Software License Management with Smart Cards

Tuomas Aura

Helsinki University of Technology

Dieter Gollmann

Microsoft Research

Software license management with smart cards

Tuomas Aura
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O.Box 5400, FIN-02015 HUT, Finland
Tuomas.Aura@hut.fi

Dieter Gollmann
Microsoft Research
St. George House, 1 Guildhall St.
Cambridge, CB2 3NH, UK
diego@microsoft.com

Abstract

This paper describes public-key protocols for binding software licenses to tamper-resistant smart cards, for transferring licenses between cards, and for purchasing them on-line. The protocols support software distribution both through retail stores and over the Internet. The user can transfer licenses from several cards onto a single card to avoid juggling between several cards in the reader. The protocols are based on signed delegation certificates that are mostly stored outside the smart card. A smart card reader and cards capable of public-key signatures are the only new hardware needed. The protocols are easy for the user and simple to implement and analyze. We prove the security of the transfer protocol.

1 Introduction

Unlicensed use of computer software has always been a major concern for the software industry. Lately, the piracy problem has been highlighted by the introduction of the Internet as a distribution channel [11], and the rise of content industry whose products are often collectively labeled as multimedia.

Most copy-protection and license management techniques have either proven ineffective or too restrictive for users to accept them. Thus, a majority of mass-market software products today are sold without any technological protection, which leaves marketing and legal battles as the only means for the software industry to defend itself. However, current advances in technology are opening new possibilities whose impact on license management should be assessed. First, with the popularity of smart cards, intelligent hardware tokens are becoming much more affordable. Second, computer networking makes two-way communication between the customer and the software publisher more convenient.

This paper shows how these new technologies add a great degree of flexibility and ease of use to software license management and hardware-based copy protection. It is possible to use the new technique in combination both with conventional software sales through retail stores and with Internet commerce. The protocols proposed in this paper require public-key cryptography on the smart cards but otherwise they are extremely simple. The license information is in the form of signed certificates and can be managed mostly outside the smart cards.

The main threats that we address are multiple installations of software from a single-license distribution medium and production of counterfeit copies by professional pirates. These types of copying appear to have the greatest impact on the software publishers' revenues.

Another recent development, robust copyright-marking techniques such as watermarking [9], helps in resolving legal disputes over the ownership of data. However, it does not prevent copying of programs because the owner and copyright status are normally obvious from software products. A level of access control is needed to help the users make the right choice. It should be easier to buy than to copy. Also, copy-resistant physical tokens are needed to slow down the professional pirates whose aim is to mass-produce copies and market those as originals. We recognize that there are always ways to work around the protection mechanisms. What can be done is to increase the time to market for pirated copies and to ensure that pirated products cannot be sold as authentic to unsuspecting customers. If honest and security conscious users are alarmed about tampered products, they are likely to buy authentic ones instead.

The rest of the paper is organized as follows. We begin with a short introduction to copy protection with tamper-resistant modules in Sec. 2. Sec. 3 gives an overview of license transfer and Sec. 4 the protocol details. Sec. 5 continues with a protocol for on-line purchase of licenses. Techniques for strengthening the copy-protection are discussed in Sec. 6 and prevention of license theft in Sec. 7. Finally, we summarize the assumptions and advantages of the suggested protocols in Sec. 8 and list some possible extensions in Sec. 9. Sec. 10 concludes the paper. The Appendix contains a proof that the protocols cannot be subverted to copy licenses.

2 Copy protection with smart cards

The only theoretically secure copy-protection arrangement is to deliver the code in encrypted form and to decrypt and execute it inside a tamper-resistant processor [14, 15, 6]. In practice, such processors cannot be mandated and the code is exposed to insecure user equipment. Therefore, copy-protection is always to some extent security by obscurity.

In practical protection mechanisms based on a hardware token, a user license is embodied by a copy-resistant piece of hardware. The software or the operating system checks for the presence of the token and refuses to run without it.

A common type of token is a *dongle* that is inserted in a communications port on the workstation that is to run the software. If smart card readers become more common, a smart card is the obvious choice for a token. This is because the production cost of a single smart card is negligible compared to the cost of a software license. It would not be impossible to routinely distribute smart cards with all shrink-wrap software.

