A Proposed Software Protection Scheme

Ping Wang and Kwangjo Kim

Cryptology and Information Security Lab.
International Research center for Information Security (IRIS)
Information and Communications University (ICU), Korea

Abstract

The term software protection is used to describe all the methods that a software vendor can use to ensure that users can run only those copies of software that have been legally purchased. The aim of software protection is to make the effort of illegally running a program equal or more expensive than developing the program. In this paper, we will introduce concepts of entrance security and function related, which ensure cryptographic techniques really work in the software protection system. Then, we propose an efficient software protection scheme based on these concepts.

1. Introduction

Software piracy is the practice of copying and using a software product without the permission of its owner or developer. Although most computer users today are aware of that unauthorized use and duplication of software is illegal, many still show a general disregard for the importance of treating software as valuable intellectual property.

Development of technology for the prevention of software piracy is important for the software industry [6]. The advent of mobile computing will only make the problem of software piracy worse. In the future it will be commonplace for users to carry applications on a mobile computing device. However, these applications would often be

☐This work was supported by a grant No. R12-2003-004-01004-0 from the Ministry of Science and Technology, KOREA.
transferred and executed on remote compute servers that would be part of the ubiquitous computing infrastructure in the future. In light of the greatly improved computing power of modern day processors, it is acceptable to expend a fraction of this computing power on protecting software. The goal of this work is to develop an approach to prevent malicious users of the software from creating fully functioning unauthorized copies of protected software.

1.1 Various categories of software piracy include:

Soft-lifting: purchasing a single licensed copy of software and loading it onto several computers, contrary to the license terms. For example, sharing software with friends, co-workers and others.

Uploading and downloading: making unauthorized copies of copyrighted software available to end users connected by modem to online service providers and/or the Internet.

Software counterfeiting: illegally duplicating and selling copyrighted software in a form designed to make it appear legitimate.

OEM unbundling: selling stand-alone software that was intended to be bundled with specific accompanying hardware.

Hard disk loading: installing unauthorized copies of software onto the hard disks of personal computers, often as an incentive for the end user to buy the hardware from that particular hardware dealer.

Renting: unauthorized selling of software for temporary use, like you would rent a video.

1.2 Software pirates can be divided into several categories:

- Dealers selling hardware pre-loaded with illegal software.
- User organizations making unauthorized copies of software for internal use.
- "Professional" software counterfeiters.
- Competitors using unauthorized software copies to develop competing products.
- Hackers' web sites offering illegal software to users.
- Any individual who makes an unauthorized copy of someone else's software program.

The term software protection is used to describe all the methods that a software
vendor can use to ensure that users can run only those copies of software that have been legally purchased. The aim of software protection is to make the effort of illegally running a program equal or more expensive than develop the program.

While the direct objective of software protection has always been of a preventive nature, today the quality of software protection is also measured by the broadness of the solution and its ability to answer further software commerce needs. A good software protection solution will not only enable software vendors to increase their revenues, but also gain remote control over their software and its distribution and channel management. It should also enable innovative sales opportunities and flexible licensing capabilities, which give vendors the freedom to concentrate on doing business without the worries of license violations.

2. Related Work

In this section we will summary some proposals [1, 2, 3, 9, 10] for software protection and mechanisms, considering aspects like security, convenience and practical applicability.

2.1 Software Protection Methods

Two major forms of control mechanisms can be used to create the incentives for obtaining the software legally and penalties or disincentives for using the software illegally.

2.1.1 Legal and Marketing Control Mechanisms

Software license agreements and copyright law are being used to prevent software piracy. Of the two legal control mechanisms copyright law is the more important. It covers all software automatically. However, these control mechanisms do not prevent a user from inadvertently or intentionally copying the unauthorized software. The power of these mechanisms is in the legal remedies available to software suppliers against software pirates.

Software publishers and suppliers are also using various marketing and educational control mechanisms to reduce the incidence level of software piracy. These include
volume discounting, site licensing, strong focus on customer support, teaching that software theft is wrong, etc. This kind of control may prevent accidental use of unauthorized software by end users, but it will not stop software pirates.

So because marketing and legal measures do have value, you should always use them, but never rely on them. They cannot help you when it comes to individual unauthorized users who, on their own, can cause huge damage.

2.1.2 Technology-Based Control Mechanisms

Technology-based control mechanisms include all programs and devices that prevent the unauthorized use of software. This form of protection has much going for it unlike legal mechanisms it cannot simply be ignored, and it is often cheaper to implement.

