of the user is stored in software key container, a file in the client, where the key itself is encrypted by 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 [4].

Presuming that the user wants to authenticate him in a computer without his key-story file, and then he has to copy the file to this computer. It is possible that he don¡ft know whether or not there are some password crackers [6] or keystroke capture programs already installed in this computer in advance, so he is possibly at the risk of exposing his private key to a hacker. Besides, if the user does not work on his own computer, he must remember to delete the copy of key-store file after he finished the

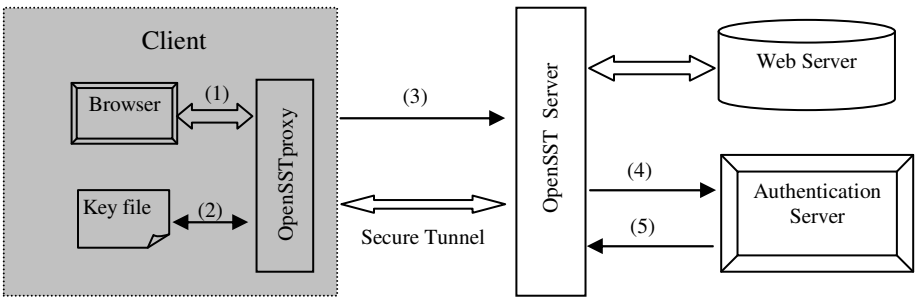

Figure 1 **An OpenSST prototype without Smart Card**

transaction. However it is also possible for people to forget to do this.

5. A new model with Smart Card

Cryptographic Smart card is an important building block in many modern security applications. In particular, its tamper-resistant packaging, low cost, inherent portability, and loose coupling to the host make it especially attractive for use as secret key storage tokens when the host cannot be trusted to itself store a secret key. In the following we will present a security-enhanced OpenSST model combined with Smart Card. In this model, authentication server is responsible for a part of functions of a Certificate Authority (CA). In [5], the author calls this part-function CA as certificate issuer (CI), it not only authenticates the user, but also issues certificate.

Figure 2 shows the architecture of the new model. Due to we introduce mainly the process between the Smart Card and OpenSST, we draw other components in broken line, and regard them whole as server.

5.1 Using Smart Card for authentication

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 [8]. This authentication can be implemented by symmetric algorithm or asymmetric algorithm.

5.1.1 Authentication with symmetric algorithm. In this model, there is no need of certificate. The solution is based on use of pseudorandom functions. The server (exactly be CI or authentication server) firstly generates a master key MK, which is used to compute the secret key on each Smart Card. Because every Smart Card has a unique serial number SN when it is manufactured. When the CI issue a Smart Card to a user, it firstly computes $K_{sc} = f_{MK}(SN)$, which is the pseudorandom function keyed by the mater key and evaluated at the serial number, then the CI stores this K_{sc} in the Smart Card. Besides, the CI should also maintain a user-list which link an user (maybe identified by an ID number) to his Smart Card serial number.

When a user wants to authenticate himself to the server, he encrypts his user ID with K_{sc} and sends this cipher together with his Smart Card SN to the server. The server will firstly use its master key MK and SN sent by the user to regenerate K_{sc} and then decrypt the cipher. At last, it compares the user ID and user¡fs SN with the record in the user list to verify the user.

The advantage of this model is:

- Symmetric algorithm is faster than asymmetric algorithm;
- For the server can recomputed the user¡fs secret key with the serial number and the master key, the server need not store a large number of secret keys for all users. Whereas the server can easily gain a secret key with a user when they need.

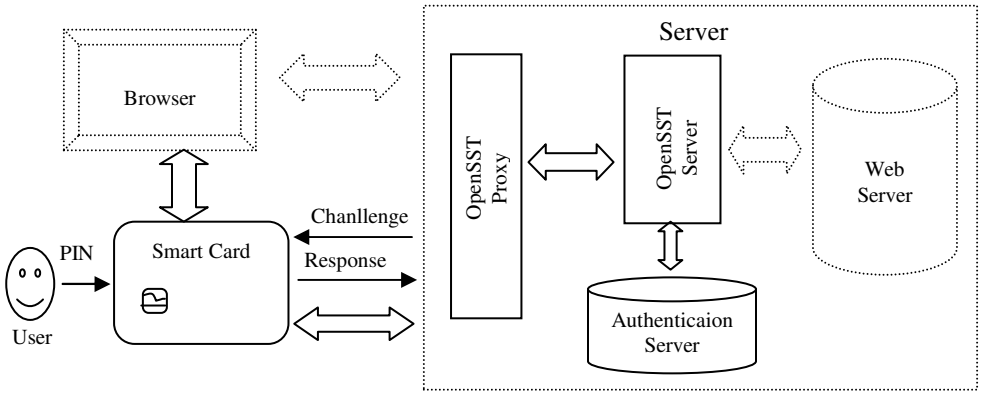

Figure 2 **A new model with Smart Card**

5.1.2 Authentication with asymmetric algorithm. In this model, the user¡fs public key pair is generated by the

authentication server, and the private key is only stored in user¡fls Smart Card At the same time, a public key certificate signed by the Certificate Issuer is also stored in the Smart Card. 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¡fls 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.

5.2 Using Smart Card for encrypting message

By the reason of Smart Card has only limited resource such as CPU, ROM and RAM etc. It cannot process data at nearly the bandwidth of the host to which it is attached. Smart Card is often used in such situation that the key stored on the card is used only occasionally and speed requirements are minimal, for example, digital signatures of message digests and challenge-response authentication protocols.

While in an application that requires large bulk encryption with Smart Card based key management, a possible scheme is that shifting work to the host processor, the principle is that it should not increase the trust requirements of old system. This scheme is difficult but interesting. Once we can implement this scheme in OpenSST, the secure tunnel can be set up directly with the secret key stored in Smart Card. A similar protocol RKEP [7] has resulted very ideal performance, which is used mainly in the area of File Encrypt System, however it prove partly that it is feasible to implement this function in OpenSST.

5.3 Benefit

When Smart Card is used in OpenSST, it will enhance OpenSST¡fls security from several aspects

First, secure storage for private keys is a unique attribute of the Smart Card [10]. Second, 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¡fls 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.

Non-repudiation is another advantage. Because all cryptographic Smart Cards are designed to ensure that a user¡fls private key never leaves the Smart Card That means the private key cannot be copied, replicated or misused by an invalid individual. So we can be extremely confident the private key used in OpenSST is always in the sole possession of the user who has done the transaction.

6. Conclusion and future work

The protocol OpenSST is still rather young and future extensions and approach supplementary researches are important. In this paper, we present a model of integrating Smart Card into OpenSST, which 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 [9]. Especially in a closed system, A PKI system satisfies almost all of the marketplace¡fls needs, 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.

7. References

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

[4] D.N .Hoover, B.N.Kausik, 1999. *Software Smart Cards via Cryptographic Camouflage*, IEEE symposium on Security and Privacy

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

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

[7] Matt Blaze, High-Bandwidth Encryption with Low-Bandwidth Smart cards. *Cambridge Workshop on Fast Software Encryption*, February 1996

[8] M.Y.Rhee, 1994. *Cryptography and Secure Communication* McGraw-Hill, Highstown, N.J. pp449-457

[9] Peter Gutmann, 2002. PKI: It¡fls not dead, just resting *in IEEE Computer society*, Vol. 8, pp 41-49

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