

基于自组装 DNA 计算的 NTRU 密码系统破译方案

张勋才^{1),2)} 牛 莹²⁾ 崔光照²⁾ 王延峰²⁾

¹⁾(华中科技大学控制科学与工程系 武汉 430074)

²⁾(郑州轻工业学院电气信息工程学院 郑州 450002)

摘 要 自组装 DNA 计算在解决 NP 问题,尤其在破译密码系统方面,具有传统计算机无法比拟的优势.文中提出了一种用自组装 DNA 计算破译 NTRU 公钥密码系统的方法.针对 NTRU 密码系统的特点,采用 DNA 瓦片编码信息,借助于瓦片间的粘性末端进行自组装,给出了求解多项式卷积运算的实现方案.在此基础上,通过引入非确定性的指派瓦片,提出了一种破译 NTRU 系统的非确定性算法.通过创建数以亿计的参与计算的 DNA 瓦片,该算法可以并行地测试每个可能的密钥,以高概率地输出正确密钥.该方法最大的优点是充分利用了 DNA 瓦片具有的海量存储能力、生化反应的巨大并行性以及组装的自发有序性.理论分析表明,该方法具有一定的可行性.

关键词 自组装;DNA 瓦片;非确定性计算;NTRU;破译;公钥密码体制

中图法分类号 TP301

Breaking the NTRU Public Key Cryptosystem Using Self-Assembly of DNA Tilings

ZHANG Xun-Cai^{1),2)} NIU Ying²⁾ CUI Guang-Zhao²⁾ WANG Yan-Feng²⁾

¹⁾(Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074)

²⁾(College of Electrical and Electronic Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002)

Abstract Computation by self-assembly of DNA is an efficient method of executing parallel DNA computing where information is encoded in DNA tiles and a large number of tiles can be self-assembled via sticky end associations. This paper shows how the DNA self-assembly process can be used for breaking the NTRU public key cryptosystem. In order to achieve this, a method for implementing the cyclic convolution product of two polynomials using self-assembled DNA computing is expounded. Then, a non-deterministic algorithmic is provided to break efficiently the NTRU public key cryptosystem. By creating billions of billions of copies of the participating DNA tiles, the algorithmic will run in parallel on all possible private keys. The computation takes advantage of non-determinism, but theoretically, each of the non-deterministic paths is executed in parallel, yielding the solution in time polynomial in the size of the input, with high probability. It presents clear evidence of the ability of molecular computing to perform complicated mathematical operations.

Keywords self-assembly; DNA tile; non-deterministic computation; NTRU; break; public key cryptosystem

1 Introduction

Given its vast parallelism and high-density storage, DNA computing approaches are employed to solve many combinatorial optimization problems^[1]. However, most of these proposals implement the computation by performing a series of biochemical reactions on a set of DNA molecules, which require human intervention at each step. Thus, one difficulty with such methods for DNA computing is the number of laboratory procedures, each time consuming and error-prone, growing with the size of the problem.

Self-assembly is the ubiquitous process by which objects autonomously assemble into complexes. It is cost-effective, versatile, and facile^[2]. Simple self-assembly schemes are already widely used in chemical syntheses^[3-4]. The relation of DNA computation to self-assembling structures developed in the 1990s, largely through the theoretical and experimental work of Winfree^[5-6], Seeman^[7], Reif^[8], and Rozenberg^[9]. Computational systems based on self-assembly have been demonstrated in both 1-D arrangements called “string tiles”^[5,10], and 2-D lattices of DNA^[6]. Other stable forms of nucleic acids include Z-DNA, non-migrating Holliday junctions, and duplexes with triple crossovers or “pseudoknots”^[7,11-12]. For 2-D self-assembly, Winfree has proposed the Tile Assembly Model (TAM)^[6,13], and has shown that the TAM is Turing-universal by showing that a tile system can simulate Wang tiles^[14], which Robinson has shown to be universal^[15]. Barish et al. have demonstrated a DNA implementation of a tile system that copies an input and counts in binary^[16]. Cook et al. used the TAM to implement arbitrary circuits^[17]. Rothmund et al. have demonstrated a DNA implementation of a tile system to compute the XOR function^[18]. The model is also employed to solve many NP-complete problems or other problems^[19-22].

This paper demonstrates to break the NTRU cryptosystem using self-assembly of DNA tiles. Such attacks have already been used for example in [23]. However, the main disadvantage of the attack is that it is not fully supported by physical experiments and the Wang tiles that are used for the attack to form 3D self-assembly have yet not been made practically. Here, we give a non-deterministic algorithm based on the self-assembly of DNA tiles to attack NTRU cryptosystem. The probability of successfully breaking an instance can be made arbitrarily close to 1 by increasing the number of self-assembling components and seeds in the com-

putation.

