E. Published Articles (2)
TOWARD A NEW ONTOLOGY OF NODELOCK LICENSING MECHANISM – PATTERNS AND APPLICABILITY

CUDDAPAH ANITHA¹, Prof. M.PADMAVATHAMMA²

¹(Department of MCA, C.R.Engineering College, Renigunta road, Tirupati-517506, Andhra Pradesh, INDIA)
²(Department of Computer Science, S.V.University, Tirupati-517501, Andhra Pradesh, INDIA)

ABSTRACT

As there lies the fact that software can be copied and since software development is costly with reference to resources, personnel, effort, intellectual property, etc., software development companies do not allow people to copy their software setups without license. This has become a major challenge in the software industry and for the developers of the software products in protecting their intellectual property rights. This paper proves the need and necessity of NodeLock Licensing Methodology during software installation by introducing various scenarios and patterns and their applicability to the patterns derived.

INDEX TERMS— Licensing Types, Licensing Methodologies, Licensing Policies, Network Connectivity, NodeLock License Management, Scenarios and Patterns.

1. INTRODUCTION

1.1. NEED FOR LICENSING?

Licensing plays an important role in installation of the software product. License agreement is required to any software product to complete its installation and execute the product. It is just like a way that a business should be run after availing grant or license or permit or registration. Software license is a legal contract between the developer of the product and the user of the product. This contract is defined for a particular period of interval which needs to be renewed after the expiration of the agreement. Like software licensing, business licensing also abide by certain functions in compliance with the government. Just like the way the business license is defined, software license is also defined based on several factors like availability, complexity, usage and so forth. Many types of licenses have been defined and the subjective cost varies with respect to developers' choice.
1.2 SOFTWARE LICENSES TYPES
These are some of the widely used software license types:

- Individual: This is a license type for a single installation.
- OEM (Original Equipment Manufacturer): A license type for software that is already installed in the hardware.
- Named User License: License type for a specific user.
- Volume License: License type supporting multiple users.
- Client Access License (CAL): License type that gives users the rights to access the services of the server.
- Enterprise Limited Use License: License type for trial versions of software.
- Enterprise (Proprietary): License type that does not require renewal and is for life long.
- Concurrent License: License type for software that requires activation by a specific number of users at a time.
- Free License: License type for free software.
- Enterprise Subscription: License type that requires renewal for every specific period.
- Node Locked: License type for workstations with specific configurations.

1.3 LICENSING METHODOLOGY
The meaning of buying and selling software is that buying, using, or selling limited or only the required features of the software is the customer. The need for providing licenses is related to the number of users the software is used by. The concept of software licenses is a new measure that has emerged in recent years. For the customer to ensure the software is used by the number of users required, some have developed software that counts the number of users using it and provides licenses based on the count. CorelDRAW is an example of such software.

1.4 LICENSING POLICIES
A Valuing a software license because easy and convenient measures with the License Provider as there is no need to use a key along with the software purchased. A license contains a license key that can be forwarded to the License Provider, which can be used by the developer of the software to purchase the software product. The key can then be transferred to different users or systems.

Different kinds of licensing are available:

a) Node-Locked Licensing
b) Concurrent Licensing
c) Per User Licensing
d) Site Licensing

e) Network Licensing
This paper focuses on Node Lock Licensing which is a most popular licensing methodology that is analogous to the software that is licensed to a particular machine. Users can be able to login to a distant machine. NodeLocked Licensing methodology fits to the machines used on workstations for a specific application. The licensing we had seen till now is for software products. But the ones said above User based license is for providing license to particular user identification. The product is installed on any number of machines and all the authorized personnel can gain access to it. Another form of licensing is the site licensing that allows user to run the software on a network from a particular site. Another two widely used licensing types are floating and network licensing. Floating licenses are mostly used in networking. These licenses allow much number of simultaneous users to run the software. These are less expensive than seat licenses as only one license purchase can serve many user needs.

2. NODELOCK LICENSE USE MANAGEMENT

A NodeLocked license is a methodology through which the software can be installed on the machine intended and the license remain valid as long as the agreement is valid. NodeLock licenses can be used for standalone, rather than client/server, applications. There are two models in license enabling. One is run time and the other is non run time enabling models. Software Distributor can chose amongst them. Distributors can enable their products using the following kinds of NodeLocked licenses:

- Simple NodeLocked licenses: A license that accepts any number of uses of the licensed product on the machine is called a Simple NodeLocked License.
- Concurrent NodeLocked licenses: Concurrent NodeLock licenses are very much similar to Simple NodeLock excepting that it limits the number of uses of the licensed product.
- Use-Once NodeLocked licenses: As with Use-Once NodeLocked licenses, license is obtained on a particular machine for a software product for a fixed period of time as an agreement during the product installation.
- Per-Server licenses: Per Server/Per-Server licenses support multiple client/server applications. The server can request to its maximum the number of licenses on behalf of its clients.
- Network Licenses: Network licenses are not fixed to a particular node; instead it is stored on network license server and will be shared among multiple clients. Various kinds of network licenses are:
  - Concurrent-offline licenses: The concurrent-offline license allows users to reserve a concurrent license for a certain number of days and to use it on a portable computer disconnected from the network.
  - Reservable licenses: A Reservable license is a network license that you can reserve for the exclusive use of a user, a group, or a node.
  - Use-once licenses: A use-once license is a network license that permits a single use of a particular licensed product within the period for which the license is valid.
  - Per-seat licenses: Vendors use per-server/per-seat licenses to enable client/server applications constructed for multiple-server solutions.