Robust mechanisms for checking the authenticity of the hardware token are based on a cryptographic key that is never stored or used outside the tamper-resistant token. The security of the mechanisms depends on two assumptions of technical intractability: it must be too expensive or time-consuming to reverse engineer the smart card in order to obtain the hidden secrets in it, and it must be equally difficult to modify the software to run without the card. Both of the assumptions present difficult problems of their own. This paper leaves them for others to solve. Tamper-resistant smart card technology is an active area of research [1, 8, 10], as is authenticated booting of software.

Unfortunately, dongles or smart cards are unpopular with users. The main objection has been that the protection mechanisms for different software packages often interfere with each other. Even if the protection mechanism for each individual product is well designed, they might become unusable together. This is a major problem for smart cards since a single card must not be allowed to monopolize the card reader. In order to prove presence of a token for different software packages, one may have to repeatedly insert different cards into the reader, an annoying practice sometimes referred to as *smart card juggling*.

We will describe a solution for binding software licenses to smart cards and for transferring them from card to card in such a way that the juggling is eliminated.

3 License transfer with delegation certificates

In order to achieve flexibility and ease of use, our goal is to allow a single smart card to act as a token for arbitrarily many software packages. The licenses are distributed on cards that the customers get bundled with each software package. Normally each card holds only a single license. With a simple procedure, the licenses on one card can be transferred onto another card. After the transfer, the “empty” card may be discarded.

Every card has a unique public-private key pair. The private key is stored on the card and never revealed to the outside. At any time when the card is in the reader, it will respond to a challenge to prove that it, indeed, has the private key corresponding to the public key. This way, the software can check that the card associated with the license is present in the reader.

It is still necessary to bind a license to the public key. A convenient way to do this is to issue a certificate to the public key of the card. The certificate will be signed by the software publisher’s master key and it will be verified with a public key incorporated in the software or in the operating system. The certificate can be stored outside the card. In fact, the card never needs to know which licenses it is certified to have.

It is crucial that the publisher’s master public key and the procedure for checking the certificate and the presence of the card are embedded in the software in such a way that the key cannot be changed and the check cannot be disabled. In general, this is not an easy task. The protection can always be removed by reverse engineering the code. In practice, however, obfuscation of the checking procedure can significantly delay the reverse-engineering process and the production of marketable copies. Section 6 describes some measures that make the marketing of the modified software less attractive after the protections have been removed.

We will now outline the mechanism for transferring licenses from one card to another. Once the license is bound to a public key, it can be given to other keys by *delegation*. The key having a license simply signs a certificate stating its willingness to give the same rights also to the key of another card. This kind of certificate with which one key delegates access rights to another one is called a *delegation certificate*. The signing is done with public-key cryptography. It is possible to use standard certificate formats and techniques [5, 2].

Unfortunately, this simple procedure is not quite enough; it would result in duplication of the license. After handing over the license, the first card must cease to function as a token. Furthermore, it must never sign another delegation certificate (at least not to delegate the same license). The simplest way to ensure this is to erase the private key from the delegating card. Erasing the key means that we must always transfer all licenses together to the same card and then discard the original card.

Thus, license transfer comprises two steps:

1. delegating the license to another card
2. disabling the delegating card as a token.

In principle, a license can be transferred an unlimited number of times. Every transfer adds a new delegation certificate to a chain that passes a growing collection of licenses from card key to card key. When the right to use a certain software package is verified, there must be a complete chain of certificates starting from a license certificates signed by the publisher's master key and ending with the public key of the card that is currently in the smart card reader. In practice, the number of transfers for a single license will be very small (usually one). The licenses cannot be transferred onto previously disabled cards and every transfer accumulates all the licenses on two cards onto a single one.

The cards need to have only two basic functions: proving the possession of a private key by responding to a challenge and signing a delegation certificate after which the card disables itself. Further management of the certificates is done outside the smart card. Sec. 4 details the transfer protocol.

The idea of transferring licenses from one smart card to another bears resemblance to the transferable digital cash of Pagnia and Jansen [12]. From the result of Chaum and Pedersen [4], we can infer that the growth of the license data with each transfer is inevitable. In our scheme, another delegation certificate will be appended to the license in each step. It is important, however, to note that the growing collection of delegation certificates is not stored in or processed on the smart cards. The smart cards act as tamper-resistant observers that guard against copying of licenses. This is equivalent to double spending prevention for digital money [3].

