Volume 2, Issue 9, September 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Enhancement of RSA Cryptography Algorithm by Membrane Computing

Salah Zaher, Amr Badr & Ibrahim Farag
Faculty of computer & information
Cairo University, Egypt
Tarek Abd Elmageed
Governmental Security Consultant
Egypt

Abstract— ***The problem of using RSA algorithm in cryptography is the long time it takes for the encryption process. To overcome the problem, RSA algorithm uses short messages. In this paper we make analysis for the RSA public key protocol in the framework of membrane computing to develop a membrane model of RSA algorithm with performance improvement. The approach has been validated by developing a simulation program in C++ of RSA algorithm using the property of parallel computing of membrane environment to enhance encryption/ decryption time of the RSA algorithm. Comparing the encryption time results using normal RSA algorithm and the developed model, it was proved that the developed algorithm is faster than the normal one by 30% approximately.***

Key words: Membrane Computing , RSA , public key, encryption, decryption.

1. Introduction

Membrane computing is a branch of natural computing, the broad area of research concerned with computation taking place in nature and with human-designed computing inspired by nature. Besides systems biology that tries to understand biological organisms as networks of interactions, and synthetic biology that seeks to engineer and build artificial biological systems, another approach to understanding nature as computation is the research on computation in living cells [1], [2]. Membrane computing abstracts computing models from the architecture and the functioning of living cells, as well as from the organization of cells in tissues, organs (brain included) or other higher order structures such as colonies of cells (e.g., bacteria).The initial goal was to learn from cell biology something possibly useful to computer science, and the area quickly developed in this direction. Several classes of computing models were defined in this context, inspired from biological facts or motivated from mathematical or computer science points of view. A number of applications were reported in the last few years in several areas biology, Bio-medicine, linguistics, computer graphics, economics approximate optimization, cryptography, etc. The models investigated in membrane computing area are called P systems .

The main components of a P system are (i) the membrane structure, (ii) the mustiest of objects placed in the compartments of the membrane structure, and (iii) the rules for processing the objects and, the membranes. Thus, membrane computing can be defined as a framework for devising cell-like or tissue-like computing models which process multisets in compartments defined by means of membranes [3].The membrane structure consisting of several membranes arranged in a hierarchal structure [4].A membrane structure is represented by a Venn diagram (or a rooted tree) and is identified by a string of correctly matching parentheses, with a unique external pair of parentheses corresponding to the external membrane, called the skin. A membrane without any other membrane inside is said to be elementary. The following Example from [5] illustrates the situation: the membrane structure in Fig.1 is identified by the string.

$\mu = [_1 [_2 [_5]_5 [_6]_6]_2 [_3]_3 [_4 [_7 [_8]_8]_7]_4]_1$

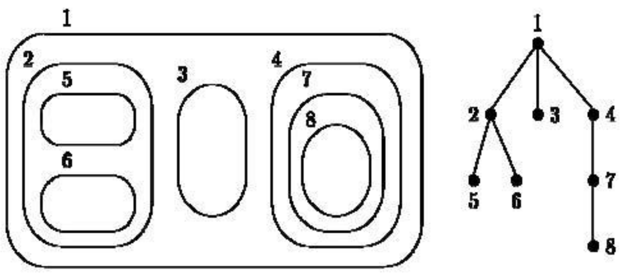


Fig.1.A membrane structure and its associated tree.

A P system with active membranes is a construct[6]

Π = (V, H, μ, w1, wm, R),

Where:

(i) M >= 1;
(ii) V is an alphabet;
(iii) H is a finite set of labels for membranes;
(iv) μ is a membrane structure, consisting of membranes, labeled (not necessarily in a one-to-one manner) with elements of H; all membranes in μ are supposed to be neutral;
(V) w1... wm are strings over V, describing the multi sets of objects placed in the regions of μ;
(vi) R is a finite set of developmental rules

(a) $[_h a \rightarrow v]_h^a$,

for $h \in H, \alpha \in \{+,-,0\}, a \in V, v \in V*$

(object evolution rules, associated with membranes and depending on the label and the charge of the membranes, but not directly involving the membranes, in the sense that the membranes are neither taking part to the application of these rules nor are they modified by them);