3. NODELOCK LICENSE – SCENARIOS AND PATTERNS

This section derives the variations on NodeLock License methodologies and its applicability to various patterns derived.
3.1. SOFTWARE TYPES
We classify software into following types based on hardware or device interactions.

![Software Types based on Hardware Interactions](image)

**Device Drivers:** A device driver is a computer program that operates or controls a particular type of device that is attached to a computer. A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware connects. Examples are printers, scanners.

**Operating System:** Computer operating system (OS) is a piece of computer software which has collection of various programs to perform computing device operation. The major task for OS is to manage the resources of the computer system. Operating systems are the core of any computer and contain the source code and services used by hardware and software. Examples include Windows XP Professional, Android, IOS etc.

**Software Applications:** Computer software, or just software, is any set of machine-readable instructions that directs a computer's processor to perform specific operations. Examples are Microsoft Word, AOL Instant Messenger, Internet Explorer, Mozilla Firefox, Adobe Photoshop, Windows XP etc.

**Firmware:** Firmware is software that is semi-permanently placed in hardware. It does not disappear when hardware is powered off, and is often changed by special installation processes or with administration tools. The memory firmware uses is very fast — making it ideal for controlling hardware where performance is important. In most of the cases Firmware establishes the communication between the core hardware of the device and the target operating system. Firmware also takes care of monitoring the health of the hardware like the temperature on the main board. Examples of Firmware include Samsung mobile firmware, ecom handheld firmware, etc.

**Hardware:** Hardware, in the computer world, refers to the physical components that make up a computer system. There are many different kinds of hardware that can be installed inside, and connected to the outside, of a computer. Examples include Graphics Cards and CD/DVD Drives etc. that go inside the computer.
Applicability of Node Lock License for various Software types:

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Firmware</th>
<th>Operating System</th>
<th>Device Drivers</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NodeLock License component of the firmware continuously monitors the health of the firmware and gets the license renewed based on</td>
<td>NodeLock License component of operating system monitors the health of the system and gets the license renewed based on</td>
<td>Device Drivers basically play a role on firmware for specific plugins and play peripherals of the device and provides smooth interaction with operating system. Considering the fact of having very innovative and expensive device peripherals, blocking the license for device drivers is also required. This can be achieved through</td>
<td>Out of all the classifications of software general software is most appropriate place where NodeLock License can be used. This can be achieved on general software based on</td>
</tr>
<tr>
<td></td>
<td>a) Health level of hardware</td>
<td>a) Hardware health</td>
<td>a) Partial or full list of hardware attributes</td>
<td>a) Partial or full list of</td>
</tr>
<tr>
<td></td>
<td>b) Fixed duration</td>
<td>b) Peripheral health</td>
<td>b) Partial or full health levels of system or environment</td>
<td>b) Partial or full health levels</td>
</tr>
<tr>
<td></td>
<td>c) Fixed usage</td>
<td>c) Fixed duration</td>
<td>c) Fixed duration</td>
<td>of system or environment</td>
</tr>
<tr>
<td></td>
<td>Example: Samsung Galaxy Note Firmware</td>
<td>Example: Windows, LINUX, Android, IOS, SYMBIAN, etc.</td>
<td>Example: Dell Network Drivers, Creative Sound drivers, VGA Graphic drivers, etc.</td>
<td>d) Fixed usage</td>
</tr>
<tr>
<td></td>
<td>B Com Firmware for industrial hand-helds</td>
<td>NodeLock License component of operating system checks for continuous patches, service packs, upgrades, and patches from various support groups and notifies the user about the license validity accordingly.</td>
<td></td>
<td>Example: MS Office, Phone Book Manager</td>
</tr>
<tr>
<td></td>
<td>The level of security (using RSA/ RMPRSA) can be customized based on firmware needs.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Example: Consider k number of hardware attributes out of total number n based on support/maintenance related needs, where m ≤ k ≤ n, where m is less secured/basic need and n is highly secured/complete need.</td>
<td>2. The level of security (using RSA/ RMPRSA) can be customized based on firmware needs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: Consider k number of system attributes out of total number n based on support/maintenance related needs, where m ≤ k ≤ n, where m is less secured/basic need and n is highly secured/complete need.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: Consider k number of peripheral attributes out of total number n based on support/maintenance related needs, where m ≤ k ≤ n, where m is less secured/basic need and n is highly secured/complete need.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Applicability of NodeLock License for software Types

3.2. NODE TYPES

NodeLock License component is available for various node types such as computer, devices and mobiles. In this paper, we discuss the variations, customizations, extensions possible on NodeLock Licensing Methodology. Since NodeLock License Methodology tightly integrates the hardware properties or device attributes like processor serial number, motherboard id, MAC address, etc here is our classification on node types based on hardware devices:
Figure 2: Node Types based on Hardware devices

Applicability of Node Lock License for various NODE types:

<table>
<thead>
<tr>
<th>NodeLock Licensing Methodology</th>
<th>Mobile</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>further goes to address specific needs of different computer variations below:</td>
<td>We found the most innovative and valuable software are getting developed for mobile devices. Hence deriving NodeLock Licensing Methodologies or patterns is also appropriate. Here is our classification:</td>
<td>We see that software are getting extended to address the specific needs apart from computer and mobile phones like</td>
</tr>
<tr>
<td>1) Server grade</td>
<td>a) Smart phone (most appropriate)</td>
<td>a) Industrial</td>
</tr>
<tr>
<td>2) PC grade</td>
<td>b) Basic phone (not appropriate)</td>
<td>b) Automation</td>
</tr>
<tr>
<td>Server grade machines will have NodeLock License interfaces for further classifications like</td>
<td>c) Moderate (unlikely appropriate).</td>
<td>c) Healthcare</td>
</tr>
<tr>
<td>a) Network-based servers</td>
<td></td>
<td>d) Home automation, etc.</td>
</tr>
<tr>
<td>b) Job servers</td>
<td></td>
<td>NodeLock Licensing Methodology can further be customized based on the software cost, support mechanisms, SLA (Service Level Agreement), etc.</td>
</tr>
<tr>
<td>c) Log servers</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.3. USAGE

In this section, we want to bring other variations of NodeLock Licensing Methodologies based on usage of the software. For each variant of the usage pattern of the software, we present the respective methodology of NodeLock License.

<table>
<thead>
<tr>
<th>S no</th>
<th>Very frequent</th>
<th>Frequent</th>
<th>Nominal</th>
<th>Rare</th>
<th>Very rare</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Very less validation on NodeLock License hardware attributes</td>
<td>Less validation on NodeLock License hardware attributes</td>
<td>Nominal validation on NodeLock License hardware attributes</td>
<td>More validation on NodeLock License hardware attributes</td>
<td>Complete validation on NodeLock License hardware attributes</td>
</tr>
</tbody>
</table>

3.4. NETWORK CONNECTIVITY

NodeLock License patterns can also be customized based on network availability on the following classifications:

1. Public network or Internet
   a) Full availability
   b) Adhoc availability

2. Private networks or VPN
   a) Full availability (sends the hardware details to private server as and when needed)
   b) Adhoc availability (collect the details and send them only when network is available)

3. Remote Networks. It is difficult and impossible to send hardware details for further verification at License provider location. Hence we should go with off line NodeLock License verification or intermediate transmitters of the data like file channel or any other similar communication scenario.

4. Bandwidth
   a) High (sends all attribute details)
   b) Medium (send priority hardware details)
   c) Low (send only important attributes)

NodeLock License component of the software collects various hardware attributes and sends these details to the License Provider for further verification based on network availability.
3.5. COST

Here we define the patterns for NodeLock License based on the software cost as following:

1) Expensive
2) Nominal
3) Low

<table>
<thead>
<tr>
<th>S no</th>
<th>Expensive</th>
<th>Nominal</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Strict license validation</td>
<td>Nominal license validation</td>
<td>Low license validation</td>
</tr>
<tr>
<td></td>
<td>required</td>
<td>required</td>
<td>required</td>
</tr>
<tr>
<td>2</td>
<td>Should follow most</td>
<td>Should follow important</td>
<td>Should follow reasonable</td>
</tr>
<tr>
<td></td>
<td>appropriate hardware</td>
<td>hardware attributes for</td>
<td>hardware attributes for</td>
</tr>
<tr>
<td></td>
<td>attributes for NodeLock</td>
<td>NodeLock License validation</td>
<td>NodeLock License validation</td>
</tr>
<tr>
<td></td>
<td>License validation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Better to consider as many</td>
<td>Better to consider good set</td>
<td>Better to consider</td>
</tr>
<tr>
<td></td>
<td>as possible attributes of the</td>
<td>attributes of the hardware</td>
<td>minimum set of attributes</td>
</tr>
<tr>
<td></td>
<td>hardware</td>
<td></td>
<td>of the hardware</td>
</tr>
<tr>
<td>4</td>
<td>We suggest 0 to 5% of</td>
<td>We suggest 0 to 20% of</td>
<td>We suggest 0 to &lt; 50% of</td>
</tr>
<tr>
<td></td>
<td>tolerance to go NodeLock</td>
<td>tolerance to go NodeLock</td>
<td>tolerance to go NodeLock</td>
</tr>
<tr>
<td></td>
<td>License verification with</td>
<td>License verification with</td>
<td>License verification with</td>
</tr>
<tr>
<td></td>
<td>reference to full set of</td>
<td>reference to full set of</td>
<td>reference to full set of</td>
</tr>
<tr>
<td></td>
<td>hardware attributes(n) such</td>
<td>hardware attributes(n) such</td>
<td>hardware attributes(n) such</td>
</tr>
<tr>
<td></td>
<td>that</td>
<td>that</td>
<td>that</td>
</tr>
<tr>
<td></td>
<td>( (n - \frac{n \times 5}{100}) \leq k \leq n )</td>
<td>( (n - \frac{n \times 20}{100}) \leq k \leq n )</td>
<td>( (n - \frac{n \times 49}{100}) \leq k \leq n )</td>
</tr>
<tr>
<td></td>
<td>where k is the number of</td>
<td>where k is the number of</td>
<td>where k is the number of</td>
</tr>
<tr>
<td></td>
<td>attributes considered for</td>
<td>attributes considered for</td>
<td>attributes considered for</td>
</tr>
<tr>
<td></td>
<td>NodeLock License</td>
<td>NodeLock License validation</td>
<td>NodeLock License validation</td>
</tr>
</tbody>
</table>

4. SOFTWARE FEATURES

NodeLock Licensing Methodology can be customizable for various softwares based on the features available/supported on the device.