4 Protocol for license verification and transfer

Each card has a unique signature key pair. The public part of the card key (CK) can be read from the card at any time while the private key is never exported outside the card. In addition to the keys, the card stores a *card certificate* signed by the software-publisher master key (PMK). The card certificate states that CK is the public key of an authentic license card. The certificate can be freely read from the card. Anyone knowing the public PMK can verify the certificate on the card and conclude that this is an authentic license card approved by the software publisher. The PMK and the card certificate will be used to ensure that licenses are transferred only to smart cards that reliably protect them from copying. Every card has to store the authentic public PMK for verifying card certificates. A card certificate is of the form

$$S_{PMK}(CK, \text{“is a license card key with production date”}, date)$$

(The notation $S_K(M)$ means the message M signed with the key K. Our view of the signature function is ideal. Implementations should follow accepted standards such as PKCS [7].)

The certificate contains the production date or serial number of the card. The licenses can only be transferred from older to newer cards. This ensures that cryptanalysis of old card keys or cracking the defenses of old tamper-resistant cards cannot be used for copying new software products. It should be noted that the card certificates and the dates on them originate from the software publisher or trusted card manufacturers, the date

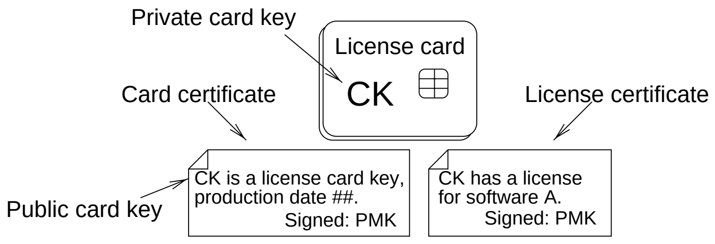
Private card key
License card
CK
Card certificate
License certificate
Public card key
CK is a license card key, production date ##. Signed: PMK
CK has a license for software A. Signed: PMK

Figure 1: A basic license card has a key pair and two certificates.

stamps are compared against each other, and the comparison is done on the tamper-resistant license cards. No clocks are needed on the smart cards and clocks in the user equipment are not relied on. Therefore, the comparison of dates is reliable. As a secondary protection, the software checking the presence of the token should also ensure that the date on the card certificate is not older than the software itself.

Normal cards originally hold only one license as they are sold in retail stores with software packages. The license is a certificate signed by the PMK that binds the right to use a certain software package to the CK. The *license certificate* is of the form

$$S_{PMK}(CK, \text{“has a license for”}, software).$$

Although it is not necessary to store the license information on the card, it is convenient to distribute licenses on the cards that come bundled with the software distribution media. This way, the software itself can be on identical media, e.g. printed CD-ROMs. For efficiency, it is best to read the card and license certificates from the card into the workstation only once when the software is installed and never refer to them on the card again. It is possible to ship several license certificates with a single card (e.g. stored on a floppy disk). This is practical when the publisher ships products directly to the users or when workstation manufacturers pre-install a standard set of software.

In addition to carrying the certificates, the card can perform two main functions: proof of identity and license transfer. The former means proving the possession of the private CK. The card does this simply by signing a challenge.

Protocol 1 (proof of identity):

1. Workstation $\rightarrow$ Card :
   “License card challenge”, N
2. Card $\rightarrow$ Workstation :
   S_{CK}(“License card response”, N)

The checking software first needs the public card key CK and the card certificate. Then, it can send a challenge to the card and verify that the card is authentic. To decide if the card has a certain license, the software follows the delegation certificates to find a chain of delegation from PMK to CK. These certificates are stored outside the card and can be written into a file or onto a floppy disk for keeping with the card.

The delegation certificates are created in the second main function of the card, the license transfer. All licenses on the card are always transferred at the same time. After the transfer, the original card can be thrown away.

The transfer protocol is very simple. Licenses on two cards will be combined onto one of them. The source card (the one to transfer from) must be the older card and the destination card (the one to transfer to) the newer one. Before the transfer, the workstation obtains the card certificate of the destination card. After that, the protocol is between the workstation and the source card only. The destination card is not involved in the communication. The delegation certificate produced in the transfer will be stored in the workstation and it will later be used together with the destination card. However, the destination card does not need to know anything about the transfer and the certificate is never saved onto the card.

Protocol 2 (license transfer):

1. Workstation $\rightarrow$ Source card :
   "Please transfer to", CK',
   $S_{PMK}(CK'$, "is a license card key with production date", $date)$
2. The source card signs a certificate and erases its private key CK.
3. Source card $\rightarrow$ Workstation :
   $S_{CK}($"I give all my licenses to", $CK')$

In the first step of the protocol, before delegating the license to the public key CK', the source card checks that the key belongs to an authentic license card. It therefore needs the card certificate for CK'. It also compares the date on the card certificate to its own production date to see that the destination card is the same age or newer.

