Nodelock License Methodology – An Architecture Preventing Software Piracy

Anitha Cuddapah, and Padmavathamma Mokkala

***Abstract*--** As the development costs to build software has been increased and can be copied or moved elsewhere, in regard to resources, personnel, effort, intellectual property, etc., there is a need to prevent and control the piracy of the software products being used [1,2]. Addressing this issue, an architecture on NodeLock licensing methodology has been introduced that could be a support for software developers in controlling their intellectual property rights. Also, the transaction complexity of NodeLock License Methodology with hypothetical Rebalanced Multi Prime RSA algorithm (RMPRSA), an extension of RSA is proved**.**

***Keywords*--** NodeLock, License Provider, License distribution.

I. Introduction

THE word software architecture [3] represents the high level structure of a software system. It can be defined as the set of structures needed to reason about the software system, which comprises the software elements, the relations between them, and the properties of both elements and relations. It also denotes the set of practices used to select, define or design software architecture. Finally, the term often denotes the documentation of a system's "software architecture". Documenting software architecture facilitates the communication between stakeholders, captures early decisions about the high-level design, and allows reuse of design components between projects. This work focuses on the need to introduce an architecture that controls software piracy which ultimately results in self satisfaction of the software developers, apart from their effort in its construction. In addition to this, the transaction complexity of the architecture is also calculated.

A. Use Case Diagram

As we know that use case diagram represents a graphical depiction of the interactions among the elements of a system. It is a methodology used in system analysis to identify, clarify, and organize system requirements [4, 5] . For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. Use case diagram looks like a flowchart. Symbols around represent the system elements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems. System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task.

Now that the use case diagram of NodeLock Licensing Methodology consists of four main components.

- **Use cases**, of which the specific roles are played by the actors within and around the system.
- The **boundary**, which defines the system of interest in relation to the world around it.
- The **actors**, usually individuals involved with the system defined according to their roles.
- The **relationships** between and among the actors and the use cases.

Here is the use case for our NodeLock licensing process.

Actors:

- License Administrator
- Software Provider/Manager
- License Provider
- Software Installation Engineer
- Software End User

Use case Steps:

- Generate/Manage Security Keys
- Distribute License Keys
- Generate License Certificates
- Software Distribution
- Software Installation/Apply Licenses
- Use Software/Features

License Administrator creates/manages public/private security keys by defining various security parameters that include the bit length, prime number combinations of algorithm (Rebalanced Multi Prime RSA) using the services of Security Key Manager and sends them to the License Provider Platform/Server. The Software Installation Engineer gets licenses from the License Provider for all the request files submitted, download/purchase softwares (with NodeLock security); install softwares and generates request files for getting licenses using public keys distributed along with software. Finally the end user of the software uses licensed softwares/ features. License Provider distributes public keys for NodeLock requests and applies private keys to generate license certificates. Software Provider/Manager gets public

Anitha Cuddapah[1] is with the CR Engg College, Renigunta Road, Tirupati, AP, India, (corresponding author's phone: 919177234548; e-mail: anithacuddapah17121@yahoo.co.in).

Padmavathamma Mokkala[2], is with S V University Tirupati,A.P., India, (author's phone:+919490362036 e-mail: prof.padma@yahoo.com).

keys for NodeLock Licenses, distributes softwares along with public keys given by the License provider.

II. Architecture

For all architectural perspectives there are various views of the architecture that are often classified as conceptual, logical, and physical views. *Conceptual views* are the most abstract and tend to be described in terms that are most familiar to the (non-IT professional) users of the system. The conceptual view is used to define the functional requirements and the business users' view of the application to generate a business model. *Logical views* show the main functional components and their relationships within a system independently of the technical details of how the functionality is implemented. The application models represent the logical view of the architecture for an application. *Physical views* are the least abstract and illustrate the specific implementation components and their relationships. Each of the elements in the physical view is implemented, normally by a design and development process, as a software or hardware system. Architects create *application models*, which are logical views of the business model, as they determine how to meet business objectives and requirements. Here we present various views of architecture for NodeLock licensing process in detail.

A. Architecture - Conceptual View

Conceptual modeling techniques, such as use case analysis, activity diagrams, process design, and business entity modeling, help to build a description of the key business processes and the data they use, in a way that emphasizes business objectives and requirements, and is free of implementation technology. In General, the conceptual view is used to define the business requirements and the business users' view of the application to generate a *business model*. In this NodeLock license process – Conceptual view, we present various sub systems and their interactions with others.

