INFORMATION LEAKAGE PREVENTED BY DATA ENCRYPTION USING SMARTCARD

D. B. Shanmugam*

P. Sathish Saravanan**

T. Karthikeyan***

N. Karthik****

Abstract:

Information leakage has recently become a serious problem and most often a result of actions inside rather than outside the system that should be protecting the information. Although system administrators have high access authority, they should not know the disk encryption keys of users because they may not be authorized to read a user's sensitive information. Information leakage from a disk in a managed network (e.g., an enterprise network) is generally prevented by encrypting either the entire disk or just the confidential files stored on it. Since this encryption uses highly secure.

Symmetric-key encryption algorithms, however, it is not easy for a user to memorize the random numbers constituting the disk encryption key. So it is important not only to encrypt the disk data but also to store the disk encryption key securely. This can be done by storing the key in a hardware token such as smart card or USB device, but there must also be some way to recover read it If the token is lost. For example, it is necessary to keep a backup copy in a safe place such as another key management server.

The encryption key should not be known by the system administrator, however, nor should it be possible for malicious users within the system to recover a user's encryption key. In this paper we present a scheme that can limit key recovery when a user loses his smart card and can do so.

* Department of MCA, Sri Balaji Chockalingam Engg College., Arni

** Department of CSE ,Sri Balaji Chockalingam Engg College., Arni

*** Department of CSE ,Sri Balaji Chockalingam Engg College., Arni

**** Department of CSE ,Sri Balaji Chockalingam Engg College., Arni

INTRODUCTION:

Transferring data from one point to another and carrying them around have become common practice, if not a habit. The end result is a mobile world that now seems to be a breeding ground for serious data thieves. Identity theft and data breach seem to hit the headlines far too often and reports of stolen laptops, lost USB tokens, financial data breach, and unlawful use of personal credentials among others, are heard of everywhere. In most of these cases, the loss is irreparable, and for companies and individuals these may mean losing the public's trust. The expansion and increasing sophistication of identity theft has undoubtedly paved the way for stronger security technologies in strategic initiatives such as e-government and e-business. Governments, financial institutions and organizations have begun turning to smart cards to secure the credentials of billions of people worldwide.

Millions of USB drives are used today as a convenient way to transferring necessary data between systems. Widespread use of smart cards as a defector standard for securing user credential has prompted the use of smart card technology for protection of mobile data on USB drives. Today smart cards can be used to generate and store encryption keys for flash drives and user authentication data.

Protection of flash memory encryption key:

Secure storage: Flash encryption keys are stored on tamper-proof smart card and can only be accessed through the smart card operating system by those with proper access rights. Flash encryption key never leaves the USB token.

PIN Block and Unblock: Smart card can block the access after predefined set of unsuccessful PIN attempts have occurred. To enable access, smart card deploys stringent PKI based challenge-response process. This prohibits any unauthorized access to the flash encryption keys and protects the authorized user.

Immunity to various software and hardware attacks:

• **Brute force and Parallel Attacks**: Brute force attacks guess the password or the encryption key. A parallel attack is a brute force attack variant in which the attacker copies the encrypted data from the stolen USB flash drive, shares the data with as many computers as possible that are under his/her control, and then puts them to work in parallel to guess the password offline and unlock the encrypted data. Smart card based integrated encryption thwarts brute force and parallel attacks as the data is never copied from device to host memory and smart card limits the number of attempts for successful authentication.

•**Cold boot:** Attack: DRAM memory is used to store data while the PC is running. After power is removed, all content is deleted in a gradual process that can take anywhere between a few seconds and up to a few minutes. If the chip is cooled by artificial means, the content can be retained for as long as 10 minutes. This characteristic of DRAM memory enables a hacker to read the memory content by cutting power and then performing a cold boot with a malicious operating system.In a smart card based approach, encryption keys are generated and stored on tamper-proof smart card and never leave the USB flash drive. Coldboot attacks on such implementation will not yield any fruitful result.