In step 2, the card signs a *delegation certificate* for CK'. The certificate is of the form $S_{CK}($"I give all my licenses to", $CK')$. After signing the certificate, the card permanently erases its own private key CK from its memory. After erasing the key, the card is not anymore able to perform Protocol 1, i.e. the proof of identity. This means that it is disabled as a license token.

Having created the delegation certificate, the card returns it to the requesting workstation in Step 3 of the transfer protocol. After the transfer, the source card is useless and it can be thrown away. In order to protect against loss of certificates, the card certificate, the license certificate and the new delegation certificate are still stored on the otherwise disabled card and can be reread an unlimited number of times.

A crucial point for the smart card implementation is that signing the delegation certificate and erasing key private key must be an atomic operation. If the operation is interrupted, for example, by cutting power from the card, the card must either complete the signing and erasure immediately after power-up, or it must return to the original state where the delegation certificate does not exist. Moreover, the production and storage of the delegation certificate on the card must be reliable because the signing cannot be repeated after the private key has been erased.

In summary, the protocol performs the two steps that make a complete transfer: delegation and disabling the old card as a token. In the workstation, the new delegation certificate will be combined with the ones both cards previously had. All these certificates are needed for use with the destination card (Fig. 4).

5 On-line software distribution

Although we cannot assume all customers or all workstations to have Internet connections, an increasing number of customers is willing to purchase software on-line. Two-way communication between the user workstation and the software publisher opens new possibilities for license management. It is necessary for the same license management system to support both traditional shrink-wrap software sales and on-line commerce.

When licenses are sold on-line, they can be personalized for each customer. The key to controlling the distribution of the licenses is to bind each license to exactly one user workstation at a time. There are plans for incorporating unique identifiers into the microprocessors in personal computers for this purpose. Because of privacy concerns, it is not clear whether such identifiers will ever be implemented be all vendors. Some copy protection products compute a fingerprint of the hardware and software configuration to identify the workstation [13]. Unfortunately, this may cause invalidation of the license when parts of the system are updated.

In our system, the card key of a smart card is a unique identifier to which the licenses are bound. The same smart cards that are sold in retail stores can be used for on-line purchases. Instead of getting the licenses with the card, the customer buys license certificates for his card from the on-line store. If the particular workstation already has a license management card, the license certificate will be issued to the public card key of that card. Most customers have recent cards e.g. from purchasing the operating system and, since the price of the smart card itself is low, empty cards without a license can be distributed free of charge.

Protocol 3 (on-line purchase):

1. Customer $\rightarrow$ Publisher :
   "I buy a license for", CK,
   $S_{PMK}(CK$, "is a license card key produced on ", $date)$
2. Publisher $\rightarrow$ Customer :
   $S_{PMK}(CK$, "has a license for", $software)$

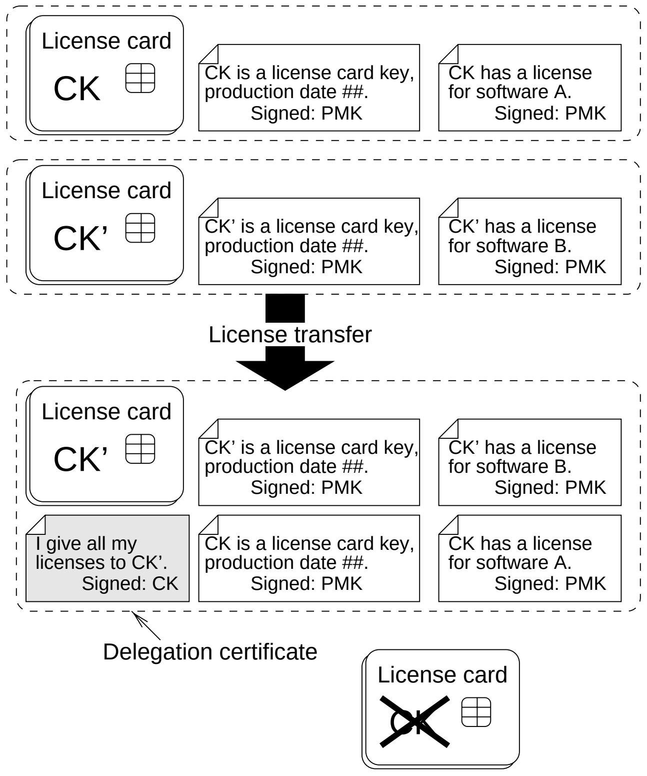
License card
CK
CK is a license card key, production date ##. Signed: PMK
CK has a license for software A. Signed: PMK
License card
CK'
CK' is a license card key, production date ##. Signed: PMK
CK' has a license for software B. Signed: PMK
License transfer
License card
CK'
CK' is a license card key, production date ##. Signed: PMK
CK' has a license for software B. Signed: PMK
I give all my licenses to CK'. Signed: CK
CK is a license card key, production date ##. Signed: PMK
CK has a license for software A. Signed: PMK
Delegation certificate
License card
CK

