Software Licence Protection and Management for Organisations

Muntaha Alawneh and Imad M. Abbadi

Abstract Most organisations have recently converted their physical assets into digital forms. This underlines the needs to have different types of software products to manage such information, and raises security concerns for protecting software products from being illegally used in organisations. This paper proposes a licence management solution that protects software products from being illegally used. The proposed scheme is based on dividing an organisation devices into dynamic domains, each of which is bound to a single software product. Each dynamic domain has a predefined number of devices that can use the dynamic domain-specific software product. This number is specified by the software provider and is stored in the software licence file. In this case a software product can be installed on multiple devices, and a device can possess multiple software products by joining multiple dynamic domains. The proposed mechanism ensures that the number of used copies of software product does not exceed the limit that is agreed with the software provider.

1 Introduction

Consumers and organisations are moving into digitising content, which becomes more convenient than physical forms. Organisations in its wider definition including private and public sectors, universities, governments and many others, have replaced their system and workflow so that everything is digitised. Digitised information requires software products to process it, stores it and enables it to be easily accessed so that it achieves organisations' main functionality.

Software providers understand the importance of providing appropriate software products that meet the current and expected future needs for managing and accessing digitised information. However, one of the main problems facing software providers

Muntaha Alawneh and Imad M. Abbadi
Information Security Group, Royal Holloway, University of London, Egham, Surrey, TW20 0EX, UK, e-mail: {M.Alawneh,I.Abbadi}@rhul.ac.uk

is that their copyright is not sufficiently protected within organisations. Many organisations abuse the weak protection for software products by using the software product on many devices they have without paying usage fees. Currently, more than one out of three software applications are pirated. It is expected that US$300 billion will be spent on PC software over the next five years. During the same period it is expected that almost US$200 billion worth will be pirated [5].

Most researches in this area focus on personal networks. Personal networks have different requirements than organisations [3, 4]; for example, an organisation has larger size, more users, different mechanisms for licence enforcement and different copyright law regulations [6]. This in turn demonstrates the importance of finding a proper solution focusing on both organisations and software providers requirements. There are few schemes attempting to address software protection for organisations; however, these schemes have many problems and security flows in addressing organisation requirements. These are discussed in section 2.

Software protection is not only for the benefits of software provider (i.e. licence enforcement), but it is also important for organisations. For example, some organisations need to securely protect their own specific-software products from getting leaked outside it and used by others, e.g. to protect their own secrets, specific design, etc. Moreover, protecting a software product from getting leaked helps, in someway or another, in protecting content. This is because leaking an organisation-specific software product enables a third party to create, using the leaked software, a forged content in the same format that could be created in the organisation.

This paper proposes a mechanism for addressing software licence management for organisations. In this scheme we analyse the main security concerns facing software providers, specifically for organisations. Next, we propose a solution for managing software licencing for organisations.

Our novel idea is based on organising an organisation devices into dynamic domains. Each dynamic domain is bound to a single software product, which is itself bound to a licence file. The licence file specifies the rules for using the software product, and it includes the maximum number of devices that can use the software product at any time. These are stored in the licence file and are agreed between software providers and organisation administrators. The licence file also specifies the dynamic domain unique identifier to which this licence file is bound.

Using dynamic domains not only provides software protection, but it also helps organisations to manage their own licences. The latter is ensured, as each software product only requires a single licence file for all devices that require using the software product. This reduces the total number of licences required per software product in an organsation, hence helps in managing software licencing, storing it, and using it. In addition, the proposed solution considers organisations needs by adding to dynamic domains other features that are required by organisations, such as: expandability and shrinking; i.e. domains can be expanded or shrink based on organisations dynamic structure and needs, devices can move between different dynamic domains and use each dynamic domain-specific software product without requiring to go through the process of ordering new licences or even to pay for new licence fees (i.e. a device can join the domain, which it needs to use its software product, or

leave the domain that it does not need to use its software product. This is conditional by having the number of devices in a domain does not exceed the limit permitted by software provider, as stored in the domain associated licence file). Moreover, the proposed solution is designed in such a way it is easy to use, and provides ease of recovery. Hence, the proposed solution satisfies both software providers and organisations needs.

