

基于 DNA 计算自组装模型的 Diffie-Hellman 算法破译

陈智华

(华中科技大学控制科学与工程系 武汉 430074)

摘 要 DNA 自组装计算模型是近年来引人关注的计算模型,已有基于自组装模型的二进制加法、乘法以及有限域中的加法和乘法的讨论.文中利用 DNA 自组装模型设计的模乘系统,实现了素数 p 的本原根 g 连续乘方后模 p 的数的排列,从而可以在线性时间内求解离散对数,为破译 Diffie-Hellman 密钥交换算法提供了新的生物方法.该模乘系统使用了 $\Theta(p)$ 种自组装类型,组装的时间复杂度为 $\Theta(p-1)$.系统最后组装结果提取出报告链后,经过 PCR 和凝胶电泳读取离散对数结果.该模型扩展了 DNA 自组装计算模型的应用,为求取离散对数提供了新思路.

关键词 DNA 计算;DNA 自组装模型;离散对数;整数排序;PCR

中图法分类号 TP301

Breaking the Diffie-Hellman Key Exchange Algorithm in the Tile Assembly Model

CHEN Zhi-Hua

(Department of Control Science and Engineering, Huazhong University of Science & Technology, Wuhan 430074)

Abstract Recent experiments have demonstrated that the simple binary arithmetic and logical operations can be computed by the self-assembly DNA tiles. This paper shows how the tile assembly process can be used to break Diffie-Hellman key exchange. In order to achieve this, the author used tile systems to construct the integers permutation from 1 to $p-1$ based on the truth table of the following numbers modular p : $g \bmod p, g^2 \bmod p, \dots, g^{p-1} \bmod p$. The output of the systems can be read by the standard sequence-reading operation that uses a combination of PCR and gel electrophoresis. Through the processes of tile systems, we can pick out the discrete logarithm result which is the secret key in the Diffie-Hellman algorithm. So the key exchange by Diffie-Hellman algorithm is unsafe. The system can be carried out in $\Theta(p-1)$ assembly time with $\Theta(p)$ tiles. The methods extend the applications of self-assembly model.

Keywords DNA computing; DNA self-assembly model; discrete logarithm; integers permutation; PCR

1 Introduction

Adleman's pioneering study^[1-2] set the stage for the new field of bio-computing research. Using the vast parallelism to do the combinatorial search among a large number of possible solutions represented by DNA strands, Adleman has solved some combinatorial search problems such as the Hamiltonian path problem^[1], and demonstrated the fea-

sibility of molecular computing approaches to solve combinatorial problems. Bio-inspired computation could be used to solve complex mathematical problems^[3-4]. With the vast parallelism and high density storage, DNA computing is employed to solve many combinatorial optimization problems. Such an elegant idea inspired people to apply DNA computing to cryptographic systems in such fields: breaking cryptosystems, encryption and decryp-

tion, hiding message and authentication, which require an enormous amount of computation.

A number of proposals have been offered for breaking conventional cryptographic systems by DNA computing. For example, Boneh^[5] and Adleman^[6] have proposed DNA computing methods to break the Data Encryption Standard (DES). Chang and Guo demonstrated that factoring the product of two large prime numbers could be completed through biological operations using a molecular computer^[7]. Furthermore, this result indicated that the cryptosystems using public-key were perhaps insecure and also presented the clear evidence of the ability of DNA computing to perform complicated mathematical operations. Gehani presented some procedures for DNA-based cryptography based on OTP^[8]. Various modified DNA steganogram methods were proposed in order to improve the security. Leier presented two different cryptographic approaches based on DNA binary strands^[9]. The first approach hid information in DNA binary strands and the second approach designed a molecular checksum. Chen proposed a novel design of DNA-based molecular cryptography^[10]. They presented Carbon nano-tube based message transformation and DNA-based cryptosystem. Clelland have demonstrated an approach of steganogram by hiding secret messages encoded as DNA strands among a multitude of random DNAs^[11].

However, most of these proposals implemented computing processes by performing a series of biochemical reactions on a set of DNA molecules, which require human intervention at each step. Thus, the difficulties of such methods for DNA computing are that the large numbers of laboratory procedures and the time consuming, which grow with the size of problems.