(b) $a[_h\,]_h^{\alpha_1} \rightarrow [_h b]_h^{\alpha_2}$

for $h \in H, \alpha_1, \alpha_2 \in \{+,-,0\}, a, b \in V$

(Communication rules; an object is introduced into the membrane, may be modified during this process; also, the polarization of the membrane can be modified, but not its label);

(c) $[_h a]_h^{\alpha_1} \rightarrow [_h\,]_h^{\alpha_2} b$,

for $h \in H, \alpha_1, \alpha_2 \in \{+,-,0\}, a, b \in V$

(Communication rules; an object is sent out of the membrane, maybe modified during this process; also, the polarization of the membrane can be modified, but not its label);

(d) $[_h a]_h^{\alpha} \rightarrow b$,

for $h \in H, \alpha \in \{+,-,0\}, a, b \in V$

(Dissolving rules; in reaction with an object, a membrane can be dissolved, leaving its entire object in the surrounding region, while the object specified in the rule can be modified);

(e) $[_h a]_h^{\alpha_1} \rightarrow [_h b]_h^{\alpha_2} [_h c]_h^{\alpha_3}$,

for $h \in H, \alpha_1, \alpha_2, \alpha_3 \in \{+,-,0\}, a, b, c \in V$

(Division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, may be of different polarizations; the object specified in the rule is replaced in the two new membranes by possibly new objects; all the other objects are copied into both resulting membranes);

(f) $[_{h_0} [_{h_1}\,]_{h_1}^{+} \ldots [_{h_k}\,]_{h_k}^{+} [_{h_{k+1}}\,]_{h_{k+1}}^{-} \ldots [_{h_n}\,]_{h_n}^{-}\,]_{h_0}^{\alpha_2} \rightarrow [_{h_0} [_{h_1}\,]_{h_1}^{\alpha_3} \ldots [_{h_k}\,]_{h_k}^{\alpha_3}\,]_{h_0}^{\alpha_5} [_{h_0} [_{h_{k+1}}\,]_{h_{k+1}}^{\alpha_4} \ldots [_{h_n}\,]_{h_n}^{\alpha_4}\,]_{h_0}^{\alpha_6}$,

for $n > k \geq 1, h_i \in H, 0 \leq i \leq n,$ and $\alpha_2, \ldots, \alpha_6 \in \{+,-,0\}$

(Division of non-elementary membranes; this is possible only if a membrane contains two immediately lower membranes of opposite polarization, + and —; the membranes of opposite polarizations are separated in the two new membranes, but their polarization can change; all membranes of opposite polarizations are always separated by applying this rule) ;

If the membrane labeled h0 contains other membranes than h1 ,..., hn specified above, then they must have neutral charges in order to make this rule applicable; these membranes are duplicated and then become part of the content of both copies of membrane h0;

P automata , which are symport/antiport P systems which accept strings: the sequence of objects (because we work with strings and symbol objects, we use interchangeably the terms “object" and “symbol") imported by the system from the environment during a halting computation is the string accepted by that computation (if several objects are brought in the system at the same time, then any permutation of them is considered as a substring of the accepted string; a variant, is to associate a symbol to each multiset and to build a string by such “marks" attached to the imported mustiest) [7].

A sequence of transitions constitutes a computation. A computation is successful if it halts; reaches a configuration where no rule can be applied. If we have an output region specified, then we count the objects present in the output regions in the halting configuration and this number is the result of computation [8].

2. RSA ALGORITHM

RSA scheme is a block cipher in which the plain text and cipher text are integers between 0 and n- 1 for some n [9].

2.1 Description of the Algorithm

RSA algorithm is specified as in [10], [11], [12], [13].

Key setup

To setup the key material, user Alice performs the following steps:

1- Each user generates a public/private key pair by selecting two large primes at random p, q.