Figure 2: License transfer = signing a delegation certificate + disabling the source card. The certificates are stored outside the cards.

The on-line store needs to see the card certificate to check that the public key CK belongs to an authentic license card.

The only limitation for this type of on-line sales is that licenses should not be sold to cards that are too old because their keys might have already been recovered by pirates. (If the pirate knows the private key of an authentic license card, he can purchase one license on-line and delegate it to any number of cards.) If the card is older than some threshold time, the customer needs to obtain a new card and move licenses from his old card onto it before buying new software on-line. The threshold time for rejecting old cards can be adjusted for new products according to the experiences from earlier releases.

Actually, it is not important how the software itself is distributed: on-line or on a CD-ROM or on some other medium. The distribution of licenses can be completely independent of the software distribution.

On-line services open new possibilities for strengthening the security of the system. If the product includes parts or services, such as updates, that are delivered over the Internet, the servers can check for the license before providing the service. The on-line server will send the user workstation a challenge. The response from the smart card is sent to the on-line service along with the public key of the smart card, the license certificate or a chain of certificates from PMK to CK, and the card certificates for all card keys in the delegation chain. The server can store fingerprints of the keys in the chain and refuse to repeatedly provide the same service for the same keys. Similarly, the same license should not be sold twice to the same card because it may indicate that the card is a clone. This improves the strength of the copy protection in case a pirate is able to recover the private key of a single card. Users of the pirated licenses will be refused on-line service.

It is for the purpose of bookkeeping at the on-line servers that we retain all the card certificates in the license transfer. If on-line services are not available or the on-line server has a database of all valid card keys, it is not necessary to store the card certificates of the disabled cards after the transfer. It suffices to keep the one belonging to the active card.

6 Enhancing the copy protection

Copy protection is never perfect. Therefore, we will consider ways of strengthening the protection. The effectiveness of these techniques depends on the nature of the product and the environment where it is mainly used.

The two most dangerous attacks against the copy protection in our license management scheme are recovering of the private key from the smart card and modifying of the software to bypass the checking for the token.

When on-line updates or other Internet services are an essential part of the product, the problem of recovered private keys is alleviated by having the servers remember the keys for which the service has already been provided (see Sec. 5). Professional pirates cannot produce fully functional copies even if they are able to crack the protections of a single card because users of the illegal copies will not be able to access the on-line services. This works because the cards have unique keys instead of one shared secret. Continuing the analogy to digital cash, the database of served card keys resembles double-spending detection by banks that keep track of spent coins.

Another way of discouraging the purchase of pirated copies is to have the users authenticate the software. The person installing or using a pirated software package should get a warning about potentially dangerous, unauthentic code. The warning can be implemented by signing the code with the publisher master key PMK or with another key held by the publisher. The public verification keys can be distributed on-line or with the operating system. If the pirates modify the software in order to remove the check for the license, the pirated copies inevitably fail the test for correct signatures. This kind of integrity check is beneficial even if copy protection is not an issue: software distributed over the Internet should be authenticated in any case.

Implementation of the integrity warning messages requires co-operation with the operating system or with a generic installation program. The warnings can naturally be avoided by modifying also these support programs. However, most business users of software are probably unwilling to tweak their operating system according to the pirate's instructions. Also, the embedded operating systems in special-purpose devices such as game consoles and multimedia terminals often cannot be modified by the user.