2 Tile Self-Assemble Model

The tile self-assembly model, which is a formal model with the self-assembly molecules constrained on a square lattice, is an extension for the tiling theory of Wang tiles^[12] that include a specific growth mechanism based on the physics of molecular self-assembly. The first experimental demonstration of computation using DNA tile assembly was in [13]. It gave a two layer, linear assembly of TX tiles that performed a bit-wise cumulative XOR computation. Barish et al. demonstrated a tile assembly system implementation that could copy an input bit and count it in binary^[14]. Cook et al. used

the tile assembly model to implement arbitrary circuits^[15]. Similarly, Rothmund et al. have implemented a DNA tile system that could compute the XOR function and result in a Sierpinski triangle^[16]. Brun proposed theoretically the systems that could compute the sum and product of two numbers using the tile assembly model^[17]. He found that the adding and multiplying can be done using $\Theta(1)$ tiles and the two computations can be carried out in linear time in tile assembly models. He then combined those systems to create two new systems with more complex behaviour to factor numbers and solve subset sum problems^[18-19].

So DNA nanostructures provide a novel methodology for us to resolve all kinds of mathematical and combinatorial problems. Now the applications of DNA tiles are more and more universal.

3 Breaking the Diffie-Hellman Algorithm in DNA Self-Assemble Models

3.1 Diffie-Hellman Problem

In the simplest form of the Diffie-Hellman scheme, everyone is assumed to know a large prime number p and one of its primitive root g . If two entities A and B wish to agree on a secret random value, then

Step 1. A chooses a random element $x \in \{1 \cdots p\}$ and sends to B the value $g^x \bmod p$, and

Step 2. B chooses a random element $y \in \{1 \cdots p\}$ and sends to A the value $g^y \bmod p$.

The random value upon which A and B have agreed is $g^{xy} \bmod p$, which both can calculate:

A : A can calculate $g^{xy} \bmod p$ from x and $g^y \bmod p$ via $(g^y)^x \bmod p = g^{xy} \bmod p$, and

B : B can calculate $g^{xy} \bmod p$ from y and $g^x \bmod p$ via $(g^x)^y \bmod p = g^{xy} \bmod p$.

The Diffie-Hellman problem and its applications have been well-studied in the world of computational cryptography. It has even gained acceptance in applied cryptography, and is used for key-agreement in such widespread protocols as SSH and TLS. Here, a molecular self-assemble model is proposed to search the secret elements x and y , then the secret random value $g^{xy} \bmod p$ can be calculated.

3.2 Models of Algorithmic Self-Assembly

The abstract of tile self-assembly model, which provides a rigorous framework for analyzing algorithmic self-assembly, was originally proposed by Rothmund and Winfree^[20]. The model considers the assembly of rigid square objects or tiles. Elementary steps in the model are simple assemblies starting from a single seed tile and boosting

by the addition of single tiles, yet the formed structures may be surprisingly complex. Each unit of assembly is a square with glues of various types on each edge. The tile “floats” on a two dimensional plane and two tiles may stick when they collide if their abutting sides have compatible glues.

Formally, a tile over a set of binding domains Σ is a 4-tuple $\{\sigma_N, \sigma_E, \sigma_S, \sigma_W\} \in \Sigma^4$ indicating respectively the binding domains of the north, east, south and west sides. A position is an element of \mathbb{Z}^2 . The set of directions $D = \{N, E, S, W\}$ is a set of four functions from positions to positions, i. e. \mathbb{Z}^2 to \mathbb{Z}^2 such that for all positions (x, y) , $N(x, y) = (x, y+1)$, $E(x, y) = (x+1, y)$, $S(x, y) = (x, y-1)$, $W(x, y) = (x-1, y)$. The positions (x, y) and (x', y') are neighbors iff $\exists d \in D$ such that $d(x, y) = (x', y')$. For a tile t , for $d \in D$, we refer to $bd_d(t)$ as the binding domain of tile t on d 's side. According to this definition, tiles may not be rotated. A special tile $empty = \{null, null, null, null\}$, represents the absence of all other tiles.

The binding domains determine the interaction between tiles. A function $g: \Sigma^2$ to \mathbb{R} , where $null \in \Sigma$, is a strength function denoting the strength of the binding domains, which may be 0, 1, or 2 (respectively called *null*, *weak*, and *strong* bonds). If $\forall \sigma, \sigma' \in \Sigma$, then $g(\sigma, \sigma') = g(\sigma', \sigma)$ and $g(null, \sigma) = 0$.

