I.J.Computer Network and Information Security, **2009, 1, 24-31**
Published Online October 2009 in MECS (http://www.mecs-press.org/)

Using P systems to Solve the Discrete Logarithm Problem used in Diffie-Hellman Key Exchange Protocol

Xiaojing Ma
Computer Department of Huazhong University of Science and Technology, Wuhan, Peoples Republic of China
lindahust@mial.hust.edu.cn

Zhitang Li and Hao Tu
Computer Department of Huazhong University of Science and Technology, Wuhan, Peoples Republic of ChinaEmail:
{leeying, tuhao}@mail.hust.edu.cn

***Abstract*—The discrete logarithm problem has been used as the basis of several cryptosystems, especially the Diffie-Hellman key exchange protocol. P systems are a cluster of distributed parallel computing devices in a biochemical type. This paper presents a P system with active membranes and strong priority to solve the discrete logarithm problem used in Diffie-Hellman key exchange protocol. To the best of our knowledge, it's the first time to solve the problem using P systems.**

***Index Terms*—P systems, Discrete Logarithm Problem, Diffie-Hellman key exchange protocol**

I. INTRODUCTION

Membrane computing (also called P systems) [1] was initiated in 1998 as a new class of distributed and parallel computing devices and much effort has been made on this research field [2]. In 2003, Thompson Institute for Scientific Information, ISI, has qualified the initial paper as "fast breaking" and the domain as "emergent research frontier in computer science". Membrane computing is inspired from the processes which take place in complex structure of a living cell. It has an attractive feature: parallelism.

The discrete logarithm problem plays an important role in cryptography. It has been used as the basis of several cryptosystems, such as the Diffie-Hellman key exchange protocol [3][9]. The problem used in Diffie-Hellman key exchange protocol can be formulated as: given a fixed primitive element *a* of a finite field *FP(p)* with a prime number *p*, and an integer number *b*, find the least positive integer *x* such that, $a^x \equiv b(\bmod p)$ $1 \le x, b \le p-1$.

However, no efficient algorithm for finding general discrete logarithms is known so far except for that in [4] which solves the problem on a quantum computer.

In this paper, we describe a P system with active membranes and strong priority to solve the discrete logarithm problem (DLP). To the best of our knowledge, it's the first time to solve DLP by membrane computing. The scheme we proposed in this paper can find the least positive index value *x* for the DLP in less than $\lfloor \log(p-1) \rfloor$ loops.

We organize the remainder of this paper as follows. Section 2 describes the backgrounds of this paper, including the discrete logarithm problem used in Diffie-Hellman key exchange protocol, and P systems with active membranes and strong priority. P systems solving the Discrete Logarithm Problem are presented in section 3. Section 4 briefly gives the conclusion.

Manuscript received January 1, 2008; revised June 1, 2008; accepted July 1, 2008.

II. BACKGROUNDS

A. *THE DISCRETE LOGARITHM PROBLEM USED IN DIFFIE-HELLMAN KEY EXCHANGE PROTOCOL*

The Diffie-Hellman key exchange protocol [3][9] is used to build a secure 2-party key distribution channel to handle the problem of secure communications.

The protocol makes use of the conjectured intractability of the Discrete Logarithm Problem. The problem can be defined as follows.

Problem. Name: Discrete Logarithm Problem.

Instance: a finite field *GF(p)* with a prime number *p*, a positive integer number *a* which is a fixed primitive element of *GF(p)*, and a positive integer number *b*. $1 \le a, b \le p-1$.

$$a^x \equiv b(\bmod p). \tag{1}$$

Output: the least positive integer *x*. $1 \le x \le p-1$.

$$x \equiv \log_a b \bmod p. \tag{2}$$

A fixed primitive element for a finite field *FP(p)* is the one whose powers build all the non-zero remainders modulo the prime number p. For example,

3^1 mod 7=3
3^2 mod 7=2
3^3 mod 7=6
3^4 mod 7 =4
3^5 mod 7 =5
3^6 mod 7 =1

The set of all the results is {1, 2, 3, 4, 5, 6} which is the same to the set of all the non-zero integers modulo 7. Thus, 3 is a fixed primitive element of the finite field *FP(7)*.

Calculation of (1) is easy while calculation of (2) is much more difficult.

We discuss the Diffie-Hellman key exchange protocol [10] briefly as follows:

Each user choose a integer number X_i from the set {1, 2, … , p-1}, and calculate

$$Y_i = a^{X_i} \bmod p. \tag{3}$$

Each user keeps X_i secret and makes Y_i public.