Authentication Counter Attack: An authentication counter tries to limit the number of security enforcing procedures that the system can perform in all the system life. Various physical attacks can be carried out on the authentication counter implementation including dumping and storing the authentication counter after decrement and fault attack on decrement.In a smart card based implementation, tamperproof storage cannot be compromised and counter decrement cannot be avoided.

Malicious Code: Malicious code can run on a PC into which a USB flash drive is inserted and can disable the encryption and protection mechanism on the USB flash drive. Malicious code can also copy data from the USB flash drive after it has been authenticated, or it can copy the user password and use it after the user logs out of the drive.

Security without compromising user experience:

Smart card based encryption of the flash memory is automatic and built-in device. It cannot be disabled by the end-user or attacker and assures protection of stored data. The user experience is not altered and the performance of the device is not affected by this integrated approach. Users still enjoy the speed and convenience of standard off-the-shelf USB drive.

Smart card based encryption has clear advantages compared to software-based and hardware-based encryption, when security of the encryption method and attack resistance is considered. It is also critical to take into account issues like performance and encryption availability before making a decision on the type of encryption to implement. The following table summarizes the key differences among available encryption approaches.

EXISTING SYSTEM:

There are two kinds of disk encryption: full disk encryption, in which all the byte data on the disk is encrypted; and file system-level encryption, in which what are encrypted are the files or directories on the disk. The Bit Locker feature of the Windows Vista OS is a popular tool for full disk encryption. It works in combination with a TPM chip that encrypts the key used in encryption and saves the encrypted key on the disk. The Secure File System, on the other hand, provides file system-level encryption.

A lot of authentication schemes based on smart cards have been proposed recently. In them a user's secret information that is shared with servers is stored in the user's smart card. It is protected by a password and by the difficulty of computing discrete logarithms. There is also a scheme that improves security by combining the use of a smart card and the Virtual Machine Monitor (VMM) . Several papers about key recovery have been published. Of the four kinds of key recovery methods (key escrow, trusted third party, commercial key backup, and key encapsulation) key encapsulation is the only one in which the key is not known to the administrator .

When key encapsulation is used, however, it is hard to confirm that the recovered key is the legitimate user's key because the administrator does not know the key in advance. This means that the encryption key can be recovered by a malicious user. Although we can easily

devise a method that uses the key with a certificate, in that case the key would be known to the administrator in advance. Key- recovery methods using a smart card have been proposed, but they are fundamentally different from the one in our scheme.

There is also a method in which secret keys are managed safely by using blind signatures and passwords . In that method a user's secret key is encrypted by the value of the blind signature and the source of the signature is encrypted by the password. Both the encrypted key and the encrypted source are kept on a local disk. This double encryption protects the password from brute force attacks.

DRAWBACKS IN EXISTING SYSTEM:

Moreover, we showed in experiments that it took about 2.6 seconds for the smart card to execute both the local authentication phase and the disk encryption key generation phase. This is a short time compared with the OS booting time.

PROPOSED SYSTEM:

Here we describe a scheme that can limit key recovery when the user's smart card is lost and can do so without the administrator knowing the key. The smart card is used for generating the key and for improving the user authentication. In our scheme the disk encryption key is not preserved anywhere and only someone who has a user's smart card and knows the user's password can decrypt that user's disk data. The smart card is used for generating the key and for improving the user authentication.

ADVANTAGES IN PROPOSED SYSTEM:

This project over comes those drawbacks. Proposed system has the following advantages.

- In this paper we described a scheme that can limit the recovered encryption key without informing the STTP (effectively, the system administrator) of the user's key. Although the STTP

maintains none of the user's confidential information, the user can recover the encryption key by cooperating with the STTP.

- To improve the user authentication.

