6. NodeLock License Methodologies - Patterns

6.1. Why licensing?

Licensing plays an important role during the 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 need to be renewed after the expiry of the agreement. Like software licensing, business licensing also abides by certain rules, i.e., it tries to comply with in compliance with the government rules. 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.

6.2. Software licenses Types

These are some of the widely used software license types.

- Individual: License type for a single installation
- OEM (Original Equipment Manufacturers): License type for software that is already installed in the hardware.
- Named User License: License type for a specific user.
- Volume: License type supporting multiple users.
- Client Access License (CAL): License type that gives a user the rights to access the services of the server.
- Trial License: License type for trial versions of software.
- Enterprise (Perpetual): License type that does not require renewal and is for life long.
- Concurrent License: License type for software that can be accessed by a specific number of users at a time.
- Free License: License type for freeware software.
- Enterprise Subscription: License type that requires renewal for every specific period.
- NodeLocked: License type for workstations with specific configurations [15, 60].

6.3. License Management Life Cycle aspects:

- License Agreement
NodeLock Licensing Methodologies Using RMPRSA Cryptography

- Grant
- Expiration of License Term
- Copying
- Modification/Changes
- Restrictions on Transfer
- Ownership
- Subscription Services
- Responsibility for Selection and Use of Software
- Limited Warranty
- Warranty Exceptions
- Intellectual Property Infringement (IP) – Copy rights, patents, trade marks etc
- License Liabilities
- License Termination[3, 61,64]

6.4. License – methodology

The meaning of buying and selling software is that buying or selling limited or the only required features of the software to the customers. Some of the softwares include patent rights and some has copyright protection. The need of providing license to the software has developed before the concept of networks came into existence. Now we shall see some of the concepts related to licensing of software products. Cost of a software license increases gradually with respect to its performance. Software license cost is also dependant on the number of licenses requested by the customer. In 1980s when personal computers became popular, personal computer vendors came out with the concept of software licensing purely based on the performance of CPU. As the networking in computers gained its popularity in 1980s, it has foreseen the need for licensing of software in its various aspects.

Honest users stick on to the terms of license agreement for the software they purchased. According to software licensing system introduced during earlier times, License Providers limit the usage of software as requested by the customers and the License Providers would keep track of the number of systems the software is installed and run for its use. This method proved to be obsolete for networked systems that include workstations, etc in many ways. Hence, new a methodology has been introduced in licensing of software products.

Now that the usage of software is more, it can be said that software is a network resource and software licenses float on it. Software cost is a function of how many users simultaneously
run the software. Cost of the license is related to the number of users using it but not with the performance measurement of CPU as was done earlier. Presently License Providers keep track of all licenses provided to the requestors of the software. License Providers also keep track of dynamic users of the software.

Software licensing is entirely different from copy protection in the sense that the licensing is provided to the requested users for their execution of the product whereas copy protection is making duplicate copies of the same product.

Availing a software license became easy and common resource with the License Provider as there is no need for a tape along with the software purchased. All that is required is a license key for the software product to be manageable. The license key can be forwarded to the requestor via email or some file transfer mechanism or by some other means. Now-a-days license for the software products get easily managed by License Providers.

6.5. Licensing Policies

The terms of agreement of license of a particular software product are defined by the developer of the product and are embodied along with the software by the License provider which can be obtained during the license installation. It is difficult to understand the types of licensing policies used by the developers of the product.

Now we go through different kinds of licensing. NodeLock Licensing is the most popular licensing methodology that is analogous to the software that is licensed to a particular machine. Users can login to a distant machine. This NodeLocked Licensing methodology fits the machines used on workstations for a specific application. The licensing we had seen till now is for software products. User based license is for providing license to a particular user identification. Here authorized personnel are considered as the unit of measure by which the product can be licensed. 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. It has a number of advantages and disadvantages in that any number of users can simultaneously access the software. Disadvantage in the sense that site license is more expensive than a license to a specific software product. Another two widely used licensing types are floating and network licensing. Floating licenses are mostly used in networking. These licenses allow a large 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. Almost all the licensing conditions are based on developer's insight during its development. It means that there is no rightly defined policy or statement for license agreement terms. License agreement also includes start date and expiration date also. Users are allowed to choose amongst the best software available and can evaluate its performance before its purchase from the License Providers. License Providers can also make license available to a particular group [12].
6.6. General Design Goals

At first, the design goal of a modern License Provider seems to be simple enough. But it is not. It has its own inherent complexity. This is because the license management has become part and parcel of the software product. If at all the license server fails due to some reason, network goes down in its entirety and the users of the product will not be able to access the software. Therefore in such a situation, License Providers should be able to use some mechanism to restart the network work after its failure.