2- Compute *N=p.q*

3- Compute $\phi(N) = (p-1)\cdot(q-1)$

4- Selecting at random the encryption key e Where $1 < e < \phi(N), \gcd(e, \phi(N)) = 1$

5- Solve the following equation to find decryption Key d where $e \cdot d = 1 \bmod \phi(N)$ and $0 \le d \le N$

6- Publish their public encryption key $KU = \{e, N\}$

7- keep secret private decryption key $KR = \{d, q, p\}$

Encryption

To send a message m < N to Alice, the sender Bob creates the cipher text c as follows:

$c \leftarrow m^e (\bmod N)$

Decryption

To decrypt the cipher text c. Alice computes $c \leftarrow m^d (\bmod N)$.

2.2 Modular Exponentiation

RSA uses fast exponential algorithm called "repeated square and multiply" the algorithm repeats the following process: dividing the exponent into 2, performing square and performing an extra multiplication if exponent is odd.

The algorithm is specified as in [14],[15].

Modular Exponentiation algorithm

INPUT x, y, n: integers with $x > 0, y \ge 0, n > 1$;

OUTPUT $x^y (\bmod n)$.

Mod_ exp (x, y, n)

1. If y = 0 return (1) ;

2. If y (mod 2) = 0 return (mod_exp(x^2(mod n), y ÷ 2, n);

3. Return (x.mod_exp ((x^2(mod n), y ÷ 2, n) ;

3. COMPACT ENCODING

The natural encoding is easy to understand and work with; however it has the disadvantage that for very large numbers the membrane systems should contain a very large number of objects, undesirable for practical reasons.

We discuss compact encodings where each object of a membrane system is represented in a more compact way, similarly to numbers in base 2 or higher. These compact encodings require notions and results from combinatorics over multisets.

To develop encoding and decoding algorithms, we start considering the number n in base b represented by using m objects. To develop encoding and decoding algorithms, we start considering the number n in base b represented by using m objects. As a first step we must determine m, the encoding length, then we look for the first (lowest) number represented using m objects. This number is $\sum_{i=0}^{m-1} N(b,i)$ the count of all numbers represented using less than m objects, and it is lower or equal to n. Thus we determine m from the following equation:

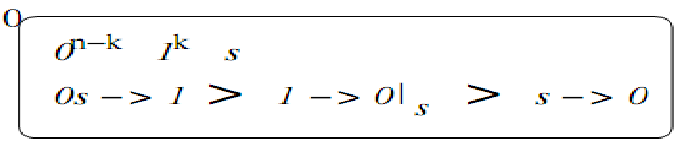
$$\sum_{i=0}^{m-1} N(b,i) - n = 0$$

TABLE1.
MOST COMPACT ENCODINGS (MCE) IN BASES 1,2,3.

Decimal	MCE1 Natural encoding	MCE_2	MCE_3
0	0^0	$0^0 1^0$	$0^0 1^0 2^0$
2	0^2	1	1
4	0^4	01	00
6	0^6	000	02
8	0^8	011	12
9	0^9	111	22
10	0^{10}	0000	000

3.1 Successor in MCE

Time complexity: O (1). The successor of a number ii this encoding is computed in the following manner: either we have an object 0 aid the rule 0s → 1, transforms this 0 in to an 1, or we have a number encoded using only objects 1 then aid the rule 1 → 0|s transforms all 1s into 0s; moreover the rule s → 0 produces an additional 0.

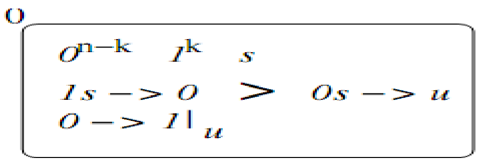


(a) Successor

3.2 Predecessor in MCE

Time complexity: O (1). The predecessor of a number is computed by turning an 1 in to a 0 by the rule 1s → 0 whenever we have objects 1; otherwise we consume one 0 by the rule 0s → u, aid transform all the other objects 0 into 1.

By rule 0 → 1|*u*.



