International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Software License Authentication: A Smart Card Approach

Tadiboina Tarun[1], Shaik Nyamathulla[2]

[1, 2]Computer Science Engineering, Guntur Engineering College, Guntur, India

Abstract: ***Software Products Licensing is a key factor in distribution of software products in IT markets, Digital Rights and Copy protection laws are days old and need an updating. software piracy is an illegal usage of software license management, this has affected very badly to software producers on an high time and great potential is lost with it, many researchers have provided their solution on unique front, we in these paper used an smart card based solution for license key management for any software product which can be authenticated on distributed networks, which can reduce to a great exert of software piracy.***

Keywords: smartcard, Licensing, software piracy, License Key management, distributed networks

1. Introduction

With the invention of many new software technologies there has been an increase in software piracy too. The problem of software piracy is of great concern to the software publisher and software distributors. A new software prevention scheme always comes with its counterpart. Cracking a serial key of the software can be classified as a Turing computable problem of automata and all these types of problems are solvable and computable which helps in spreading piracy. This paper presents a scheme for legal distribution of software with dual security. The dual security consists of optimal utilization of software and seclusion of software restrictions from the software itself. The paper also presents a concept to stop piracy by applying efforts in two directions. In one direction hardware tokens (smart cards) are integrated along with the software token (software serial keys). In the other direction optimal utilization techniques of the software in the customer organization is achieved. The optimal utilization [1, 7] of the software is an ethical way to prevent software piracy. The software piracy can be prevented in an organization by using the software in a best and most optimized way. Therefore, this paper presents a scheme involving management of license keys among the user machines needing the license key for executing the software dynamically. The isolation of software and its restrictions or instructions for execution also results in minimizing the software piracy. If the software execution details for example the serial key needed for execution of the software is embedded in the software then there is a high probability for a pirate to crack the serial key and then use this patch for making copies of the software, thus illegally spreading the software. So this paper forwards a way in which the software is devoid of the mechanism or procedure to run the software. The license keys needed for execution of the software may be present at different location and it can be isolated hardware like smart cards [4] or other plugging.

2. The Role of Coordinator in Distributed System

In this model [3] an organization tries to keep the information about the specified software on a single machine (considered as coordinator) and the complete management of the dynamic distribution of the software license is to be done on the same machine. The selection of the coordinator is done arbitrary or by executing the election algorithms. If in any case the coordinator goes down than any other machine is voluntary elected as the coordinator to provide uninterrupted functioning. In this methodology the organization cannot use the software on the number of computers, exceeding the number of license purchased but this methodology provides an ethical way for optimal uses of the software in the network of an organization. The coordinator has a counter which keeps a check on number of license keys that are reserved at a particular instant of time to various users. Besides this the coordinator machine maintains a list of active clients which are executing the software and a separate list for the waiting clients.

Subsequently when any other machine requires software to execute, it broadcast same message request packet dedicated for that specific port in the network and waits for its response. The coordinator listens the request messages of these clients and sends them back a response message indicating the presence of coordinator. In this process the coordinator gets the IP address of the client machines and the client machines gets the IP address of the coordinator machine and now here after they can communicate to each other by unicasting the message packets. In this stage the client will get an inactive copy of the software which needs a runtime license key to come in the active and working state. At this time the client sends the message packet to the coordinator demanding a dynamic license key. The coordinator keeps the status of the copy of the currently active software on the various client machines. Here two cases arise:-

Case 1: If the number of machines executing the software is less than the number of license purchased –in this case a message packet containing the encrypted license key is transferred back to the client. On receiving the license key electronically the client's software application turns into active or working state. This information is updated in the list of currently active clients available with the coordinator. It also updates the counter and decreases it by one.

Case 2: If the number of machines executing the software is greater than the number of license purchased—in this case after the client has demanded the license key from the coordinator, a message is sent to the client by the coordinator which ask the client whether to wait or quit. If the client waits for the key then it is put into the waiting list. The concept of waiting list works on the principal of FIFO.

Number of request messages can be reduced to achieve better efficiency by introducing the concept of hierarchical coordinators in the different sub networks of the customer organizations.

3. Proposed Methodology

3.1. Overview

Suppose there are two entities – software publisher (software developer) and customer. Software developer produces Software and Smart Card. Smart card is used for storing license keys [2] of the software which is made available to the customer. Smart card is provided along with the software to the customer such that there exists a unique relationship between the software and smart card provided. Each smart card can be read using the card reader at the time of software installation [5]. As the software is installed on a machine it needs a key for its execution which is obtained from the smart card.