Most of the buyers of the software products, generally the users or the customers of the software are not aware of the use of license, in its terms. During the installation of the product, the user just signs the contract that specifies the terms of license agreement to get the software run on the system. So, the developers of the License Providers should state the need for license agreement of the product. The License Providers should allow the application to be run on other workstation. As most of the workstations today lie on large networks, the client-server management should not be a failure on the net.

6.7. Industry Standards:

Even though, FLEXlm proved its existence in license marketing with its acceptances by many of the vendors, many of the companies are moving toward new API as industrial standard for their product. Software License Working Group developed by Apollo, Highland Software and other introduced an API for license management, modeled after NetLS. FLEXlm supported this API but it has not grown as an industrial standard for licensing. SCD applications are also called as SPARCware. It ensures that all applications can be executed on any SCD system. The conforming system can use the name described in the specification. OSF, the Open Software Foundation, tried to define new standards for license management. DME, the Distributed Management Environment Highland Software is viewed to be a failure and there were no licenses for it. GLOBE trotter is another API, but not a widely used one. License Service API (LSAPI), is an identification for license management supported by Microsoft, Novell, Highland Software, HP, DEC and many more [62].

The ratio to the number of license management API s is more compared to the software vendors. It is required to propose an industrial standard API for license management. Also it is impossible to decide the exact standards for license but truly the fact lies in the hands of developers and it is a matter for them to decide.

6.8. License Use Management:

NodeLocked Licenses: A NodeLocked license is a methodology through which the software can be installed on the machine intended and the license remains valid as long as the agreement is valid. NodeLock licenses can be used for standalone, rather than client/server,
There are two models in license enabling. One is run time and the other is non run time enabling model. Software Distributor can choose between these two[64,65].

If the choice of the Software Distributor is non-runtime-based enabling, then there is no need for the License Use Runtime to be installed. Just a password is required for this which is available in the NodeLock file. The software application checks the NodeLock file to make sure a valid license is available[65].

If the choice is made for runtime-based enabling, NodeLocked license is done by the NodeLocked license server. The NodeLocked license server is managed through the Basic License Tool, BLT which enables you to view and update information about the NodeLocked licenses on the machine and get reports about their use.

Distributors can enable their products using the following kinds of NodeLocked licenses:

- Simple NodeLocked licenses
- Concurrent NodeLocked licenses
- Use-once NodeLocked licenses
- Per-server licenses

6.9. **Simple NodeLocked Licenses**

A license that accepts any number of uses of the licensed product on the machine is called a Simple NodeLocked License. These licenses are Simple NodeLocked licenses and are ideal for Software Distributor use products. An example of such a licensed product is word processor. Suppose for an instance that the Software Distributor enabled the product using non-runtime-based enabling, the following diagram shows the sequence of activities that happen whenever user attempts to run the application.

![Figure 6.1 Simple NodeLock License](image)
6.9.1. Using a NodeLocked License (Non-Runtime-Based Enabling)

Whenever user attempts to run the product, the software checks to see whether the software is installed in the machine or not. If it is installed in the machine, then the product runs or it is distributor enabled.

6.9.2. Concurrent NodeLocked Licenses

Concurrent NodeLock licenses are very much similar to Simple NodeLock excepting that they limit the number of uses of the licensed product. An instance of such a product is a client / server based product where the server identifies the number of users of the licensed software product and prompts for license to each use. Software Distributors can enable the product as either vendor managed or customer managed. If it is customer managed then the customer needs to specify the number of licenses acquired.

6.9.3. 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. Whenever the product is initiated, one license will be expended. The main use or advantage of these licenses is to distribute demonstrable versions of the products. Use-Once licenses are useful to the distributors at times when the user’s request on a software product exceeds the number of concurrent licenses defined on it. Also the products are so defined that when a concurrent license on a particular product is not available, then the user can opt for use-once license on the product until the concurrent license on the product is freed by some customer on the product. This license can be used as both distributor-based or customer-managed. The following pic shows the sequence of steps that happen when an end user invokes NodeLock licensed application with runtime-based enabling.

![Figure 6.2 Using a NodeLocked License – Runtime Enabling](image-url)
Now that the user invokes the software product

1. The product requests license from the NodeLock licensed server for the local system to run the software.
2. The NodeLock licensed server verifies the license on that machine and confirms that the user is authorized to run the product.
3. If the product is not NodeLock enabled but is implemented as soft stop policy, then the server checks user file against this.
4. Finally the NodeLock license server returns the license request status to the product.

The software product can be run after getting the license verified with the NodeLock server. If no license is available i.e. NodeLock license or a soft stop license then the server returns the status and the product cannot be run. With no such licenses, depending on how the distributor enabled the product, it can be run.

