International Journal of Grid and Distributed Computing
Vol.6, No.5 (2013), pp.17-28
http://dx.doi.org/10.14257/ijgdc.2013.6.5.02

A DNA Sticker Algorithm for Parallel Reduction over Finite Field GF(2^n)

Yongnan Li[1,2], Limin Xiao[1,2], Li Ruan[1,2], Guangchao Yao[1,2] and Ke Xie[1,2]

[1] State Key Laboratory of Software Development Environment, Beihang University, Beijing, 100191, China
[2] School of Computer Science and Engineering, Beihang University, Beijing, 100191, China

liyongnan.buaa@gmail.com {xiaolm, ruanli}@buaa.edu.cn, {yutianzuijin, xieke89}@cse.buaa.edu.cn

Abstract

This paper proposes a DNA sticker algorithm for parallel reduction over finite field GF(2^n). This algorithm is suitable for some specific finite fields defined with trinomials or pentanomials. We use one binary finite field GF(2^{163}) which is recommended by National Institute of Standards and Technology (NIST) to describe the details about our algorithm. The solution space of 2^{325} cases could be figured out within 3059 DNA steps. This work also presents clear evidence of ability of DNA computing to perform complicated mathematic operations for elliptic curve cryptosystem over finite field GF(2^n).

Keywords: *Parallel reduction, Sticker model, DNA computing, Finite field GF(2^n)*

1. Introduction

DNA computing is a new technique of simulating biomolecular structure of DNA instead of the traditional silicon-based computer technology and the concept of DNA computing was creatively developed by Leonard Adleman of the University of Southern California in [1]. Two major advantages of DNA computing are huge memory capacity and high parallelism which, in data processing, leads molecule computing procedures to solve mathematic hard problems by a brute force searching for the answer sequences.

In recent years, plenty of researches have been focused on attacking cryptosystem such as Diffie–Hellman cryptosystem, RSA, DES and so on. In [2], some DNA-based algorithms designed to solve the problem of discrete logarithms demonstrated that the Diffie–Hellman public-key cryptosystem were perhaps insecure. Work in [3] proved the RSA cryptosystem could be decoded with the linear steps by designing five DNA-based algorithms. A DNA sticker algorithm for bit-substitution was proposed in [4] to indicate that block cipher systems with a 64-bit key were perhaps insecure.

In this paper, we propose a novel DNA sticker algorithm for parallel reduction over finite field GF(2^n) which could compute the solution space of 2^{325} possibilities in 3059 DNA operations. The reduction is one of the major mathematic operations for elliptic curve cryptosystem over finite field GF(2^n). This algorithm provides evidence of the ability of DNA computing to perform intractable computation problems for elliptic curve cryptosystem. This cryptosystem could be proved insecure if it is possible to accomplish modular-multiplication (another time consuming operation) in linear DNA operations. The finite field GF(2^{163}) is taken as an example to describe the details about our algorithm. Firstly, we simplify the

ISSN: 2005-4262 IJGDC

classical reduction over GF(2^{163}) based on the characteristic of binary finite field. Then sticker model which was introduced firstly in [5] as an abstract non-autonomous model for DNA computing is adopted to design the algorithm by considering different cases about every bit of the input parameter.

The rest of this paper is organized as follows. Next section introduces background on sticker model and reduction over finite field GF(2^n). Section 3 presents our DNA sticker algorithm for parallel reduction over GF(2^n). The computation complexity is discussed in Section 4. Section 5 introduces some related works. Last section provides a conclusion and a brief discussion of potential future research ideas.

2. Background

2.1. Sticker model

The idea of our algorithm proposed in this paper is rooted in the sticker model, in which DNA strands are considered as strings over the binary alphabet{0,1}. As depicted in Figure 1, each subsection of memory complex is defined as 0 by annealing no sticker, while the sticker-annealed subsection is defined as 1. For more details about the encoding of memory complex, please refer to [6-9].