1) Full list of features
2) Partial list or packages (time to time tolerance)
3) Configurable or customizable (groups and subgroups)

Here we are presenting the novel pattern of NodeLock Licenses to customize the feature or package (or group of features) level tolerance which can be further customized by License Provider. In this pattern we also claim that the tolerance can be changed from time to time, device to device and further possible variations.

Another claim on top of base claim would be classifying the hardware attributes into groups and subgroups (and further groups) and customizing the tolerance at the respective groups or subgroups.

Combining 2 and 3 extension claims together with base claim would form further variations based on need.
5. NEW VERSIONS AND UPGRADEITIONS

Here we are deriving the patterns of NodeLock License methodology for softwares based on software versions, service packs, and feature up gradations, patches and version upgrades as following:

1) Independent versions
2) Service packs or patches
3) Version upgrades

NodeLock License can be customized to define the tolerance levels at various versions of software. Also further classification can be done on variations of independent versions and upgradable versions. For example, the upgradable software version(n) can have the derived NodeLock License patterns from previous versions n-1, n-2, ............etc. Also this can further lead to specific customizations based on need. In general service packs and patches won't require any variations from base version of the software installed on the device. Also we recommend combinational patterns out of software types, node types, usage, network availability, cost, features, versions, service packs, patches and or many more similar classifications.

6. CONCLUSIONS

These NodeLock Licenses are always specific to a particular node or a system to which software is to be installed and operated. During the license generation process, it is part of the installation procedure that the terms and conditions of the license should be agreed upon by the user which is present in the license certificate and managed by the License Provider Server. Though NodeLock License Methodology is most appropriate for software licensing, there is no right pattern or solution available to address the specific needs of the target environment. To address this problem in implementing NodeLock License Methodologies everywhere outside connecting environments, we proposed many variations of NodeLock Licensing along with target security needs. Also, we suggested the best possible variations of the tolerances on hardware attributes, software features and other customizations.

7. ACKNOWLEDGEMENTS

It is my privilege to extend thanks for the continued support and deemed effort of Mr. J. Lokanatha Reddy for this innovation to see the dawn today.

REFERENCES

We have done a detailed prior art on various artifacts including white papers, journals, patents, etc and came to know that these patterns are completely unique and there is no such research held before. Hence we are happy to inform you that these patterns are innovative and unique outcome of our research and hence no references have been attached.
Cooperation, coordination, and communication play key roles in the effective functioning of a network. The cooperation and coordination between different nodes in the network ensure smooth data flow and efficient resource utilization. Communication is essential for exchanging information and coordinating activities. In this context, the model has been designed to facilitate cooperation, coordination, and communication among nodes.

To achieve this, various strategies have been adopted. These include:

1. **Establishing a common protocol**: A standard communication protocol is used to ensure that all nodes understand the data formats and exchange information in a consistent manner.
2. **Implementing a distributed control mechanism**: This allows nodes to take decisions based on local information, reducing the reliance on a central authority.
3. **Enforcing cooperation through incentives**: Nodes are motivated to cooperate by offering rewards or penalties. This encourages nodes to contribute to the network effectively.
4. **Utilizing feedback mechanisms**: Feedback from the network is used to adjust strategies and improve performance over time.

These strategies help in enhancing the network's efficiency and reliability. By addressing the challenges of cooperation, coordination, and communication, the network can achieve optimal performance.

**NPRA System Application**

Performance Analysis of NodeLock Licensing Methodology with

NPRA System Application
Performance Analysis Of NodeLock Licensing Methodology With RMPRSA Cryptography

NodeLock Licensed software cannot be installed nor run on more than one system and any unpermitted access to the software will automatically and strictly be monitored.

\textbf{Definition.} A Node\textit{Locked} License allows a single instance of an application to run on a specific machine. Node\textit{Locked} Licenses are directly tied to the hardware of the machine on which the licensed application license is installed. Node\textit{locked} licenses do not require a license server, because they are uncounted. It is also referred to as local licensing or seat per machine licensing.

\section*{1. Working Of Node Locked License Management System}

The diagram below illustrates the functionality and steps involved in Node\textit{Locked} License Methodology.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{node_lock_diagram.png}
\caption{Node\textit{Locked} License Management System}
\end{figure}

Working procedure of the node \textit{Lock} License can be seen from the following steps.

1. Consider a computer B requesting for a node locked licensed software. A request file is generated for its request that includes the hardware serial numbers of the computer B.
2. The request file is encrypted using the extended RSA algorithm called the Multi Prime Rebalanced RSA and sent to the License Provider’s site for the license generation to install the node lock licensed software.
3. On the receipt of the encrypted request file at the License Generator Server side, it is decrypted using its private key.
4. Now the license file is generated for the software with the hardware details of the computer B.
5. The license file that has been generated by the server is encrypted using the public key and sent to the client’s machine.
6. It is then decrypted at the client’s site using the private key of the client.
7. Now the license generated by the Server is compared and verified with the local license generated from hardware details. If the local license is similar to the license sent by the Server, then the software could be installed on computer B else an exception is raised.

The corresponding Activity diagram of Node\textit{Locked} License methodology.
II. Rebalanced Multi Prime RSA

ABOUT REBALANCED MULTI PRIME RSA

RMP RSA involves a public key and a private key same as basic RSA. The public key can be known by everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted in a reasonable amount of time using the private key.