Instead of completely disabling the check for the token, pirates may try to modify the smart card reader or its driver software in such a way that several workstations can share one reader. The challenges and responses could be transferred over the network between a single reader and a large number of verifiers. To prevent such modifications, the checking software should have direct access to the smart card reader hardware so that it can trust the responses to be from a local source. Sometimes, especially if the operating system consists of replaceable modules or layers, it may be impossible to prevent tapping between the card and the verifier. Even then, the attack is only possible if the modification to the operating system is available and if the user organization is willing to install the patches on all workstations. Other possible defenses include binding the licenses to workstation identities and limiting the number and frequency of answered challenges per license. Such measures, however, imply unique processor identifiers, on-card timers, counters, and much more complex protocols that are beyond the scope of this paper.

There is one effective technique for checking the presence of the token which requires some adjustment for our public-key protocols. That is, the software on the distribution media can be encrypted and the license token should contain the key for decrypting it. If the software is never stored outside the computer memory in decrypted form, it cannot be loaded without the token. (Naturally, this is just a way of obscuring the check for the tokens. The program in the insecure computer memory can be read and saved with special tools and skillful reverse engineering.) Dongles sometimes carry a secret key for decrypting the code [13]. If we want to use this technique in our license transfer scheme, we have to pass the decryption keys to the destination card and erase them from the source card as a part of the license transfer. The keys can be transferred by encrypting them with the public key of the receiving card.

7 Preventing license theft

New technology often creates new types of vulnerabilities that are beyond our prior experience. When the software licenses are bound to small, tangible objects, a new threat emerges: theft of licenses. And what is most disconcerting, the transfer protocol could be misused to steal licenses electronically over the network. Luckily, theft can be prevented with simple password protection.

The physical theft of license cards may be a problem anywhere where untrusted persons have physical access to the workstations. The standard protection against the theft of smart cards is that the card requires the user to enter a password after it is inserted into the reader. The card refuses to work unless activated with the correct password. This effectively prevents the use of stolen license cards. The password can be distributed on paper with the card. For convenience, the users should be able to disable the password feature in environments where theft is not a major threat.

Even with the password protection, there is still the danger that someone removes the card not for his own profit but to cause damage to the owner. Vandalism is a problem for public-access computers in places like universities and libraries. The card could be protected by enclosing the card reader inside the workstation casing or by using lockable special-purpose readers.

A more interesting scenario is that the thief transfers the license onto his own card. A hacker could even break into the computer from the network and invoke the transfer procedure without having physical access and without exposing himself to much danger of being identified. Again, a separate one-time password should be required by the card before the transfer. Since the transfer is activated only once for each card, passive sniffing for the passwords does not benefit an attacker. In theory, the hacker could take over the transfer process after the user has entered the password and replace the destination card certificate with his own. Therefore, license transfers should be done on a trusted workstation, preferably off-line. An alternative protection that prevents attacks from the network is a physical write-protect switch on the card that must be shifted to allow the transfer.

8 Evaluation

The main goal in the development of our license management scheme was to make the use of hardware tokens user-friendly. In particular, we have solved the problem of smart-card juggling. Although the user must insert a smart card into the reader, it is not any more necessary to periodically switch between cards. Licenses of several software packages are transferred onto a single card.

The license transfer is an extremely simple procedure for the user. He inserts the two cards into the smart card reader, the newer card first. (If the order is wrong, he is asked to reinsert the first card.) He may be asked to provide a floppy disk for backing up the certificates.

The system requires the user to have a smart card reader on every workstation and the license card must be in the reader for most of the time. Beyond the card reader, no changes to existing hardware (e.g. Internet connection, secure processors or hardware identity numbers) are needed. After the initial investment in the readers, the marginal cost of protecting each new product is small. Since the certificates are stored and handled mostly outside the cards, the storage and computational capacity needed in the smart cards is bounded.

We have seamlessly integrated shrink-wrap and on-line sales of software licenses. The license management does not require any changes to existing software distribution channels. In particular, incorporating a smart card into shrink-wrap software packages does not increase the workload at retail stores or require users to have network connections.

The security of the system relies on two non-trivial assumptions. First, the smart card must be tamper resistant in the sense that the private key cannot be recovered from the card. Recovering the key of even one card makes it possible for a professional pirate to sell counterfeit licenses. With on-line distribution of licenses, the pirate must crack a new card when the previously cracked card becomes so old that the on-line store refuses to sell new licenses to it. If software is not sold on-line, the pirate must recover a new key for each new software product or version. It depends on the state of the tamper-resistant smart card technology how long it takes to analyze a card. Service can be denied to those users of pirated software who try to utilize on-line updates and services associated with the products.