Technological mechanisms use an encryption process or other protective measures to protect the software. The most common approaches are: making copy-resistant distribution disks, access locks, hard-coded numbers in computer memory, software resident inside a ROM chip and copy protection security devices, also known as dongles. This form of control will prevent accidental use of unauthorized software by the end user. In some cases these methods will also prevent software pirates, depending on the strength of the control mechanism.

Technological protection comes in two forms: software and hardware, the latter being a much stronger mechanism, aimed at combating software pirates. Today, software-based copy-protection is associated with license management systems and license files. The strength of these systems however, is in their licensing capabilities. The fact is that many software-based copy protection vendors offer a hardware-based copy-protection key when it comes to enhancing the protection level of their security solutions.

2.2 Hardware-Based Software Protection Keys

Hardware-based copy protection systems offer the best, most proactive solution for software vendors. These systems offer the most secure solution while placing a minimum burden, if any, on the software user. There are two major types of hardware-based solutions, which differ in strength: EEPROMs and the much stronger ASIC-based solutions.

Electrical Erasable Programmable Read-Only Memory (EEPROM) is a standard memory chip that can be purchased off the shelf. Dongle vendors that use these chips
will often mask them in an attempt to physically conceal their identity and the known technology.

EEPROM contents are almost always readable via software and they generally contain plain data that make it possible to emulate the process with a software patch.

EEPROM-based copy-protection keys are "dumb" devices that let you store data. This is because validating the presence of the device in order to determine whether the user is authorized is summed up in a simple operation of reading what is stored on the EEPROM and comparing it to an expected value.

2.3 ASIC-Based Devices

Application Specific Integrated Circuit (ASIC) is a chip designed for a special application and cannot be purchased from just any electronics store. An ASIC can be pre-manufactured for a special application or it can be custom manufactured (typically using components from a "building block" library of components) for a particular customer application. This is an extremely secure technology because only the company that designed and manufactured it knows the technology. Microcontrollers also fall under this category.

Both ASIC-based and microcontroller-based keys are referred to as "intelligent" devices because they can deploy encryption. ASICs have an on-chip encryption engine designed within their logic.

Microcontrollers activate an encryption algorithm that is burned into an internal EEPROM.

2.4 Encryption the Heart of Software Protection

The way to validate that a user is legally using a software application is to protect it with a hardware-based key and to verify that the key is connected to the user's computer during the application's runtime.

There are two ways to perform the verification:

Send the key a query and check the response; if the response is as expected, then the key is present. This approach is fundamentally insecure. Checking for an expected response can be easily hacked and removed leaving the application bare from protection.

The other method, most secure, is to "use" the key (as opposed to checking it) to decrypt encrypted strings or text and to deploy those within the application. As a result
the application will run properly, if at all, only when the strings are decrypted properly.

3. Our Proposed Scheme

As it is usual in other fields of information security, in software protection there are no completely secure solutions. The objective of a software protection scheme is to make the attack to the scheme difficult enough to discourage dishonest users.

The new scheme is based on the truth that, those schemes based on tamperproof hardware tokens that have been proposed in the literature have been analyzed concluding that all of them are based on the check of the presence of the token and are therefore vulnerable to code modification attacks. Although the popularization of smart cards and their evolution in storage and processing capacity may lead us to consider them the most appropriate choice for our scheme, it is expensive and inconvenience to use smart card for software protection. However, our design does not depend on this technology and, consequently, our solution deal with the problem of vulnerable to code modification attacks.

A secure software protection scheme can be designed using just tamperproof hardware tokens [4, 5]. In this scheme some sections of the software to be protected can be substituted by functionally equivalent sections to be processed in the main code. In this way, the protected software is divided and will not work unless it cooperates with the right subfunction. Code modification attacks will not succeed in this case. In fact, the only possible attack is to analyze the variant and functions during running. If we include enough functions, with enough importance in the main code, the attack described could become impractical.

This scheme allow the distribution of the protected software using Internet because of the easy way to register online and distributed with the software. With the purpose of avoiding the aforementioned problems we will introduce the cryptography as the second building block of our software protection scheme.

3.1 Modeling the entities

It is important to adopt a clear and extensible model for the system entities and their relationship with other entities. The basic principle of the model is to clearly separate
and identify the three core entities: Users, Content, and Rights as shown in Figure 1. Users can be any type of user, from a rights holder to an end-consumer. Content is any type of content at any level of aggregation. The Rights entity is an expression of the permissions, constraints, and obligations between the Users and the Content. The primary reason for this model is that it provides the greatest flexibility when assigning rights to any combination or layering of Users and Content. The Core Entities Model also does not constrain Content from being used in new and evolving business models.