3.2. Methodology

Software developer produces the software which is same for every customer. Each customer is provided the software and a smart card. Smart card contains a list having two entries – id number and the license keys: public license key (KL) and private license key (KL -1). When the software is installed on a machine then at the time of installation we need to transfer license keys from smart to the Random Access Memory (RAM) of the machine and then only execution of software can occur on the machine. But now a very substantial question arises on the security of transmission of license keys from smart card to RAM. The procedure of secure transmission of license keys follows as – When the card producer produces the card then at the time of manufacturing it embeds license keys, Card OS which handles the storage of license keys and a random number - 'x' on it. At the time of installation of software there occurs an execution of an application which generates a random number – 'y'. Now the most commonly used algorithm for key exchange occurs – Diffie Hellman key Exchange between smart card and RAM (software).

3.3. Establishing unique relationship between smart card and software [8]

For this purpose Diffie Hellman Key Exchange algorithm is proposed. Each card bears unique x, p, g. At the time of installation of software at the computer, it generates y and store it in a read only file at the computer. System gets the value of p, g and x from smart card. Calculate K1 = gx mod p and K = k1 y mod p on machine and store the value of k in smart card. At the time of execution: x, k, p, g are retrieved from smart card. Calculate K2 = gy mod p and K' = k2 x mod p on the machine. If K = K' then the list of license keys containing a pair (public and private key) is transferred from smart card to the coordinator machine else no transfer occurs.

4. Scenario at the Coordinator Machine

Now the coordinator machine say 'B' contains a list of public and private keys needed by the clients for the execution of their software. A client say 'A' wishes to establish a logical connection with B and requires a onetime session key to protect the data (license keys) transmitted over connection.

There are two methods for such establishment of logical connection for secure transmission of license keys:

a) Using a third party as a Key Distribution Centre (KDC).
b) Using a decentralized key control mechanism.

The above two methods generates a session key (KS) for secure transmission between client and coordinator.

A. Using Key Distribution Centre

Client (A) and coordinator (B) both share their private keys KA and KB with KDC. The following steps show the functioning of KDC. **1)** Request is sent to KDC along with a nonce the request message has identity of A and B. **2)** KDC sends a message to 'A' encrypted with private key Ka (so that only 'A' can read it). KS Session key EKb (KS, IDA) is sent to B as it is to establish connection and prove A's identity. **3)** Client 'A' stores session key KS and sends EKb (KS||IDA) to coordinator (B). Coordinator now knows session keys KS, knows other party-the client (A) and knows that information is generated by KDC (because Kb is known only to itself & KDC). **4)** B sends nonce and public license key KL to A using encryption by KS. **5)** A responds with K1 and f(N2) encrypted using KS. A has also stored KL with itself. **6)** Coordinator (B) sends private license key KL -1 to client 'A' encrypting it using KS.

B. Using decentralized key control mechanism

In this case we do not use any third party (KDC) because if we take KDC then it should be trusted. Here we use the concept of Master keys-MKm therefore, if there are 'n' machines in a system then we need n(n-1)/2 master keys. Steps for session key generation and distribution of license keys along with the authentications— **1)** Client issues request to B for session key along with N1. **2)** N1 Coordinator responds message encrypted using master key- MKm. **3)** Client returns nonce using new session keys- KS. **4)** Coordinator returns KL and new nonce N2 using KS. **5)** Client returns KS and f(N2). **6)** Then at coordinator side the matching occurs for the second key in the pair of public & private key for KL to find KL -1 which is sent using KS. Every coordinator has to maintain 'n' master keys corresponding to n clients at the time when a particular machine is declared as coordinator

1, 2, 3 Session key generation & distribution, 3, 4, 5 Authentication steps, 6 license key distribution