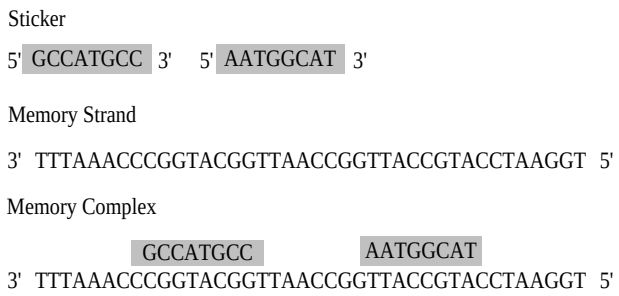


Figure 1. An example of sticker system, which is encoded information 01010

The basic operations of sticker model are listed as follows:

Merge: $Merge(T;T_1,T_2,\cdots,T_n)$. This operation combines memory complexes of the tubes $T_1,T_2,\ldots,T_n$ into a test tube T.

Separate: $Separate(T;i,T_1,T_2)$. This operation separates all strands in test tube T into two different test tubes T_1 and T_2 based on their i^{th} substrands. The memory complexes in test tube T in which the i^{th} substrand is 1 are assigned to the first test tube T_1 and the other ones in which the i^{th} substrand is 0 are assigned to the second test tube T_2.

Set: $Set(T,i)$. This operation sets the i^{th} substrand of every memory complex in test tube T as on.

Clear: $Clear(T,i)$. This operation sets the i^{th} substrand of every memory complex in test tube T as off.

The test tube containing the initial set of strands consists of memory complexes with same format and length.

2.2. Reduction algorithm

Binary finite field, in which carry bits do not need to be propagated, is one of the major mathematical set for constructing elliptic curve cryptosystem. Two specific fields defined with trinomials or pentanomials are the best selections for fast reduction. As demonstrated in Table 1, NIST recommended five reduction polynomials from GF(2^{163}) to GF(2^{571}) in [10].

Table 1. Five reduction polynomials

Number	Modular
1	$f(z)=z^{163}+z^7+z^6+z^3+1$
2	$f(z)=z^{233}+z^{74}+1$
3	$f(z)=z^{283}+z^{12}+z^7+z^5+1$
4	$f(z)=z^{409}+z^{87}+1$
5	$f(z)=z^{571}+z^{10}+z^5+z^2+1$

The following algorithm designed for the 32-bit word length computer is the classical one for computing fast reduction over finite field GF(2^{163})[11]. The highest power exponent of dividend polynomial C(Z) is 324 and C[i] represents the i^{th} word of C(Z), from (*32i-1*) bit to *31i* bit.

Reduction over GF(2^{163}) (word length=32)

Input： $C(Z)=C_{324}Z^{324}+\cdots+C_1Z^1+C_0$

$f(Z)=Z^{163}+Z^7+Z^6+Z^3+1$

Output： $C(Z) \bmod f(Z)$

1. for *i* from 10 to 6, repeat:
   1.1 $T \leftarrow C[i]$
   1.2 $C[i-6] \leftarrow C[i-6] \oplus (T << 29)$
   1.3 $C[i-5] \leftarrow C[i-5] \oplus (T << 4) \oplus (T << 3) \oplus T \oplus (T >> 3)$
   1.4 $C[i-4] \leftarrow C[i-4] \oplus (T >> 28) \oplus (T >> 29)$
2. $T \leftarrow C[5] >> 3$
3. $C[0] \leftarrow C[0] \oplus (T << 7) \oplus (T << 6) \oplus (T << 3) \oplus T$
4. $C[1] \leftarrow C[1] \oplus (T >> 25) \oplus (T >> 26)$
5. $C[5] \leftarrow C[5] \& 0x7$
6. return $(C[5], C[4], C[3], C[2], C[1], C[0])$

3. Theoretical Model of Parallel Reduction

3.1. Simplification of Reduction over GF(2^{163})

In order to design the DNA sticker algorithm for parallel reduction over GF(2^{163}), we could firstly simplify the original algorithm according to the characteristic of this finite field. Every monomial, for instance, whose index is larger than 162, is equivalent to a corresponding polynomial over GF(2^{163}).

$$\begin{cases} Z^{163} \equiv Z^7 + Z^6 + Z^3 + 1 \quad (\mathrm{mod}\ f(x)) \\ Z^{164} \equiv Z^8 + Z^7 + Z^4 + Z \quad (\mathrm{mod}\ f(x)) \\ \quad \vdots \\ Z^{324} \equiv z^{168} + Z^{167} + Z^{164} + Z^{161} (\mathrm{mod}\ f(x)) \end{cases} \tag{1}$$

Then coefficients of polynomial C(Z) are deduced as shown in the follow equations (2), (3), (4), (5), (6) and (7). To design the DNA sticker algorithm and differ from C[i], we use parameter X[i] to represent the i^{th} bit in this subsection.