6.9.4. Per-Server Licenses

Per-Server licensing procedure is similar to concurrent licenses but can be converted to Per-Seat license based on the requirement. Per Seat/ Per-Server licenses support multiple client/server applications. The server can request to its maximum the number of licenses on behalf of its clients. Each server is associated with a specific number of clients for which it can request license to its extreme. All the client licenses will be stored temporarily in the server and are granted when requested. A number of servers are available to grant license to the clients. A client can get grant to more than one license if it requests for license from more than one server. Thus a client can avail more than one license[66]. To overcome such anomalies, it is advised to use Per-Server license in certain environments where:

- A client can connect to only one server at a time.
- Client can use the product infrequently and for limited amount of time.

It is essential to convert to per-Seat if all or most of the clients are connecting multiple application servers. All the unused client licenses are stored to a repository. Even if a client connects to multiple application server, he can avail only one license. Unlike simple and concurrent licenses, Per Server licenses are available to only customer-managed products. The following figure shows the sequence of activities that happen when the user runs a product that is Per-Server license enabled and Per-Seat being disabled and the server is NodeLock enabled.
Now that the client runs the product

1. The client server requests a per-server license from the NodeLock licensed server.
2. The NodeLock license server checks the database for the license and user authorization. If license is available, then the product would be run else the server checks if the soft license is available.

3. The NodeLock licensed server now returns the status of the request to the client server. If license found, or the permission grant for soft stop license is noticed, the product runs.

If no license is granted to the user, the NodeLock license server checks how the product is enabled by the distributor and decides whether the product could be run or not.

6.9.5. Network Licenses

Unlike the licenses discussed above, Network licenses are not fixed to a particular node; instead they are stored on network license server and will be shared among multiple clients. Various kinds of network licenses are:

- Concurrent licenses
- Concurrent-offline licenses
- Reservable licenses
- Use-once licenses
- Per-seat licenses [67]

Software Distributors can enable the products using the above kinds of network licenses.

6.9.5.1. Concurrent Licenses

A concurrent license is a network license that can be temporarily granted to run the licensed application on a client.
When the product is running, that license remains unavailable to other users of the product. When the product stops running the license is returned to the server, where it becomes available to other users.

Concurrent licenses allow as many users to run a licensed software application simultaneously as there are valid licenses for the product available from the network license servers in your licensing environment.

A typical use of concurrent licenses is for products with relatively expensive licenses that each user will use only part of the time. The customer orders fewer licenses than there are users to optimize use of the licenses. Such applications may be client/server applications for which the client is enabled, or non-client/server applications. Vendors can use concurrent licenses for both vendor-managed and customer-managed products. The following figure shows the steps that happen when an end user invokes an application with concurrent licenses.

Using a Concurrent License
The user invokes the application.

1. The application requests a license from the network license server.
2. The network license server checks its license database for an available license and the user file for authorization.
3. The network license server returns the status of the request to the application. If a license was found and granted, the application runs.
4. If a network license was not found, and the application implements the soft stop policy, the application requests a soft stop license from the central registry license server.
5. The central registry license server checks its database for a soft stop license and the user file for authorization.
6. The central registry license server returns the status of the request to the application. If a soft stop license was granted, the application runs.

If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

6.9.5.2. 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. Only authorized users can exploit this feature. The users must be defined by the license administrator at the customer site. The portable computer can use supported Windows platforms only, excluding Windows Millennium.

A concurrent-offline license is installed on the network license server. It is similar to a normal concurrent license because it can also be used as a normal concurrent license. For the duration of the reservation, the license remains unavailable to other users of the product. The license becomes available to other users when one of the following occurs:

- The reservation period expires
- The license is returned to the server from the portable computer

Concurrent-offline licenses can be enrolled and administered only on a server running License Use Management Version 4.6.7 (or later), by means of a Basic License Tool 4.6.7 (or later). When the concurrent-offline license has been reserved for use, a copy is installed on a portable computer connected to the network. This copy of the concurrent-offline license is called an offline-NodeLocked license. When the offline-NodeLocked license has been installed on the portable computer, the portable computer can be disconnected from the network.

The following figure shows the steps that happen when an end user invokes an application with concurrent-offline licenses.

![Figure 6.5 Using a Concurrent-Offline License](image-url)
The user invokes the application.

1. The application looks for an offline-NodeLocked license in its NodeLocked file. If a license is found and is valid, the application runs.
2. If the application does not find a license in the NodeLocked file and the portable computer can reach a License Use Management network license server, the application requests a concurrent-offline license from it.
3. The network license server checks if the license has an authorization for the user, group, or for the target ID of the machine, and validates the password provided by the user.
4. If the authorization is for a concurrent-offline license, the server creates an offline-NodeLocked license and copies it to the NodeLocked file of the portable computer. The license is marked as in use on the server.
5. The application checks whether the NodeLocked file contains a valid license and the application starts. The application can now run without a connection to the License Use Management network license server (that is, the application can run on a portable computer).