2 Molecular Tiling and Self-Assembly Processes

DNA nanostructures provide a programmable methodology for bottom-up nanoscale construction of patterned structures, utilizing macromolecular building blocks called DNA tiles based on branched DNA. These tiles have sticky ends that match the sticky ends of other DNA tiles, facilitating further assembly into larger structures known as DNA tiling lattices.

2.1 DNA Tile

Computation and self-assembly are connected by the theory of tiling, of which Wang tiles^[14] are a prime example. The theory of Wang tiles showed that square tiles with colored edges could emulate a Turing machine, if they are allowed to assemble in a way that would cover the plane, according to additional rule that edges of the same color have to face each other. Branched DNA molecules provide a direct physical motivation for the TAM. There is also a logical equivalence between DNA sticky ends and Wang tile edges. Winfree and Seeman^[10] introduced “DNA tiles” to demonstrate the feasibility of computing through the self-assembly of DNA tiles. The most relevant constructs for DNA computing by self-assembly of DNA tiles include the double-crossover (DX)^[24] and triple-crossover (TX)^[25] complexes. The structures of TAO and TAE are given in Fig. 1 (Fig. 1(a) shows TAO tiles are formed by the annealing of four DNA single strands and they have four sticky ends at the four corners. The strand colored Black passing from the bottom left to the top right of the tile. When TAO tiles join together diagonally (see Fig. 1(b)), and we have a single DNA strand passing through the assembly from bottom left to top right (shown in bold). Fig. 1(c) shows the structure of the TAE resembles the TAO in that it contains an even number of helical half-turns between crossover points linking each pair of helices. These 3 reporter segments will be used for building up a long strand which records inputs and outputs for the entire assembly. Rotated TX tile in (d) has only two sticky ends. These two sticky ends are so designed that when these attach with the sticky ends of two neighboring TAE tiles, the tile gets rotated (by an angle close to $90^\circ \pm$) relative to the plane of the TAE tiles. This rotated tile can fit in the small gap left in the center when four TAE tiles attach together. Therefore, the TAE tiles and these rotated tiles can assemble to form a closely packed two-dimensional lattice structure as shown in Fig. 1(d).

Notice that when the rotated tile joins with two TAE tiles Fig. 1(e), a single DNA strand (shown in bold) passes through the middle of the three tiles). We abstract each DNA tile as a square with label. Each label indicates a particular kind of sticky end. Two sticky ends that can complement in the Watson-crick sense and ligating are represented by identical labels. Each tile can have from one to six labels. Non-labeled corners indicated the absence of sticky ends (See Fig. 1). We can use different sticky ends to denote different symbols or values. Therefore, tiles constructed using different sequences can store different values or symbols. The tile attached to this tile would have the complementary sticky end encoding the same value or symbol. In this way, we can pass information from one tile to its adjoining tile.

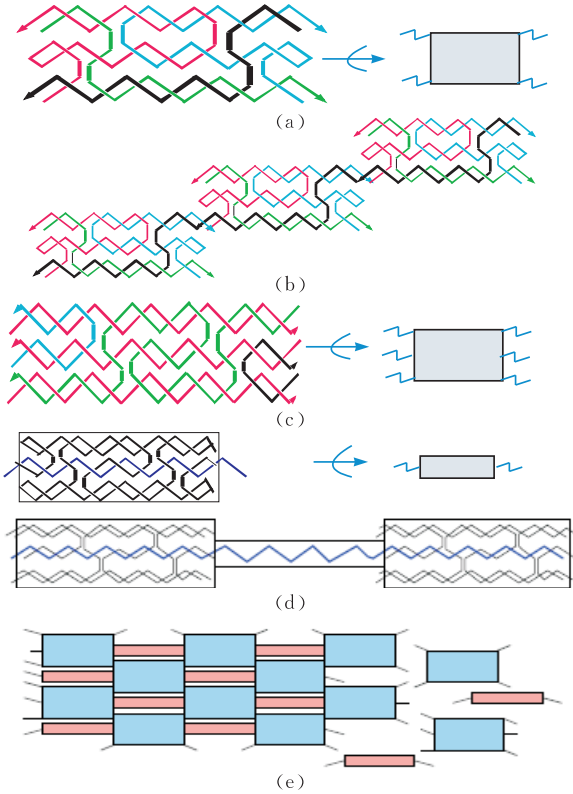


Fig. 1 TX tiles

2.2 Molecular Self-Assembly Processes

Molecular Self-assembly is the ubiquitous process by which simple objects autonomously assemble into intricate complexes. It has been suggested that intricate self-assembly processes will ultimately be used in circuit fabrication, nano robotics, DNA computation, and amorphous computing^[26-27].

