IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. VIII (Mar-Apr. 2014), PP 35-44
www.iosrjournals.org

Performance Analysis of NodeLock Licensing Methodology with RMPRSA Cryptography

Cuddapah Anitha[1], Prof.M.Padmavathamma[2]
[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:*There are many means of unauthorized access of software products being applied. To control this access towards copying the software products on various systems, software developers generally define certain restrictions by controlling its operation being used in a particular environment. Despite of this, it has become a major challenge in the software industry and for the developers of the software products in protecting their intellectual property rights. Hence, in an attempt to prevent piracy in the upholding environment, this paper introduces a novel innovation against the piracy called NodeLock Licensing Methodology with RMPRSA Cryptography.*

Keywords:*Decryption, Encryption, Key generation RSA, RMPRSA Cryptography*.

I. Introduction

As there lies the fact that software can be copied from any source to any destination without involving cost, people, and effort 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. So it is necessary to devise a mechanism to ensure that whosoever possesses license or paid royalties alone should be able to access the software and that the license holders should also be legitimate users. This will ensure that the licensed software should not be run on more number of systems than specified. There are many means of gaining unauthorized access to software products being used. To control this access to copying the software products on various systems, software developers adopt various control measures that include:

1) placing a limit on the product being used in a particular environment,

2) making the product unusable after a predetermined period of time,

3) developing a mechanism so as to generate a key to complete its installation on user's site.

4) to generate a call automatically to the License Provider's site each time the product is run for its installation.

To address this unauthorized access to software products everywhere outside connecting environments, we propose NodeLock Licensing along with target security needs by extending RSA with multi prime support called Multi Prime Rebalanced RSA and performance analysis results when compared to RSA with respect to key generation, encryption and decryption. The present work consists of 5 sections. The first section talks about the introduction, functionality, activity diagram in reference to the flow of information among various components of the system. Section 2 talks about the innovation, the NodeLock Licensing Methodology algorithm with RMPRSA cryptography and defines its key generation, encryption and decryption parameters. Section 3 talks about the performance analysis of NodeLock Licensing Methodology algorithm with RMPRSA cryptography over RSA, observations and averaging of key generation, encryption and decryption and their comparison graphs. Section 4 talks about the conclusions and future scope.

In particular the present RMPRSA algorithm is applied to a NodeLock License while generating the keys for the installation of software on user's site These Node Lock Licenses are always specific to a particular node or a system in 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 in an encrypted form and managed by the License Provider Server. This agreement between the software developer and the user of the software is permanent and is valid till the expiry of license as indicated in the certificate. This even introduces a security issue that addresses the duplication of the license information by copying it on another system and, hence, with the Node Lock License, the entire installation process can be controlled and gets completed in regard to the validation of encrypted hardware details as is in our algorithm. The users of the software licensed with Node Lock should be aware that the License Provider monitors the usage of the software and that any unauthorized access to the software installation procedure would not be entertained which is the main concern of the software development organization providing node lock licenses. This fact enforces a constraint on the users of the software that

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.

Definition. A *NodeLocked License* allows a single instance of an application to run on a specific machine. NodeLocked Licenses are directly tied to the hardware of the machine on which the licensed application license is installed. Node-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*.

1.1. Working Of Node Locked License Management System

The diagram below illustrates the functionality and steps involved in NodeLocked License Methodology.