Let T be a finite set of tile-types containing the empty tile. A configuration of T is a map from \mathbb{Z}^2 to T . For $t \in T$, $\Gamma_t^{(x,y)}$ is the configuration such that $\Gamma_t^{(x,y)}(i, j) = t$ iff $(i, j) = (x, y)$ and empty otherwise.

A tile system is a quadruple $\mathbb{S} = (T, S, g, \tau)$, where T, g are as above and S is a set of super tiles called seed supertiles which assembly begins with, τ is the temperature.

If A is a configuration, then in the system \mathbb{S} , a tile t can be attached to A at position (x, y) and produce a new configuration A' iff:

- 1) $(x, y) \notin A$,
- 2) $\sum_d \in Dg(bd_d(t), bd_{d-1}(A(d(x, y)))) \geq \tau$,
- 3) $\forall (u, v) \in \mathbb{Z}^2; (u, v) \neq (x, y) \Rightarrow A'(u, v) = A(u, v)$, and
- 4) $A'(x, y) = t$.

That is, a tile can be attached to a configuration only in empty positions and if the total strength of the appropriate binding domains on the tiles in neighboring positions meets or exceeds the temperature τ .

Given a tile system $\mathbb{S} = (T, S, g, \tau)$, a set of tiles Γ , and a seed configuration $S: \mathbb{Z}^2$ to Γ , if the

above conditions are satisfied, one may attach tiles of T to S . Configurations produced by repeated attachments of tiles from T are said to be produced by \mathbb{S} on S . If this process terminates, the configuration is called the final configuration. If all possible final configurations are identical, the tile system \mathbb{S} is said to produce a unique final configuration on S .

3.3 Modular Multiplication Using Self-Assemble

DNA self-assemble methods could be used to search the random element x when we know the value g, p , and g^x . We construct a DNA self-assemble system $\mathbb{S} = (T, S, g, \tau)$ to form the integers permutation from 1 to $p-1$ based on the truth table of the following numbers modular p : $g \bmod p, g^2 \bmod p, \dots, g^{p-1} \bmod p$.

A seed configuration S is designed, if conditions are satisfied, the tiles in the set T can be attached to S one by one until the final configurations produce. The final configurations can be extracted, and then according to the value g^x , and g^y , we extract the corresponding length DNA sequence by PCR, the length just maps to the corresponding random element x or y . The system use $\Theta(1)$ types to construct Γ and $\Theta(p)$ to construct T . The correctness of the system can be referred to the unique final configuration corollary^[21], which states that if all the tiles in a system have unique east-south or west-south binding domain pairs, that system always produces a unique final configuration on some seed configurations.

The set of Γ is designed as: $\Gamma = \{S = \langle s, null, null, null \rangle, E = \langle e, null, null, null \rangle, G = \langle g, null, null, null \rangle, SL = \langle null, 1, s, null \rangle, ER = \langle null, null, e, 1 \rangle\}$. The 5 kinds of tiles are used to calculate the modular multiplication. Fig. 1(a) shows a graphical representation of Γ , the value in the middle of each tile t represents that tile's $v(t)$ value. These tiles with one binding domains, are denoted as bd , and let $bdN(t)$ be the binding domain. The seed configuration for modular multiplication is shown in Fig. 1(b), in which the primitive root g acts as continuous multiplier. Two special tiles are denoted as the symbols ‘S’ (with a strong bond, and three null bonds) and ‘E’ (with a weak bond, and three null bonds) respectively, are called boundary tiles, which are used to denote the start and the end of input numbers. They set limits on the extent of the calculation or patterning, and will facilitate a modular approach to the process. The strength of the binding domains for side with double line is set 2. These are the forms we store all inputs in our system.

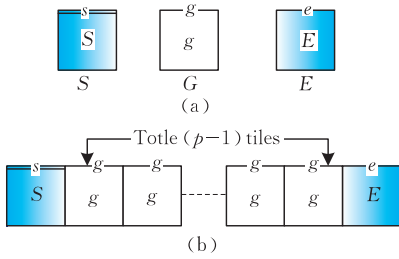
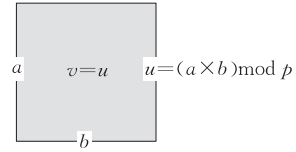
Fig. 1 g as the continuous multiplier

Fig. 2 shows the calculation tile's form used in the system. In the calculation tile, the west side stores the value of faciend, the south side is the multiplier, and result of $u = (a * b) \bmod p$ is stored in the east side. The strength of the binding domains for side with single line is set 1. The calculation tiles arrange figures in calculating results, which will be explained in the following sections.