2 Problem Definition

Software products licensing are, typically, charged based on number of devices, or, sometimes, on number of users, which use a software product. Although there are different techniques trying to enforce the rules included inside software licences; however, most of these techniques have many security flows, also, some of them have usability limitations (as described below). Moreover, these techniques have been abused many times, e.g. an organisation could buy one licence for a software product, and then illegally installs it (using the same licence) on unlimited number of devices; see, for example, [1, 2]. This is a clear breach of copyright law, and certainly software providers are not happy with such mechanisms.

Each software product, typically, has a licence-agent that regularly checks the availability and validity of a proper licence for the associated software product. Licences are protected using software-only techniques, a combination of both software and hardware protection techniques, and/or deterrent measures. In the remaining part of this section we discuss these techniques, which are used by software providers for protecting their own software products. In addition, we discuss other issues related to managing software licences for organisations.

2.1 Software-only Techniques

A software-only technique is based on having a software agent that is installed on a consumer device, and which requests a serial key or a password to enable the accessing of an associated software product. This serial key/password is provided by the software provider to the consumer after paying a proper licencing fees, and which needs to be inserted by the consumer whilst installing the software product. The licence-agent protects the serial key and stores it somewhere inside the consumer device. It then regularly checks the availability and the validity of this key to authorise the associated software to run.

The serial key/password is either bound to a single device, or it can work on any device. In the latter case, the software product is easier to be hacked; for example all devices in an organisation can use the same serial key to run a software product, which is originally bought to work on a single device. In the earlier case a serial key is bound to a permanent factor inside a device. Such a technique is imple-

mented by some vendors such as Sun Microsystems [12] and weblogic [15]. This typically would be based on either a device hardware-id or IP address. However, such a mechanism has been attacked, as the hardware-id (after the system starts up) is stored in unprotected area inside the device memory, which can be bypassed or changed [1]. Binding the serial number with an IP address could also be easily hacked; a machine could have multiple network cards with different IP addresses, so a network card (which should not be connected to the main network to prevent address conflict) could be configured to have the right IP address that the software checks before starts up; for example, a company can buy a software product to work on a single machine that has a predefined IP address, and later on, the company can configure all its PCs to have two network interfaces each has two IP addresses. The first interface to have the same IP address used for licencing, and which needs to be disconnected from the main network (to prevent address conflict). The second network interface to have a public IP address and is connected to the main network.

2.2 Software and Hardware Techniques

Other solutions are mainly based on combinations of both software and hardware mechanism. Although these mechanisms are much secure than software-only techniques; however, they still have security flows and usability limitations. Such techniques are mainly based on using a tamper-resistant component storing, for example, a serial key that the licence-agent checks every time it runs. A common example of this type what is know by a 'dongle', which is "a small hardware device that connects to a computer and acts as an authentication key for a particular piece of software" [16]. Using a dongle does not solve the defined problem, as it is not robustly and securely integrated with computer devices [11]. Moreover, and most importantly, dongles are not practical and more expensive to have. This is because a dongle is a software-product specific, and a device, typically, has multiple software products from different vendors each requiring a specific-dongle to be connected to a device port all the time a software is running. This raises serious usability limitations for small devices. Also a device normally has a limited number of ports that are usually used for other purposes; e.g. connecting a printer or scanner, so it is not practical to have multiple software products using this technique on a device.

2.3 Deterrent Measures

Other software vendors, such as Oracle [9], do not enforce licences using cryptographic techniques; i.e. this licensing mechanism relies upon deterrent measures, which is based on copyright law enforcement.