A tiling is an arrangement of a few tiles that fit together perfectly in the infinite plane. Programming DNA self-assembly of tilings amounts to

the design of the pads of DNA tiles; ensuring that only the adjacent pads of neighboring tiles are complementary so that tiles assemble together as intended. The use of pads with complementary base sequences allows the neighbor relations of tiles in the final assembly to be intimately controlled; thus, the only large-scale superstructures formed during assembly are those that encode valid mappings of input to output. The progress of self-assembly of DNA tilings can be carried out by the following four steps^[28]: Mixing the input oligonucleotides to form the DNA tiles; Allowing the tiles to self-assemble into superstructures; Ligating strands that have been co-localized, then performing a single separation to identify the correct output.

3 Preliminaries

3.1 Introduction of NTRU

We give a brief description of the NTRU public key cryptosystem proposed in [28], which is based on embedding messages in a polynomial ring, \mathbb{R} . The ring is defined such that the multiplication operation of the polynomials is wrapped around the degree of the polynomial rather than expanding the degree of the polynomial. Further, each coefficient of the polynomial is an integer that is reduced modulo certain parameters, after every math operation. The notation for the ring is given as: $\mathbb{R} = \mathbb{Z}[X]/(X^n - 1)$.

The sets \mathcal{L}_f , \mathcal{L}_g , \mathcal{L}_r , and \mathcal{L}_m are subsets of \mathbb{R} . Two parameters p and q are chosen so that they are relatively prime and the private key $f \in \mathcal{L}_f$ is taken so that it is invertible modulo q . The inverse will be denoted by f_q , and using a random polynomial $g \in \mathcal{L}_g$, the public key is set to $h \equiv pf_q \otimes g \pmod{q}$.

Note that f and g are “randomly” chosen in \mathbb{R} , but have a specific amount of coefficients that are 0, +1 and -1. The polynomial f has d_f coefficients equal to +1, $(d_f - 1)$ coefficients equal to -1, and the rest equal to 0. The polynomial g has d_g coefficients equal to +1, d_g coefficients equal to -1, and the rest equal to 0.

To encrypt a message $m \in \mathcal{L}_m$, we choose a random $r \in \mathcal{L}_r$ and compute $e \equiv r \otimes h + m \pmod{q}$, e is our encrypted message.

To decrypt encrypted message e , we should have precomputed the polynomial f_p . In order to decrypt e , we compute: $a \equiv f \otimes e \pmod{q}$; where we choose the coefficients of a in the interval from $-q/2$ to $q/2$. Now, treating a as a polynomial

with integer coefficients, we recover the message by computing: $f_p \otimes a \pmod p$.

3.2 Cyclic Convolution Product Operations

According to section 3.1, we need to calculate the product of two polynomials. We know, the NTRU uses the ring of truncated polynomials $\mathbb{R} = \mathbb{Z}[X]/(X^n - 1)$ combined with the modular arithmetic. An element $F \in \mathbb{R}$ can be written as a polynomial or a vector:

$$F = \sum_{i=0}^{n-1} F_i x^i = [f_0, f_1, \dots, f_{n-1}].$$

We write \otimes to denote multiplication in \mathbb{R} . This star multiplication is given explicitly as a cyclic convolution product.

$$G = F \otimes H = \sum_{i=0}^{n-1} f_i * H * x^i \pmod{(X^n - 1)}.$$

We will denote $H * x^i \pmod{(X^n - 1)}$ by H_i . The basic operation here, is computing $H * x^i \pmod{(X^n - 1)}$ from a given number H ($H_0 = H$). Here, the addition of the coefficients is done modulo q . We have $H * x \pmod{(X^n - 1)} = (h_{n-2} * x^{n-1} + \dots +$

$h_1 * x^2 + h_0 * x) + h_{n-1}$, i. e., the coefficient of H have been rotating 1 bit left. Similarly $H_{i+1} = H_i * x \pmod{(X^n - 1)} = (h_{i-2}^i * x^{n-1} + \dots + h_1^i * x^2 + h_0^i * x) + h_{n-1}^i$, i. e., the coefficient of H_i have been rotating 1 bit left. Then, we can calculate hierarchically. Begin to calculate from the lowest layer, the first layer is H_0 , the second layer can compute by $H_0 * x \pmod{(X^n - 1)}$, and called H_1 , the rest may be deduced by analogy.

We represent the number H as a string of DNA tiles, with each tile storing one bit of H . The tile structure for H would be as shown in Fig. 2 and this consists of TAE tiles with rotated TX tiles in between. Each tile has two sticky ends on its top, the left one encoding the value of the bit h_i stored in the tile and the right one encoding the value 0 as the i th bit of the initial partial sum S_{-1} . Two special tiles encoding the symbols ‘SV’ and ‘RE’ respectively, are used to denote the start and the end of number.