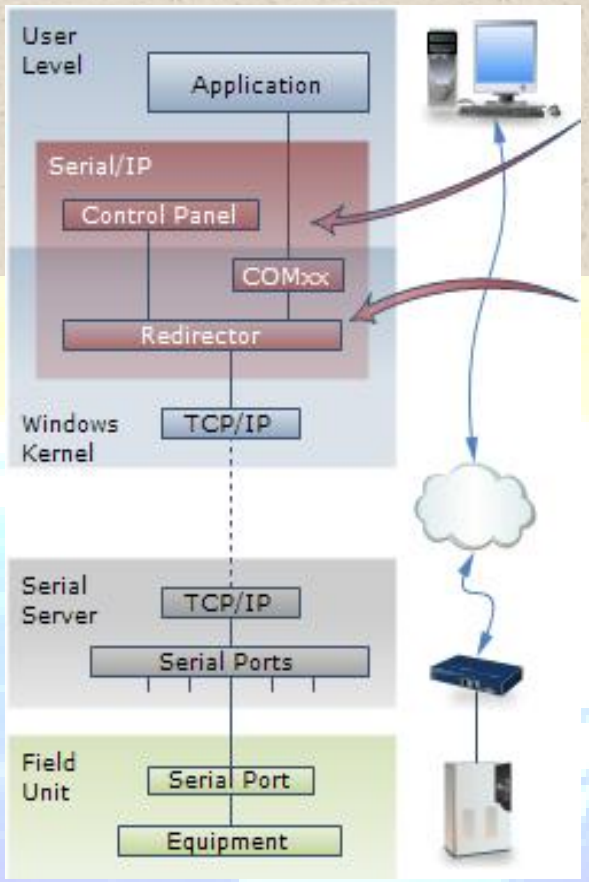
PROBLEM DEFINITION:

Information leakage has recently become a serious problem. Because a user's disk might contain a lot of confidential information, it should be encrypted and the encryption key protected securely. Disk security has been improved by storing the encryption key in a hardware token such as a smart card or USB device.

There must be some way to recover the encryption key when the token is lost, but to prevent information leakage the encryption key should not be known by the system administrator and should not be able to be recovered by malicious users inside the system.

Here we describe a scheme that can limit key recovery when the user's smart card is lost and can do so without the administrator knowing the key. The smart card is used for generating the key and for improving the user authentication.

Serial port driver:



The serial port driver handles any I/O devices that behave like serial ports, including those based on 16450 and 16550 universal asynchronous receiver-transmitter (UART) chips and those that use direct memory access (DMA). Many hardware platforms have devices of this type, including ordinary 9-pin serial ports, infrared I/O ports, and PC Card serial devices, such as modems. If multiple types of serial ports exist on a hardware platform, you can either create several different serial drivers, one for each serial port type, or create several different lower layers and link them to a single upper layer. This creates one multipurpose serial driver. Creating separate drivers can simplify debugging and maintenance because each driver supports only one type of port. Creating a multipurpose driver, such as the sample serial port driver, is more complex but gives a small memory savings. RTS (Request to Send)

- This signal is asserted (logic '0', positive voltage) to prepare the DCE device for accepting transmitted data from the DTE device. Such preparation might include enabling the receive

circuits, or setting up the channel direction in half-duplex applications. When the DCE is ready, it acknowledges by asserting Clear to Send.

- DTR (DTE Ready) - This signal is asserted (logic '0', positive voltage) by the DTE device when it wishes to open a communications channel. If the DCE device is a modem, the assertion of DTE Ready prepares the modem to be connected to the telephone circuit, and, once connected, maintains the connection. When DTE Ready is deasserted (logic '1', negative voltage), the modem is switched to "on-hook" to terminate the connection.
- OUT2 - controls interrupt servicing for the UART: OUT2=0 blocks the interrupt generated by the UART. This allows the user to prevent UART interrupts from reaching the computer's priority interrupt controller, while still generating them within the UART.
- CTS (Clear to Send) - This signal is asserted (logic '0', positive voltage) by the DCE device to inform the DTE device that transmission may begin. RTS and CTS are commonly used as handshaking signals to moderate the flow of data into the DCE device.