Fig. 3(a) shows the two boundary tiles used in the tile system and each tile's name is written on

Fig. 2 a is the faciend, b is the multiplier, $i = (a \times b) \bmod p$ and the tile's value $v = u$

its top. The strength of the binding domains for side with double line is set 2. Fig. 3(b) shows the seed configuration S for the continuous multiplying the primitive root g modular p . Fig. 3(c) shows a sample of seed configuration which encodes the $y = 3^x \bmod 7$. The start tile (boundary tile) which is described as the starting point is on the left side. Note that only one tile may be attached to this configuration because $\tau = 2$. Fig. 3(d) gives the final configuration for the example, and the report strand gives us the permutation: $1 \times 3 \bmod 7, 1 \times 3^2 \bmod 7, \dots, 1 \times 3^6 \bmod 7$.

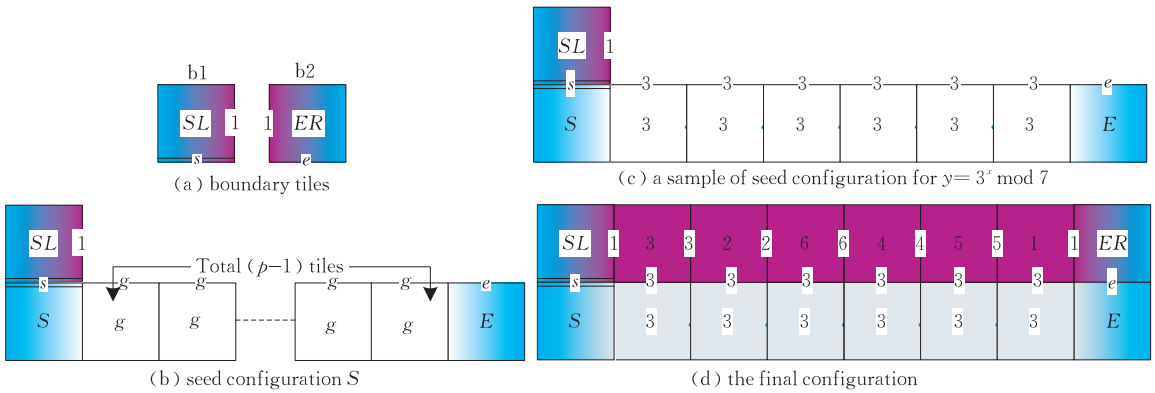


Fig. 3 Modular multiplication tile system

Lemma 1. Let $\Sigma = \{1, 2, \dots, p-1\}$ and T be the set of tiles defined by Fig. 1(b), let $g=1, \tau=2$ and S be a seed configuration. Let $y = g^x \bmod p$, here the prime number p determines the set's scale of T . Let $\mathbb{S} = (T, S, g, \tau)$, then, there must exist some $(x_0, y_0) \in \mathbb{Z}^2$ such that: $S(x_0 + 1, y_0 - 1) = E$, $S(x_0 - p, y_0 - 1) = S$, $S(x_0 - p, y_0) = SL$; for all $i \in \{1, 2, \dots, p-1\}$, $bd_N(S(x_0 - i, y_0 - 1)) = g$, and other positions $(x, y), (x, y) \notin S$. Then S produces a unique final configuration F based on S and can gives us the permutation of modular p multiplication, and the assembly time is $\Theta(p-1)$.

Proof. Consider the tile system $\Sigma = \{1, 2, \dots, p-1\}$. Let $\Gamma = \{S = \langle s, null, null, null \rangle, E = \langle e, null, null, null \rangle, G = \langle g, null, null, null \rangle, SL = \langle null, 1, s, null \rangle, ER = \langle null, null, e, 1 \rangle\}$.