**Note:** At the end of the reservation period, the copy of the license stored in the NodeLocked file of the portable computer expires (the license can also be returned by the user before the reservation period expires). The license can be used by another user both as a normal concurrent license and as a new offline-NodeLocked license.

### 6.9.5.3. 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. The reservation is for a specified time period. A reservable license that has been reserved is called a reserved license. A reservable license that has not been reserved is called an unreserved license.

When a reserved license is granted from the network, the license is stored on the workstation where the licensed application is running. Thereafter, the license can be used on the workstation, even if the workstation is disconnected from the network, until the reservation expires.

A typical use of reservable licenses is for the client part of a client/server application that is likely to run on a portable computer that is often disconnected from the network. Another typical use is for a compiler being used in software development. During a build process involving many compilations, it is more efficient to reserve a compiler license for a day or two than to make a separate request for a compiler license for every compilation.

You can reserve some of the reservable licenses for an application and leave others unreserved. Unreserved licenses are treated like concurrent licenses.

Vendors can use reservable licenses for both vendor-managed and customer-managed products.
6.9.5.4. Using a Reservable License

1. The user invokes the application.
2. The application requests a reserved license from the NodeLocked license server.
3. The NodeLocked license server checks its database for a reserved license. This is a license that you reserved for the user and that was granted to the user, and stored on the local machine, in response to a previous request.
4. The NodeLocked license server returns the status of the request to the application. If a license was found, the application runs.
5. If the NodeLocked license server does not find a license, the application requests a reserved license from the central registry license server. This is a license that you have reserved for this user, group, or workstation.
6. The central registry license server checks its database for a reserved license and the user file for authorization.
7. The central registry license server returns the status of the request to the application. If a reserved license was found and granted, it is stored in the NodeLocked license server's database, and the application runs.
8. If a reserved license was not found, the application requests a reservable license from the network license server. This is a reservable license that you have not reserved for anyone.
9. The network license server checks its license database for a reservable license and the user file for authorization.
10. The network license server returns the status of the request to the application. If a reservable license was found and granted, the application runs.
11. If a reservable license was not found, and the application implements the soft stop policy, the application requests a soft stop reservable license from the central registry license server.
12. The central registry license server checks its database for a soft stop reservable license and the user file for authorization.
13. The central registry license server returns the status of the request to the application. If a soft stop license was granted, the application runs.
If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

### 6.9.5.5. **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. Every time the product is started, one license is consumed.

A typical use of use-once license is to distribute promotional or demonstration versions of software. Vendors also provide use-once licenses for their products to supplement concurrent licenses during times when user demand for those products exceeds the number of available concurrent licenses. The vendor designs the product so that when all concurrent licenses for the product are in use, a user can request an available use-once license [68]. Vendors can use use-once licenses for both vendor-managed and customer-managed products.

The following figure shows the steps that happen when an end user invokes an application with use-once network licenses.

![Figure 6.7 Using a Use-Once Network License](image)

1. The user invokes the application.
2. The application requests a license from the network license server.
3. The network license server checks its license database for an available license and the user file for authorization.
4. The network license server returns the status of the request to the application. If a license was found and granted, the application runs, and one license is subtracted from the number of available use-once licenses.

If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available[43].

### 6.9.5.6. **Per-Seat Licenses**

Vendors use per-server/per-seat licenses to enable client/server applications constructed for multiple-server solutions. Both per-server and per-seat licenses make it possible for the server of a licensed client/server application to request licenses on behalf of
its clients without the need for the application clients to be license-enabled. With per-seat licensing, unused application client licenses are kept in a central repository, which all the application servers share. They also share a central list of application clients to which a license has been assigned. When a license is assigned to an application client, that assignment is permanent. If an application client connects to multiple application servers, it is assigned only one license. You will probably want to use per-seat, rather than per-server, licenses in an environment where application clients connect to multiple application servers. Per-seat licenses are valid only for customer-managed use products. The following figure shows the steps that happen when an end user invokes an application with per-server/per-seat licenses when per-seat has been enabled. In the figure, the application server is license-enabled.

The application client user invokes the application.