(b) Predecessor

3.3 Multiplication MCE

<u>*Time complexity*</u>: O (n1· n2) = O (n2) if n1 = n2 =n. We implement multiplication in a similar manner to addition, coupling a predecessor with an adder. The idea is to provide the first number to a predecessor, and perform the addition iteratively until the predecessor reaches 0. The evolution is started by the predecessor working over the first number. The predecessor activates the adder by passing a communication token. The adder is modified to use an extra backup membrane which always contains the second number.

When the adder is triggered by the predecessor, it signals the backup membrane which supplies a fresh copy of the second number to the adder, and a new addition iteration is performed. At the end of the iteration.

At the end the adder sends out a token to the predecessor. The procedure is repeated until the predecessor reaches.

4. RSA ENCRYPTION IN MEMBRANE COMPUTING

The MCE multiplication can be used in calculating the encryption equation of RSA:

C ← m e (mod N)

Using Modular Exponentiation as dividing the exponent into 2, performing square and performing an extra multiplication with membrane computing system.

The evolution process of the P system for RSA encryption is shown in Fig. 2.

The P system model of RSA encryption can be described as follows.

Π = (V, H, μ, w1, wm, R),

M= 4

H= a b c

u=[1[2]2[3]3[4]4]1

V1= {[x,1,1]}

V2= {[x,2,1],[x,2,2]}

V3= {[x,3,1]}

V4= {[x,2,2],[x,1,1],[x,2,1]}

R1= {[x,1,1]}

R2= {([x,2,1]^2) % [x,2,2]}

R3= {BitShiftRight([x,3,1])}

R4= {([x,1,1]*[x,2,1]) % [x,2,2]}

A modular power function with three parallel threads as described in Fig. 3 is used in simulating the encryption equation using the property of parallelism of membrane computing. The pseudo-code of the function is described as follows:

```
Function modular_pow(base, exponent, modulus)
 result := 1
 while exponent > 0
 if (exponent mod 2 == 1):
 result := (result * base) mod modulus
 exponent := exponent >> 17
 base = (base * base) mod modulus
 return result
```

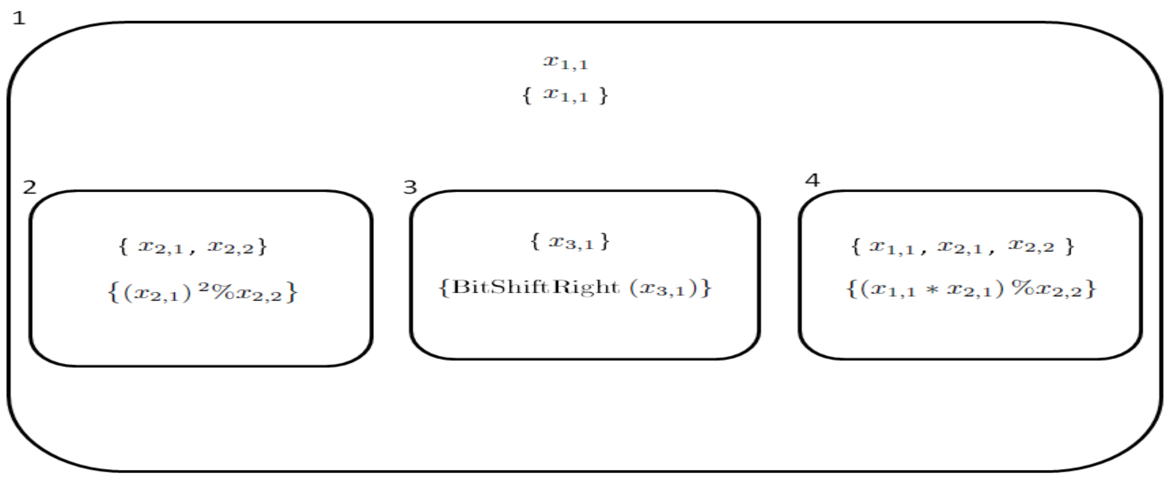
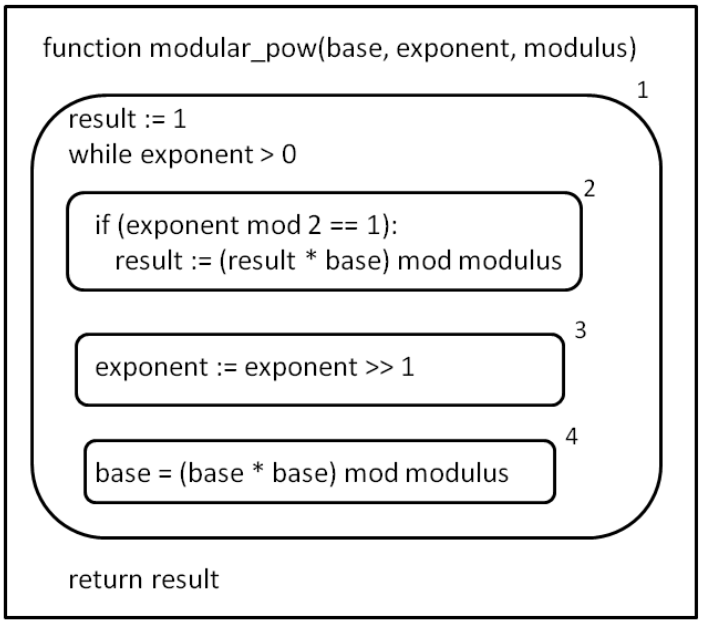