A smart card, chip card, or integrated circuit card (ICC), is in any pocket-sized card with embedded integrated circuits which can process data. This implies that it can receive input which is processed — by way of the ICC applications — and delivered as an output. It is simple plastic card, just at the size of a credit card, with a microprocessor and memory embedded inside is a smart card. Beside its tiny little structure it has many uses and wide variety of applications ranging from phone cards to digital identification of the individuals.

Applications:

- ✓ Identity of the customer,
- ✓ Library card, e-wallet,
- ✓ Keys to various doors, etc...

Here only one card can be issued to an end-entity for all these applications. Smart cards hold these data within different files, and, as you will read, these data is only visible to its program depending on the operating system of the card. These data files are arranged in a file system much like a Linux directory structure.

i) Contact Smart card

As smart cards have embedded microprocessors, they need energy to function and some mechanism to communicate, receiving and sending the data. Some smart cards have golden plates, contact pads, at one corner of the card. This type of smart cards is called *Contact* Smart Cards. The plates are used to supply the necessary energy and to communicate via direct electrical contact with the reader. When you insert the card into the reader, the contacts in the reader sit on the plates.

ii) Contact less Smart card

Some smart cards do not have a contact pad on their surface. The connection between the reader and the card is done via radio frequency (RF). But they have small wire loop embedded inside the card. This wire loop is used as an inductor to supply the energy to the card and communicate with the reader. When you insert the card into the readers RF field, an induced current is created in the wire loop and used as an energy source. With the modulation of the RF field, the current in the inductor, the communication takes place.

iii) Memory cards

The most common and least expensive smart cards are memory cards. This type of smart cards contains EEPROM (Electrically Erasable Programmable Read-Only Memory), non-volatile memory. Because it is non-volatile when you remove the card from the reader, power is cut off, card stores the data. You can think of EEPROM, inside, just like a normal data storage device, which has a file system and managed via a microcontroller (mostly 8 bit). This microcontroller is responsible for accessing the files and accepting the communication. The data can be locked with a PIN (Personal Identification Number), your password. PIN's are normally 3 to 8 digit numbers those are written to a special file on the card. Because this type is not capable of cryptography, memory cards are used in storing telephone credits, transportation tickets or electronic cash.

ENCRYPTION PROCEDURE:

Principles of Data Encryption

While there are many good reasons to encrypt data, there are many reasons not to. Encryption does not solve all security problems, and may even make some problems worse. The following sections describe some misconceptions about encryption of stored data.

Principle of Encryption Does Not Solve Access Control Problems

Most organizations need to limit data access to those who have a need to know. For example, a human resources system may limit employees to viewing only their own employment records, while allowing managers of employees to see the employment records of subordinates. Human resource specialists may also need to see employee records for multiple employees. This type of security policy limiting data access to those with a need to see it is typically addressed by access control mechanisms. Oracle Database has provided strong, independently-evaluated access control mechanisms for many years. It enables access control enforcement to an extremely fine level of granularity through its Virtual Private Database capability.

Because human resource records are considered sensitive information, it is tempting to think that all information should be encrypted for better security. However, encryption cannot enforce granular access control, and it may actually hinder data access. For example, an employee, his manager, and a human resources clerk may all need to access an employee record. If all employee data is encrypted, then all three must be able to access the data in un-encrypted form. Therefore, the employee, the manager and the HR clerk would have to share the same encryption key to decrypt the data. Encryption would therefore not provide any additional security in the sense of better access control, and the encryption might actually hinder the proper or efficient functioning of the application. An additional issue is that it is very difficult to securely transmit and share encryption keys among multiple users of a system.

WORK DESCRIPTION:

Information leakage has recently become a serious problem and most often a result of actions inside rather than outside the system that should be protecting the information. Although system administrators have high access authority, they should not know the disk encryption keys of users because they may not be authorized to read a user's sensitive information.