2.4 Other Issues

In addition to the problems associated with each technique described in the previous sections, these techniques mainly focus on enforcing licences rules on a single device (except for the case of site licence, where a licence can be used on any device). This raises serious licence manageability problems for organisations, as an organisation, usually, has hundreds or even thousands of devices each run multiple software products from different vendors. Hence an organisation ends up with thousands, and even tens of thousands, of different licences each of which is bound to a single device and a single software product.

From the above we can see the importance of finding an acceptable solution for the problem of software licence management for organisations. In order to find a practical ground, such solutions should satisfy organisations, software providers, and copyright law requirements.

3 Dynamic Domains

Software providers need a solution which solves the problems defined in section 2. Using dynamic domains, which can be reallocated dynamically between organisation devices, helps to solve these problems. A dynamic domain is a domain consisting of one or more devices chosen from the organisation devices, each dynamic domain is bound to a single software product. The number of devices in a dynamic domain must not exceed the number of devices that can use the software product bound to that domain, as specified in the licence file which is provided by the software provider. This ensure that the maximum number of devices using the assigned software product do not exceed the maximum permitted number of devices agreed with the software provider. Each dynamic domain has a unique identifier, and a unique symmetric key. The dynamic domain symmetric key is used to protect the software product inside the dynamic domain devices. This key is only available inside devices member of the domain, so that only these devices can access the software product bound to the domain.

The dynamic domain creation process is performed by an organisation authorised security administrators, who choose devices that need to be bound to one or more dynamic domains. This binding is performed using a master control device, which needs to be trusted by software providers. The master control device intermediates the communication process between software providers and devices in an organisation that is going to use a software product. In addition, the master control device enforces the limits inside the licence file by ensuring the number of devices assigned to a dynamic domain does not exceed the authorised number of devices in the licence file, which are provided by the software provider whose software product is binded to the dynamic domain. These are explained in detail in section 5.

4 Proposed Model

In this section we describe the main entities constituting the proposed model.

4.1 Hardware Requirement

4.1.1 Devices.

Devices are commercial off-the-shelf PC hardware enhanced with trusted computing technology as defined by the Trusted Computing Group (TCG[1]) specifications [13, 14]. TCG compliant trusted platforms (TP) are not expensive, and are currently available from a range of PC manufacturers, including Dell, HP and Intel [10].

4.1.2 TCG Overview.

TPM: The TCG specifications require each TP to include an additional inexpensive hardware chip to establish trust in that platform. This chip is referred to as the Trusted Platform Module (TPM), which has protected storage and protected capabilities. The TPM is typically implemented as a processing engine that is separate from the TP's main processing environment.

Protected Storage: The TP protects all secret keys required by devices. Stored secrets are only released after the platform's software state has been measured and checked. Storage, and retrieval are carried out by the TPM. Therefore, if a software process relies on the use of secrets, it cannot operate unless it and its software environment are correct. The latter ensures that the software process operates as expected. Once a TPM has been assigned an owner, it generates a new Storage Root Key pair (SRK), which is used to protect all TPM keys. The private part of the SRK is stored permanently inside the TPM. Other TPM objects (key objects or data objects) are protected using keys that are ultimately protected by the SRK in a tree hierarchy structure. The entries of a TPM platform configuration registers (PCRs), where integrity measurements are stored, are used in the protected storage mechanism. This is achieved by comparing the current PCR values with the intended PCR values stored with the data object. If the two values are consistent, access is then granted and data is unsealed.

Attestation: Establishing trust in a TP is based on the mechanism that is used for measuring, reporting and verifying platform integrity metrics. TP measurements are performed using the RTM (Root of Trust for Measurement), which measures software components running on a TP. The RTS (Root of Trust for Storage) stores these measurements inside TPM shielded locations, which is referred to as the Platform Configuration Registers (PCR). Next, the RTR (Root of Trust for Reporting) mech-

[1] http://www.trustedcomputinggroup.org

anism allows TP measurements to be reliably communicated to an external entity in the form of an integrity report. The integrity report is signed using an AIK (Attestation Identity Key) private key, and is sent with the appropriate identity credential. This enables a Verifier to be sure that an integrity report is bound to a genuine TPM[2].