When the users U_i and U_j need to communicate securely, they can calculate the secret key to protect the communication according to (4).

$$K_{ij} = a^{X_i X_j} \bmod p. \tag{4}$$

The user U_i can get K_{ij} by computing

$$K_{ij} = Y_j^{X_i} \bmod p = (a^{X_j} \bmod p)^{X_i} = a^{X_i X_j} \bmod p. \tag{5}$$

U_j can get the secret communication key in a similar way. It is difficult for the other users to calculate the secret communication key of U_i and U_j, since the problem to calculate $\log_a Y_i \bmod p$ and $\log_a Y_j \bmod p$ are both Discrete Logarithm Problem.

B. P SYSTEMS

P systems are a cluster of distributed parallel computing devices in a biochemical type, inspired by the structure and functioning of living cells [1][5]. The basic model consists of a hierarchical structure composed by several membranes. The essential ingredient of P systems is membrane structure. A membrane structure consists of several membranes which are hierarchically embedded in a main membrane. Each membrane delimits one region containing some objects. And a P system has evolution rules for objects and input-output prescription. An elementary membrane is one which does not contain other membranes. The space out of the membrane is called environment.

Starting from an initial configuration, the objects evolve according to the evolution rules. The evolution rules are inspired by the reactions which happen in a living cell. All the rules are applied in maximally parallel manner. That is, all the rules which can be applied must be applied, and all the objects which can involve must be involved.

We give an example of a general P system.

$$\Pi = (V, \mu, \omega_1, ..., \omega_8, (R_1, \rho_1), (R_2, \rho_2), ..., (R_8, \rho_8), 8)$$
$$V = \{a, b, d, e, f, g, h, s\}$$
$$\mu = [_1[_2[_3[_7]_7[_8]_8]_3[_4]_4]_2[_5]_5[_6]_6]_1$$
$$\omega_3 = \{a^4, b^3, f\}$$
$$\omega_5 = \{h, s\}$$
$$\omega_7 = \{g, d\}$$
$$\omega_1 = \omega_2 = \omega_4 = \omega_6 = \omega_8 = \lambda$$
$$R_3 = \{a \to a(e, in_8), r_1 : bf \to f, r_2 : f \to \delta\}$$
$$\rho_3 = \{r_1 > r_2\}$$
$$R_1 = R_2 = R_4 = R_5 = R_6 = R_7 = R_8 = \varnothing$$
$$\rho_1 = \rho_2 = \rho_4 = \rho_5 = \rho_6 = \rho_7 = \rho_8 = \varnothing$$

(1) V is the alphabet of all the objects used in the P system.

(2) μ expresses the membrane structure of the P system. The membrane structure of this P system can be illustrated in Fig. 1, where the membrane structure can be expressed in the form of $[_1[_2[_3[_7]_7[_8]_8]_3[_4]_4]_2[_5]_5[_6]_6]_1$ and Fig. 2 too.

(3) ω_i, $1 \le i \le 8$ is strings from V^* representing the set of objects present in region i. In Fig. 1, a^4 indicates that there are four occurrences of object a in membrane with label "3".

(4) R_i, $1 \le i \le 8$ is the set of evolution rules associated with the region i. In R_3, there are a evolution rule $a \to a(e, in_8)$ which produce an objects a and sends an object e to the region 8 from one object a. Each paired objects b and f evolve to one object f under $bf \to f$. $f \to \delta$ makes the object f disappeared and dissolves the membrane.

(5) ρ_i $1 \le i \le 8$ is the set of priority relationship over R_i. $\rho_3 = \{r_1 > r_2\}$ means that the rule r_1: $bf \to f$ has priority over the rule r_2: $f \to \delta$. Firstly, the object f will evolve according to r_1. The rule r_2 will not be used unless r_1 cannot be used.

(6) The "8" in Π means the output region of the P system, and the result of the computation is collected in the output region. In this example, the output is the number of objects present in the region 8.

This P system can calculate 4^2. We explain the computation procedure as follows:

In the initial configuration, there are four occurrences of object a, three occurrences of object b and an object f.

At the first step, the rule $bf \to f$ "absorbs" one object b with the help of the object f, and $a \to a(e, in_8)$ produces four objects e and sends them to the region 8.

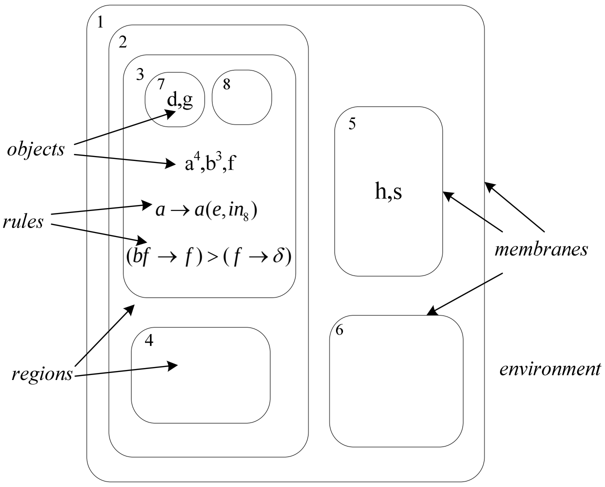


Figure 1. An example of membrane structure

The rule $f \to \delta$ is not used according to $(bf \to f) > (f \to \delta)$.