1. The application server requests a per-seat license from the NodeLocked license server. This is a license that has already been granted to the user on a previous request and stored on the local machine.
2. The NodeLocked license server checks the NodeLocked license database for such a license.
3. The NodeLocked license server returns the status of the request to the application server. If a per-seat license was found, the application runs.
4. If no per-seat license was found on the NodeLocked license server, the application server requests a per-seat license from the central registry license server.
5. The central registry license server checks whether a license is already being used by the requesting application client, possibly granted through another application server. In such a case the application can start without having a new license granted. Otherwise, the central registry license server checks whether a per-seat license is available. If so, it grants the license and records the application client identifier. If no per-seat license is found, but the application implements the soft stop policy, the central registry license server checks for a soft stop license.
6. The central registry license server returns the status of the request to the application server.
7. If a per-seat or soft stop license was granted, the application sends a shadow copy of the granted per-seat license to the NodeLocked license server.
8. The NodeLocked license server adds the shadow copy to the NodeLocked license database.
9. The NodeLocked license server returns the status of the request to the application server, and the application runs.

If no license can be granted, depending on how the vendor enabled the product, the application may return information to the end user, or it may run even with no license available.

6.9.6. Hardware-Based Node Locked Licenses

Hardware-based Node Locked licenses use a combination of a physical hardware key and a special code that, when installed along with the software, allows Research Systems products to run in their licensed mode. The hardware key, or Hasp, must be installed on the same machine as the software. Both the hardware key and the license code are required in order to run the software in licensed mode. Traditionally, Research Systems used hardware-based Node Locked licenses for machines running Microsoft Windows and the Macintosh OS. Macintosh systems continue to use these licenses, and the details on installing both the hardware key and the license code are discussed in the Macintosh Installation Guide. Microsoft Windows systems can use both hardware-based and software based Node Locked licenses as well as floating (Client/Server) licenses. If you are unsure about which type of license you have purchased for your Windows system, consult the Research Systems Product Installation Form you received with your purchase, or ask your Research Systems sales representative.

6.9.7. Software-Based Node Locked and Floating (Client/Server) Licenses

Both software-based Node Locked and client/server licenses use a license file containing information based on a unique host ID from the machine or machines to be licensed. For some types of these licenses a program called a license manager runs on the system to determine whether the proper license for a given product is available. If the proper license is present and not already in use, the license manager allows the software to run. For some types of Node Locked licenses, including those used on Windows systems, the license manager is not used. Client/Server licensing has several advantages. Since the license manager can operate over a network, many computers can share a limited number of licenses for a Research System product without the need for the physical hardware key. For Unix and VMS, Client/Server licensing may allow the software to run on a remote computer; the real computational work may occur on a powerful server, while results are displayed on inexpensive terminals. In order to gain the ability to administer licenses over a network in this way, however, the license manager software itself must be installed and run, and the central repository of license information must be maintained in a location accessible over the network.
6.9.8. What is the License Manager?

Research Systems products that use Client/Server licensing use the Globetrotter Software FlexLM license manager. The license manager runs all the time your system is running, and waits for a request from a Research Systems product. When a request arrives, the license manager checks the license file for the system to determine if:

1. a license for the product being requested is installed on the system, and
2. the license is not already in use.

If a license is available, the software will run. Before you run one of these Research Systems products, you must provide us with information so that we can create a license file, which you must install on your system. You must then start (or arrange to start automatically) the license manager itself.

6.9.9. How do Client/Server Licenses Work?

The license file, which Research Systems provide, encodes information about your system along with a description of which products are licensed and how many copies of each can be run simultaneously. Each license you purchase provides a certain number of license units; when a license is requested, a certain number of these license units are “checked out” and become unavailable to other users until the software exits. Different products and different platforms “cost” different numbers of license units in order to run. Depending on the total number of license units you have purchased, you will be able to run different numbers of concurrent sessions of the Research Systems product. For example, while a product running on a Sun Ultra workstation may require ten license units in order to run, the same product running on a Linux system may require six units. This means that if you have ten units available, you will be able to run either a single session on a Sun Ultra or a single session on Linux systems. If you have six units available, you will be able to run a session on a Linux or Windows system, but not on a Sun Ultra.

6.9.10. Where is License Information Located?

For all systems except VMS, license information is located in a file named license.dat, located in the license subdirectory of the Research Systems product directory on your system. For example, on most Unix systems, this means that the license file is located in the /usr/local/rsi/license directory. You can locate the license file in other locations if you set the appropriate environment variables, logical names, or preferences, depending on the type of system you are using. See the chapter in this volume that discusses the platform on which your license manager runs for complete details.
6.9.11. Multiple Products Using the License Manager

Since use of the FlexLM license manager is not unique to Research Systems, the possibility exists that you will need to run software from two or more software vendors that use FlexLM licensing. In this case, the potential for license server conflict exists.