Information leakage from a disk in a managed network (e.g., an enterprise network) is generally prevented by encrypting either the entire disk or just the confidential files stored on it. Since this encryption uses highly secure symmetric-key encryption algorithms, however, it is not easy for a user to memorize the random numbers constituting the disk encryption key. So it is important not only to encrypt the disk data but also to store the disk encryption key securely. This can be done by storing the key in a hardware token such as smart card, but there must also be some way to recover it if the token is lost.

Here we describe a scheme that can limit key recovery when the user's smart card is lost and can do so without the administrator knowing the key. The smart card is used for generating the key and for improving the user authentication. In our scheme the disk encryption key is not preserved anywhere and only someone who has a user's smart card and knows the user's password can decrypt that user's disk data.

Information leakage has recently become a serious problem. Because a user's disk might contain a lot of confidential information, it should be encrypted and the encryption key protected securely. Disk security has been improved by storing the encryption key in a hardware token such as a smart card or USB device. There must be some way to recover the encryption key when the token is lost, but to prevent information leakage the encryption key should not be known by the system administrator and should not be able to be recovered by malicious users inside the system. Here we describe a scheme that can limit key recovery when the user's smart card is lost and can do so without the administrator knowing the key. The smart card is used for generating the key and for improving the user authentication.

Related work:

There are two kinds of disk encryption: full disk encryption, in which all the byte data on the disk is encrypted; and file system-level encryption, in which what are encrypted are the files or directories on the disk The Bit Locker feature of the Windows Vista OS is a popular tool for full disk encryption .It works in combination with a TPM chip that encrypts the key used In encryption and saves the encrypted key on the disk. The Secure File System [3], on the other hand, provides file system-level encryption.

A lot of authentication schemes based on smart cards have been proposed recently. In them a user's secret information that is shared with servers is stored in the user's smart card. It is protected by a password and by the difficulty of computing discrete logarithms. There is also a scheme that improves security by combining the use of a smart card and the Virtual Machine Monitor (VMM) .Several papers about key recovery have been published. Of the four kinds of key recovery methods (key escrow, trusted third party, commercial key backup, and key encapsulation) key encapsulation is the only one in which the key is not known to the administrator .

When key encapsulation is used, however, it is hard to confirm that the recovered key is the legitimate user's key because the administrator does not know the key in advance. This means that the encryption key can be recovered by a malicious user. Although we can easily devise a method that uses the key with a certificate, in that case the key would be known to the administrator in advance. Key-encryption key with the help of a smart card. We assume that the smart card is also an identification card (ID card),so the encryption key is linked to the user's identity. Thus only a specific user who has the smart card is able to start up the OS in the user's client PC. We also assume that a PKI such as remote authentication is used with the smart card. The private key, the public key, the public key certificate, and the signature calculation software in the public key cryptosystem are stored in this card.

Basic policy:

In this section we describe basic policy for constructing our scheme. We assume that the scheme is used in managed network such as an enterprise network.

1. Realization of full disk encryption:

An advantage of full disk encryption (encrypting all byte data on the disk) is that everything, including the swap space and the temporary files, is encrypted and the decision of which files to encrypt is not left to users [1]. Our scheme uses a symmetric-key encryption algorithm such as AES because it is a high-security algorithm and can encrypt/decrypt a disk quickly. In a full disk encryption, the OS is encrypted in a hard disk. So some program would start up the OS by decrypting the hard disk data.

2. Use of a Virtual Machine Monitor:

For a full disk encryption, it is necessary for a program to decrypt the disk data before starting up the OS. We use a Virtual Machine Monitor (VMM) as this program. The VMM encrypts/decrypts the disk data by using it own encryption engine by which the disk access data is compulsorily hooked. The VMM first authenticates the smart card locally by using the public key authentication method. When the card is authenticated, the VMM can acquire the disk encryption key and load it into the VMM memory. Note that we do not put secret information such as the disk encryption key directly into the VMM. The validity of the signature calculation in the smart card can be guaranteed at the same time that the card is being authenticated.