1
$x_{1,1}$
$\{ x_{1,1} \}$
2
$\{ x_{2,1}, x_{2,2} \}$
$\{(x_{2,1})^2 \% x_{2,2}\}$
3
$\{ x_{3,1} \}$
$\{\text{BitShiftRight}\ (x_{3,1})\}$
4
$\{ x_{1,1}, x_{2,1}, x_{2,2} \}$
$\{(x_{1,1} * x_{2,1}) \% x_{2,2}\}$

Fig.2. The evolution process of the P system for RSA encryption.



```
function modular_pow(base, exponent, modulus)
    1
    result := 1
    while exponent > 0
        2
        if (exponent mod 2 == 1):
            result := (result * base) mod modulus
        3
        exponent := exponent >> 1
        4
        base = (base * base) mod modulus
    return result
```

Fig. 3. Parallel threads description.

5. EXPREMINTAL RESULTS

The performance of the proposed algorithm of RSA using membrane computing is simulated using software written in C++(using visual c++)

We compared the performance to the performance of normal RSA on the same platform.

We used 3.2 giga hertz Intel processor with one giga byte ram. The results is shown in table 2.

The encryption time in seconds was recorded as a function of number of decimal digits of the two primary numbers products (N) in decimal digits.
The results is graphed as in Fig. 4.

TABLE 2

COMPARISON BETWEEN ENCRYPTION TIME OF NORMAL RSA AND MEMBRANE RSA

Number of Decimal Digits N(digits)	**Average Normal Time T_n (ms)**	**Average Membrane Time T_m (ms)**
400	5803	3993
500	11060	7581
600	19562	13432
700	30950	22261
800	45595	30504
900	65733	49094
1000	89092	66142
1100	122650	85723
1200	153061	110348
1300	195699	135689
1400	240490	169713