Let the seed configuration $S: \mathbb{Z}^2 \rightarrow \Gamma$:

$$\begin{cases} S(1, -1) = E; \\ S(-p, -1) = S, S(-p, 0) = SL; \\ \forall i \in \{1, \dots, p-1\}, S(-i, -1) = g; \\ \text{For all other } (x, y) \in \mathbb{Z}^2, S(x, y) = \text{empty}. \end{cases}$$

It is clear that there is only one position where a tile may be attached to S , and after that tile attached there will also only be a single position where a tile may be attached. By induction, for $\forall t \in T$, the duple $\langle bd_S(t), bd_W(t) \rangle$ is unique, and because $\tau = 2$, it follows that \mathbb{S} produces a unique final configuration on S . Let that configuration be F .

$\forall t \in T$, the output is the product of its front tile's output and the primitive root g modular p , that is $\forall 1 \leq i \leq p-1, bd_E(F(-i, 0)) = (bd_E(F(-i-1, 0)) \times g) \bmod p$. When the right boundary tile $ER = \langle null, null, e, 1 \rangle$ is attached to the position $(1, 0)$, the self-assemble process will

terminate. It indicates that $\mathbb{S}=(T,S,g,\tau)$ can produce a unique final configuration on S and give the permutation of $1\times g \bmod p, 1\times g^2 \bmod p, \cdots, 1\times g^{p-1} \bmod p$. Because of $\tau=2$, only a tile with two neighbors may attach at any time in this system. And then no tile may be attached to S until its left neighbor has. Thus the assembly time to give us the permutation is $p-1$ steps. \square

3.4 Extracting the Report Strand

To implement our method of modular multiplication, we have to find suitable nucleotide sequences for encoding all the symbols used in our computation. And we have to ensure that these sequences (and their complementary sequences) are

sufficiently different from each-other. TAE tiles are used in our tile system. Fig. 4(a) shows the abstract structure. Each tile is made up of six DNA strands, and there are six sticky ends that can complement and ligate together in the Watson-Crick sense. Each tile can have from one to six sticky ends. Non-used sticky ends can be bent into loops so that they are not used as sticky ends.

Note that, a single strand of DNA (shown in Fig. 4(b)) passes through each tile assembled and thus contains all integers stored as tile’s value. This single strand of DNA is called reporter strand, and is also the result strand we need.

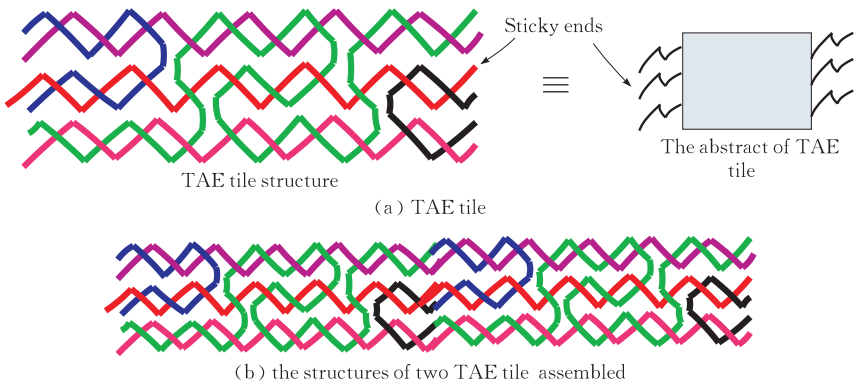


Fig. 4 Structures of two TAE tile assembled

At the end of self-assemble process, ligase is added to seal the joints (see Fig. 5) and then, the temperature of the chamber is increased to break up the tile structure into single strands. The result strand is then extracted by affinity purification using the complement of the ‘RES’ sequence. The ‘RES’ nucleotide sequence would contain the site of a restriction enzyme at the end, so that the ‘RES’ portion can be cut-off from the result strand by applying the appropriate restriction enzyme. As a result we will get an ssDNA containing all the encoded integers in some permutation. This strand can then, be used in further computations.

To output the result of the computation we would use a modification of the standard sequence-reading operation that uses a combination of PCR and gel electrophoresis. Using PCR we can obtain strands of different lengths representing the different exponent x in the result strand. According to the known values of the primitive root g and the result value y , we start PCR. The part of report strand “ $g-y$ ” can be amplified by PCR method by putting the “ g ” as a forward primer and “ y ” as a reverse primer. The needed string could be gel purified since its length would be different from the remaining sequence. The corresponding needed string “ $1\times g \bmod p, 1\times g^2 \bmod p, \cdots, y = 1\times g^x \bmod p$ ” can be extracted, see Fig. 6(a).