$$\begin{cases} X[168] = X[168] \oplus X[324] \\ X[167] = X[167] \oplus X[323] \oplus X[324] \\ X[166] = X[166] \oplus X[322] \oplus X[323] \\ X[165] = X[165] \oplus X[321] \oplus X[322] \\ X[164] = X[164] \oplus X[320] \oplus X[321] \oplus X[324] \\ X[163] = X[163] \oplus X[319] \oplus X[320] \oplus X[323] \\ X[162] = X[162] \oplus X[318] \oplus X[319] \oplus X[322] \end{cases} \tag{2}$$

$$\begin{cases} X[161] = X[161] \oplus X[317] \oplus X[318] \oplus X[321] \oplus X[324] \\ \quad \vdots \\ X[7] = X[7] \oplus X[163] \oplus X[164] \oplus X[167] \oplus X[170] \end{cases} \tag{3}$$

$$X[6] = X[6] \oplus X[163] \oplus X[166] \oplus X[169] \tag{4}$$

$$\begin{cases} X[5] = X[5] \oplus X[165] \oplus X[168] \\ X[4] = X[4] \oplus X[164] \oplus X[167] \\ X[3] = X[3] \oplus X[163] \oplus X[166] \end{cases} \tag{5}$$

$$\begin{cases} X[2] = X[2] \oplus X[165] \\ X[1] = X[1] \oplus X[164] \\ X[0] = X[0] \oplus X[163] \end{cases} \tag{6}$$

In (3), all equations could be expressed as (7):

$$X[i] = X[i] \oplus X[i+156] \oplus X[i+157] \oplus X[i+160] \oplus X[i+163] \tag{7}$$

The index of final polynomial result must be lower than 163, so the monomials power exponent varying from 168 to 163 should be reduced again. Consequently, the final result is obtained by combining (2), (3), (4), (5) and (6) as Table 2, in which Y[i] represents coefficient of result, shows.

Table 2. Parallel reduction over GF(2^{163})

Result	Input
Y[162]	X[162] ⊕ X[318] ⊕ X[319] ⊕ X[322]
Y[161:13]	X[161:13] ⊕ X[317:169] ⊕ X[318:170] ⊕ X[321:173] ⊕ X[324:176]
Y[12]	X[12] ⊕ X[168] ⊕ X[324] ⊕ X[169] ⊕ X[172] ⊕ X[175]
Y[11]	X[11] ⊕ X[167] ⊕ X[323] ⊕ X[168] ⊕ X[171] ⊕ X[174]
Y[10]	X[10] ⊕ X[166] ⊕ X[322] ⊕ X[167] ⊕ X[324] ⊕ X[170] ⊕ X[173]
Y[9]	X[9] ⊕ X[165] ⊕ X[321] ⊕ X[166] ⊕ X[323] ⊕ X[169] ⊕ X[172]
Y[8]	X[8] ⊕ X[164] ⊕ X[320] ⊕ X[165] ⊕ X[322] ⊕ X[168] ⊕ X[171]
Y[7]	X[7] ⊕ X[163] ⊕ X[319] ⊕ X[164] ⊕ X[321] ⊕ X[167] ⊕ X[170]
Y[6]	X[6] ⊕ X[163] ⊕ X[319] ⊕ X[322] ⊕ X[320] ⊕ X[166] ⊕ X[169]
Y[5]	X[5] ⊕ X[165] ⊕ X[321] ⊕ X[322] ⊕ X[168] ⊕ X[324]
Y[4]	X[4] ⊕ X[164] ⊕ X[320] ⊕ X[321] ⊕ X[167] ⊕ X[323]
Y[3]	X[3] ⊕ X[163] ⊕ X[319] ⊕ X[320] ⊕ X[166] ⊕ X[322]
Y[2]	X[2] ⊕ X[165] ⊕ X[321] ⊕ X[322]
Y[1]	X[1] ⊕ X[164] ⊕ X[320] ⊕ X[321] ⊕ X[324]
Y[0]	X[0] ⊕ X[163] ⊕ X[319] ⊕ X[320] ⊕ X[323]

3.2. Sticker Algorithm

As are so often interpreted in other molecular algorithms, we must first describe the DNA strands that will be used in our algorithm. A memory strand X representing the input data and a memory strand Y representing the result are shown in Figure 2(a) and Figure 2(b), and 488 stickers are complementary to subsections of the two memory strands as illustrated in Figure 2(c). To solve the solution space of reduction over finite field GF(2^{163}), we assume that test tube T_1 contains enough memory complexes, covering all possible inputs and consisting of memory strands X and stickers X_i, and that test tube T_2 contains sufficient memory strands Y in which all subsections are set as off.