The keys for the RMP RSA algorithm are generated the following way:

Choose two distinct set of prime numbers p and q, where p is a product of first set of prime numbers (p1 ... pm), and q is a product of second set of prime numbers (q1 ... qn).

For security purposes, the prime product p1..pm and q1..qn should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.

Here is the unique computation of RMPRSA to strengthen the security:

$$P = (4p+1)$$ and $$Q = (4q+1)$$

Now, perform the prime factor adjustments for P and Q as follows:

$$P' = P$$ and $$Q' = Q$$,

where $$P'$$ represents process of adjusting the integer P to next immediate possible prime number and in the same way Q.

Now, rest of the process remains same as basic RSA key generation.

Compute $$n = PQ$$,

$$n$$ is used as the modulus for both the public and private keys. Its length, usually expressed in bits, is the key length.

Compute $$\phi(n) = (p-1)(q-1) = (P-1)(Q-1)$$, where $$\phi$$ is Euler's totient function.

Choose an integer $$e$$ such that $$1 < e < \phi(n)$$ and gcd(e, $$\phi(n)$$) = 1, i.e. $$e$$ and $$\phi(n)$$ are co-prime.

$$e$$ is released as the public key exponent. $$e$$ having a short bit-length and small Hamming weight results in more efficient encryption – most commonly 216 + 1 = 65,537. However, much smaller values of $$e$$ (such as 3) have been shown to be less secure in some settings. Determine d as $$d^{-1} = e \pmod{\phi(n)}$$, i.e., d is the multiplicative inverse of e (modulo q(n)). This is more clearly stated as: solve for d given d * e = 1 (mod q(n)). Also, this is often computed using the extended Euclidean algorithm. d is kept as the private key exponent.

The public key consists of the modulus n and the public (or encryption) exponent e. The private key consists of the modulus n and the private (or decryption) exponent d, which must be kept secret. p, q, and n are also kept secret because they can be used to calculate d.

Alice transmits her public key (n, e) to Bob and keeps the private key secret. Bob then wishes to send a message M to Alice. He first turns M into an integer m, such that 0 < m < n by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to M.
$c \equiv m^e \pmod{n}$.

This can be done quickly using the method of exponentiation by squaring. Bob then transmits $c$ to Alice.

Alice can recover $m$ from $c$ by using her private key exponent $d$ via computing

$m \equiv c^d \pmod{n}$.

Given $m$, she can recover the original message $M$ by reversing the padding scheme.

RMP RSA algorithm is considered to be much slower as compared to DES and other symmetric cryptosystems. There is a lot of research work done to increase the speed of RSA algorithm. Here is one approach known as Multi $P$-Prime RSA algorithm which is being studied to improve the RSA decryption speed. The decryption speed of RSA can be increased thanks to the Chinese remainder theorem. Instead of a modulus such as $N = P \cdot Q$, we can use more primes. For example, $N = (4^4p_1p_2\ldots p_k+1)(4^4q_1q_2\ldots q_m+1)$. However, with a bit length of 1024, it is not secure anymore to use a decomposition of more than three primes.

RMP RSA technique was introduced by Collins who modified the RSA algorithm so that it consists of $k$ primes $p_1, p_2,\ldots, p_k$ instead of the traditional two primes $p$ and $q$. Classically, an RSA modulus has been composed from two primes. However, there are very practical reasons why using more than two primes might be preferred.

- The primes are smaller and key generation takes less time despite there being more of them.
- Private key operations take less time if one uses the Chinese Remainder Theorem. Using three primes vs. two primes gives a theoretical speedup of 9/4. A speedup of 1.8 to 2.0 has been achieved in practice.

The key generation, encryption and decryption algorithm are described as given below:

**Key Generation of RMP RSA**

The parameter $k$ indicates the number of primes to be used in key generation algorithm. The public and private key pairs can be generated as follows:

1. Select $k$ primes at random, each of which is $n/3$ bits in length.
2. Set $p = (4^4p_1p_2\ldots p_k+1)(4^4q_1q_2\ldots q_m+1)$ and $q = (4^4p_1p_2\ldots p_k+1)(4^4q_1q_2\ldots q_m+1)$.
3. Pick randomly an odd integer, $e (1 < e < q)$, such that $\text{gcd}(e, q(n)) = 1$.
4. After that find an integer, $d (1 < d < q)$, such that $d = e^{-1} \pmod{q(n)}$. This is part of our private key and must be kept secret.
5. Select an integer, $e (1 < e < q)$, such that $\text{gcd}(e, q(n)) = 1$ and $e$ and $q$ are relatively prime.
6. Find an integer, $d (1 < d < q)$, such that $e \cdot d = 1 \pmod{q(n)}$. This is part of our private key and must be kept secret.
7. The public key is $e$ and $n$, or $(e, n)$, and the private key is $d$ and $n$, or $(d, n)$.

**Encryption of RMP RSA**

Once we have generated a public/private key pair, we can encrypt a message with the public key with the following steps:

1. Take the message $M$ to represent a piece of plaintext. In order for the algebra to work properly, the value of $m$ must be less than the modulus $n$, which was originally computed as $p_1 \cdot p_2 \cdot \ldots \cdot p_k$. Long messages must therefore be broken into small enough pieces that each piece can be uniquely represented by an integer of this bit size and each piece is then individually encrypted.
2. Calculate the ciphertext $C$ using the public key containing $e$ and $n$. This is calculated using the equation $C = (M^e \pmod{n}) \mod{n}$.