Second, the checking for the token in the software package must be obscured in such a way that the check cannot be disabled. Since the public key of the software publisher (PMK) is used for the checking, one should not be able to change this key in the code. Like tamper-resistant cards, obscured software can be analyzed with time and resources. In this case, it is probably the easier line of attack. We suggest discouraging the use of modified software by issuing warnings to the user.

When these basic assumptions are satisfied, the license transfer protocol itself is fairly robust. The transfer process cannot be interrupted to prevent erasure of the old license because the private key is erased as soon as the delegation certificate has been signed. We state formally the claim that the protocols do not allow copying of licenses (see the Appendix for the proof):

Proposition 1: The number of keys with valid licenses is at most equal to the number license certificates signed by PMK.

Moreover, licenses are not easily lost because the delegation certificates can be reread from the card at any later time and the certificates are backed up on the workstation hard disks or on floppy disks.

In summary, the license management system prevents multiple use of one license and it increases significantly the work of professional pirates. Although the user must keep the license card in the smart card reader, it is much more convenient than having separate tokens for each product.

9 Protocol extensions

Our license management protocol can be extended in several ways to increase its flexibility for users.

Although the protocol is fairly robust against accidental loss of licenses, there should be some off-line recovery mechanisms in case the license card is damaged or the delegation certificates are lost. The software producer can be generous with replacing cards and lost licenses. If a log is kept of the customers who receive a replacement certificate or smart card, the number of customers willing to cheat to get one extra copy of the product is likely to be small.

Although we have designed the license transfer protocol with local transfer in mind, the protocol itself has no restriction for remote transfer over a network. If this is implemented, the customer can transfer licenses over large distances without waiting to get the physical license card in mail. The remote transfer cannot be abused so that several remote users would pool to share a license (with at most one user at a time) because a new smart card is needed for each transfer.

In order to have only a single card per workstation, software publishers must co-operate. All card must use the same protocol and meet the same standard of tamper-resistance. Fortunately, it is not necessary to have all publishers share the master key PMK. Instead, the products of each publisher can check for a delegation chain starting from its own master key. These publisher keys, however, cannot be used for signing the card certificates because the safety of the cards affects all publishers. For this purpose, another layer of delegation can be added: a trusted agency holding a master key that will certify card manufacturers' master keys. The manufacturer will include its certificate on the card and sign the card key CK with its own key. This allows accreditation of new manufacturers at any time.

Naturally, the use of the delegation certificates for license management is not restricted to smart cards. The same ideas could be used with any intelligent hardware tokens as long as the tokens are capable of processing public-key signatures and their cost is low enough so that they can be discarded. Different physical implementations of the tokens can be mixed as long as they follow the same protocols. Non-discardable physical tokens such as dongles and chips embedded in the computer hardware work well but licenses should be only transferred onto them, not from them. One possibility is to have one embedded token per workstation and to use smart cards only for license distribution.

Finally, an alternative to having a card for each workstation is to let each user carry a personal license card. That naturally leads to using the card as a general-purpose identifier for the user. Our protocols are equally well suited for many other purposes such as maintaining personal key rings for smart-card locks. However, such applications are beyond the scope of this paper.

10 Conclusion

We have described protocols for binding software licenses to tamper-resistant smart cards, for transferring them between cards and for buying licenses on-line. There must be smart card readers at the workstations but no network connection or other changes to existing hardware are needed. The protocols support software distribution both through retail stores and over the Internet. The user can transfer licenses from several cards onto a single card so that juggling between several card in the reader is eliminated. The transfer protocol is easy and intuitive for the user. The smart cards must be able to process public-key signatures. In other respects, the protocols are simple both to implement and to analyze. Most of the data involved is stored outside the smart card. The protocols may also have applications in other system where smart cards are used for storing credentials.

11 Acknowledgments

Tuomas Aura's work was funded by Helsinki Graduate School for Computer Science and Engineering (HeCSE) and Academy of Finland and it was done mostly at UC Davis Computer Security Laboratory.

References

[1] Ross Anderson and Markus Kuhn. Tamper resistance — a cautionary note. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, pages 1–11. USENIX Association, November 1996.

[2] Tuomas Aura. Distributed access-rights management with delegation certificates. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Distributed and Mobile Objects*, LNCS. Springer, 1999.

[3] Stefan Brands. Untraceable off-line cash in wallets with observers. In *Advances in Cryptology - Proceedings of CRYPTO '93*, volume 773 of *LNCS*, pages 302–318, Santa Barbara, 1993. Springer-Verlag.