Fig. 2 Input tiles for the number H

Then, we can construct such tiles which would assemble on top of this sequence of tiles to form another sequence of tiles representing the bits of $H_1 = H * x \pmod{X^n - 1}$. This is done using the tiles shown in Fig. 3. Here, $H(x)$, $G(w)$ and $Q(y, m)$ denote nucleotide sequences encoding the value x, w, y and m respectively. Notice the tiles (with six sticky ends) floating above the tile assembly for H . The sticky ends of these tiles are labeled by variables x, y, w and m of which x, y and m can take any value from $\{-1, 0, 1\}$. Only those tiles having compatible sticky ends could join together. Thus the tile which assembles on top of the two tiles storing h_{n-1} and h_{n-2} respectively, would have $x = h_{n-2}$ and the same y value as the ‘M’ tile adjacent to it (which is equal to h_{n-1}). This tile then computes the value which is the $(n-1)$ th bit of H_1 . This operation is shown in Fig. 4(b). Here the labels on the sticky ends represent the information carried by them. We denote by $[x]$ the nucleotide sequence encoding the value x and by $[\bar{x}]$ the complementary sequence.

Thus, the tile representing the $(i-1)$ th bit of H , passes the value of the bit to the i th tile in the upper layer through the sticky end in its top left corner. The i th bit tile in the upper layer gets the

value of the $(i-1)$ th bit, h_{i-1} as the value of the i th bit of H_1 . The value of y is passed to each tile from the tile on its left, through the sticky end in the middle. The value of y is initially obtained from the leftmost tile of the H sequence as h_{n-1} , and is passed to the upper layer through the tile labeled ‘M’.

On repeating this simple operation, we can get $H_2 = H * x^2 \pmod{(X^n - 1)}$, on top of H_1 and $H_3 = H_2 * x \pmod{(X^n - 1)}$ on top of H_2 and so on. Thus, we would have an assembly of multiple

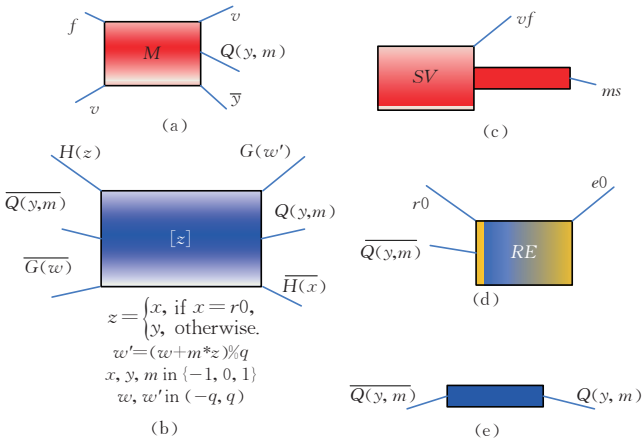


Fig. 3 DNA tiles required for computing cyclic convolution product

layers of DNA tiles, where the i th row would represent H_i . Now we have to perform an addition of all such H'_i s.

For this we maintain a partial sum, S_i at each layer, such that $S_{-1} = 0$ and $S_i = S_{i-1} + f_i * H_i$. The bits of S_{i-1} are passed to the i th layer through the bottom right sticky end of the tiles, but, in order to calculate the corresponding bits of S_i , the

value of f_i should be available. The solution is to construct a diagonal tile assembly representing the bits of F (as shown in Fig. 4(c)), and allow this to join to the left side of our earlier assembly such that the tile storing the i th bit of F is attached to the i th row. The value of f_i would be passed along the row, through the sticky ends in the middle, along with the y value.