At the second step, $bf \to f$ "absorbs" one another object b with the help of the object f, and $a \to a(e, in_8)$ produces another four occurrences of object e. Now there are eight occurrences of object e in region 8.

The third step is similar to the second step, and there are twelve occurrences of object e in the region 8.

At the fourth step, $a \to a(e, in_8)$ produces another four objects e and sends them to the region 8. There are sixteen (4^2) occurrences of object e in the region 8. At the same time, there is no object b left in the region 3, so $bf \to f$ cannot be used. Thus, the rule $f \to \delta$ is used to dissolve the membrane with label "3". There is no rule in the region 2, so the computation is completed.

The hierarchical structure of membranes can also be presented by a rooted tree, as shown in Fig.2 which describes the membrane structure from Fig.1.

C. P SYSTEMS WITH ACTIVE MEMBRANES

There are many variants of P systems, one of which is P systems with active membranes [6][7]. P systems with active membranes are obtained by including rules for membrane division.

On the other hand, noting the fact that certain reactions are more active than others in a living cell, [1][8] consider a priority relationship on the set of rules in a given region.

Now, we describe the P systems with active membranes and priority. It is a construct expressed in (6).

$$\Pi = (V, H, \mu, \omega_1, ..., \omega_m, R, \rho, e) \qquad (6)$$

(1) V is a alphabet which contains all the alphabet used in the P system;

(2) H is a set of labels for membranes.

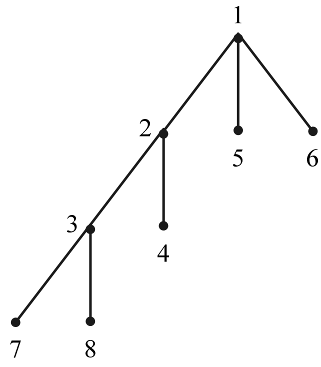


Figure 2. The tree describing the membrane structure from Fig.2.

(3) μ is the initial membrane structure consisting of m membranes.

(4) $\omega_1, ..., \omega_m$ are strings over V, describing the objects placed in the m regions of μ.

(5) R is a finite set of evolution rules, including rules for membrane division, of the following forms ($for\ h_0, h_1, h_2, \in \mathrm{H},\ \alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$, $\mathrm{x,y} \in \mathrm{V}^*$, $\mathrm{a,b,c} \in \mathrm{V}$):

i. $[_h \mathrm{x} \to y]_h^{\alpha}$

(Object evolution rules, all the object pairs that can form string x are replaced by the corresponding object pairs in string y.)

ii. $\mathrm{a}[_h\,]_h^{\alpha_1} \to [_h b]_h^{\alpha_2}$

(Communication rules, object a is sent into the membrane, simultaneously, a evolves to object b and the charge of the membrane can be modified.)

iii. $[_h \mathrm{a}]_h^{\alpha_1} \to [_h\,]_h^{\alpha_2} b$

(Communication rules, object a is sent out of the membrane, modified to b and the charge of the membrane can be modified too.)

iv. $[_h \mathrm{a}]_h^{\alpha} \to b$

(Membrane dissolving rules, the membrane is dissolved, a evolves to object b)

v. $[_h \mathrm{a}]_h^{\alpha_1} \to [_h b]_h^{\alpha_2} [_h c]_h^{\alpha_3}$

(Elementary membrane division rules, the membrane is divided into two membranes with the same label, maybe of different charge; object a is placed into two new membranes maybe replaced by new objects b and c; each new membrane has a copy of all the other objects.)

vi. $[_{h_0}[_{h_1}]_{h_1}^{+}[_{h_2}]_{h_2}^{-}]_{h_0}^{0} \to [_{h_0}[_{h_1}]_{h_1}^{0}]_{h_0}^{0}[_{h_0}[_{h_2}]_{h_2}^{0}]_{h_0}^{0}$

(Membrane division rules for non-elementary membranes, membrane h_0 has two membranes of opposite charge which are separated into two new membranes with label "h_0"; and they have neutral charge.)

(6) ρ is a set of priority relations for evolution rules. There are two types of priority relation: weak priority and strong priority. For rules with strong priority, if a rule R_1 has priority over a rule R_2, and R_1 can be applied, then R_2 can not be used, regardless of whether there are objects suitable for R_2.