4.2 Master Control Device

The master control device is a trusted device that has all TP features, as defined in section 4.1. The master controller is a single logical entity, although its implementation may be a distributed one[4]. Each organisation has a specific master control device in charge of managing the organisation dynamic domains and all devices membership in each dynamic domain. The master control device has the following main functionalities.

- The master control device communicates with third parties, i.e. software providers, for downloading software products associated with proper licence files. The licence file, associated with the software product, contains a limit specifying the total number of devices that can use the software product. The master control device enforces this limit
- Creating and managing dynamic domains. This includes the following:
  - Securely generating and storing each dynamic domain-specific unique identifier, protection key, and a public key list which includes the public keys for all devices member in the dynamic domain.
  - Attesting to the execution environment status of devices added to a dynamic domain, ensuring they are trusted to securely store dynamic domain keys and execute as expected.
  - Adding devices to a dynamic domain by releasing the dynamic domain-specific key (i.e. the software protection key) to devices member of the dynamic domain.
- Managing software licencing, by ensuring each software product is bound to a single dynamic domain that has a maximum number of devices does not exceed the number of devices in the licence file associated with the software product.

5 Process Workflow

The workflow of the proposed system is divided into the following phases.

[2] One might argue that the device states might change after getting attested. This is solved by using the new generation of Intel/AMD hardware technology that stops DMA or by using Virtualisation technology as has been described in [10].

5.1 Master Control Device Initialisation

This section describes the process of initialising a master control device, which establishes the dynamic domains. The first time a master control device is initialised, the master control device instructs the organisation security administrators to provide their authentication credentials. The master control device then stores in its protected storage[3] the authentication credentials of the organisation security administrators associated with its trusted execution environment state (which is stored in the TPM's PCR based on TCG specification; see, for example, section 4.1). The authentication credential is used to authenticate security administrators before using the master control device. The master control device is used each time the security administrators want to create, expand, shrink or change a dynamic domain.

5.2 Buying Software Licences

This section describes the process of buying and downloading software products, which involves the following steps (figure 1 summarises the protocol for this stage).

1. The organisation administrators need to specify the number of licences the organisation need for a software product, say X (the number of devices in a dynamic domain that will use this software product should not exceed the value of X).
2. An organisation then negotiates the price with the software provider, for X licences. If the organisation agrees on a price, a formal contract is established between the software provider and the organisation that specifies the software product terms and conditions of usage and the maximum number of devices permitted to use the software product.
3. Next, the organisation administrators instructs the master control device to send a request to the software provider to download the software product. The software provider and the master control device exchange each other certificate, extracts the signature verification key from the certificates, and checks that it has not been revoked, e.g. by querying an Online Certificate Status Protocol (OCSP) service, [8]. If so, the software provider attests to the execution status of the master control device based on TCG specifications; see, for example, section 4.1. If the attestation shows that the master control device is trusted, the software provider encrypts the software product with a symmetric key k_S, and creates a licence file containing a one-way hash value of the encrypted software product. This is to bind the software product with the licence file. The licence file also contains the following: the software product encryption key k_S encrypted using the master control device public key, the value of X, the soft-

[3] We mean by storing data in a protected storage is protecting data using the SRK, which its private key part is stored inside the TPM. The protected data is then stored in an unprotected storage (see section 4.1).

ware product identifier *id*, and other usage rules. The software provider signs the licence file and sends the encrypted software product associated with the licence file to the organisation master control device.