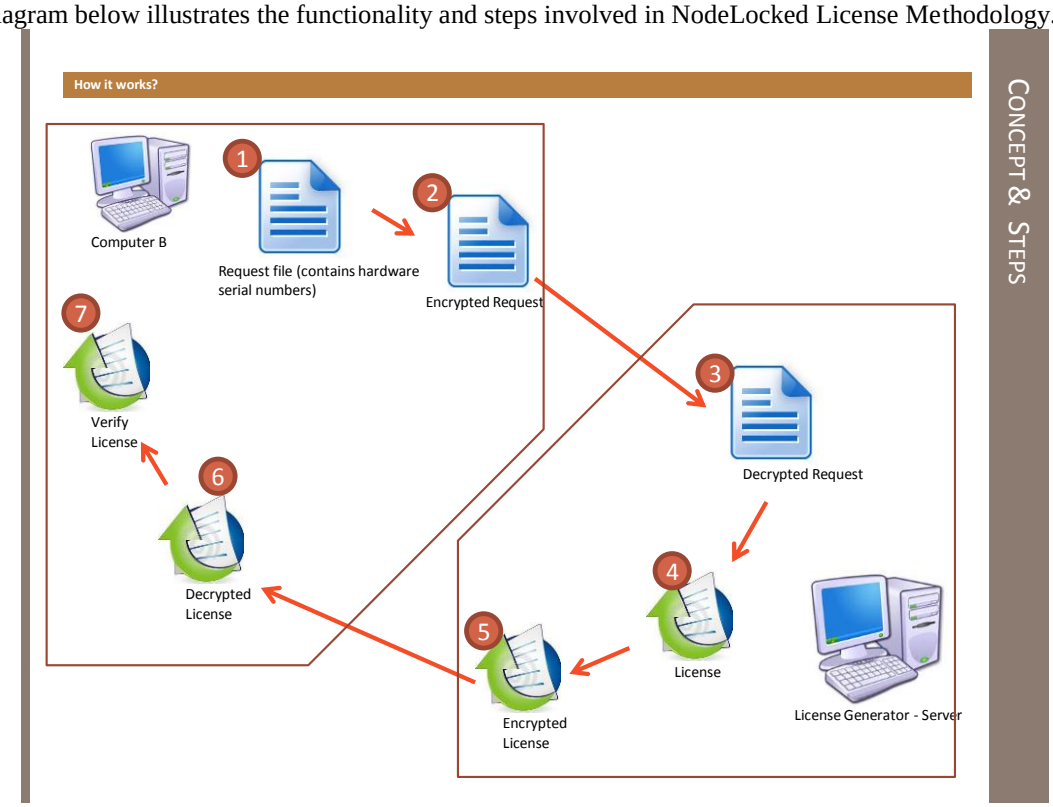

Fig 1: NodeLock License Management System

Working procedure of the node 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 NodeLock License methodology:

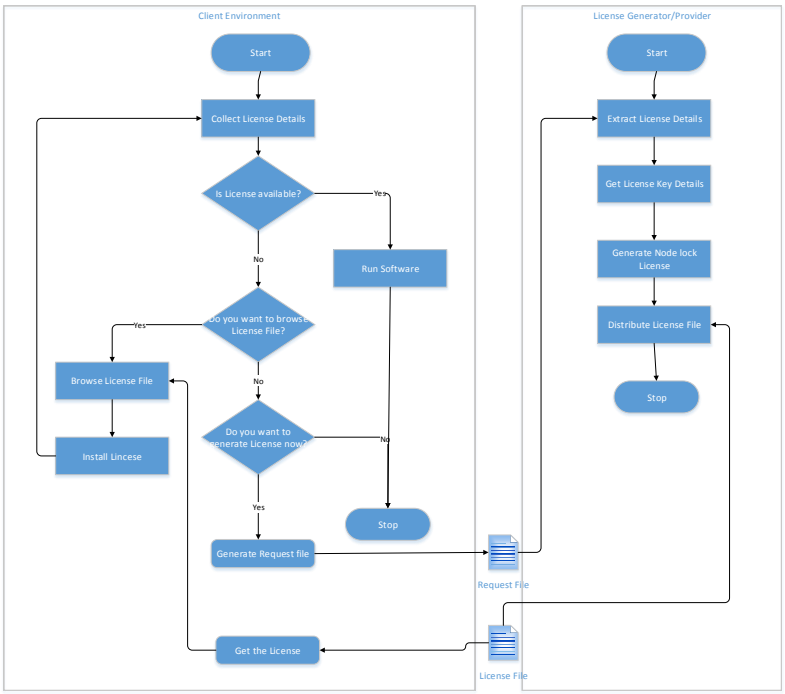

Fig 2. The Activity diagram for NodeLock License Methodology

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….pn), and q is a product of second set of prime numbers (q1…qm).