3. Use of smart card

The smart card is used for generating the key and for improving the authentication. Because we use a symmetric-key encryption algorithm, it is difficult for a user to memorize the disk encryption key (e.g., a 128-bit random number). Our scheme therefore generates the encryption key with the help of a smart card. We assume that the smart card is also an identification card (ID card), so the encryption key is linked to the user's identity. Thus only a specific user who has the smart card is able to startup the OS in the user's client PC. We also assume that a PKI such as remote authentication is used with the smart card. The private key, the public key, certificate, and the signature calculation software in the public key cryptosystem are stored in this card.

4. Use of Semi-Trusted Third Parties

The administrator and the trusted management server together constitute a semi-trusted third party (STTP) that issues legitimate identification and smart cards, recovers lost encryption keys,

and prevents the information stored in its database from being falsified. It need not, however, prevent the leakage of a user's secret information, such as the encryption key, because it does not have any of the user's confidential information. The only secret information that the STTP has is its own private key.

5. Recovery of disk encryption key

If you have lost your smart card, you might not be able to decrypt your disk data because the disk encryption key is stored in your smart card. We therefore need to have some way to recover the key when the smart card is lost. One drawback of most key recovery methods, however, is that they require the STTP to know the user's key. Although there are also some methods in which the user's key need not be known to the STTP, with them it is difficult to ensure that a key can be recovered only by the legitimate user. This means that the encryption key can be recovered by a malicious user. We therefore want to recover the key without letting the STTP know what the key is.

Our scheme:

In this section we explain the detailed protocol of a scheme following the basic policy described.

1. Premises

The premises of our scheme are these:

1. The STTP issues the public key certificate and the certificate revocation list, and it prevents this information from being falsified while it is in the STTP's database.

2. The smart card is tamper resistant, and the confidential information is not stolen from the smart card itself or while it is being transferred between the smart card and the client PC.

3. The calculation algorithms (e.g., signature generation)

in the smart card are not falsified.

4. The STTP can freely alter the data in user's smart card by using the STTP's privileged password, but the STTP cannot read that data.

5. The user can read data other than his private key in his smart card by using his password, but he cannot alter it.

6. The code of the VMM in the client PC is not falsified, and the STTP does not steal the user's private information in the VMM.

7. We do not deal with the situation in which the user who has lost his smart card forgets his password.

2. Notations- In this description of the notations used in our scheme, | *n* | and | | *i n* are assumed to be more than 1024 bits.

Ui :User i

PWi : Ui's password

(e,d,n) : RSA keys of STTP

(ei,di,ni) : RSA keys of Ui

R,ri : Random numbers(<|n|)

ai,bi : Random numbers(<|n|/2)

certi : Public certificate of Ui

CERT : Public key certificate of STTP

CRL : Certificate revocation list

C : Challenge

Ski : Disk Encryption key of Ui

h(.) : Hash function(e.g SHAI)

3. Protocol

The protocol in our scheme is composed of the following six phases.

A. Initial phase

The STTP generates its RSA keys, *CERT*, and *CRL*. It also generates the random numbers *R* that are the common public information of the system.

B. Registration phase (1)

The user connects his smart card with the STTP and initializes the card (see Figure 1). In this part of the registration phase the user *i U* sets his RSA keys, *i cert* , and *i pw* in his smart card. The STTP makes each user's public information (mod) *I Rdi n* by using *i U* 's smart card and preserves the value along with *i cert* .

C. Registration phase (2)

i U connects the client PC (VMM) to the STTP with a secure channel, and makes information that is necessary for the key recovery in the VMM (see Figure 2). In this part of the registration phase the VMM generates both (mod *i*) *Raibi n* and (*a b*)*e* (mod *n*) *i i* in cooperation with the STTP. Note that neither the VMM nor the STTP knows the value of *i i a b* .

D. Local authentication phase

The client PC (VMM) confirms the validity of the smart card in a local network. We can confirm that the smart card was legitimately issued by the STTP. In this phase we use a blind signature for both the generation of the disk encryption key and the authentication. The procedure is shown in Figure 3. 108 Authorized licensed.