4. The master control device verifies the software provider signature, and verifies the content is bound to the licence file by recomputing a one-way hash value of the received encrypted software product and comparing it with the one stored in the licence file. If the verifications succeed, the master control device signs the licence file, and then stores the encrypted software product associated with the signed licence file. Before installing the software product into devices, the master control device should first bind the software product to a specific dynamic domain. This binding could be performed by storing the dynamic domain specific identifier *i* in a specific field inside the software product licence file. This filed is exclusive for only one dynamic domain identifier, which ensures that each software licence is bound to a single dynamic domain.

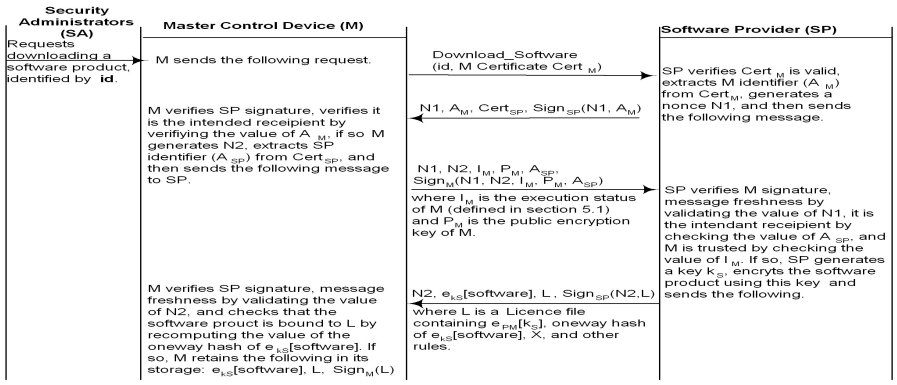

Fig. 1 Buying Software Licence Protocol

5.3 Dynamic Domain Establishment

Whenever an organisation wishes to install a software product into a set of devices, it must do the following (figure 2 1 summarises the protocol for this stage).

1. The organisation system administrators decide how many devices need to use a specific software product at this stage, say N. N would be the initial size of a dynamic domain, and it should not exceed the value of X stored in the licence file (it can be less than that).
2. The organisation system administrators decide which devices that will use the software product; the selection process is based on organisation needs, for example, a dynamic domain could consist of devices owned only by managers

layer, seniors layer, or mixed between different layers. These devices constitute the dynamic domain.
3. The security administrators instruct the master control device to create a new dynamic domain. The master control device then authenticates the organisation security administrators, e.g. using a password.
4. If authentication succeeds, the master control device instructs the security administrators to provide the number N, the public keys of devices that will be in the dynamic domain, and the identifier of the software product *id* that will be used on this dynamic domain.
5. The master control device verifies that the software licence is not bound to an existing domain, by verifying a field in the licence file showing whether it is used by another domain or not, as described in section 5.2, point (4).
6. If the above succeeds, the master control device then securely generates a dynamic domain specific symmetric key k_D, and a dynamic domain specific identifier i. The master control device creates a public key list for this domain consisting of the provided public keys. It then ensures that the size of the public key list equals to N, and verifies that the value of N does not exceed the value of X. k_D and i are associated with the public key list and the value of N, and then stored in the master control device protected storage and bound to a trusted execution environment based on TCG specifications; see, for example, section 4.1. The dynamic domain specific identifier i is also stored in the software product licence file. This is to bind the software product licence with a specific dynamic domain, and to make sure each software licence is bound to a single dynamic domain. The master control device then decrypts the software product encryption key (as stored in the licence file; see section 5.2 point 3), and re-encrypts it using the dynamic domain key k_D and stores the result in the licence file.

5.4 Adding Devices into a Domain and Software Installation

This section describes the process for adding a device into a dynamic domain and the process of software installation, which are performed as follows (in this section we describe the process using bull technique; i.e. a device sends a join request to the master control device. The same process applies using push technique; i.e. when a master control device sends a join domain requst to all devices in the domain. Which way to go for depends on the organisation policy).