For security purposes, the prime product p1..pn and q1..qm 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 $\varphi(n) = \varphi(p)\varphi(q) = (P - 1)(Q - 1)$, where φ is Euler's totient function.

Choose an integer e such that $1 < e < \varphi(n)$ and $\gcd(e, \varphi(n)) = 1$; i.e. e and φ(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 $d-1 \equiv e \pmod{\varphi(n)}$, i.e., d is the multiplicative inverse of e (modulo φ(n)). This is more clearly stated as: solve for d given $d \cdot e \equiv 1 \pmod{\varphi(n)}$. 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 φ(n) must also be 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 message M to Alice. He first turns M into an integer m, such that $0 \leq m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to

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

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.
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 the approach known as Multi 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 Q, we can use more primes. For example N =(4*(p1.p2….pn)+1)(4*(q1.q2…qm)+1). However, with a bit length of 1024, it is not secure anymore to use a decomposition of more than three primes.

RMPRSA technique was introduced by Collins who modified the RSA algorithm so that it consists of k primes p1, p2,……,pk 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.
a) The primes are smaller and key generation takes less time despite there being more of them.
b) 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 RMPRSA

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:
i) Select k primes at random, each of which is n/3 bits in length.
ii) Set n = (4*(p1*p2*……*pn)+1)*(4*(q1*q2*q3*…qm)+1) and
φ (n) = ((4*(p1*p2*……*pn)+1)-1)*((4*(q1*q2*q3*…qm)+1)-1)
iii) Pick randomly an odd integer, e ($1 < e < \varphi$), such that gcd (e, φ (n)) = 1.
iv) After that find an integer, d ($1 < d < \varphi$), such that d = e -1 mod φ (n). This is part of our private key and must be kept secret.
v) Select an integer, e ($1 < e < \varphi$),
such that gcd (e, φ) = 1 and e and φ are relatively prime.
v) Find an integer, d ($1 < d < \varphi$), such that e * d = 1 (mod φ). This is part of our private key and must be kept secret.
vi) The public key is e and n, or (e, n), and the private key is d and n, or (d, n).

ENCRYPTION OF RMPRSA

Once we have generated a public/private key pair, we can encrypt a message with the public key with the following steps.
i) 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 p1 * p2 *…..* pk. 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.
ii) Calculate the cipher text C using the public key containing e and n. This is calculated using the equation C = (M exp e) mod n.

DECRYPTION OF RMPRSA

Once we have generated a public/private key pair, we can encrypt a message with the public key with the following steps.
Finally, we can perform the decryption procedure with the private key using the following steps.
i) The decipher first computes Mi for $1 \leq i \leq k$ such that Mi = Ci di mod pi where Ci = C mod pi
ii) Next the message M can be obtained as M = Cd mod n by applying Chinese Remainder Theorem.
iii) Calculate the plain text M using the public key containing e and n. This is calculated using the equation M = (C exp e) mod 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 1. Equivalently, key length = ceiling(log2(n+1)). The other convention, sometimes used, is that the key length is the number of bytes needed to store *n* multiplied by eight, i.e. ceiling(log256(n+1))*8.
The following table is taken from NIST's Recommendation for Key Management. It shows the recommended comparable 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

i. CLIENT: STEP1

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

$$m=StrFun\ (h_1, h_2, h_3, h_4,.....h_n)$$

Where $h_1,h_2,h_3,.....h_n$ 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 details/notations in a single message context.