As we know, the security of the Diffie-Hellman key exchange protocol lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. Here is an example, key exchange is based on the use of the prime number $p=7$ and a primitive root of p , in this case $g=3$. The entities A and B select secret keys X_A and X_B , respectively. Each computes his or her public key:

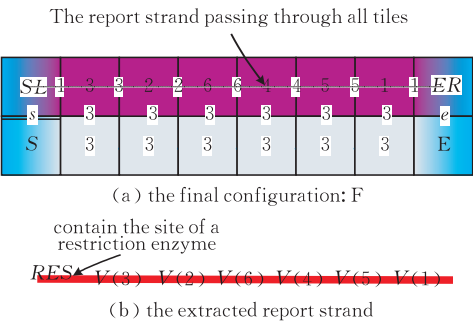


Fig. 5 Extracting the report strand

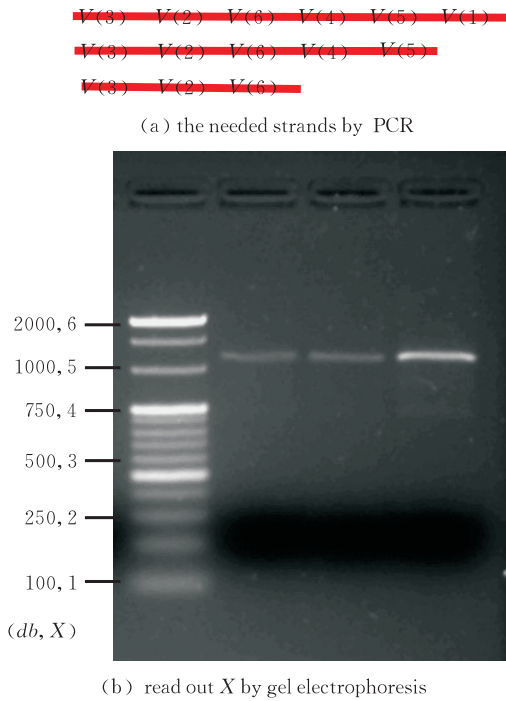


Fig. 6 Read out X by gel electrophoresis after PCR

$Y_A = 5$ and $Y_B = 6$. After they exchange public keys, each one can compute the common secret key: $K = (Y_B)^{X_A} \bmod p = (Y_A)^{X_B} \bmod p$. In fact, we could get the p , g , Y_A and Y_B easily. We use the DNA tile systems and biological technologies to get the X_A and X_B . In this case, $p = 7$, $g = 3$ and $Y_A = 5$, the part of report strand “3–5” can be amplified by PCR method by putting the “3” as a forward primer and “5” as a reverse primer. The needed string could be gel purified since its length would be different from the remaining sequence, see Fig. 6(b). Based on the relation of sequence length and x , we can read out the $X_A = 5$. Then $p = 7$, $g = 3$ and $Y_B = 6$, we repeat the same programs and read out the $X_B = 3$. So we can compute their common secret $K = 6^5 \bmod 7 = 5^3 \bmod 7 = 6$.

Through the tile system, PCR and gel electrophoresis, we can break the Diffie-Hellman algorithm.

4 Conclusion

DNA self assembly is expected to be useful in information security. Our method for breaking Diffie-Hellman algorithm extends the technique used by LaBean et al.^[22] for binary addition and XOR. The advantage of our methods is that once the initial strands are constructed, each permutation operation is processed very fast through the self-assembly and the output can be directly passed as input to another computation. The only time consu-

ming operation is the reading of the output.

The permutation of $1 \times g \bmod p$, $1 \times g^2 \bmod p, \dots, 1 \times g^{p-1} \bmod p$ can be achieved by the self-assembly system. The tile systems use $\Theta(p)$ input tiles and compute in $\Theta(p)$ steps. We extract the report strand in the final configuration, then use PCR and gel electrophoresis to read out the discrete logarithms. Furthermore, we can compute the common secret key used in Diffie-Hellman scheme.