[4] David Chaum and Torben Pryds Pedersen. Transferred cash grows in size. In *Advances in Cryptology - Proceedings of EUROCRYPT '92*, volume 658 of *LNCS*, pages 390–407. Springer-Verlag, May 1992.

[5] Carl M. Ellison, Bill Franz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylönen. Simple public key certificate. Internet draft, IETF SPKI Working Group, March 1998.

[6] Amir Herzberg and Shlomit S. Pinter. Public protection of software. *ACM Transactions on Computer Systems*, 5(4):371–393, November 1987.

[7] Burt Kaliski and Jessica Staddon. PKCS #1: RSA cryptography specifications, version 2.0. Internet draft, IETF Network Working Group, September 1998.

[8] Davis P. Maher. Fault induction attacks, tamper resistance, and hostile reverse engineering in perspective. In *Proc. 1st International Conference on Financial Cryptography FC '97*, volume 1318 of *LNCS*, pages 109–121, Anguilla, British West Indies, February 1997. Springer Verlag.

[9] Nasir Menon and Ping Wah Wong. Protecting digital media contect. *Communication of ACM*, 41(7):35–43, July 1998.

[10] Mondex. Mondex home page, 1998. URL:http://www.mondex.com.

[11] Information Technology Association of America. Intellectual property protection in cyberspace: towards a new consensus. ITAA discussion paper, 1998.

[12] Hans-Henning Pagnia and Ralph Jansen. Towards multiple-payment schemes for digital money. In *Proc. 1st International Conference on Financial Cryptography FC '97*, volume 1318 of *LNCS*, pages 203–215, Anguilla, British West Indies, February 1997. Springer Verlag.

[13] John Phipps. Physical protection devices. In Derrick Grover, editor, *The protection of Computer Software - its technology and applications*, British Computer Society (BCS) Monographs in Informatics, chapter 3. Cambridge University Press, 2nd edition, 1992.

[14] George B. Purdy, Gustavus J. Simmons, and James A. Studier. A software protection scheme. In *Proc. 1982 Symposium on Security and Privacy*, pages 99–103, Oakland, California, April 1982. IEEE Computer Society Press.

[15] Steve R. White. ABYSS: A trusted architecture for software protection. In *Proc. 1987 IEEE Symposium on Security and Privacy*, pages 38–51, Oakland, California, April 1987. IEEE Computer Society Press.

Appendix (proof of protocol security)

We consider the licenses for a single software product.

Definition 1: A key CK *has a valid license* if the private part of the key CK is unerased, and there is a card certificate signed by PMK and issued to the public part of CK, and a *certificate chain*:

a license certificate signed by PMK to CK_0,
a delegation certificate signed by CK_0 to CK_1,
a delegation certificate signed by CK_1 to CK_2,
$\vdots$
a delegation certificate signed by CK_{k-1} to CK_k

such that $CK_k = CK$. □

This forms a valid license because the verifier checks for these conditions before allowing the use of the software. It is possible that $k = 0$, i.e. there are no delegation certificates. We ignore the check for the other card certificates (of $CK_0 \dots CK_{k-1}$) and for the production dates because they have effect only if some other assumption is broken.

Assumption 1: PMK only issues license and card certificates to authentic card keys. An authentic card key only issues delegation certificates to keys with a card certificate. □

Assumption 2: For every authentic card key CK, exactly one of the following holds:

1. CK has signed no delegation certificates.
2. CK has signed exactly one delegation certificate and the private key CK has been erased. □

The second assumption follows from the policy of erasing the private key immediately after signing a delegation certificate.

Proposition 1: The number of keys with valid licenses is at most equal to the number license certificates signed by PMK. □

Proof: The subjects of license certificates (CK_0) are always authentic card keys. An authentic card key only delegates to a key with card certificate and such keys are authentic card keys (Ass. 1). Consequently, all keys in the chains starting from license certificates are authentic card keys. The authentic card keys delegate to at most one other key and a key that has delegated is itself erased (Ass. 2). Thus, the certificate chains starting from license certificates do not branch and only the last key in a chain can be unerased.

If there would be more keys with valid licenses than license certificates, there should be some two valid licenses whose corresponding certificate chains (Def. 1) begin with the same license certificate but end in two different unerased keys. However, this is not possible since the chains do not branch and only the maximal-length chains end in unerased keys.