2. Compute C_1(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}, ...p_{1n1} and q_1 be the product of n_2 randomly chosen distinct primes q_{11}, q_{12}, ...$q_{1n2.}$

i.e, $p_1 = \prod^{n1}_{i=1}(p_{1i})$ and $q_1 = \prod^{n2}_{i=1}(q_{1i})$

Let $P_1 = (4p_1+1)$, $Q_1 = (4q_1+1)$, and $N_1= P_1Q_1$

Compute Euler's Totient function of N_1.

$$Now\ \Phi(N_1)= (P_1-1)(Q_1-1)$$

Chose an integer e_1, where $1<e_1<\Phi(N_1)$, such that $GCD(e_1, \Phi(N_1)) = 1$
The pair (N_1, e_1) is the public key1, this shall be distributed along with the software or program for which the Node Lock license is required.
For this message m ϵ Z_{N1}, the cipher text is computed as

$$C_1=m^{e1} \bmod N_1$$

3. Compute C_2(encrypted message) from m using multi-prime rebalanced RSA.

Let p_2 be the product of n_3 randomly chosen distinct primes p_{21}, p_{22}, ...p_{1n3} and q_2 be the product of n4 randomly chosen distinct primes q_{21}, q_{22}, ...$q_{2n4.}$

i.e, $p_2 = \prod^{n3}_{i=1}(p_{2i})$ and $q_2 = \prod^{n4}_{i=1}(q_{2i})$

Let $P_2 = (4p_2+1)$, $Q_2 = (4q_2+1)$, and $N_2= P_2Q_2$

Compute Euler's Totient function of N_2.

$$Now\ \Phi(N_2)= (P_2-1)(Q_2-1)$$

Chose an integer e_2, Where $1<e_2<\Phi(N_2)$, such that $GCD(e_2, \Phi(N_2)) = 1$
The pair (N_2, e_2) is the public key2, this also shall be distributed along with the software or program for which the Node Lock license is required.
For this message m ϵ Z_{N2}, the cipher text is computed as

$$C_2=m^{e2} \bmod N_2$$

4. Create a Request file/content by combining the results of step2 and step3.

req = length of C_1 +C_1 + length of C_2 + C_2

Here is the simple protocol to merge the contents of encrypted hardware details C_1 and C_2

ii. SERVER: STEP2

5. After receiving the Request file at server(license generator) side, extract the contents of C_1 and C_2

req = length of C_1 +C_1 + length of C_2 + C_2

6. Compute $d_1 = e_1^{-1} \bmod \Phi(N_1)$, the private key1 is the pair (N_1, d_1)

For the encrypted message $C_1 \epsilon Z_{N1}$, the plaintext is recovered by computing $m_1 = C_1^{d1} \bmod N_1$.

7. Compute $d_2 = e_2^{-1} \bmod \Phi(N_2)$, the private key2 is the pair (N_2, d_2)

For the encrypted message $C_2 \epsilon Z_{N2}$, the plaintext is recovered by computing $m_2 = C_2^{d2} \bmod N_2$.

8. Compare the plaintexts of m_1 and m_2 and verify whether the request file/ content is valid.

Isvalidrequestfile = (m_1 equals to m_2)

Or

If ($m_1 = m_2$) then isvalidrequestfile = true;

elseisvalidrequestfile = false;

i.e., $m = m_1 = m_2$

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 ($m_1 \neq m_2$) then If $f(m_1) \neq f(m_2)$

This is Node Lock License concept.

In other terms, for any two hardware details m_1 and m_2, $f(m_1) \neq f(m_2)$.

10. Compute C_3(encrypted cipher text from message f(m), using multi prime re-balanced RSA.

Let p_3 be the product of n5 randomly chosen distinct primes p_{31}, p_{32}, …….p_{3n5} and q_2 be the product of n6 randomly chosen distinct primes q_{31}, q_{32}, …….q_{3n6}.

i.e, $p_3 = \prod^{n5}_{i=1}(p_{3i})$ and $q_3 = \prod^{n6}_{i=1}(q_{3i})$

Let $P_3 = (4p_3+1)$, $Q_3 = (4q_3+1)$, and $N_3 = P_3Q_3$

Compute Euler's Totient function of N_3.

$$\text{Now } \Phi(N_3) = (P_3-1)(Q_3-1)$$

Chose an integer e_3, Where $1<e_3<\Phi(N_3)$, such that $GCD(e_3, \Phi(N_3)) = 1$