**Decryption of RMP RSA**

Once we have generated a public/private key pair, we can encrypt a message with the public key with the following steps:

1. Take the ciphertext $C$ and raise it to the $d$th power modulo $n$, where $d$ is the private key.
2. The ciphertext $C$ can be obtained as $M = C^d \pmod{n}$ by applying the Chinese Remainder Theorem.
3. Calculate the plaintext $M$ using the public key containing $e$ and $n$. This is calculated using the equation $M = (C^e \pmod{n} \cdot n)$.

**RMP RSA - Key Lengths**

When we talk about the key length of an RMP RSA key, we are referring to the length of the modulus, $n$, in bits. The minimum recommended key length for a secure RMP RSA transmission is currently 1024 bits. A key length of 512 bits is now no longer considered secure, although cracking it is still not a trivial task for the likes
of you and me. The longer your information is needed to be kept secure, the longer the key you should use. Keep up to date with the latest recommendations in the security journals.

There is small one area of confusion in defining the key length. One convention is that the key length is the position of the most significant bit in $n$ that has value '1', where the least significant bit is at position $n_i$. Equivalently, key length $= \lceil \log(2n + 1) \rceil$. The other convention, sometimes used, is that the key length is the number of bytes needed to store a multiplied by eight, i.e., $\lceil \log(256(n + 1)) \rceil$. For key sizes for symmetrical block ciphers (AES and Triple DES) and the RSA algorithm. That is, the key length you would need to use to have comparable security.

**NODELOCK LICENSE METHODOLOGY - ALGORITHM AND PROTOCOL**

**1. CLIENT: STEP1**

1. Collect node specific (hardware) details and form a message by concatenating ids, serial numbers, sequence numbers, etc.

   \[
   m = \text{StrFun}(h_1, h_2, h_3, h_4, \ldots, h_n)
   \]

   Where $h_1, h_2, h_3, \ldots, h_n$ are hardware/device specific details and $m$ is a message of current context of hardware details, and StrFun() is a string manipulating function, which will accommodate hardware detailsnotations in a single message context.

2. Compute $C_i$ (encrypted message) from $m$ using multi-prime rebalanced RSA.

   Let $p_2$ be the product of $n_1$ randomly chosen distinct primes $p_{11}, p_{12}, \ldots, p_{1n}$ and $q_1$ be the product of $n_2$ randomly chosen distinct primes $q_{11}, q_{12}, \ldots, q_{1n}$.

   \[i.e., p_2 = \prod_{j=1}^{n_1}(p_{1j}) \quad \text{and} \quad q_1 = \prod_{j=1}^{n_2}(q_{1j})\]

   Let $P_1 = (4p_2 + 1)$, $Q_1 = (4q_1 + 1)$, and $N_i = P_1Q_1$.

   Compute Euler's Totient function of $N_i$

   \[
   \Phi(N_i) = (P_1 - 1)(Q_1 - 1)
   \]

   Chose an integer $e_i$, where $1 < e_i < \Phi(N_i)$, such that $\gcd(e_i, \Phi(N_i)) = 1$.

   The pair $(N_i, e_i)$ is the public key for which the Node Lock license is required.

   For this message $m \in Z_{N_i}$, the cipher text is computed as

   \[
   C_i = m^{e_i} \mod N_i
   \]

3. Compute $C_i$ (encrypted message) from $m$ using multi-prime rebalanced RSA.

   Let $p_1$ be the product of $n_1$ randomly chosen distinct primes $p_{11}, p_{12}, \ldots, p_{1n}$ and $q_2$ be the product of $n_4$ randomly chosen distinct primes $q_{11}, q_{12}, \ldots, q_{1n}$.

   \[i.e., p_1 = \prod_{j=1}^{n_1}(p_{1j}) \quad \text{and} \quad q_2 = \prod_{j=1}^{n_4}(q_{1j})\]

   Let $P_2 = (4p_1 + 1)$, $Q_2 = (4q_2 + 1)$, and $N_i = P_2Q_2$.

   Compute Euler's Totient function of $N_i$

   \[
   \Phi(N_i) = (P_2 - 1)(Q_2 - 1)
   \]

   Chose an integer $d_i$, where $1 < d_i < \Phi(N_i)$, such that $\gcd(d_i, \Phi(N_i)) = 1$.

   The pair $(N_i, d_i)$ is the public key for which the Node Lock license is required.

   For this message $m \in Z_{N_i}$, the cipher text is computed as

   \[
   C_i = m^{d_i} \mod N_i
   \]
Performance Analysis Of NodeLock Licensing Methodology With RMRSA Cryptography

4. Create a Request file/content by combining the results of step 2 and step 3.
   req = length of C1 + length of C2 + length of C2 + C3
   Here is the simple protocol to merge the contents of encrypted hardware details C1 and C3

II. SERVER: STEP 2
   A) After receiving the Request file at server (license generator) side, extract the contents of C1 and C2
   i.e., m1 = C1, m2 = C2
   B) Compute d1 = e1⁻¹ mod \( \Phi(N_1) \), the private key 1 is the pair (N1, d1)
   C) Decrypt the encrypted message \( C_1 \text{ in } Z_{N1} \); the plaintext is recovered by computing \( m1 = C_1^{d1} \text{ mod } N_1 \)
   D) Compute d2 = e2⁻¹ mod \( \Phi(N_2) \), the private key 2 is the pair (N2, d2)
   E) Use the encrypted message \( C_3 \text{ in } Z_{N2} \); the plaintext is recovered by computing \( m2 = C_3^{d2} \text{ mod } N_2 \)