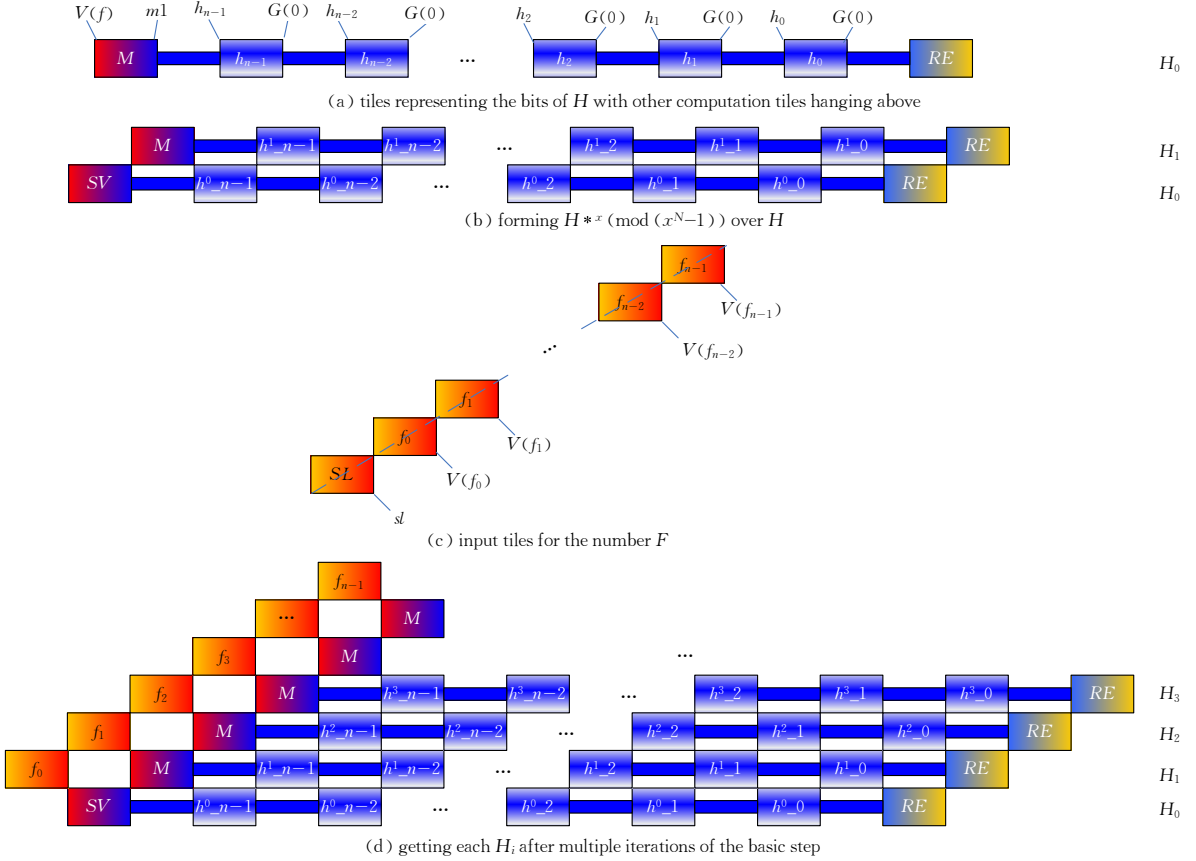


Fig. 4 Implementing the basic step of computing $H * x \pmod{(x^n - 1)}$ from H

The tile shown in Fig. 3(b) is the computation tile which computes the bits of H_i as explained before. But, it also computes the partial sum bits as w_0 , using the value of partial sum bit w that it receives from the lower layer through the sticky end at its bottom left. The value of y and $m = f_i$ are passed from one tile to another in the same row, using the bridge tiles shown in Fig. 3(e). The value of f_i is initially passed to the layer representing H_i , through the tile labeled ‘ M ’ shown in Fig. 3 (a), to which the i th bit tile of F would be attached. Finally, the tiles shown in Fig. 3(d) are used for adding a ‘ RE ’ at the end, while computing H_i from H_{i-1} .

At this stage, the sticky ends of the final layer tiles contain the result of the multiplication. In order to output the result in the form of a DNA

strand, we require some output tiles of the form shown in Fig. 5.

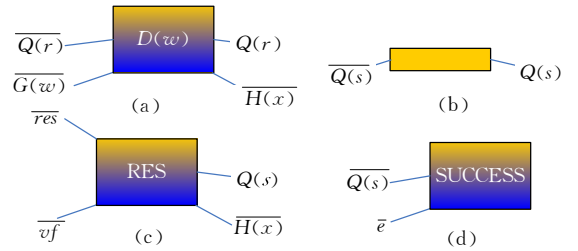


Fig. 5 Output tiles: These tiles are used to store the result of the computation

4 Breaking the NTRU

Perhaps of greatest interest is the harnessing of the massive parallelism of DNA information storage to break cryptosystem. In [29], the au-

thors have studied different possible attacks on the NTRU cryptosystem. We know, giving all parameters (N, p, q) , the sets $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$, and a public key h , an attacker can recover the private key by trying all possible $f \in \mathcal{L}_f$ and testing if $f \otimes h \pmod q$ has small entries, or by trying all $g \in \mathcal{L}_g$ and testing if $g \otimes h^{-1} \pmod q$ has small entries. It is the huge (exponential) number of different possible f that makes the problem difficult on a conventional computer. From this section, we will attempt to break the NTRU public key cryptosystem by using non-deterministic algorithms.

4.1 Non-Deterministic Algorithm

Here we propose a way to perform the computing procedure above on molecular substrate using 2D DNA self-assembly. By creating billions of copies of the participating DNA tiles, we expect that the procedure will run in parallel on all possible f . The f will be created dynamically as part of the assembly. In effect, that will make the computation time linear in the size of the problem, while pushing the exponential dimension of the problem into the large number of DNA assemblies, and thus into space occupied by the DNA molecules. If there is a satisfying assignment, we expect that at least one of these parallel computations will discover it.