For example, if R_1: $[_1 FF \rightarrow \lambda]_1^0$ and R_2: $[_1 FG \rightarrow H]_1^0$, the current multiset is *FFFG* in membrane "0" of neutral charge, then rule R_1 can be used, but R_2 cannot be used even there are objects *FG* suitable for R_2. On the contrary, R_2 can be used if the relationship between R_1 and R_2 is a weak priority.

(7) $e \in V$. The number of the copies of object *e* sent out to the environment is the computation result.

III. Solving the Discrete Logarithm Problem Using P System

We discuss in this Section on how to apply P systems to solve the DLP. Recall the formulation of DLP in section 1. We wish to solve the problem shown in (1).

We construct a P system with active membranes and strong priority.

$$\Pi = (V, H, \mu, \omega_1, \omega_2, \omega_3, R, \rho, X)$$

$$V = \{A, B, C, D, F, G, H, H', X, \lambda, Z, Z', Z''\} \cup \{k_i, k_i', k_i'', t_i, f_i \mid 1 \le i \le \lfloor \log(p-1) \rfloor\}$$

$$H = \{1, 2, 3\}$$

$$\mu = [_1[_2[_3]_3^0]_2^0]_1^0$$

$$\omega_1 = \{C, X, G^b\} \cup \{k_i'' \mid 1 \le i \le \lfloor \log(p-1) \rfloor\}$$

$$\omega_2 = \{A^a\}$$

$$\omega_3 = \varnothing$$

$R = \{$

$$R_0 : A[_3]_3^0 \rightarrow [_3 A']_3^0$$

$$R_1 : [_3 k_i'']_3^0 \rightarrow [_3 k_i']_3^0$$

$$R_2 : [_3 k_i']_3^0 \rightarrow [_3 k_i]_3^0 \quad \text{for } 1 \le i \le \lfloor \log(p-1) \rfloor$$

$$R_3 : [_3 A']_3^0 \rightarrow [_3 BD]_3^0$$

$$R_4 : [_3 BBC \rightarrow C]_3^0$$

$$R_5 : [_3 BC \rightarrow C]_3^0$$

$$R_6 : [_3 D]_3^0 \rightarrow [_3 DE]_3^0$$

$$R_7 : [_3 k_i]_3^0 \rightarrow [_3 t_i]_3^+ [_3 f_i]_3^- \quad \text{for } 1 \le i \le \lfloor \log(p-1) \rfloor$$

$$R_8 : [_3 Ct_i \rightarrow CZ'']_3^+ \quad \text{for } 1 \le i \le \lfloor \log(p-1) \rfloor$$

$$R_9 : [_3 X \rightarrow XX]_3^+$$

$$R_{10} : [_3 Cf_i \rightarrow CXZ'']_3^- \quad \text{for } 1 \le i \le \lfloor \log(p-1) \rfloor$$

$$R_{11} : [_3 X \rightarrow XX]_3^-$$

$$R_{12} : [_3 D \rightarrow \lambda]_3^+$$

$$R_{13} : [_3 D \rightarrow \lambda]_3^-$$

$$R_{14} : [_3 E \rightarrow F]_3^+$$

$$R_{15} : [_3 E \rightarrow F^a]_3^-$$

$$R_{16} : [_3 k_i \rightarrow k_i'']_3^+ \quad \text{for } 1 \le i \le \lfloor \log(p-1) \rfloor$$

$$R_{17} : [_3 k_i \rightarrow k_i'']_3^- \quad \text{for } 1 \le i \le \lfloor \log(p-1) \rfloor$$

$$R_{18} : [_2[_3]_3^+[_3]_3^-]_2^0 \rightarrow [_2[_3]_3^0]_2^0[_2[_3]_3^0]_2^0$$

$$R_{19} : [_3 F^p \rightarrow \lambda]_3^0$$

$$R_{20} : [_3 Z'' \rightarrow Z']_3^0$$

$$R_{21} : [_3 FG \rightarrow H]_3^0$$

$$R_{22} : [_3 Z' \rightarrow Z]_3^0$$

$$R_{23} : [_3 GZ \rightarrow G]_3^0$$

$$R_{24} : [_3 FZ \rightarrow F]_3^0$$

$$R_{25} : [_3 HZ]_3^0 \rightarrow \lambda$$

$$R_{26} : [_3 H \rightarrow GF]_3^0$$

$$R_{27} : [_3 F]_3^0 \rightarrow [_3]_3^0 A$$

$$R_{28} : [_2 X]_2^0 \rightarrow [_2]_2^0 X$$

$$R_{29} : [_1 X]_1^0 \rightarrow [_1]_1^0 X$$

$\}$

$$\rho = \{R_{4,5} > R_7, R_4 > R_5, R_{23,24} > R_{25}, R_{25} > R_{26}, R_{19} > R_{21} > R_{24} > R_{28}, R_{19\sim27} > R_1, R_2\}$$

For every prime number p $\Pi = (V, H, \mu, \omega_0, \omega_1, \omega_2, R, \rho)$ is a deterministic P system and the evolution generated by this system will stop after several steps. In the last step, the number of object *X* sent to the environment is the index *x*. We prove this by describing the process of evolution generated by $\Pi = (V, H, \mu, \omega_0, \omega_1, \omega_2, R, \rho)$.