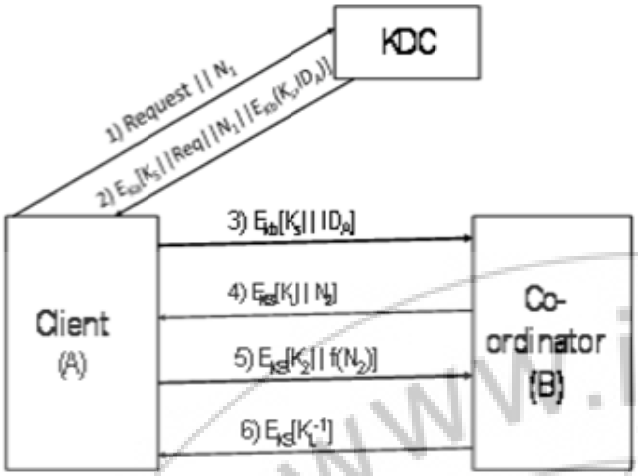

Figure 1: Working of KDC

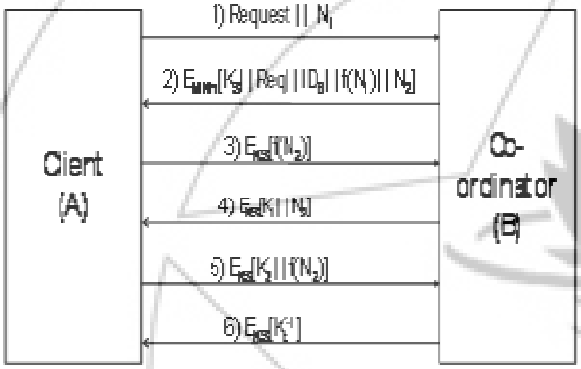

Figure 2: Decentralized keys distribution

5. Future Scope

Aura and Gollmann recently proposed an elegant software license management scheme with tamper-resistant smart cards [2]. In their scheme, software licenses are bound to smart cards, and software cannot be run without the appropriate card being in the card reader. They designed protocols for transferring licenses between cards, so that customers only need one card to use all of their software. Their scheme is fine in a home environment, but it is not scalable and flexible enough for large company based license management. One drawback of their scheme is that once a smart card transfers its licenses to another card, the original card is destroyed (''zero'ed out'') and licenses cannot be transferred back. *The user can have several software licenses all of which can be bounded to one card [2], to avoid juggling several cards in and out of the card reader. This technique can be coupled with the concept of single coordinator or hierarchical coordinators [3] in the network of the customer organizations so that optimal utilization of software can also be achieved there and it would certainly increase the degree of piracy prevention too [6].*

6. Conclusion

In these paper smart cards can be utilized as a hardware token which may be integrated to software token (serial keys) to achieve higher degree of security and so by this way it would be an initiative to stop software piracy. We proposed an enhanced and ethical solution to software license management based on tamper-resistant smart cards. We presented public-key protocols for binding software licenses to smart cards and transferring licenses between cards.

References

[1] Leili Noorian & Mark Perry, Autonomic Software License Management System: an implementation of licensing patterns. 2009 Fifth International Conference on Autonomic and Autonomous Systems. IEEE 978-0-7695-3584-5/09
[2] Mikhail J. Atallah, Jiangtao Li, Enhanced Smart-card based License Management. IEEE International Conference on E-Commerce (CEC'03)0-7695-1969-5/03 IEEE (2003)
[3] Vineet Sharma, Dr. S.A.M. Rizvi, S. Zeeshan Hussain, Distributed software and license key management "an initiative to stop software piracy", UBICC, Ubiquitous Computing & Communication Journal, South Korea. Vol. 4, ISSN 1992-8424 (2009)
[4] Rankl, W, W. Effing Smart Card Handbook. John Wiley & Sons. ISBN 0-471-96720-3. Guthery, Scott B.(1997)
[5] Timothy M.Jurgensen, SmartCard Developer's Kit. Macmillan Technical Publishing. ISBN 1-57870-027-2
[6] Yawei Zhang, Lei Jin, Xiaojun Ye Dongqing Chen, Software Piracy Prevention: Splitting on Client. International Conference on Security Technology IEEE 978-0-7695-3486-2/08 (2008)
[7] Zhengxiong Hou, Xingshe Zhou, Yunlan Wang Software License Management Optimization in the Campus Computational Grid Environment. Third International Conference on Semantics, Knowledge and Grid 0-7695-3007-9/07 © IEEE (2007)
[8] Diffie–Hellman key exchange algorithm http://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Author Profile

Tadiboina Tarun Obtained the B.Tech degree in Computer Science and Engineering (CSE) from **Chalapathi Institute of Technology**, Mothadaka, Guntur District. At present i am persuing the M.Tech in Computer Science and Engineering (CSE) Department at **Guntur Engineering College**, Yanamadala, Guntur District.

Shaik Nyamathulla obtained the B.Tech. degree in Computer Science and Information Technology(CSIT) from **Hi-Point college of Engg and Tech,** Hyderabad in 2009 and M.Tech from Guntur Engineering College in 2013. He has 5 years of teaching experience and working in Computer Science and Engineering (CSE) Department at **Guntur Engineering College**, Yanamadala, Guntur District.