SCIS 2004 The 2004 Symposium on
Cryptography and Information Security
Sendai, Japan, Jan.27-30, 2004
The Institute of Electronics,
Information and Communication Engineers

On the Design and Implementation of a Secure Time-Stamping Service

Chung-Huang Yang[1] Chih-Ching Yeh[2] Li-Ling Hsu[3]

Abstract— As the Internet grows in scale almost every year, security measures are expected to become all the more important on the Internet. A Time-stamping authority (TSA) is a trusted authority which provides a proof that a datum existed before a particular time. In this research, we implemented a RFC-3161 compliant time-stamping service over the Internet. The TSA server software was implemented on Linux platform using C language while the TSA client software was implemented on Windows platform using C++Builder tool. Our TSA server is also equipped with an USB reader and an off-the-shelf smart card to store and retrieve private key of the TSA.

Keywords: Time-stamping, implementation, digital signature, PKI, IC card

1 Introduction

A time-stamping authority (TSA) [1,2,3,4] is a usually operated as trusted third party which provides a proof that a datum existed before a particular time. The time-stamping service (TTS) become a part of public-key infrastructures (PKI) [5,6] and is a crucial part of many secure network applications such as Kerberos, online stock trading, etc.

As documents or hash values of documents are send to TSA for time-stamping, the TSA server will response with a time-stamping token in order to indicate that the (hashed) datum existed at a particular point in time. There are many situations where we need to certify the date and time that some data was created or modified [7]. The TSA will link time-stamping token in a tamper-evident chain and it is computationally impossible for anyone to insert a document previously unseen in the middle of the chain [8,9].

[1]Institute of Information and Computer Education, National Kaohsiung Normal University, 116, Ho Ping First Road, Kaohsiung 802, TAIWAN, http://crypto.nknu.edu.tw/, chyang@computer.org

[2]dexteryeh@mail3.cy.edu.tw

[3]Dept. of Information Management, National Kaohsiung First University of Science and Technology, 2, Juoyue Road, Nantz District, Kaohsiung 811, TAIWAN, karenhsu@ccms.nkfust.edu.tw

IC cards [10] are much more difficult to duplicate than magnetic strip cards and cryptographic functions can be implemented inside these cards. With substantial cost reductions and the ability to handle multiple applications on a single card, the IC cards are about to enter a period of the rapid growth.

In this paper, we present our effort in the design and implementation of CA/PKI on the PC Windows environment. Borland C++Builder 6 [11] is used as a software tool to develop Internet-based TTS client. The open source software provided by the OpenSSL project [12,13] is used as a middleware to build a TTS server on the Linux environment. To safeguard private key, our TSA server is equipped with an off-the-shelf IC card with USB reader from Omnikey [14] and TTS response certificates are signed by private key stored on IC cards. The communication protocol between TTS clients and TTS server is followed IETF RFC3161 standard [15].

2 Implementation of TSA Server

OpenSSL [15,16] is an open-source implementation of SSL/TLS protocols for Linux and Windows platforms. It also contains basic cryptography functions, such as message encryption and digital signature. We installed and set up OpenSSL software on a Linux machine with extra C codes to provide RFC3161-compatible

time-stamping format so that they will act as TSA. Figure 1 shows system diagram of our time-stamping service (TTS) over the Internet.

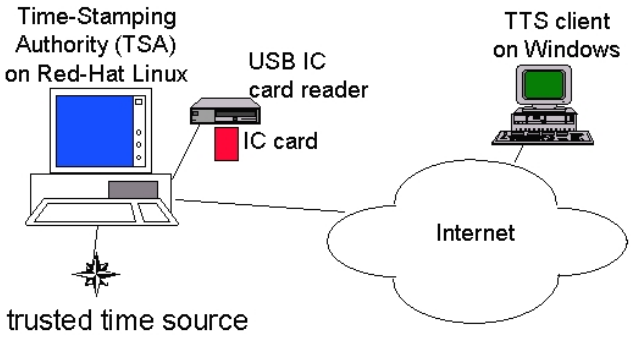

Figure 1: System Architecture of the TTS

The operation of a TSA will be required to have accurate time source. Our TSA is equipped with a PCI-interface Time Processor from TrueTime, Inc. (acquired by Symmetricom Inc. in 2002) [16].

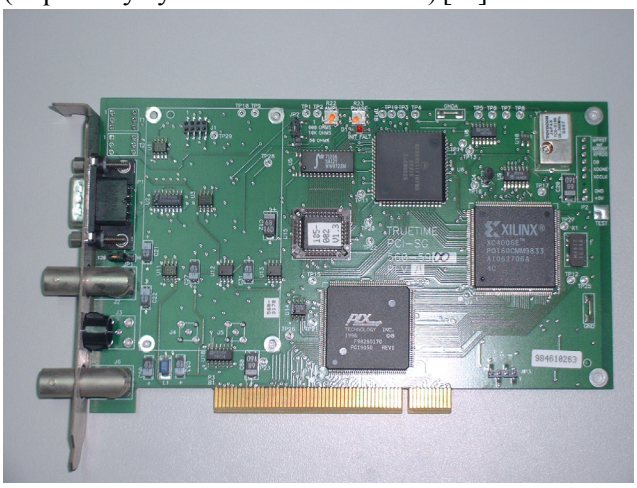

Figure 2: Time Processor Used in Our TSA

TSA CA fundamentally performs the generation of time-stamping certificates that are digitally signed by the TSA's private key. At present, we use an IC card with the Infineon SLE 4418 [17] memory chip to store TSA's private key. This IC card chip contains 1024 bytes of EEPROM memory and its contents are protected by 4 hexadecimal digits, the programmable security code (PSC). The USB IC reader with Linux driver is the Omnikey's CardMan reader [18].