1. From each device in the public key list, the organisation security administrators sends a join domain request to the master control device to install the dynamic domain specific key. This request includes the dynamic domain specific identifier i identifying which domain to join.
2. The master control device and the joining device mutually authenticates each other conforming to the three-pass mutual authentication protocol described in

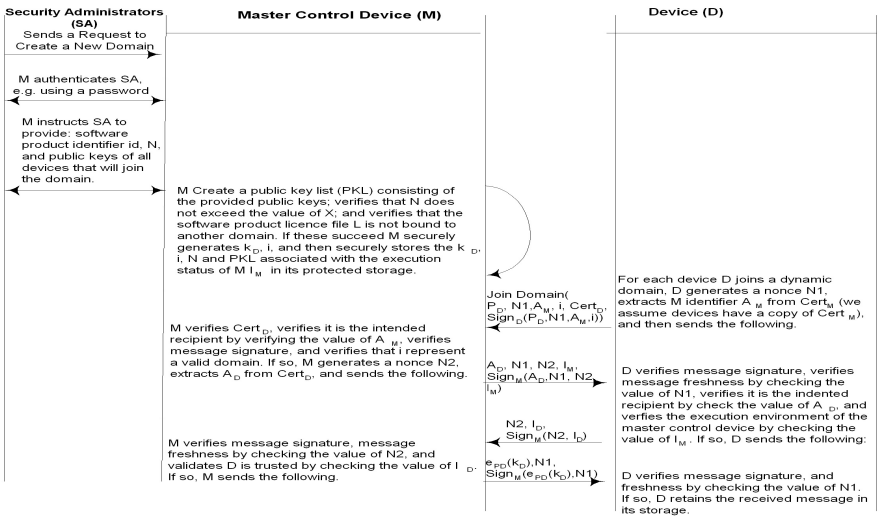

Fig. 2 Domain Establishment and Adding Devices Protocols

[7]. The master control device then attests to the execution environment of the joining device and validates its trustworthiness; as described in section 4.1.

3. If the joining device execution environment is trusted, the master control device checks if the device's public key is included in the public key list for the dynamic domain (as specified in step (1) above). If so, it securely releases the dynamic domain specific key to the device.
4. The device stores the domain key in its protected storage, and binds it to a specific execution environment. This device is now part of the domain, as it possesses a copy of the domain key and its public key matches the one stored in the master control device.
5. Now, all devices member of the domain can download from the master control device the encrypted software product associated with the licence file, which is bound to the domain. All these devices have a copy of the dynamic domain-specific key k_D. Therefore, these devices can decrypt the software encryption key, which is stored inside the licence file encrypted with the key k_D. These devices can then decrypt the software product and access it.

6 Domain Management

In order for a solution to be accepted and be widely used, it should adapt with organisations dynamic structure; for example, an organisation might need to change its strategy, layout, business work flow, and/or replace its devices. In this section we discuss how the proposed scheme covers these requirements, i.e. removing a

device from a dynamic domain, adding a device into a dynamic domain, and key revocation.

6.1 Domain Shrinking

An organisation might need to use a software product on fewer number of devices than it is currently use, or it might need to replace its devices for several reasons, e.g. a hardware failure and the device cannot be recovered, or replace the device with newer technology. In these cases the organisation should still have the right to use the software product on other devices by adding them to the domain, as long as the number of devices in a domain N does not exceed the value of the maximum number of devices X that can use a software product, as stored in the licence file.

The way to remove a device from a dynamic domain is as follows. The master control device needs to attest to the execution status of the device ensuring it is trusted to remove the dynamic domain key from its storage (based on TCG specifications; see, for example, section 4.1). If the device is trusted, the master control device (for each dynamic domain) instructs the device to delete the dynamic domain key. The master control device then removes this device public key from the public key list of the dynamic domain, and decrements the value of N. On the other hand, if the execution status of the device is not trusted, the master control device will not remove this device; i.e. it will not decrement the value N and will not remove the device public key from the dynamic domain-specific public key list.