- Security Key Manager
- License Key Distributor
- License Certificate Generator
- Software Distributor
- NodeLock Enabled Software Module
- Software License Installer

Security Key Manager module acknowledges to License Key Distributor with acceptance of security parameters. License Key Distributor generates multiple private and public key combinations from the given security parameters using Security Key Manager module. License Administrator defines the security parameters including the bit length, prime number combinations of RSA extension algorithm RMPRSA using Security Key Manager module. Security Key Manager module holds the responsibility of maintaining all these security parameters. This module interacts with License Key Distributor module to supply the security configuration. These can be a list of security groups configured to address various types of security requirements. License Key Distributor module issues private keys to the License Certificate Generator module and public keys to the Software Distributor. One of the key users of NodeLock License methodology is License Provider who defines the license distribution settings using License Key Distributor module. License Certificate Generator module asks for the security parameters and private key combinations from License Key Distributor. Software Distributor takes the key details of public key encryption from License Key Distributor and integrates the software installer with NodeLock Security configuration. Software Distributor holds the responsibility of distributing the NodeLocked enabled software to all requested Software + Licensing Client Platform which is then issued to the Software License Installer. Software License Installer then returns the required set of public key encrypted license file along with NodeLock request file to the License Certificate Generator. License Key Distributor module generates distributable private/public secret key combinations along with advanced key distribution methodologies that includes private key encrypted license file and NodeLock License file and distributes to Software License Installer of Software + Licensing Client Platform.

B. Architecture: Physical View

The physical view depicts the system from system engineer's point-of-view. It is concerned with the topology of software components on the physical layer, as well as the physical connections between these components [6]. This view is also known as the deployment view.

In this NodeLock license process – Physical view, various sub systems and their deployment views are presented

- Desktops and clients
  - General Software
  - Software License Installer
- License Management System
  - Security Manager
  - Key Distributor
  - License Provider
- Software Deployment Server
  - Software License Integrator
  - Software Distribution Manager
- Data Management

The deployment view of NodeLock license generation architecture is in fig4.

The Client checks to see if the license is available to install the required software. If it is not available, then a request is made to the Software License Installer for the license. The Software License Installer then generates the request file that includes the hardware details (examples include process id, hard disk registered number and mother board id) of the client's computer or device details that contains the information like manufacturer information, device id and sends it to the License Management System through email, file sharing or by content sharing. Local license is generated from the hardware details of the client system. The License Management System includes various subsystems like Security Manager, Key Distributor, and License Provider. The Security Manager subsystem provides necessary parameters for providing security to the keys that are to be generated. The Key Distributor subsystem provides the public and private key combinations with required security parameters and finally the License Provider subsystem generates the license file and forwards it to the Software Deployment Server. The Software

Deployment Server has various subsystems; two of the most commonly used subsystems are Software License Integrator and Software Distribution Manager. The Software License Integrator verifies the license generated by the License Provider with the local license generated from the hardware details. If these two licenses are valid, then the software is installed with the license else an exception arises.

C. Architecture: Implementation View

The system from a programmer's perspective is concerned with software management. This view is also known as the implementation view. It uses the UML Component diagram to describe system components. UML Diagrams used to represent the development view include the Package diagram. In this NodeLock license process – Implementation view, we present various sub systems and their development views

- License Manager
  - Security Manager
  - Key Distributor
  - License Provider
- Software Deployment Server
  - Software + License Integrator
  - Software Distribution Manager
- Desktop-Client License Environment
  - Software License Installer
  - NodeLock enabled General Software
- Data Manager

The implementation view of NodeLock License Generation architecture module includes various components like Security Manager, Key Distributor and License Provider. The Security Manager module has the responsibility of maintaining security parameters like bit length, prime numbers set etc. This module interacts with the Key Distributor module in supplying the security configuration which can be a list of security groups configured to address various types of security requirements. The Key Distributor implementation component is responsible for collecting security parameters from Security Manager to generate public and private key combinations. The License Provider component accepts the request from Software License Installer component in the form of NodeLock request file or packet. Now to generate the NodeLock License for the end customer, it collects the private key security details from the Key Distributor implementation module. The Software Deployment Server module includes the components like Software License Integrator and Software Distribution Manager. The Software License Integrator implementation component collects the key details of public key encryption from Key Distributor Implementation component of License Manager and integrates the Software Installer with NodeLock Security configuration. The Software Distribution Manager component holds the responsibility of taking the NodeLock enabled software from Software License Integrator component and distributes the software to all requested client-side vendors. The Desktops - Clients Environment module includes the components like Software License Installer and General Software (NodeLock License enabled) components. The Data Manager module is for archiving the security public/private keys and also the license details generated by License Provider subsystems. It has seamless integration with License Manager to supply the data generated in the past. The Software License Installer component of the Desktops - Clients Environment module generates the license request file or packet from the currently installed client platform's hardware configuration including MAC– id, base board serial number and other key hardware attributes and sends the generated request file to the License Provider implementation component of the License Manager for further validation of NodeLock License or Certificate Generation.

III. Transaction Complexity Of Architecture/ Implementation

In this section, the complexity and performance of various components with respect to NodeLock License architecture and implementation using RMPRSA algorithm, an extension of RSA algorithm [7] is calculated. Later we present the overall analysis of all these components. At last, the performance of RMPRSA with respect to simple RSA and their comparison is given. Lastly, various graphical scenarios related to normal condition, once an exception has arisen due to machine or node or hardware related issues, the scenario when an exception occurred due to license corruption or some related issues, an exception due to license shipment, say to a new node, have been discussed.