The pair (N_3, e_3) 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 $m \epsilon Z_{N3}$, the cipher text is computed as

$$C_3 = f(m)^{e3} \bmod N_3$$

iii. CLIENT: STEP3

11. After receiving the license key at the client (license validator) side, decode the contents of cipher text C_3.

License key = $f(m) = C_3^{d3} \bmod N_3$ (this is license key decoding)

12. Define function g such that f(m) identically equal to g(m).

i.e., $f(m_1)$ and $g(m_1)$ shall always give similar output

and for m_1 and m_2, $f(m_1)$ identically equal to $g(m_2)$

13. Get the hardware details or device properties (h) and see whether license is valid

Isv3alidlicense = (f(m) identically equal to g(m))

Or

If f(m) identically equal to g(m) then

Isvalidlicense = true

Else isvalidlicense = false

14. Register the product on successful validation of Node Lock License key.

III. Performance Analysis Of NodeLock License 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

In colloquial language 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, the mode, 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.

i. 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.

Key Gene		Ob1	Ob2	Ob3	Ob4	Ob5	Average
128	Simple RSA	0.6279	0.2723	0.4874	0.383	0.7367	0.50146
	RMP RSA	696.0648	146.6199	201.9249	221.2803	352.5945	323.6969
256	Simple RSA	0.3127	0.9729	0.8728	0.9041	0.7694	0.76638
	RMP RSA	74.5695	1055.932	791.0222	415.6621	1087.56	684.9491
512	Simple RSA	0.6693	1.279	0.6645	1.4296	0.8608	0.98064
	RMP RSA	465.2311	1186.79	999.986	1479.969	952.5758	1016.91
1024	Simple RSA	2.6682	2.3708	2.8077	2.5103	2.2572	2.52284
	RMP RSA	1102.666	1167.503	951.7881	981.1397	472.9549	935.2104

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.

Encryption		Ob1	Ob2	Ob3	Ob4	Ob5	Average
128	Simple RSA	7.8973	8.107	8.5627	7.8429	8.1739	8.11676
	RMP RSA	8.5022	8.9048	5.4619	6.7631	8.7485	7.6761
256	Simple RSA	15.5854	16.1727	15.641	14.873	13.4838	15.15118
	RMP RSA	15.6311	13.8779	13.2091	11.2121	10.3994	12.86592
512	Simple RSA	28.57	31.0088	25.4308	31.0126	30.1239	29.22922
	RMP RSA	17.8682	19.4078	15.8503	15.7093	17.0822	17.18356
1024	Simple RSA	57.141	54.1771	46.8991	45.5109	44.8252	49.71066
	RMP RSA	29.2374	30.4164	29.633	30.2389	30.5468	30.0145

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.

Decryption		Ob1	Ob2	Ob3	Ob4	Ob5	Average
128	Simple RSA	9.0503	9.6112	5.9932	7.4266	9.3211	8.28048
	RMP RSA	1.9642	1.6408	1.5085	1.7221	0.78	1.52312
256	Simple RSA	16.2283	14.9231	14.2427	12.2063	10.8267	13.68542
	RMP RSA	1.7299	1.789	2.1042	1.8164	0.895	1.6669
512	Simple RSA	18.6295	20.2763	16.6265	16.5832	19.3476	18.29262
	RMP RSA	1.7039	2.0137	1.8083	1.802	1.7953	1.82464
1024	Simple RSA	30.0661	31.234	30.6878	31.3711	31.3831	30.94842
	RMP RSA	1.9113	2.0469	1.5422	1.8862	1.9656	1.87044

Table 0. 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.