6.2 Domain Expansion

An organisation can expand a dynamic domain as long as the value of N does not exceed the value of X. In this case, the master control device instructs the security administrators to enter the public keys of the new devices. The master control device then add the number of the new devices to N. The master control device check the new value of N is still less than or equal the value of X. If so, the master control device securely stores the new value of N and updates the public key list with the added values, and finally it allows the new devices to join the domain as described in section 5.4.

6.3 Key Revocation

Hacking a dynamic domain specific key only affects the dynamic domain-specific software product protection. As a precautionary measure, security administrators need to revoke the domain key, and generate a new domain key, which can be done

as follows. The security administrators instruct the master control device to change the key for a specific dynamic domain. The master control device then authenticates the organisation security administrators. If authentication succeeds, the master control device generates a new domain-specific key, and then decrypt the domain-specific software protection key stored in associated licence file with the old domain key, and re-encrypts it with the new domain key. The master control device then re-install this key and the licence file on domain devices; the master control device identifies devices using their public keys, which are securely stored inside the master control device, as described in section 5.3. For each device, the master control device releases the licence file, and the new value of the domain key encrypted using the device public key. The device replaces the old licence file, and stores the domain key in its protected storage and binds it to the same execution environment used for the old key, as it has already been verified as trusted; see section 5.4 point (3).

7 System Analysis

In this section we discuss the pros of using dynamic domains, which are summarised as follows.

- Software Protection and Licence Enforcement. A software is protected from being abused in organisations, as each software product is bound to a single domain with a maximum number of devices must not exceed the number of devices allowed to use the software product. This latter number is specified in the licence file associated with the software product. Having a trusted master control device enforces this limit. Each domain has a unique key, which is used to encrypt the software protection key. This encrypted key is stored in the software-specific licence file. The domain key is securely stored in domain devices, so only domain devices can decrypt the software protection key and access the software product. Moreover, the domain key is bound to a trusted execution environment that should work as expected. Hence, these devices are trusted to enforce the rules stored inside the associated licence file.
- Licence management. By using dynamic domains a software product is protected with a single licence file shared between a set of devices, rather than having a device-specific licence for each software product. Hence, this reduces the total number of required licences, and so it eases licence management.
- Flexibility. This is realised as follows.
  - As it is known, organisations have different layers, e.g. managers, seniors. In addition, organisations are organized in different business processes, e.g. a newspaper type of organisation has an editorial work flow, a publishing work flow, and page layout. A dynamic domain can contain devices from a single layer, or from different layers, based on organisation requirements. This provides an organisation the flexibility to layout its software products on devices based on the organisation functionality.

- An organisation can dynamically move devices between dynamic domains based on changes in its needs. For example, if an organisation requires to change its layout, say after one year, this might require software re-allocation. Assuming dynamic domains are not implemented, some software licences require renewing based on redistributing software products on different organisation devices. This is because a software licence would typically be bound to a device hardware id or an IP address (as discussed in section 2). On the other hand, by using dynamic domains, an organisation does not require renewing software licences. In this case, when a device is reallocated to be used by a new layer (i.e. different business process) that require different software products than it already has, it can join all dynamic domains where the new software products are bound, as long as the number of devices in each domain does not exceed the domain-specific number X stored in each domain licece file. The device also needs to remove all softwares it no longer needs to give chance for another device to use it.
- As we said earlier, the security administrators can make a dynamic domain with a number of devices less than the number of devices allowed to use the software licence, which is bound to the domain. We propose this feature, to add more flexibility and to cover organisations requirements. Having the number of devices less than the number of licences allows the organisation administrators to add devices in future time i.e. if the organisation changes its layout by moving devises between different layers, or if the organisation expanded. For example, system administrators could buy a 50 user licence for a software product and install it on 30 devices (still has the right to use the remaining 20 user devices at a later time), If the corporation expanded after 6 months and decided to add 10 more devices to use that software, the organisation can use the 10 licences as it has the right to use them, also, it is still has the rights to use the remaining 10 licences. One reason for doing this is cost, as the more number of devices that can use a software lincece the cheaper the licence would be per device. In addition, going into the process of buying another licence would, typically, require repeating the process of buying licences and many other procurement procedures, which we manage to eliminate in our solution.
- Removing a device from a domain does not mean loosing the licence associated with the device, as dynamic domains provide the flexibility to shrink domains and re-allocate licences to new devices that can replace the leaving one (now or in the future).