8. Compare the plaintexts of m1 and m2 and verify whether the request file/content is valid.
   invalidrequestfile = (m1 equals to m1)
   Or
   If \( (m1 \neq m2) \) then invalidrequestfile = true;
   otherwise invalidrequestfile = false;
   i.e., m1 = m2 = m3
   9. Compute the license key/content/file from original message m
      License = \( f(m) \)
      Here \( f(m) \) is defined, such that no two hardware messages give the same license
      i.e., if \( (m1 \neq m2) \) then \( f(m1) \neq f(m2) \)
      This is Node Lock License concept.
      In other terms, for any two hardware details m1 and m2, \( f(m1) \neq f(m2) \).

10. Compute \( C_i \text{ (encrypted cipher text from message } f(m) \text{, using multi prime re-balanced RSA.} \)
    Let \( p_i \) be the product of \( n5 \) randomly chosen distinct primes \( p_{11}, p_{12}, \ldots, p_{15} \) and \( q_i \) be the product of \( n5 \) randomly chosen distinct primes \( q_{11}, q_{12}, \ldots, q_{15} \).
    i.e., \( p_i = \prod_{1}^{n5}(p_{11}), q_i = \prod_{1}^{n5}(q_{11}) \)
    Let \( P_i = (4p_i+1), Q_i = (4q_i+1), \) and \( N_i = P_iQ_i \)
    Compute Euler’s Totient function of \( N_i \).

    \[ \Phi(N_i) = (P_i - 1)(Q_i - 1) \]

    Choose an integer \( r_i \), where \( 1 < r_i < \Phi(N_i) \), such that \( \text{OCD}(r_i, \Phi(N_i)) = 1 \)
    The pair \( (N_i, r_i) \) is the public key 3; this also shall be distributed along with the software or program for which the Node Lock license is required.
    For this message \( m \in Z_{N_i} \), the cipher text is computed as

    \[ C_i = f(m)^{r_i} \text{ mod } N_i \]

III. CLIENT: STEP 3
11. After receiving the license key at the client (license validator) side, decode the contents of cipher text \( C_i \)
    License key = \( g(m) = C_i^{r_i} \text{ mod } N_i \) (this is license key decoding)
12. Define function \( g \) such that \( f(m) \) identically equal to \( g(m) \)
    i.e., \( f(m) \) and \( g(m) \) shall always give similar output and for \( m1 \) and \( m2 \), \( f(m1) \) identically equal to \( g(m2) \)
13. Get the hardware details or device properties (h) and see whether license is valid
    invalidlicense = \( f(m) \) identically equal to \( g(m) \)
    Or
    If \( f(m) \) identically equal to \( g(m) \) then
    invalidlicense = true
    Else invalidlicense = false
14. Register the product as successful validation of Node Lock License key.

www.ourjournals.org 40 | Page
III. Performance Analysis Of NodeLock Licensing Methodology Algorithm With RMPRSA Cryptography And RSA

Here we come across the performance measures of RSA and RMPRSA and the time required for the key generation, encryption and decryption.

a. OBSERVATIONS AND AVERAGING

The term "average" usually refers to the sum of a list of numbers divided by the size of the list, in other words the arithmetic mean. However, the word "average" can confusingly be used to refer to the median, i.e., the middle value or some other central or typical value. In statistics, these are all known as measures of central tendency. Thus the concept of an average can be extended in various ways in mathematics.

1. KEY GENERATION: OBSERVATIONS AND AVERAGING

To improve the security levels, randomization techniques have been applied in co-prime computation process, so it makes sense to take multiple observations, and take an average of each process step.

Following table shows the various observations taken for Key generation process of both Sample RSA and RMPRSA and last Column shows the average values.

<table>
<thead>
<tr>
<th>0.8279</th>
<th>0.2723</th>
<th>0.4674</th>
<th>0.243</th>
<th>0.7367</th>
</tr>
</thead>
<tbody>
<tr>
<td>690.004</td>
<td>146.6199</td>
<td>201.9249</td>
<td>221.2803</td>
<td>332.9045</td>
</tr>
<tr>
<td>0.3127</td>
<td>0.9729</td>
<td>0.8728</td>
<td>0.9041</td>
<td>0.7004</td>
</tr>
<tr>
<td>74.5693</td>
<td>1055.932</td>
<td>791.0222</td>
<td>413.6621</td>
<td>1087.96</td>
</tr>
<tr>
<td>0.6971</td>
<td>1.279</td>
<td>0.6145</td>
<td>1.699</td>
<td>0.8809</td>
</tr>
<tr>
<td>465.2311</td>
<td>1186.79</td>
<td>999.9896</td>
<td>1649.569</td>
<td>953.2715</td>
</tr>
<tr>
<td>3.4002</td>
<td>2.3708</td>
<td>2.3077</td>
<td>3.5103</td>
<td>2.3572</td>
</tr>
<tr>
<td>1102.4866</td>
<td>1167.503</td>
<td>951.7801</td>
<td>981.1397</td>
<td>472.9549</td>
</tr>
</tbody>
</table>

Table 1. Average values for Key Generation Process of RSA and RMPRSA

II. ENCRYPTION: OBSERVATIONS AND AVERAGING

Following table shows the various observations taken for Encryption process of both Sample RSA and RMPRSA and last Column shows the average values.