3 Implementation of TSA Client

We developed a TTS client software on Windows platforms which interfaces with TTS server through HTTP protocol over the Internet. The client software, developed using C++Builder version 6.0 [11] tool from Borland Software Corp. C++Builder is an object-oriented, visual programming environment, it provides tools to develop, test, debug, and deploy applications, including a large library of reusable components, a suite of design tools, application and form templates, and programming wizards.

A TTS client allows creating a time-stamping request by selecting a file and choosing a message-digest algorithm (such as MD5 or SHA-1), then sends the time-stamping request to the TSA. Basic operation of TTS client is illustrated in Figure 3.

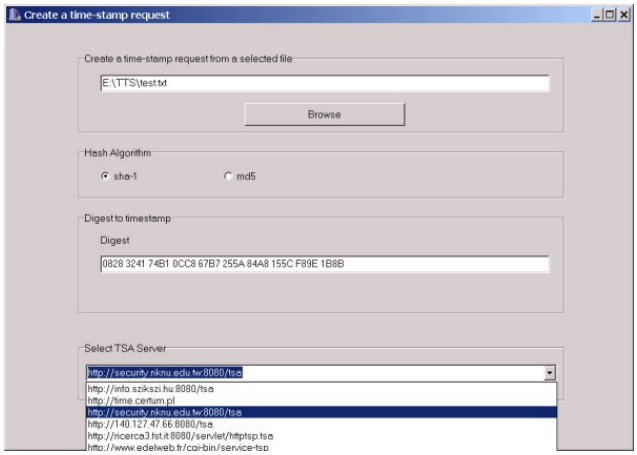

Figure 3: Creation of a TTS Request with TTS Client Software

Upon receiving time-stamping response from the TTS server, our TTS client software could parse and display the response in a meaningful manner. The TTS client software could also save time-stamping response and verify it against presentation of the original (hashed) datum on which the time-stamping request had been sent to TTA. Figure 4 displays the client software during verifying.

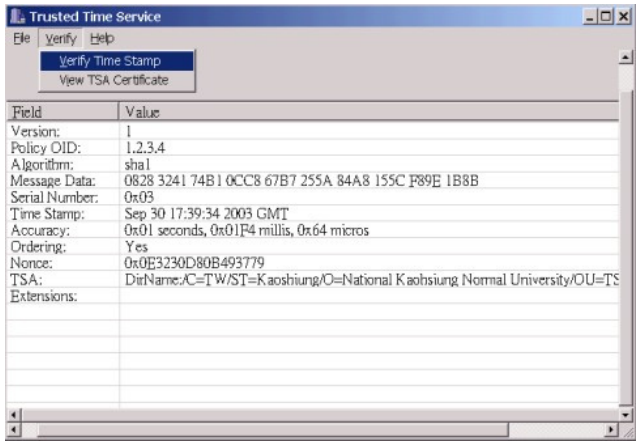

Figure 4: Verification of a TTS Response

4 Conclusions

PKIs will become more and more prevalent in the near future while time-stamping service (TTS) is a part of PKI. A secure, certifiable, and auditable TTS solution will be useful in e-government and e-commerce applications. In this paper, we have described our preliminary results on implementing RFC3161-compatible TTS client and server software.

References:

[1] S. Haber and W.S. Stornetta, "How to Time-Stamp a Digital Document," *Journal of Cryptology*, Vol. 3, No. 2, 1991, pp. 99-111.

[2] S. Haber, B. Kaliski, and W. Stornetta, "How Do Digital Timestamps Support Digital Signatures?," *Cryptobytes*, Vol. 1, No. 3, pp 14-15, RSA Laboratories, Autumn 1995.

[3] A. Kakura and S. Naito, "A Secure and Trusted Time Stamping Authority," *Internet Workshop*, 1999, pp. 88-93.

[4] H. Massias and J.J. Quisquater, "Time and Cryptography," *TIMESEC Technical Report*, 1997

[5] C. Adams and S. Lloyd, *Understanding Public-Key Infrastructure*, Macmillan Technical Publishing, 1999.

[6] A. Nash, W. Duan, C. Joseph, and D. Brink, PKI: *Implementing and Managing E-Security*, McGraw-Hill, 2001.

[7] Datum.com, "The Importance of Time," http:// www.trusted-time.com/

[8] Surety.com, "Digital Notary Service Technical Overview," http://www.surety.com/

[9] P.A.S. Ward and D.J. Tayler, "A Hierarchical Cluster Algorithm for Dynamic, Centralize Timestamps," *International Conference on Distributed Computing Systems,* 2001, pp. 585 -593.

[10] W. Rankl and W. Effing, *Smart Card Handbook*, 2nd edition, John Wiley & Sons, 2000.

[11] Borland Software Corp. C++Builder version 6.0, http://www.borland.com/cbuilder/

[12] OpenSSL Project, http://www.openssl.org

[13] Viega, M. Messier, and P. Chandra, *Network Security with OpenSSL*, O'Reilly, 2002.

[14] Omnikey CardMan Desktop USB2020, Omnikey AG, http://www.omnikey.com/en/ produkt_details.php3?produkt=1&variante=3

[15] C. Adams, et al, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol," IETF RFC 3161, August 2001.

[16] Symmetricom Inc., PCI Time and Frequency Processor, http://www.symmetricom.com/products/ product.php/107

[17] Infineon Technologies AG, SLE 4428 Chip, http://www.infineon.com/cgi/ecrm.dll/ecrm/ scripts/prod_ov.jsp?oid=15066&cat_oid=-9520

[18] Omnikey CardMan Desktop USB2020, Omnikey AG, http://www.omnikey.com/en/ produkt_details.php3?produkt=1&variante=3