Our design is described at the algorithmic level. It attempts to simulate a non-deterministic algorithm for the problem. Non-determinism implies that at some steps the algorithm makes a non-deterministic choice (as if some oracle could tell the algorithm what the right choice is). In our case, this is simulated by an exponential number of DNA assemblies, expecting to cover all possible choices. The algorithm is given below. Notice that step 4 is the non-deterministic step.

```
Nondeterministic  $(F, H)$ 
{
  1.  $G_0 = 0$ ;
  2.  $H(-1) = H(0) = H$ ;
  3. for  $(i = 1, 2, \dots, n-1)$  {
  4. Assign a value  $(-1, 0 \text{ or } 1)$  to variable  $f_i$ 
  5. Computing  $H(i) = G_0 + f_i * H(i-1) * x$ 
  6. Computing  $G_0 = G_0 + f_i * H(i-1) * x$ 
  7. If  $G_0$  are satisfied
  8. Then return and output  $G_0$ .
  9. Else return failure.
}
```

The idea can be carried out almost directly by assembly of DNA tiles. The input is coded as a concatenation of tiles representing the H and F

(now, the values of F are null). This input structure is reproduced in billions and is mixed with a DNA solution that already contains tiles from a fixed library to be described shortly. The appropriate tiles will self-assemble on this input layer. Values are assigned to the variables f_i in a random manner. Use the massive parallelism of DNA to compute the convolution product of H with all the possible F . Each assembly is testing one possible assignment to the variables F . The input DNA structure and a successful assembly computation are shown in Fig. 6. Note that, a single strand of DNA (shown in black), passes through each tile of H and thus contains an encoding $D(h_i)$ of each bit h_i of H . This single strand of DNA can be used to represent the number H . Reading of the operation result is done by the reporter strand method. Among all the results, the one consisting only of 0, 1 or -1 is the private key. From Fig. 4, we can find out, when calculating the product, the value of F is given definitely. Here, the value of F is appointed at random. So, the ‘ M ’ tile is replaced with ‘ V ’ tile. The structure of ‘ V ’ tile is described in Fig. 6(a). The value of ‘ V ’ tile will be appointed at random in the course of assembly.

Let us now look at an example computation using our method. For simplicity we will only consider $(n, p, q, H) = (5, 3, 16, -x^4 - 2x^3 + 2x + 1)$. We take $F = x^4 + x - 1$, then, $G = F \otimes H = x^3 - x$. Fig. 6(b) shows the initial stage of the computation, there is only one slot where a tile can bind. This tile will be an assignment tile (‘ V ’ tile). Now, we add the other tiles shown in Fig. 3 and ‘ V ’ tile required for computation and allow them to anneal together. Fig. 6(c) shows the pre-final stage of the computation when the last layer representing H_4 has been computed. At this stage, the sticky ends of the final layer tiles contain the result of the multiplication. In order to output the result in the form of a DNA strand, we add some output tiles of the form shown in Fig. 5. On adding these tiles, and allowing them to anneal, we get the final tile assembly as shown in Fig. 6(d) (the number on the bottom right corner of the tile denotes the assemblage order). On adding Ligase to seal the bonds, we will have a single strand of DNA passing through the tiles in the final output layer, which encodes the result of the computation. This single strand begins with the unique nucleotide sequence labeled ‘ res ’. Therefore, the F is satisfied with this assignment if and only if the success symbol appears in the result DNA strand.