The main idea of this P system to solve DLP is enumerating the value of index. Using membrane division, the system can check 2^n values of the index ($1\sim2^n$) in *n* loops. Each loop can be divided into three stages. We generate enough objects *F* at stage one; In stage two we get the remainder whose value is equal to the number of the object *F* left after using R_{21}. And in stage three the system checks if the remainder is equal to *b*. If it is equal to *b*, then the object *X* will be output to the environment, otherwise, we go to the next loop.

Let us take a closer look at how one step of the loop is performed from the initial configuration. The process with only those critical rules and key objects is briefly demonstrated in Fig. 2.

In the initial configuration, the number of the object *G* in membrane "3" corresponds to the modular *b*. The number of the object *X* corresponds to the index *x* whose value is one in the initial configuration. The number of the object *A* in membrane "2" corresponds to the element *a*. Moreover, the objects k_i ($1 \le i \le \lfloor \log(p-1) \rfloor$) in membrane "3" are used as counters, which control the membrane division.

According to the priority relationship $R_{4,5}>R_7$, the rules R_0~R_6 are used. The objects *A* of number *a* are sent into the membrane "3", and we get the objects *E* of the number a^2. Simultaneously, by using the rule R_7, the "electrically neutral" membrane "3" is divided into two separate copies of opposite charge. Further, the object k_i evolves to t_i and f_i with t_i placed into the copy of positive charge and f_i placed into the copy of negative charge respectively. The number of the object k_i is $\lfloor \log(p-1) \rfloor$ corresponding to the number of loop times.

Next, R_8~R_{18} are all applied in one step. R_{12} and R_{13} consume all the objects *D*. R_8~R_{11}, R_{14}, and R_{15} generate a^2 objects *F* and *2* objects *X* in membrane of label "1" with positive charge, correspondingly, a^3 objects *F* and *3* objects *X* in membrane of label "3" with negative charge. In the same step, by using R_{18}, the membrane "2" is divided into two copies of neutral charge.

Now the charge of membranes with label "3" turn to neutral again, and according to the priority relationship $R_{19}>R_{21}>R_{24}>R_{28}$, R_{19} is the next rule to be used. By using R_{19}, the number of the object *F* in those two membrane of label "3" are decreased to $a^2(mod\ p)$ and $a^3(mod\ p)$ respectively. Then, R_{21} is used to transform *FG* into *H*. If there is a membrane of label "3" which does not contain *F* or *G* after R_{21} has been applied, in other words, $a^2 \equiv b(mod\ p)$ or $a^3 \equiv b(mod\ p)$, this one will be dissolved by R_{25} and all the objects *X* in it will be sent to the environment by R_{28},R_{29}. Otherwise, the objects *F* and *G* would be changed back to *A* by R_{26} and R_{27}. Hence the rule R_0 can be used again, and it is the beginning of the next loop.

In this way, in stage one of the i-th ($1 \le i \le \lceil \log(p-1) \rceil$) round, there would be 2^i membranes of label "3" contained by a membrane of label "2". And in each membrane "2" the objects *A* of number $a^{j}(modp)(1<j<p-1)$ would be sent into the membrane of label "1" and generate $a^{2j}(modp)$ objects *F* (note that $(a^{j}(modp))^{2}(modp) \equiv a^{2j}(modp))$, 2j objects *X* in membrane "1" of positive charge, $a^{2j+1}(modp)$ objects *F*, 2j+1 objects *X* in membrane "1" of negative charge. Then, through the operation in stage two and stage three, we have checked the value of the index from 2^i to 2^{i+1}-1 in the i-th round. Finally, we exhaustively search all the possible values of