► Using a software-specific dynamic domain provides better protection for software products. For example, hacking the key of a dynamic domain affects only the protection of a single software product, i.e. it does not cause a global impact on other software products protection.
► Ease of Recover. Using a software-specific dynamic domain provides ease of recovery for a hacked software product. For example, hacking the key of a dynamic domain requires the recovery of only one dynamic domain.

8 Conclusion

In this paper we propose a solution to protect software products and manage licence files used by organisations. The proposed solution uses dynamic domains, consisting of devices owned by an organisation, which can be dynamically reallocated between dynamic domains, following the organisation needs. The proposed solution ensures that software products are protected from being illegally used.

References

1. spoofing hostids, 2005. http://blogs.sun.com/relling/entry/spoofing_hostids.
2. Spoofing time and space with DTrace, 2005. http://blogs.sun.com/relling/entry/spoofing_time_and_space_with.
3. Imad Abbadi. Digital asset protection in personal private networks. In *8th International Symposium on Systems and Information Security (SSI 2006), Sao Jose dos Campos, Sao Paulo, Brazil*, November 2006.
4. Imad Abbadi. Digital rights management using a master control device. In I. Cervesato, editor, *ASIAN '07: Proceedings of the 12th Annual Asian Computing Science Conference Focusing on Computer and Network Security*, volume 4846 of *Lecture Notes in Computer Science*, pages 126–141. Springer-Verlag, Berlin, December 2007.
5. BSA and IDC Global Software. 2005 piracy study, 2005. http://www.bsa.org.
6. Natali Helberger, Nicole Dufft, Stef van Gompel, Kristof Kerenyi, Bettina Krings, Rik Lambers, Carsten Orwat, and Ulrich Riehm. Digital rights management and consumer acceptability. Technical report, DG Information Society, December 2004. http://www.indicare.org/soareport.
7. International Organization for Standardization. *ISO/IEC 9798-3, Information technology — Security techniques — Entity authentication — Part 3: Mechanisms using digital signature techniques*, 2nd edition, 1998.
8. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol — OCSP. RFC 2560, Internet Engineering Task Force, June 1999.
9. Oracle, 2007. http://www.oracle.com.
10. A. Sadeghi. Trusted computing — special aspects and challenges. In V. Geffert et al., editor, *SOFSEM*, volume 4910 of *Lecture Notes in Computer Science*, pages 98–117. Springer-Verlag, Berlin, 2008.
11. A. Sadeghi, M. Wolf1, C. Stble, N. Asokan, and J. Ekberg. Enabling fairer digital rights management with trusted computing. In J. Garay et al., editor, *Information Security, 10th International Conference*, volume 4779 of *Lecture Notes in Computer Science*, pages 53–70. Springer-Verlag, Berlin, 2007.
12. Sun Microsystems Inc. Licensing center, 2007. http://www.sun.com/software/licensingcenter/.
13. Trusted Computing Group. *TPM Main, Part 1, Design Principles. Specification version 1.2 Revision 94*, 2006.
14. Trusted Computing Group. *TPM Main, Part 2, TPM Structures. Specification version 1.2 Revision 94*, 2006.
15. Weblogic, 2007. http://www.bea.com.
16. Wikipedia. Dongle, 2007. http://en.wikipedia.org/wiki/Dongle.