In the interest of protecting customer data or securing trade secrets many companies are modifying their mechanisms of transferring data across the Internet. There are a number of things to consider when improving the security of data transfer procedures. These include:

· User Authentication – FTP has traditionally used clear text passwords. This weakens the security as someone can pick up the password that is used and use it later to get access to the data. Remote system identification verification is used to prevent hijacking of the packets by a system masquerading as the destination. Data privacy (encryption) is maintained so that no intermediate system can use the data. Data security (integrity or tamper prevention) is ensured to prevent modification to the data while it is in transit. Preservation of data format. Different operating systems may store data in different formats. It is desirable to have a defined interchange format when this is the case. FTP has traditionally done a very good job of performing data interchange between systems.

· Ease of use. A mechanism that requires extra steps or is not easy to use will encourage users to take short cuts that may not preserve the desired security when they are in a hurry. A variety of mechanisms are discussed below. These mechanisms are: separate encryption, SFTP (Secure Shell File Transfer Program), FTP over Secure Shell (SSH), IPSec, Virtual Private Networks, and FTP over Transport Layer Security (TLS). Each mechanism has arguments for and against it so no single one can be declared the solution to all problems.

6.9.13. Separate Encryption before using FTP

Encryption of the data by a separate program before the transfer was probably the first method used to solve this problem. Though this method is readily available, it doesn't solve all of the problems. This method doesn't protect the user's password, so someone spying on the transmission could get access to the data after it has been decrypted unless a separate mechanism is used to limit the reusability of passwords. The requirement for manual encryption could cause problems when the user is in a hurry or discovers that there is a file that is needed that wasn't encrypted before the transfer session was started.

A Comparison of Secure File Transfer Mechanisms Process Software Secure File Transfer Mechanisms systems. It is possible that the data also needs to be converted either before or after it is encrypted. While this may be taken care of by the encryption program, it is something to take into account when evaluating this method. Separate encryption may not provide data integrity. There are no mechanisms for the server to certify that it is the intended system in this method. This is the weakest method[69].
SFTP (SSH File Transfer Program)
SFTP is widely available for a number of platforms and it solves the problems of securing the user’s password and provides data encryption and integrity on the fly. SSH (which SFTP uses as an authentication and data transport mechanism) also authenticates the server involved through the exchange of keys. SSH keys are privately maintained and require external acceptance upon first use or prior transfer through an alternate method. Since SFTP only uses a single TCP connection to exchange both commands and data, it does not have the problems with firewalls that FTP can have. Unfortunately SFTP doesn't always preserve the file format when different operating systems are involved. The SFTP protocol was originally specified as a binary file access protocol. Though a text transfer mechanism was added in later revisions of the specification, not all implementations support it, particularly the most popular (OpenSSH). The protocol also provides a mechanism for arbitrary file characteristics to be passed on the file open command, but this is not highly used. The SSH community has allowed the draft specification to expire as it was felt that the group didn’t have the necessary expertise to standardize a file access protocol. While there are many implementations available, not all of them will update the protocol level that they support if they don’t feel that their market has a use for it. This method provides acceptable functionality for many users though the user may have to experiment to deal with text file format conversion issues.

FTP over SSH
SSH can be used to create a secure tunnel between two systems. It is possible to have one end of this tunnel point to an FTP server and provide a secure channel for FTP transfers. Some SSH servers and clients recognize the FTP PORT and PASV commands and replies and can provide protection for the data channel as well. To use this method an SSH connection must be established between the two systems before the FTP connection is established, which adds inconvenience or uses resources even when there are no transfers being done. With this method SSH provides data privacy and integrity, server identification verification and privacy for the user password. FTP provides any data format conversion that is necessary between the two systems.

IPSec (and FTP)
IP Security (IPSec) provides secure communications (authentication, integrity, confidentiality) over IP-based networks between systems. Not all systems have IPSec available. Even when systems have it available, configuring differing types of systems to work together can be a challenge. Since this needs to be configured on a per system basis it may lack flexibility when destinations or sources change frequently. Since IPSec protects the individual packets sent between nodes it can present a problem if one of the nodes is operating behind a NAT device that does not support IPSec NAT traversal. Depending upon how it is configured, IPSec can provide both data integrity and data security. If data
protection is desired, then it is necessary to configure IPSec to encrypt all traffic between the two systems as FTP may use an arbitrary data port for the transfer. Configuring IPSec to encrypt all data transferred between two addresses may expend CPU cycles where it isn’t desired. Since IPSec is implemented in the lower layers of the IP protocol use of it can lead to a high amount of CPU cycles being used at interrupt or kernel levels. IPSec generally requires configuration by a system administrator and therefore users may be delayed in performing transfers while waiting for new configurations to be entered and tested. IPSec provides host authentication while doing key negotiation and during data transfer.