128 Bit Mode	Simple RSA	RMP RSA
Key Generation	1	324
Encryption	8	8
Decryption	8	2

Table 4. Comparison matrix of Simple RSA and RMPRSA

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

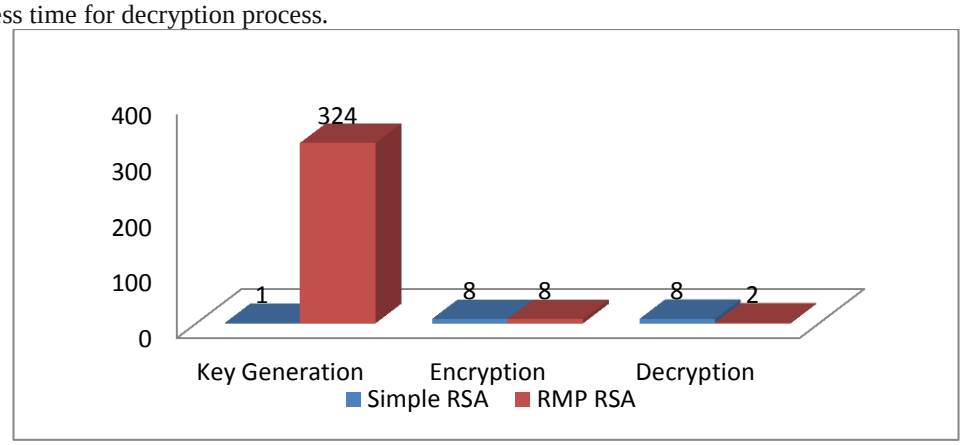

Figure 3. Graphical inferences drawn on performance with 128 bit

ii. COMPARISON AND GRAPHS – 256 BIT

Following table shows the comparisonFigure0—1 matrix of Simple RSA and RMP RSA for Key generation, Encryption, Decryption process steps individually under 256 Bit mode.

256 Bit Mode	Simple RSA	RMP RSA
Key Generation	1	685
Encryption	15	13
Decryption	14	2

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.

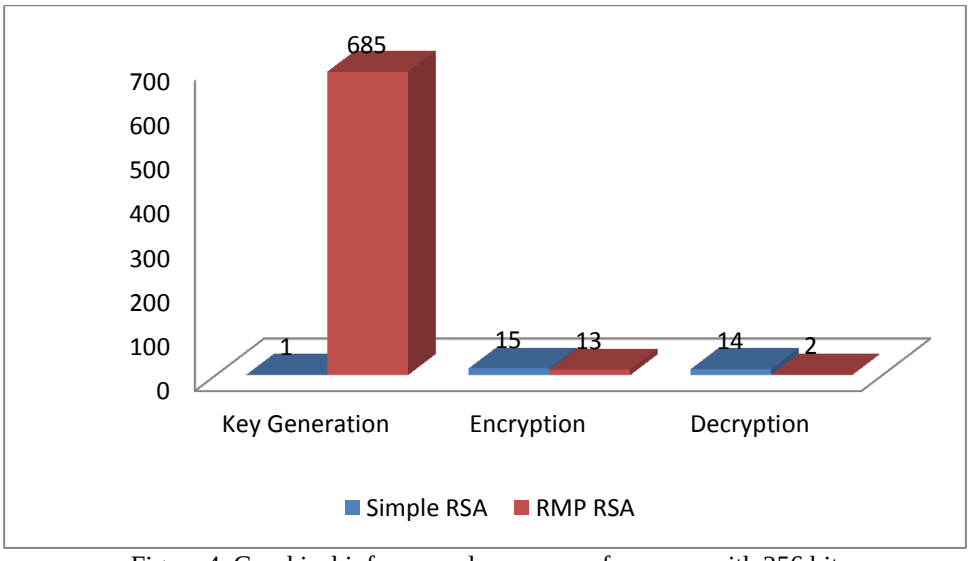

Figure 4. Graphical inferences drawn on performance with 256 bit

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.

512 Bit Mode	Simple RSA	RMP RSA
Key Generation	1	1017
Encryption	29	17
Decryption	18	2

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.