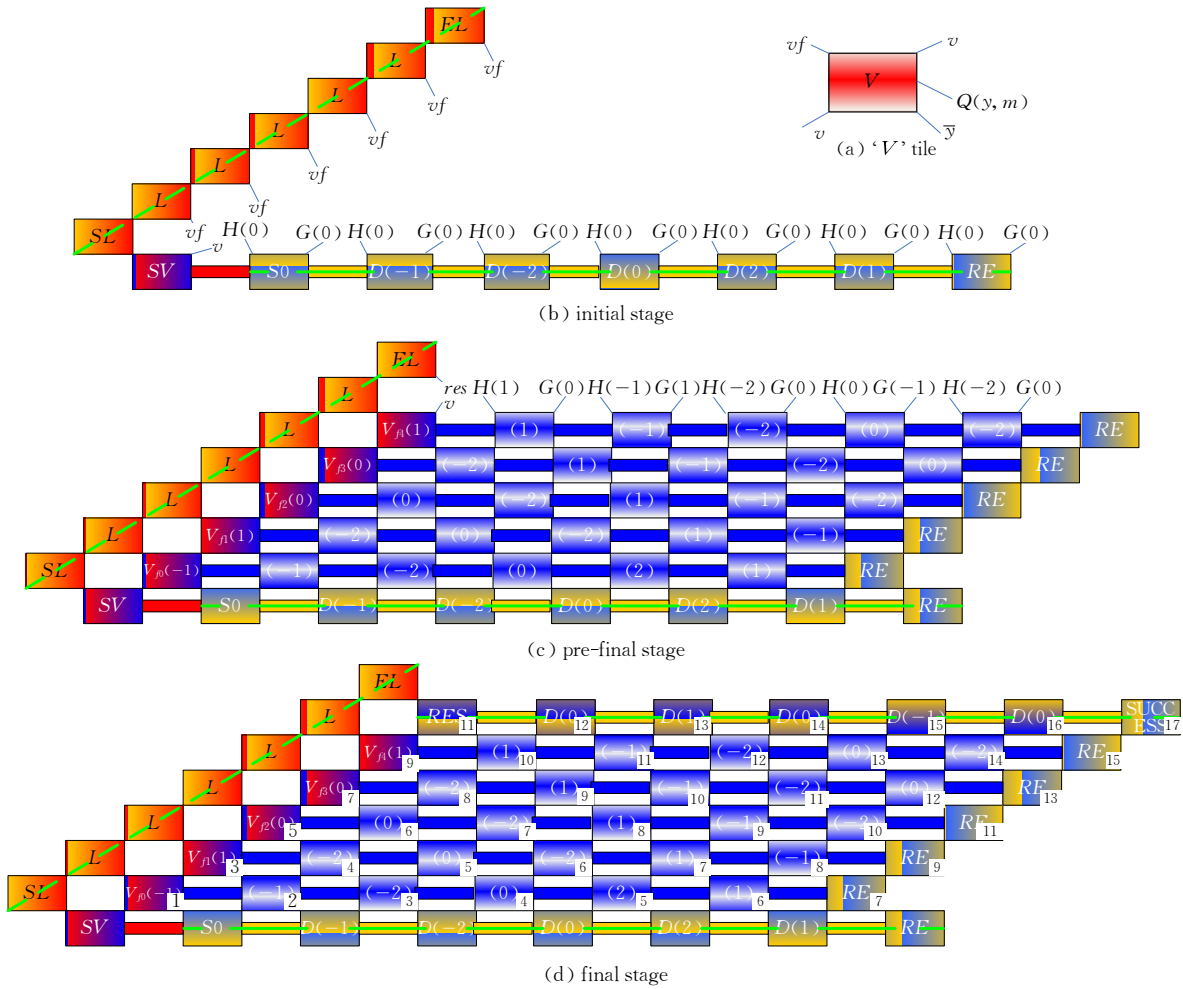


Fig. 6 Input tiles and a successful assembly computation

The actual implementation detail is not discussed here since they fall outside of the scope of this paper. However, we believe that we make no arbitrary hypotheses. In fact, our work is based on the assumptions and achievements that come with DNA tiling computation in general.

4.2 Complexity of System

The complexity of the design is considered in terms of computation time and computation space. It is obvious from the example given that the computation time T is equal to depth (diagonal) of the assembly. In fact, it is:

$$T = (n + 3) + (n + 1) + 3 = 2n + 7 = O(n).$$

It follows directly from the assembly time corollary in [19].

The space S taken for each assembly is the area of the assembly:

$$S = (n + 3) * (n + 2) + 1 = O(n^2).$$

5 Discussion and Conclusion

As the massive parallelism of DNA computing, it has the unexampled dominance in solving

difficult problems, especially for NP complete problems, than silicon computer. DNA self-assembly is expected to be useful in various applications. In this study, we presented an algorithmic design for breaking the NTRU public key cryptosystem using DNA self-assembly. It extends the technique used by Rana Barua et al. for finite field arithmetic. This method mapping variables of the problem to DNA tiles, through encoding the sticky ends of the tiles properly, molecule can hybridize in term of Watson-Crick complement. The advantage of our method is that once the initial strands are constructed, each operation can compute fast parallel through the process of DNA self-assembly without any participation of manpower and the output of one computation can be directly passed as input to another computation.

A final detail that is crucial for the success of the algorithm. The concentrations of assignment tiles corresponding to variable f_i , tiles for $f_i \in \{-1, 0, 1\}$, have to be equal so that there is equal chance of assigning each value. If this is not the

case, there might be assignments that will never be explored because of this discrimination in assigning values.

The attack is also limited by the size of q , which determines the number of different tiles. A typical value for q is 64 or 168, which means that more than 4000 different tile classes are needed. Another limitation of the algorithm, which is common for most DNA computation, comes from the fact that the exponential dimension of the problem has been pushed into physical space (volume) occupied by the DNA molecules. This will eventually become a restrictive factor. The number of DNA tiles to be used in the computation cannot be expected to be much more than the Avogadro number (about 10^{28}). Therefore, this kind of attack is roughly limited to 2^{80} different possibilities for g .

The field of DNA self-assembly holds tremendous promise. Our ultimate goal is to test the design(s) experimentally. If the molecular and supramolecular word can be controlled at will, then it is possible that in the future molecular computers will be the clear choice for performing massively parallel computations. We hope that this paper helps to demonstrate that molecular computing is a technology worth pursuing.