Virtual Private Networks (and FTP)

A Virtual Private Network uses encryption to provide secure communication between two systems. It may do it at network layer 2, by creating a logical wire between the two systems. In this case all network traffic passes over this logical wire, whether or not it is destined for the system at the other end. Or it may create it at network layer 3 by encrypting and encapsulating packets that traverse a particular route. A VPN can also be created with external (router) hardware being configured to encrypt the data between specified addresses. Not all systems support the creation of a virtual private network (VPN), and not all systems that do offer compatible mechanisms. The VPN setup mechanism may or may not provide a method of verifying the identity of the remote system. A VPN may end up carrying (and encrypting) data that does not need to be carried over a secure network connection if it operates at layer 2, or if routing characteristics direct all traffic through it. This method also has the potential problem of limited adaptability to quick changes.

FTP over TLS (FTPS)

Transport Layer Security (TLS, RFC 2246) is commonly used to secure data transferred between web browsers and servers (https). TLS is also known as Secure Socket Layer (SSL). FTP over TLS is specified by RFC 4217 and uses TLS to add password privacy and server verification to FTP. It also makes privacy for data transfers available. The command channel is protected during the user authentication procedure and may be set to clear after setting file transfer protection requirements to let firewalls and NAT devices learn about the FTP data channel and open the necessary ports to allow the data to be exchanged. The data channel may be set to private, which provides both data security and integrity. FTP was designed to properly handle ASCII and binary file transfers so it does well when different system types are involved. The data encryption takes place after the data is converted to standard interchange formats so conversion of file formats is not an issue. Since the data protection features are integrated into FTP the CPU cycles necessary to provide it are expended at user level. This functionality is available for many systems, but may not be available for all as it requires integration with the FTP utility. TLS provides server authentication with keys that may be either self-signed or signed by a trusted authority. Servers may be configurable to require secure data transfers. FTP over TLS requires an
explicit request for encryption and server authentication so use of the secure channel can be optional. There are a number of methods available for providing security to data transfers. It is hard to say that a particular one is the best, as functionality and availability of implementations are a key concern. A system that provides a variety of methods has a higher possibility of allowing data to be exchanged in a usable format with necessary security and a minimum amount of inconvenience [69].

6.10. NodeLock License – Scenarios and Patterns

Let’s go in detail about deriving the variations on NodeLock license methodologies for above mentioned areas and specifications:

6.10.1. Software Types

We classify software into following types based on hardware or device interactions.

![Figure 6.9 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 [70]. 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, eom 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 NodeLock 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>NodeLock License component of the firmware continuously monitors the health of firmware and gets the license renewed based on a) Health level of hardware b) Fixed duration c) Fixed usage</td>
<td>NodeLock License component of operating system monitors the health of the system and gets the license renewed based on a) Hardware health b) Peripheral health c) Fixed duration. d) Fixed usage</td>
<td>Device Drivers basically play a role on firmware for specific plug 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</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 a) Partial or full list of hardware attributes b) Partial or full health levels</td>
</tr>
</tbody>
</table>
2. The level of security (using RSA/RMPRSA) can be customized based on firmware needs. Example: Consider \( k \) number of hardware attributes out of total number \( n \) based on support/maintenance related needs, where \( m \leq k \leq n \), where \( m \) is less secured/basic need and \( n \) is highly secured/complete need.

| Firmware for industrial hand-helds | SYMBION, etc. Also NodeLock License component of operating system checks for continuous patches, service packs upgradation, heart fixes from various support groups and notifies the user about the license validity accordingly. | required. This can be achieved through a) Hardware attributes of the peripherals b) Peripheral health levels c) Fixed duration d) Fixed usage Example: Dell Network Drivers, Creative Sound drivers, VGA Graphic drivers, etc. | The level of security (using RSA/RMPRSA) can be customized based on firmware needs. Example: Consider \( k \) number of peripheral attributes out of total number \( n \) based on support/maintenance related needs, where \( m \leq k \leq n \), where \( m \) is less secured/basic need and \( n \) is highly secured/complete need. | of system or environment c) Fixed duration d) Fixed usage Example: MS Office, Phone Book Manager |

| 2. | The level of security (using RSA/RMPRSA) can be customized based on firmware needs. Example: Consider \( k \) number of peripheral attributes out of total number \( n \) based on support/maintenance related needs, where \( m \leq k \leq n \), where \( m \) is less secured/basic need and \( n \) is highly secured/complete need. | | The level of security (using RSA/RMPRSA) can be customized based on firmware needs. Example: Consider \( k \) number of partial/full system attributes out of total number \( n \) based on support/maintenance related needs, where \( m \leq k \leq n \), where \( m \) is less secured/basic need and \( n \) is highly secured/complete need. | |

*Table 6.1 Applicability of NodeLock License for software Types*
6.10.2. Node Types