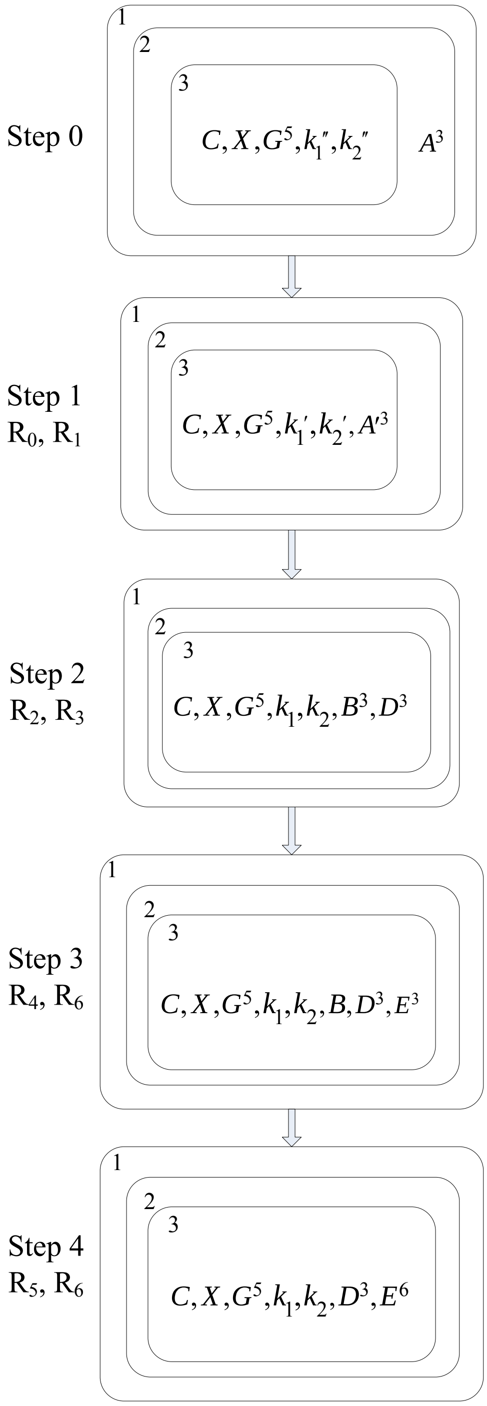


Figure 3. The evolution process of the P system solving the Discrete Logarithm Problem (step0~4)

x and can get the correct index value for the DLP in less than $\lfloor \log(p-1) \rfloor$ loops.

To explain the evolution process more clearly but, more importantly, to prove this P system, we give an example, to calculate: *x*=$\log_3 5$mod7. *a*=3, *b*=5, *p*=7.

We explain the evolution process as follows:

1. Step 0

   In the initial configuration, there are five occurrences of object *G* in the membrane of label

"3" and three occurrences of object *A* in the membrane of label "2".

2. Step 1

   According to the rules R_0 and R_1, all the three objects *A* will go into the membrane "3"and evolve to the object A', and the objects k_1'' and k_2'' evolve to k_1' and k_2'.

3. Step 2

   Under R_2 and R_3 every object A' in the membrane "1" evolve to two objects *B* and *D*, and the objects k_1' and k_2' evolve to k_1 and k_2.

4. Step 3

   Two objects *B* and one object *C* evolve to one object *C* under the evolution rule R_4. The three occurrences of object *D* produce three occurrences of object *E* according to R_6.

5. Step 4

   According to the priority relationship $R_{4,5}>R_7$, the last object *B* and the object *C* evolve to one object *C* under the evolution rule R_4. The three occurrences of object *D* produce another three occurrences of object *E* according to R_6.

The "Step 0-4" of the evolution process solving this problem is presented in Fig.3. "Step 0" shows the initial configuration.

6. Step 5

   There is no object *B* left, so R_7 can be used now. R_7 divides the membrane "3" into two separate copies of opposite charge. At the same time, R_6 produces another three occurrences of object *E*. Now there are nine occurrences of object *E* in both of the membranes with label "3".

7. Step 6

   R_8~R_{18} "absorb" the objects *D*, t_1, and f_1, produce one object Z'' in both of the membranes with label "3". What's more, the objects *X* and *E* evolve to two objects *X* and nine objects *F* in the membrane with positive charge, and three objects *X* and 27 objects *F* in the membrane with negative charge.

8. Step 7

   R_{19} consumes the objects *F*, and the numbers of the objects *F* left in the two membranes "3" are 2 and 6, which are the remainder modulo 7. R_{20} transforms object Z'' to Z'.

9. Step 8

   R_{21} transforms the paired objects *G* and *F* into object *H*. R_{22} transforms object Z' to Z.

10. Step 9

    Since there are object *G* left in membrane "3" with positive charge and object *F* left in membrane with negative charge, R_{23} and R_{24} delete the object *Z* according to the priority relationship $R_{23,24}>R_{25}$. R_{25} cannot be used to dissolve any membrane, and R_{26} transforms the object *H* back to object *F* and *G*, so the computation continues.

The "Step 5-9" of the evolution process solving this problem is presented in Fig.4.