References

- [1] Adleman L M. Molecular computation of solutions to combinatorial problems. *Science*, 1994, 266: 1021-1024
- [2] Lehn J M. Supramolecular chemistry. *Science*, 1993, 260: 1762-1763
- [3] Adleman L M, Cheng Q, Goel A, Huang M, Kempe D, Moisset P, Rothmund P. Combinatorial optimization problems in self-assembly//*Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. Montreal, Canada, 2002: 23-32
- [4] Abelson H, Allen D, Coore D, Hanson C, Homsy G, Knight T, Nagpal R, Rauch E, Sussman G, Weiss R. Amorphous computing. *Communications of the ACM*, 2002, 43(5): 74-82
- [5] Winfree E, Eng T, Rozenberg G. String tile models for DNA computing by self-assembly//*Proceedings of the 6th International Workshop on DNA-Based Computers*. Leiden, The Netherlands, 2000: 65-84
- [6] Winfree E. Algorithmic self-assembly of DNA [Ph. D. dissertation]. California Institute of Technology, Pasadena CA, 1998
- [7] Seeman N C. DNA nanotechnology: Novel DNA constructions. *Annual Review of Biophysics and Biomolecular Structure*, 1998, 27: 225-248
- [8] Reif J H. Computing: Successes and challenges. *Science*, 2002, 296: 478-479
- [9] Rozenberg G, Spaink H. DNA computing by blocking. *Theoretical Computer Science*, 2003, 292: 653-665
- [10] Winfree E, Liu F, Wenzler L A, Seeman N C. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 1998, 394: 539-544
- [11] Mao C, Sun W, Seeman N C. Designed two-dimensional DNA holliday junction arrays visualized by atomic force microscopy. *Journal of the American Chemical Society*, 1999, 121(23): 5437-5443
- [12] 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
- [13] Chen H L, Goel A. Error free self-assembly with error prone tiles//*Proceedings of the 10th International Meeting on DNA Based Computers*. Milan, Italy, 2004
- [14] Wang H. Proving theorems by pattern recognition I. *Bell System Technical Journal*, 1961, 40: 1-42
- [15] Robinson R M. Undecidability and nonperiodicity for tilings of the plane. *Inventiones Mathematicae*, 1971, 12(3): 177-209
- [16] Barish R, Rothmund P, Winfree E. Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano Letters*, 2005, 5(12): 2586-2592
- [17] Cook M, Rothmund P, Winfree E. Self-assembled circuit patterns//*Proceedings of the 10th International Meeting on DNA Based Computer*. Milan, Italy, 2004: 91-107
- [18] Rothmund P, Papadakis N, Winfree E. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2004, 2(12): 2041-2053.
- [19] Brun Y. Arithmetic computation in the tile assembly model: Addition and multiplication. *Theoretical Computer Science*, 2006, 378: 17-31
- [20] Brun Y. Nondeterministic polynomial time factoring in the tile assembly model. *Theoretical Computer Science*, 2008, 395(1): 3-23
- [21] Brun Y. Solving NP-complete problems in the tile assembly model. *Theoretical Computer Science*, 2008, 395(1): 31-46
- [22] Zhang X C, Wang Y F, Chen Z H, Xu J, Cui G Z. Arithmetic computation using self-assembly of DNA tiles: Subtraction and division. *Progress in Natural Science*, 2008, accepted
- [23] Pelletier O, Weimerskirch A. Algorithmic Self-Assembly of DNA Tiles and its Application to Cryptanalysis//*Proceedings of the GECCO-2002*. New York, USA, 2002: 139-146
- [24] Li X, Yang X, Qi J, Seeman N C. Antiparallel DNA double crossover molecules as components for nanoconstruction. *Journal of the American Chemical Society*, 1996, 118: 6131-6140
- [25] Liu D, Park S, Reif J, LaBean H T. DNA nanotubes self-assembled from triple-crossover tiles as templates for conductive nanowires//*Proceedings of the National Academy of Science (PNAS)*. USA, 2004, 101: 717-722
- [26] Carbone A, Seeman N C. Molecular tiling and DNA self-assembly//Jonoska N, Paun G, Rozenberg G eds. *Proceedings of the Aspects of Molecular Computing*. LNCS 2340. Berlin: Springer-Verlag, 2004: 61-83

[27]

Reif H J, LaBean H T, Seeman C N. Challenges and applications for self-assembled DNA nanostructures//Proceedings of the 6th International Workshop on DNA-Based Computer. Leider, The Netherlands, 2000; 173-198

[28]

Hoffstein J, Pipher J, Silverman J H. NTRU: A ring based public key cryptosystem//Proceedings of the Algorithm Number Theory. Portland, OR, 1998; 267-288



ZHANG Xun-Cai, born in 1981, Ph.D. candidate. His research interests include bio-computer and information security.

NIU Ying, born in 1982, M. S. candidate. Her research interests include bio-computer and algorithmic design.

CUI Guang-Zhao, born in 1959, Ph. D. , professor. His research interests include bio-computer, neural networks, and system engineering.

WANG Yan-Feng, born in 1973, Ph. D. , associate professor. His research interests include bio-computer and bioinformatics.