In order to compute the multiple operations of XOR for every bit, we modify the DNA operation: $Separate(T, i, T_1, T_2)$. Take dividing X_0X_1 as an example to show the concrete steps, two DNA operations with probe X_0 and probe X_1 are carried out to divide the memory complexes in test tube T into two parts: T_1 and T_2. The two DNA operations are expressed as $Separate(T, X_0X_1 = 01, T_1)$ and $Separate(T, X_0X_1 = 10, T_1)$. All memory complexes in which the two substrands are '01' or '10' are assigned to T_1 and the other ones are assigned to another test tube T_2 by defaut.

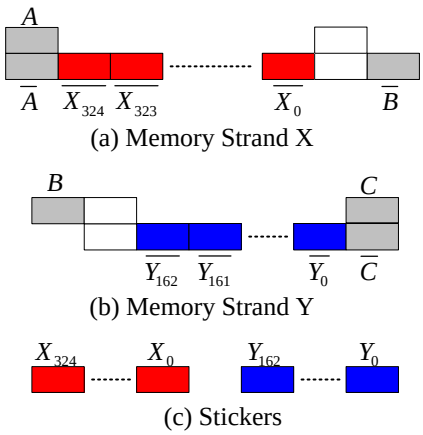


Figure 2. 2 memory strands and 488 stickers used in the proposed algorithm

The processing procedure of our algorithm is described as follows:

Reduction (T, T_1, T_2, 163)

$Merge(T, T_1, T_2)$

// computing the highest bit 162

$Separate(T, X_{162}X_{318}X_{319}X_{322} = 0001, T_1)$

$Separate(T, X_{162}X_{318}X_{319}X_{322} = 0010, T_1)$

$Separate(T, X_{162}X_{318}X_{319}X_{322} = 0100, T_1)$

$Separate(T, X_{162}X_{318}X_{319}X_{322} = 1000, T_1)$

$Separate(T, X_{162}X_{318}X_{319}X_{322} = 0111, T_1)$

$Separate(T, X_{162}X_{318}X_{319}X_{322} = 1011, T_1)$

$Separate(T, X_{162}X_{318}X_{319}X_{322} = 1101, T_1)$

$Separate(T, X_{162}X_{318}X_{319}X_{322} = 1110, T_1)$

$Set(T_1, Y_{162})$

$Merge(T, T_1, T_2)$

// 155 loops

for i from 161 *to* 7

// 16 possible inputs that the total count of '1' bit is an odd integer

$Separate(T, X_i X_{i+156} X_{i+157} X_{i+160} X_{i+163} = 00001, T_1)$

$Separate(T, X_i X_{i+156} X_{i+157} X_{i+160} X_{i+163} = 00010, T_1)$

$\vdots$

$Separate(T, X_i X_{i+156} X_{i+157} X_{i+160} X_{i+163} = 11111, T_1)$

$Set(T, Y_i)$

$Merger(T, T_1, T_2)$

od

// 6 loops, Y_i is considered as input in Separate operation

for i from 12 *to* 7

$Separate(T, Y_i X_{i+312} = 10, T_1)$

$Separate(T, Y_i X_{i+312} = 01, T_1)$

$Set(T_1, Y_i)$

$Clear(T_2, Yi)$

$Merge(T, T_1, T_2)$

od

for i from 10 *to* 7

$Separate(T, Y_i X_{i+314} = 10, T_1)$

$Separate(T, Y_i X_{i+314} = 01, T_1)$

$Set(T_1, Y_i)$

$Clear(T_2, Yi)$

$Merge(T, T_1, T_2)$

od

// 64 possible inputs that the total count of '1' bit is an odd integer

$Separate(T, X_6 X_{163} X_{319} X_{322} X_{320} X_{166} X_{169} = 0000001, T_1)$

$\vdots$

$Separate(T, X_6 X_{163} X_{319} X_{322} X_{320} X_{166} X_{169} = 1111111, T_1)$