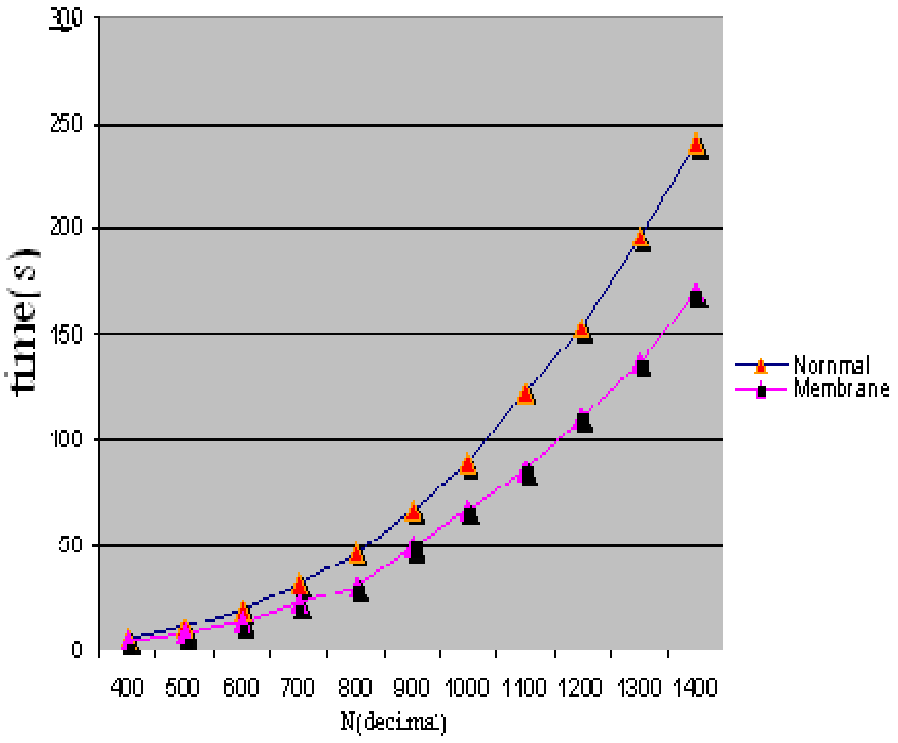


Fig.4 Comparison between encryption times of normal RSA and membrane RSA

6. CONCLOUSION AND FUTURE WORK

In this paper, we propose an analysis of RSA using the membrane computing environment. We make a comparison with normal RSA and our proposed system can improve the performance of the encryption time of the algorithm. the proposed algorithm also enhance the security of RSA by using large prime numbers and large exponents in addition to proving the ability of membrane computing in performing massive calculations with high performance, in relatively short time.

REFERENCES

[1] D. Endy. Foundations for engineering biology. Nature , pp 438:449 – 453, (2005).

[2] J. Dassow, G. P_AUN "Journal of Universal Computer Science, vol. 5, no. 2, pp 33-49", (1999).

[3] O. H. Ibarra, G. Paun “Membrane Computing: General View” The European Academy of Sciences, (2007).

[4] G. Paun, “Membrane Computing: An introduction” Springer Verlag, Berlin, ISBN: 3-540-42601-4, (2002)

[5] P. Sosik, Alfonso Rodriguez-Patton “Membrane computing and complexity theory: A characterization of PSPACE” Journal of Computer and System Sciences73, pp.137–152, (2007).

[6] H. Adorna, G. Paun, M. J. PEREZ- JIMENEZ " On Communication Complexity in Evolution- Communication P Systems "Romanian Journal Of Informartion " Volume 13, Number 2, pp.113-130, (2010).

[7] G. P˘aun, M. J. Perez Jimenez "Solving Problems in a distributed Way in Membrane Computing: DP systems" Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. V, No. 2, pp. 238-250, (2010).

[8] G.Paun “Application of Membrane Computing” Springer- Verlag, Berlin, ISBN: 3-540- 25017-4. (2002).

[9] W. Stallings" Cryptography and Network Security Principles and Practices", fifth Edition, ISBN:13 978-0-13-705632-3 (2011).

[10] C. Paar .jan pelzl "understanding cryptography "Spring Verlag ,Berlin , ISBN : 978 -3-642 – 04100 - 6 , (2010) .

[11] B. Esslinger "The cryptool script cryptography , mathematics and more", available at [http: //www.cryptool.org] (2010).

[12] B. Schneier "Applied cryptography Protocol , Algorithms and Code in C" ISBN:13 978-0-047-1117094 (1996), .

[13] J. Katz and y. lindel "introduction to modern cryptography", ISBN: 978-1-58488-551-1, (2008).

[14] W. mao "modern cryptography, theory and practice ", ISBN 0-13-066943-1, (2004).

[15] R. Oppliger "Contemporary Cryptography" Artech House , ISBN 978-1-60807-145-6, (2011).