11. Step 10

    In this step, all the objects *F* evolve to object *A* and be sent out of each membrane "3" by R_{27}. Thus, both membrane structures in membrane "1" are similar to the initial configuration, and the next loop begins.

12. Step 10~n

    These steps are similar to the step 0~7. Now we get four membrane structures composed of membranes with label "2" and "3" ($[_2[_3]_3]_2$). We focus on the second membrane structure in which the number of object *F* is the same to the number of object *G*.

13. Step n+1,

    R_{21} transforms the paired objects *G* and *F* into object *H*. R_{22} transforms object Z' to Z.

14. Step n+2

    There is no *F* or *G* left in the region "3", so the rule R_{23} and R_{24} cannot be applied. According to the priority relationship $R_{25}>R_{26}$, R_{25} is used in the second membrane structure. R_{25} dissolves the membrane, and release all the objects in region "3" to the region "2".

15. Step n+3, Step n+4

    R_{28} and R_{29} send *X* to the environment. The number of the objects *X* is the results: 5.

The "Step 10-n+4" of the evolution process solving this problem is presented in Fig.5. Fig.6 shows the whole evolution process of membrane structure clearly.We can conclude that the Discrete Algorithm Problem can be solved in at most two loops for *p*=7.

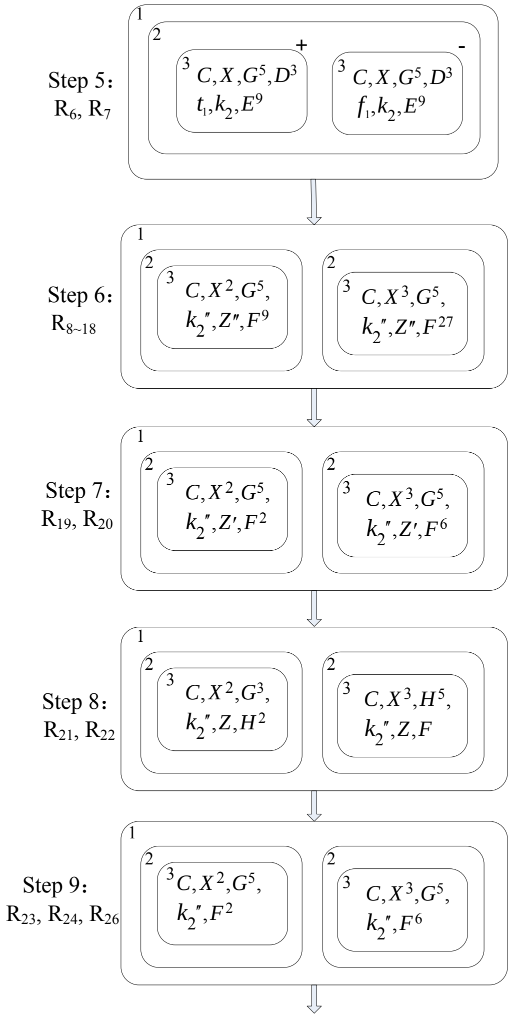


Figure 4. The evolution process of the P system solving the Discrete Logarithm Problem (step5~9)

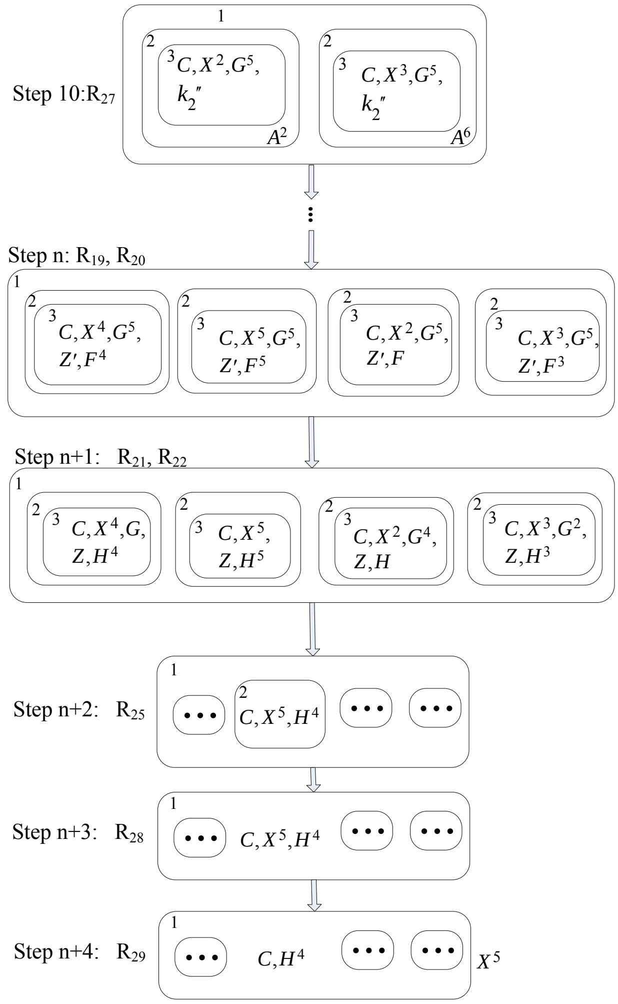


Figure 5. The evolution process of the P system solving the Discrete Logarithm Problem (step10~n+4)