<table>
<thead>
<tr>
<th>7.8973</th>
<th>8.167</th>
<th>8.5627</th>
<th>7.8429</th>
<th>8.1779</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.9022</td>
<td>8.8948</td>
<td>3.4619</td>
<td>6.7851</td>
<td>8.9083</td>
</tr>
<tr>
<td>15.6311</td>
<td>13.8779</td>
<td>13.2091</td>
<td>11.2121</td>
<td>10.3094</td>
</tr>
<tr>
<td>25.57</td>
<td>31.0084</td>
<td>25.4306</td>
<td>31.0136</td>
<td>20.1129</td>
</tr>
<tr>
<td>17.9662</td>
<td>19.4078</td>
<td>15.8503</td>
<td>15.7093</td>
<td>17.0622</td>
</tr>
<tr>
<td>57.141</td>
<td>54.1771</td>
<td>48.9991</td>
<td>45.5109</td>
<td>44.0253</td>
</tr>
<tr>
<td>29.2374</td>
<td>30.4164</td>
<td>29.633</td>
<td>30.2369</td>
<td>30.3468</td>
</tr>
</tbody>
</table>

Table 2. Average values for Encryption Process of RSA and RMPRSA

III. DECRYPTION: OBSERVATIONS AND AVERAGING

Following table shows the various observations taken for Decryption process of both Sample RSA and RMPRSA and last Column shows the average values.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.9042</td>
<td>1.6409</td>
<td>1.3083</td>
<td>1.7221</td>
<td>0.76</td>
</tr>
<tr>
<td>1.7999</td>
<td>1.709</td>
<td>2.1039</td>
<td>1.8165</td>
<td>0.986</td>
</tr>
<tr>
<td>1.7039</td>
<td>2.8137</td>
<td>1.0083</td>
<td>1.0852</td>
<td>1.7052</td>
</tr>
<tr>
<td>30.0861</td>
<td>31.5234</td>
<td>28.4876</td>
<td>31.3713</td>
<td>32.3023</td>
</tr>
<tr>
<td>1.9113</td>
<td>2.9409</td>
<td>1.3422</td>
<td>1.8062</td>
<td>1.9056</td>
</tr>
</tbody>
</table>

Table 3. Average values for Decryption Process of RSA and RMPRSA
b. COMPARISON AND GRAPHS

i. COMPARISON AND GRAPHS - 128 BIT

Following table shows the comparison matrix of Simple RSA and RMP RSA for Key generation, Encryption, Decryption process steps individually under 128 Bit mode.

<table>
<thead>
<tr>
<th></th>
<th>Simple RSA</th>
<th>RMP RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Gen</td>
<td>1</td>
<td>124</td>
</tr>
<tr>
<td>Enc</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Dec</td>
<td>6</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4. Comparison matrix of Simple RSA and RMP RSA

And the inference is RMPRSA takes more time for key generation, almost same time for Encryption process and very less time for decryption process.

Figure 3. Graphical inferences drawn on performance with 128 bit

ii. COMPARISON AND GRAPHS - 256 BIT

Following table shows the comparison matrix of Simple RSA and RMP RSA for Key generation, Encryption, Decryption process steps individually under 256 Bit mode.

<table>
<thead>
<tr>
<th></th>
<th>Simple RSA</th>
<th>RMP RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key Gen</td>
<td>15</td>
<td>483</td>
</tr>
<tr>
<td>Enc</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Dec</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table 5. Comparison matrix of RSA and RMPRSA with 256 bit

And the inference is RMPRSA takes more time for key generation, slightly less time than basic RSA for Encryption process and very less time for decryption process.

Figure 4. Graphical inferences drawn on performance with 256 bit
III. COMPARISON AND GRAPHS - 512 BIT
Following table shows the comparison matrix of Simple RSA and RMP RSA for Key generation, Encryption, Decryption process steps individually under 512 Bit mode.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1017</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 6. Comparison matrix with 512 bit sequence
And the inference is RMPRSA takes more time for key generation, less time than basic RSA for Encryption process and very less time for decryption process.

![Graphical representation of 512 bit comparison](image)

Figure 5. Graphical inferences drawn on performance with 512 bit

IV. COMPARISON AND GRAPHS - 1024 BIT
Following table shows the comparison matrix of Simple RSA and RMP RSA for Key generation, Encryption, Decryption process steps individually under 1024 Bit mode.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>975</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Table 7. Comparison matrix with 1024 bit sequence
And the inference is RMPRSA takes more time for key generation, less time than basic RSA for Encryption process and very less time for decryption process.

IV. Conclusions
In this paper, we have presented NodeLock and Licensing Methodology for protecting sensitive data during NodeLock Licensed software installation. Generally sensitive data will be stored on disk in the form of hidden files. The RMPRSA algorithm that is defined for protecting the sensitive data during the NodeLock Licensed software installation is based on considering hardware attributes of the client’s machine. Thus sensitive data is encrypted and is stored in the License Providers’ site and hence no copy of it is available on the disk in user’s site.

Acknowledgements
My sincere thanks to Mr. J. Lokanatha Reddy, one of the contributors of the innovation, towards his continued support for the entire work to see its light today.

References
Performance Analysis Of NodeLock Licensing Methodology With RMPRSA Cryptography