Fig. 1. Core Entities Model

3.2 Code Modification Attacks and Entrance Security

Among the previous proposed solutions that rely on some hardware component, one of the most popular consists in the use of hardware tokens that are difficult to duplicate, which are connected through some communications port to the computer running the software. The protected software checks the presence of the token and refuses to run if the check fails. Examples of this kind of systems are hardware keys or dongles. These systems usually have the problem of the incompatibility between tokens of different applications.

When the tokens are smart cards, as it is expected that the computer will include just one card reader, the user must continuously change the card, a problem known as card juggling that represents a serious inconvenience.

The check of the presence can be done in different ways: the simplest is to read a value from the communications port, but, commonly, to avoid that the interception of the communication in that port allows the attacker to replicate the token, the software will send a value (called challenge) that the token has to process, the software can predict
the result that the token must send back. In any case, whatever the check is, it is not hard to bypass this protection, as the access to the communications port or the reader are easily found in the executable code. The check can then be bypassed obtaining a completely functional copy of the software as figure 2 shows. This process can even be automated by specially designed programs called "patches".

We introduce Entrance Security to represent these kinds of problem, which avoid cryptographic techniques by code modification. To solve entrance security, should we introduce function related concept, which guarantee cryptographic techniques really work.

```
... ...
OUT data
IN rep
    --respOk is computed
    CMP resp,respOK
    JZ continue
    HALT
    :continue
        --normal processing
        --continues
...  ...
```

**original code**

```
...  ...
OUT data
IN rep
    JMP continue
    --respOk is computed
    CMP resp,respOK
    JZ continue
    HALT
    :continue
        --normal processing
        --continues
...  ...
```

**bypassed code**

*Fig.2. Code modification to bypass the check of the presence of the token.*

### 3.3 Function Related

*Function Related* means related the "Check value" with the function of the code or sub-modules. There are two steps for Function Related:

First step: compute "Check value", such as CRC value, Hash value, or Signature value.

Second step: bind the "Check value" with function code or sub-modules.

With this method, any code modification can be detected by check value, also bypass the check of the present of token is useless because the check value involved function of the code or sub-modules of the software. Thus we can achieve the aim of software protection, to make the effort of illegally running a program equal or more expensive than developing the program.

By binding software modules with the check value, we ensure that an adversary cannot steal the entire software. Given that the adversary cannot compute the check
value, he or she must construct these modules by observing their runtime interaction with the check value. However, we construct the sub-modules such that they are hard to identify because they are formed through program slices whose function is hard to determine. The number of variables that form the sub-modules and the relationships among the values as defined by the main code are both unknown to an adversary. Thus, it is not possible to easily determine the functions performed by the sub-modules so constructed.

4. Concluding Remarks

Our approach for prevention of software piracy through tamperproof appears to be similar to research by Sander, T. and Tschudin, C. F. [8]. However, their work is based on application splitting which is inefficient and suffer from runtime observing. In [2] a network of security units, called guards, work together to detect changes to the binary. The guards essentially perform checksum on parts of the binary to detect if the software has been modified. But this method can not prevent detecting and removing the guards and hence vulnerable to code modification attack. Another approach described in [7] provides a mechanism that redundantly tests for changes in the executable code, as it is running, and reports modifications. It's far from efficient and also do not provide protection against code modification.

We have described a robust software protection scheme based on entrance security and cryptographic techniques. Related schemes based on tamperproof hardware tokens that have been proposed in the literature have been analyzed concluding that all of them are based on the check of the presence of the token and are therefore vulnerable to code modification attacks. Considering that the new scheme is not only based on that check, new scheme also bind the check value with function code or sub-modules, therefore code modification is not a potential attack. We have also introduced possible alternative applications of the scheme. Hence, we can conclude that the advantages of the presented scheme are robustness against different attacks (bypassing the check, code substitution and attacks to the license management protocols), confidence for the user, efficient use of the computational resources, free distribution and copy of the software, selective license transfer, control of the expiration of the licenses and
We presented a novel approach for protecting software from being stolen. Tools to produce protected software automatically from unprotected executable programs, applet protection and payment integration are under development. We are studying the possibilities that the combination of function hiding techniques [8] with our scheme.

[REFERENCES]