A. License Provider

License Provider accepts requests for license generation from the License Installer in the form of NodeLock request file that includes the hardware details of the client machine onto which the NodeLocked software is to be installed. On the receipt of NodeLock request file, the License Provider under various security levels issued by the Security Manager generates the license keys encrypted using RMPRSA protocol as license file and forwards it to the Software Distribution Manager. From the above scenario, we could compute the complexity and performance of the License Provider as below.

Below are architectural transactions:

1. Provide security levels
2. Generate license keys
3. Distribute license keys

The 'Provide security levels' architectural step has following implementation transactions

a) Take multiple primes for p
b) Take multiple primes for q
c) Rebalance P using (4p+1)
d) Rebalance P using (4q+1)
e) Adjust P, so that P is prime and P>=(4p+1)
f) Adjust Q, so that P is prime and Q>=(4q+1)

Generate license keys architectural step has following implementation transactions

a) n=PQ
b) $\varphi = (P-1)(Q-1)$
c) Compute public key (E,N)
d) Compute private key (D,N)

Distribute license keys architectural step has following implementation transactions

a) Distribute private key (E,N)
b) Distribute private key (D,N)

Transaction summary of License Provider process:

Architecture = O(3)

Implementation = O(12)

B. Software provider

Software Provider component holds the responsibility of collecting the license file from the License Provider and integrates the software installer with the NodeLock security configuration and distributes the software to all requested client-side vendors.

Now from the above scenario, we could compute the performance and complexity of the Software Provider as below.

Number of transactions:

1. Take/ embed/inject public keys.
2. Prepare software setup along with license keys.
3. Distribute software setup.

Transaction summary of Software Provider process:

Architecture = O(1)

Implementation = O(3)

C. Software installation

Software Installation component generates the request file from currently installed client platform's hardware configuration including MAC-id, Base board serial number, other key hardware attributes and forwards the generated request file to the License Provider for the license file.

Calculation of the complexity and its performance is as shown below.

Number of transactions:

1. Install software
2. Generate request file with node (hardware) based attributes.
3. Encrypt with public keys.
4. Send request file for license generation.

Transaction summary of Software Installation process:

Architecture = O(1)

Implementation = O(4)

D. License installation

On the receipt of request file, License Provider decrypts the request file using the private key and verifies the hardware details and generates the license file encrypted with private key and sends to the Software Distributor on the client side via email, telephone, file sharing or by some other means. The License Installer component also is involved in the above process. License Installer component collects the key details of public key encryption of License Provider and integrates the software installer with NodeLock security configuration.

The performance evaluation and the complexity of the component is as below.

Number of transactions:

1. Decrypt request file using private key.
2. Verify node (hardware) attributes.
3. Generate license file and encrypt private key protection.
4. Send the license file to the client node (hardware).
5. Decrypt the license file with public key.
6. Verify the license on node basis.
7. Install the license on successful verification.

Transaction summary of License Installation process:

Architecture = O(1)

Implementation = O(7)

E. Launch software

This component browses the net for the required license file. If available, setup file is run and the software is installed on the machine. Else, the request file is sent to the License Provider server via email or file transfer requesting for the license file. On the receipt of the license file, the software is installed on the client's machine. This scenario is used in the computation of performance and the complexity of the component.

Number of transactions:

First Time:

1. Search the license.
2. If not available, go to License Installation process.
3. Launch the software.

Second Time:

1. Search the license.
2. Launch the license.

First Time:

Transaction summary of Launch Software process:

Architecture=O(3)

Implementation=O(9)

Subsequent levels:

Transaction summary of Launch Software process:

Architecture = O(2)

Implementation = O(2)

F. License un-installation

License Un-installation component verifies if the current license exists in the client's machine. If it is, encrypt it with public key and delete all nodes. Send the encrypted license file to the License Provider for de-allocation of the product. With these events the performance and the complexity of the License Uninstallation component is determined.

Number of transactions:

1. Verify current license.
2. Encrypt current license with public key.
3. Delete all license files on the node.

Send encrypted license file (to server) for NodeLock (de_allocation)

Transaction summary of License Uninstallation process:

Architecture = O(1)

Implementation = O(4)

IV. Conclusions

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. NodeLock License Methodology is most appropriate for eradicating software piracy, To address this problem in implementing NodeLock License Methodologies everywhere outside connecting environments, we proposed an architecture of NodeLock Licensing along with targeting security benefits with hypothetical RMPRSA algorithm.

ACKNOWLEDGMENTS

It is my privilege to extend thanks to Mr. J. Lokanatha Reddy for this ideology and new innovation.

REFERENCES

[1] http://www.basis.com/sites/basis.com/advantage/magv3n2/globetrotter.html
[2] http://www.basis.com/sites/basis.com/advantage/mag-v3n2/globetrotter.html
[3] http://www.freebase.com/m/0cyls
[4] http://inditakr.blogspot.in/2013/10/use-case-diagram.html
[5] http://savedwebhistory.org/k/use-case-diagram-example
[6] http://www.enterprise-architecture.info/Images/Documents/Microsoft%20Architecture%20Overview.pdf
[7] http://simple.wikipedia.org/wiki/RSA_algorithm