NodeLock License component is available for various node types such as computer, devices and mobiles. In this research, 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, mother board id, MAC addresses, etc here is our classification on node types based on hardware devices:

![Node Types based on Hardware devices](image)

**Figure 6.10 Node Types based on Hardware devices**

Applicability of NodeLock License for various NODE types:

<table>
<thead>
<tr>
<th>Computer</th>
<th>Mobile</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeLock Licensing Methodology</td>
<td>We found the most innovative and valuable</td>
<td>We see that softwares are getting extended to address</td>
</tr>
<tr>
<td>further goes to</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

138
NodeLock Licensing Methodologies Using RMPRSA Cryptography

address specific needs of different computer variations below:

1) Server grade
2) PC grade

Server grade machines will have NodeLock License interfaces for further classifications like
a) Network_based servers
b) Job servers
c) Log servers
d) Real_time servers
e) Support servers
f) Storage servers
g) Processing servers, etc.

PC grade machines have analog interactions, customized for
a) Desktops
b) Laptops
c) Tablets
d) Hand-held computers, etc.

softwares are getting developed for mobile devices. Hence deriving NodeLock Licensing Methodologies or patterns is also appropriate. Here is our classification:
a) Smart phone(most appropriate)
b) Basic phone(not appropriate)
c) Moderate(unlikely appropriate).

the specific needs apart from computer and mobile phones like
a) Industrial
b) Automation
c) Healthcare
d) Home automation, etc.

NodeLock Licensing Methodology can further be customized based on the software cost, support mechanism, SLA (Service Line Agreement), etc.

<table>
<thead>
<tr>
<th>Softwares are getting developed for mobile devices. Hence deriving NodeLock Licensing Methodologies or patterns is also appropriate. Here is our classification:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smart phone (most appropriate)</td>
</tr>
<tr>
<td>Basic phone (not appropriate)</td>
</tr>
<tr>
<td>Moderate (unlikely appropriate)</td>
</tr>
</tbody>
</table>

PC grade machines have analog interactions, customized for
a) Desktops
b) Laptops
c) Tablets
d) Hand-held computers, etc.

<table>
<thead>
<tr>
<th>PC grade machines have analog interactions, customized for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktops</td>
</tr>
<tr>
<td>Laptops</td>
</tr>
<tr>
<td>Tablets</td>
</tr>
<tr>
<td>Hand-held computers, etc.</td>
</tr>
</tbody>
</table>

Table 6.2 NodeLock License for various Node Types

6.10.3. Usage

In this research 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>
6.10.4. 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 (collects the details and sends 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 (sends 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.

6.10.5. Cost

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

1) Expensive
2) Nominal [9]
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 required</td>
<td>Nominal license validation required</td>
<td>Low license validation required</td>
</tr>
</tbody>
</table>
NodeLock Licensing Methodologies Using RMPSRA Cryptography

<table>
<thead>
<tr>
<th></th>
<th>Should follow most appropriate hardware attributes for NodeLock License validation</th>
<th>Should follow important hardware attributes for NodeLock License validation</th>
<th>Should follow reasonable hardware attributes for NodeLock License validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Should follow most appropriate hardware attributes for NodeLock License validation</td>
<td>Should follow important hardware attributes for NodeLock License validation</td>
<td>Should follow reasonable hardware attributes for NodeLock License validation</td>
</tr>
<tr>
<td>3</td>
<td>Better to consider as many as possible attributes of the hardware</td>
<td>Better to consider good set of attributes of the hardware</td>
<td>Better to consider minimum set of attributes of the hardware</td>
</tr>
<tr>
<td>4</td>
<td>We suggest 0 to 5% of tolerance to go for NodeLock License verification with reference to full set of hardware attributes(n) such that ((n – (n \times 5)/100) \leq k \leq n), where (k) is the number of attributes considered for NodeLock License validation</td>
<td>We suggest 0 to 20% of tolerance to go for NodeLock License verification with reference to full set of hardware attributes(n) such that ((n – (n \times 20)/100) \leq k \leq n), where (k) is the number of attributes considered for NodeLock License validation</td>
<td>We suggest 0 to &lt; 50% of tolerance to go for NodeLock License verification with reference to full set of hardware attributes(n) such that ((n – (n \times 49)/100) \leq k \leq n), ((n – (n \times 50)/100) &lt; k \leq n), where (k) is the number of attributes considered for NodeLock License validation</td>
</tr>
</tbody>
</table>

Table 6.4 NodeLock License Methodology based on cost

6.10.6. Software Features

NodeLock Licensing Methodology can be customized 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 License 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.
6.10.7. New Versions and Upgradations

Here we are deriving the patterns of NodeLock License methodology for softwares based on software versions, service packs, feature upgradations, 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.