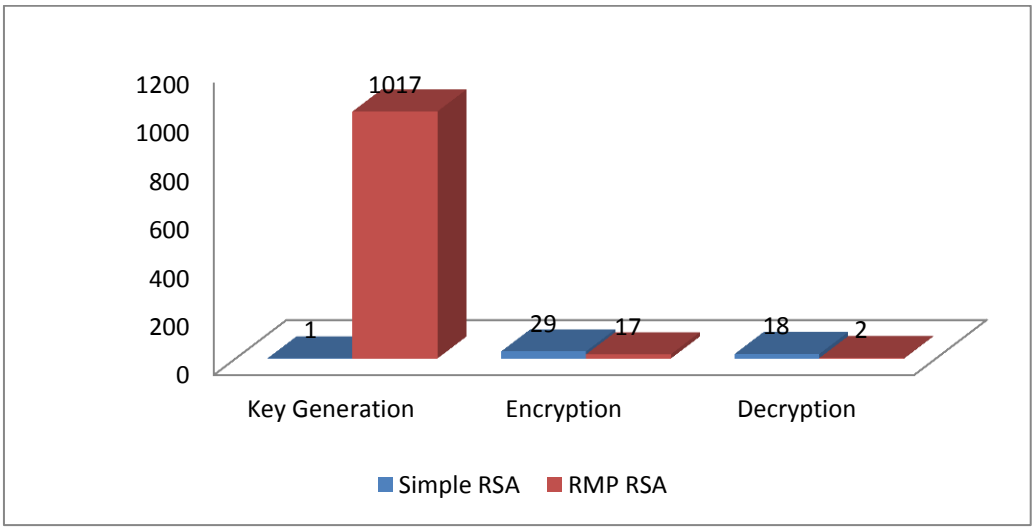

Figure 5. Graphical inferences drawn on performance with 512 bit

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.

1024 Bit Mode	Simple RSA	RMP RSA
Key Generation	3	935
Encryption	50	30
Decryption	31	2

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 NodeLocked Licensing Methodology for protecting sensitive data during NodeLock Licensed software installation. Generally sensitive data will be stored on disks 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

[1]. B.Persis Urbana Ivy, PurshotamMandiwa, Mukesh Kumar, " A modified RSA cryptosystem based on 'n' prime numbers", International Journal of Engineering And Computer Science ISSN:2319-7242 Volume 1 Issue 2 Nov 2012 page No. 63-66.

[2]. Deepak Garg, SeemaVerma, " Improvement over Public Key Cryptographic Algorithm", IEEE International Advance Computing Conference.
[3]. G.A.V.Rama Chandra Rao, P.V.Lakshmi, and N.Ravi Shankar, "A Novel Modular Mutiplication Algorithm and its Application to RSA Decryption", IJCSI International Journal of Computer Science *ssues, Vol 9, Issue 6, No 3, November 2012, ISSN (Online): 1694-0814*.
[4]. Hung-Min Sun, Mu-En Wu, M.JasonHinek, Cheng-Ta Yang, Vincent S. Tseng, "Trading Decryption for Speeding Encryption in Rebalanced-RSA", The Journal of Systems and Software, Elsevier Inc.,0164
[5]. Klaus Hansen, TroelsLarsen and Kim Olsen, " On the efficiency of Fast Variants in Modern Mobile Phones", IJCSIS, vol 6, no 3, 2009, ISSN 1947-5500.
[6]. Lalit Singh, Dr. R.K.Bharti, " Comparative Performance analysis of Cryptographic Algorithms" International Journal of Advanced Research in Computer Science and Kaminski, Mark Perry, "Open Source Software Licensing Patterns", Computer SciSoftware Engineering, Volume 3, Issue 11, November 2013, ISSN: 2277 128X
[7]. MayankJhalani, Piyush Singh, GauravShrivastava, " Enhancement over Variant of Public Key Cryptography (PKC) Algorithm, International Journal of Engineering Technology and Advanced Engineering, ISSN 2250-2459, ISO 9001:2008 Certified Journal, volume 2, Issue 12, December 2012.
[8]. SushmaPardhan, Birendra Kumar Sharma, *"An Efficient RSA Cryptosystem with BM-Prime Method",* " International Journal of Information & Network Security(IJINS), Vol 2, No 1, February 2013, pp. 103-108, ISSN: 2089-3299.
[9]. Wang Tao, " Evaluation and Construction of Individual credit evaluation system Based on third party e-commerce transaction platform", 2010 International Conference on E-business and E-Governance, IEEE DOI 10.1109/ICEE.2010.79.