E. Disk encryption key generation phase

When a local authentication succeeds, the VMM executes the following procedure.

1. The VMM derives the legitimate disk encryption key *i sk* of size | *i* | *n* by calculating *I cdi r* (mod *i*) *n* : ( )*I a b h pw d*

$sk = R\ i\ i\ (\ i\ )\ i \bmod n$.

2. The VMM divides the disk encryption key *i sk* by the block size of the symmetric-key encryption algorithm, and then uses some chopped keys from the head of byte data of *i sk* . For instance, when the symmetric-key encryption algorithm is AES-128 you can get eight disk encryption keys because | |= 1024 bits *i n* . In this case you use only the first key, *i*1 *sk* . *ski* = *ski*1 | *ski* 2 |….

F. Key-recovery phase

When you accidentally lose your smart card, you execute the key-recovery phase by connecting your client PC (VMM) to the STTP with a secure channel. After recovering the key, you discard it and start from registration phase (2) again.

1. Purpose

Our scheme uses a smart card with a low processing ability, so its processing time might be long because two kinds of phases (the local authentication phase and the disk encryption key generation phase) are executed every time the client PC is used. We therefore measured the processing time required for each function in both phases in order to confirm that the smart card can complete the processing within a reasonable time.

2. Circumstances

We used as the client PC a ThinkPad X60 (CPU: Core 2 Duo 2GHz, Memory: 1GB), used as the smart card an eLWISE (NTT Communications), and used as the smart card reader an ASE drive IIIE (Athena Smartcard Solutions). This smart card is equipped with a CPU, RAM, and ROM and corresponds to PKCS#11. The software we used was the OS Linux Fedora Core6, a smart card library group, the encryption library OpenSSL 0.9.8b, the multiple-precision arithmetic library GMP 4.1.4-9, and the virtual machine monitor QEMU 0.8.2.

3. Method and Results

In the local authentication phase and the disk encryption key generation phase, we measured the processing time for the six items listed in Table 1. Items 1 and 4 were executed in the smart card, and the others were executed in the VMM. The measurement was conducted by inserting the *gettimeofday* function in the source code.

Table 1: Experimental results

Measurement items Time,

1. Acquisition of user's public key certificate 2140 ms

2. Verification of user's public key certificate 19.4 ms

3. Generation of challenge 19.2 ms

4. Generation of blind signature 420 ms

5. Verification of blind signature 11.7 ms

6. Generation of disk encryption key 0.202 ms

7. Total time 2610 ms.

Each of the times for the items listed in Table 1 is the average of five measurements. The total time is discussed in subsection. The processing times for acquiring the user's public key certificate and for generating the blind signature were both comparatively long, and that for acquiring the public key certificate was the longest. Additionally, the processing time for generating the blind signature contains not only the calculation but also the transfer of the signature data. That is, the transfer between the smart card and the client PC took longer than the signature calculation in the smart card.

1. Limitation of recovered key

The value of (mod *i*) *Rdi n* can be calculated only from *i U* 's smart card because that is the only place where the private key *di* is stored. That is, the value of (mod *i*) *Rdi n* is link to *i U* through *certi* (see Figure 1). And the recovered disk encryption key *Raibih*(*pwi*) *di* (mod) *i n* is calculated from the (mod *i*) *Rdi n* that the STTP stores (see Figure 4). That is, the recovered encryption key is link to the value of (mod *i*) *Rdi n* . The recovered encryption key is therefore linked to *i U* . This means that the STTP can limit *i U* 's recovered encryption key to the value of *Raibih*(*pwi*)*di* (mod) *i n* if the STTP authenticates *i U* who lost his smart card.