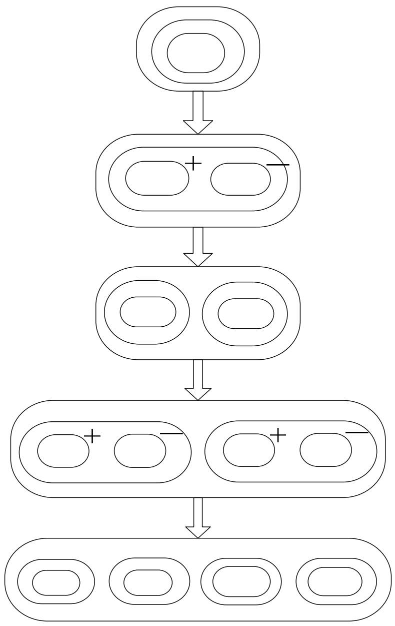


Figure 6. The evolution process of the P system solving the Discrete Logarithm Problem

IV. CONCLUSION

We present a P system with active membranes and strong priority to solve the Discrete Logarithm Problem used in Diffie-Hellman key exchange protocol for the first time. In future work, we will further improve the performance of this system.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grand No. 60573120, and by the National High Technology Research and Development Program of China(863 Program) under Grand No. 2007AA01Z420, and by the Key project sponsored by Natural Science Foundation of Hubei Province under Grand No. 2008CDA021.

REFERENCES

[1] G. Paun, "Computing with Membranes," *Journal of Computer and System Sciences*, vol. 61, 2000, pp. 108-143.
[2] "Home - The P Systems Webpage";

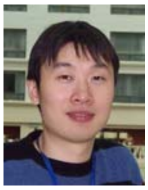
http://ppage.psystems.eu/.
[3] W. Diffie and M.E. Hellman, "Multiuser cryptographic techniques," *IEEE Transactions on Information Theory*, 1976.
[4] P.W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer" *SIAM J.Sci.Statist.Comput. 26 (1997) 1484*
[5] G. Ciobanu, G.P. aun, and M.J. Pérez-Jiménez, eds., *Applications of Membrane Computing*, Springer-Verlag, 2005.
[6] A. Obtulowicz, "On P systems with Active Membranes Solving Integer Factorizing Problem in a Polynomial Time," C. Calude et al., ed., Springer-Verlag, 2001, p. 267—286.
[7] G. Paun, "P systems with active membranes: attacking NP-complete problems," *Journal of Automata, Languages and Combinatorics*, vol. 6, 2001, pp. 75-90.
[8] G. Paun, Membrane Computing: An Introduction, Springer, 2002.
[9] Whit Diffie and Martin Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory,* IT-22(6):644-654, November 1976.
[10] A. Evans, Jr., W. Kantrowitz, and E. Weiss, A improved algorithm for computing algorithm in *GF(p)* and its cryptographic significance. *IEEE Transactions on Information Theory,* vol. 24, 2003, pp. 106-110.

Xiaojing Ma was born in Jinzhai, China in 1983, and received her B.E. degree from Wuhan University of Science and Technology, Wuhan, China, 2003. She is currently a PhD student in the Department of Computer Science and Technology at Huazhong University of Science and Technology. Her research interests include: cryptography, network security and P systems.

She was a Software Engineer of 263 On-Line Communications Inc. She has published several papers in the areas of multicast key management and P systems.

Zhitang Li was born in Jianli, China in 1951, and received his M.E. degree in Computer Architecture from Huazhong University of Science and Technology, Wuhan, China, 1987, and PhD degree in Computer Architecture from Huazhong University of Science and Technology, Wuhan, China, 1992. His research interests include computer architecture, network security and P2P networks.

He is currently the director of China Education and Research Network (CERNET) in Central China. He was a vice president of Department of Computer Science and Technology, Huazhong University of Science and Technology, China. He has published more than one hundred papers in the areas of computer architecture, network security and P2P networks.



Hao Tu was born in Wuhan, China in 1977, and received his B.S. and PhD degrees in computer science and technology from Huazhong University of Science and Technology, Wuhan, China, in 1999 and 2008. His research interests include: computer and network security, network measurement.

He is currently a lecturer in the Network Center, Huazhong University of Science and Technology.