The bottleneck of the self-assembly system is that the types of basic input tiles is linear correlation with the prime number p . The needed input tile types will increase with increasing p . When p is large enough, it is very difficult to find enough distinct types tile to carry out computation. If the improved tile system can break the bottleneck, the system would be more meaningful.

The nanotechnology and biological technology hold tremendous promise. But many technical hurdles will have to be overcome before complex algorithmic self-assembly can be developed into a practical commercial technology. If the molecular word can be controlled at will, it may be possible to achieve vastly better performance for information storage and information security.

References

- [1] Adleman L M. Molecular computation of solutions to combinatorial problems. *Science*, 1994, 266: 1021-1024
- [2] Adleman L M. Computing with DNA. *Scientific American*, 1998, 279(2): 54-61
- [3] Liu L Q, Liu G W, Xu J, Liu Y C. Solid phase based DNA solution of the coloring problem. *Progress in Natural Science*, 2004, 14(5): 104-107
- [4] Pan L Q, Carlos M V. Solving multidimensional 0-1 knapsack problem by P systems with input and active membranes. *Journal of Parallel and Distributed Computing*, 2005, 65(12): 1578-1584
- [5] Boneh D, Dunworth C, Lipton R J. Breaking DES using a molecular computer//*Proceedings of the 1st DIMACS Workshop on DNA Based Computers*. 1995: 37-65
- [6] Adleman L M, Rothmund P W K, Roweis S, Winfree E. On applying molecular computation to the data encryption standard//*Proceedings of the 2nd Annual Meeting on DNA Computers*. 1996: 10-12
- [7] Chang W L, Guo M Y, Michael S H. Fast parallel molecular algorithms for DNA-based computation: Factoring integers. *IEEE Transactions on Nanobioscience*, 2005, 4(2): 149-163
- [8] Gehani A, LaBean T H, Reif J H. DNA-based cryptography//*Proceedings of the 5th DIMACS Workshop on DNA-Based Computers*, MIT, 1999: 233
- [9] Leier A, Richter C, Banzhaf W, Rauhe H. Cryptography with DNA binary strands. *Biosystems*, 2000, 57: 13-22

- [10] Chen J. A DNA-based biomolecular cryptography design// Proceedings of the 2003 International Symposium on Circuits and Systems: ISCAS 2003, 2003: 822
- [11] Clelland C T, Risca V, Bancroft C. Hiding messages in DNA microdots. *Nature*, 1999, 399: 533-534
- [12] Wang H. Proving theorems by pattern recognition I. *Bell System Technical Journal*, 1961, 40: 1-42
- [13] Mao C, LaBean T H, Reif J H, Seeman N C. Logical computation using algorithmic self-assembly of DNA triple-cross-over molecules. *Nature*, 2000, 407: 493-496
- [14] Barish R, Rothmund P, Winfree E. Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano Letters*, 2005, 5(12): 2586-2592
- [15] Cook M, Rothmund P, Winfree E. Self-assembled circuit patterns//Proceedings of the 10th International Meeting on DNA Computer. Milan, Italy, 2004: 91-107
- [16] Rothmund P, Papadakis N, Winfree E. Algorithmic self-assembly of DNA sierpinski triangles. *PLoS Biology*, 2004, 2(12): 2041-2053
- [17] Brun Y. Arithmetic computation in the tile assembly model: Addition and multiplication. *Theoretical Computer Science*, 2006, 378(1): 17-31
- [18] Brun Y. Nondeterministic polynomial time factoring in the tile assembly model. *Theoretical Computer Science*, 2008, 395(1): 3-23
- [19] Brun Y. Solving NP-complete problems in the tile assembly model. *Theoretical Computer Science*, 2008, 395(1): 31-46
- [20] Rothmund P, Winfree E. The program-size complexity of self-assembled squares//Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC). Portland, 2001: 459-468
- [21] Rozenberg G, Spaink H. DNA computing by blocking. *Theoretical Computer Science*, 2003, 292(3): 653-665
- [22] LaBean T H, Winfree E, Reif J H. Experimental progress in computation by self-assembly of DNA tilings//Proceedings of the 5th International Meeting on DNA Based Computers (DNA5). MIT, Cambridge MA, 1999: 123-140



CHEN Zhi-Hua, born in 1976, Ph. D. candidate, lecturer. Her research interests include DNA-based information security and intelligent control algorithm.