2. Impersonation attack by STTP

The RSA private key *di* , the hash value of the password (*i*) *h pw* , and the value of *i i a b* —all corresponding to *i U* —are needed to generate the disk encryption key, but neither the STTP nor an attacker knows these values. The STTP does not maintain any of the user's secret information. The STTP can get the values of (mod *i*) *Raibi n* and (mod *i*) *Rdi n* in the registration phase but, because of the secrecy of the RSA cryptosystem, it cannot get *i i a b* or *di* from these values. Therefore neither the STTP nor an attacker impersonating the STTP can get the user's disk encryption key *i U* cannot get *i i a b* or *di* . So the user cannot get the disk encryption key without using his smart card.

3. Revocation of disk encryption key

When the disk encryption key of some user is revoked, the user must not be able to freely use his encryption key. The STTP can revoke the RSA private key *di* by using the *CRL*. The disk encryption key includes user's private key *di* . Therefore the STTP can revoke *i U* 's disk for encryption key by revoking *di* . As a result, the user whose encryption key is revoked cannot decrypt his disk data even with his smart card. Of course he is also unable to use the PKI authentication with the same card after revocation. Note that it is necessary to have the *CRL* stored in the VMM updated.

4. Protection of recovered key:

The disk encryption key should not be known by anybody even after it is recovered. In our scheme the STTP cannot derive *i U* 's encryption key after it is revoked because the STTP does not know the hash value (*i*) *h pw* . Also, the person other than STTP does not know (*i*) *h pw* by the same token. The encryption key is thus kept secret even after the key recovery phase.

5. Time until the OS has booted up

The time until the OS has booted up is the total time required for the local authentication phase, the disk encryption key generation phase, and the OS booting. Our experimental results showed that execute both the local authentication phase and the disk encryption key generation phase took about 2.6 seconds. On the other hand, booting up the OS Windows XP) on the QEMU took about 60 seconds on the same machine. So the processing time for both phases was less than 5% of the time required for booting up the OS. We therefore think that the time which is required for both phases is short enough to be practical.

Smartcard Solutions:

Smartcard solutions are used in wireless communication, loyalty systems, banking Pay TV and government ID. These are used to provide strong authentication in e-business. These solutions are used with standard non-secured PCs. Consumers, vendors and financial institutions need to know that the transactions, documents and identities are authentic. DES and Triple-DES algorithms are the most used encryption methods in data security for the Smartcard solutions.

References:

- Z. Chen, Java CardTM Technology for Smart Cards Architecture and Programmer's Guide, Addison-Wesley,MA, USA, 2000.
- Transcards,GIE Sesam Vitale, Transcards Project, URL:http://www.sesamvitale.fr/html/projets/transcards/tcd accueil eng.htm, last accessed: 2002.
- Netlink, GIE Sesam Vitale, Netlink Project, URL:http://www.sesamvitale.fr/html/projets/netlink/index.html,last accessed: 2002.
- Netcards, GIE Sesam Vitale, Trans-European Healthcare Facility Service for Mobile Citizens, URL:http://www.sesamvitale.fr/html/projets/netcards/index eng.htm, lastaccessed: 2002.
- Netcards Project, Netcards Consortium, Trans-European Healthcare Facility Service for Mobile Citizens, URL: http://www.netcards-project.com/, last accessed: 2003.
- R. Novak, G. Kandus, D. Trcek, Further development of a smart-card based health care information system in Slovenia, in: Presented at the Fifth International Congress on Conference and Exhibition on Cards Applications in Health Care: Health Cards'99, Milan, Italy, 1999.

Authors Details:

D.B.Shanmugam M.C.A.,M.Phil.,

Asst. Professor, Department of MCA, Sri Balaji Chockalingam Engg College Arni.

T.Karthikeyan MCA.,M.Phil.M.E.,

Asst. Professor, Department of CSE, Sri Balaji Chockalingam Engg .College Arni.

P.Sathish Saravanan.B.Tech.,M.Tech.,

Asst. Professor, Department of CSE,Sri Balaji Chockalingam Engg .College Arni.

N.Karthik.,B.E.,M.Tech.,

Asst. Professor, Department of CSE,Sri Balaji Chockalingam Engg .College Arni.

IJMRA