$Set(T_1, Y_7)$

$Merge(T, T_1, T_2)$

for i from 5 *to* 3

// 32 possible inputs that the total count of '1' bit is an odd integer

$Separate(T, X_i X_{i+160} X_{i+316} X_{i+317} X_{i+163} X_{i+319} = 000001, T_1)$

$Separate(T, X_i X_{i+160} X_{i+316} X_{i+317} X_{i+163} X_{i+319} = 000010, T_1)$

$\vdots$

$Separate(T, X_i X_{i+160} X_{i+316} X_{i+317} X_{i+163} X_{i+319} = 111110, T_1)$

$Set(T_1, Y_i)$

$Merge(T, T_1, T_2)$

od

for i from 2 *to* 0

$Separate(T, X_i X_{i+163} X_{i+319} X_{320} = 0001, T_1)$

$Separate(T, X_i X_{i+163} X_{i+319} X_{320} = 0010, T_1)$

$Separate(T, X_i X_{i+163} X_{i+319} X_{320} = 0100, T_1)$

$Separate(T, X_i X_{i+163} X_{i+319} X_{320} = 1000, T_1)$

$Separate(T, X_i X_{i+163} X_{i+319} X_{320} = 0111, T_1)$

$Separate(T, X_i X_{i+163} X_{i+319} X_{320} = 1011, T_1)$

$Separate(T, X_i X_{i+163} X_{i+319} X_{320} = 1101, T_1)$

$Separate(T, X_i X_{i+163} X_{i+319} X_{320} = 1110, T_1)$

$Set(T_1, Y_i)$

$Merge(T, T_1, T_2)$

od

// last procedure for computing Y_1 and Y_0

for i from 1 *to* 0

$Separate(T, Y_i X_{i+323} = 01, T_1)$

$Separate(T, Y_i X_{i+323} = 10, T_1)$

$Set(T_1, Y_i)$

$Clear(T_2, Yi)$

$Merge(T, T_1, T_2)$

od

$return(T)$

Firstly, the highest bit of the result, the 162nd bit of memory strand 'Y', which is particularly concerned with four bits of the input parameter – the 162nd bit, the 168th bit, the 169th bit and the 322nd bit, is initialized by executing eight '**Separate**' operations, one '**Set**' operation and one '**Merge**' operation. All of the 162nd bits of memory strand 'Y' in test tube T_1, where every case of the input data that leads the result of 162nd bit to value '1' has been considered, are assigned into '1'.

Next, 155 bits of memory strand 'Y' – from the 161st bit to the 7th bit – could be calculated through a chain of similar procedures considering different inputs for every bit and computing the first five bits of the inputs firstly. And apparently there would be still 6 bits of memory strand 'Y', in which one or two input bits has not been regarded in the above procedures, where the first five input bits of the 155 bits are exhibiting like a series. It would be an assistance to compute the final result of the 6 lower bits, from the 12th bit to the 7th bit, since the later consideration of the remaining cases would simplify the execution. In fact, in the entire procedures of computing the 6 lower bits, these substrands of memory complex (from Y_{12} to Y_0) acts as the temporary results while computing the first five inputs and then acts as the input to consider the final value.

And then, 64 possible cases of 7 inputs for the calculation of the 6th bit of memory strand 'Y' should be considered. So are the procedures of computing 3 bits (the 5th bit, the 4th bit and the 3rd bit), all of which consists of 32 possible inputs.

Lastly, the lowest 3 bits are figured out by computing 4 inputs firstly and then regarding the remaining one input for the 1st bit and the 0th bit, much the way we deal with the 155 bits of memory strand 'Y' in the second step.

4. Complexity Discussion

This DNA sticker algorithm needs 2^{325} memory strands X, 2^{163} memory strands Y, $2^{325}/2$ stickers X_{324}, X_{323},…, X_0 , $2^{163}/2$ stickers Y_{162}, Y_{161},…, Y_0 and 488 probes. Therefore, the space complexity of this algorithm is

$$\begin{cases} S_{memory\ strand} = 2^{325} + 2^{163} \\ S_{sticker} = \left(2^{325} + 2^{163}\right)/2 \\ S_{probe} = 488 \end{cases} \tag{8}$$

According to the above algorithm, the total number of every DNA operation could be obtained by simple arithmetical calculations. The time complexity is

$$\begin{cases} S_{separate} = 2696 \\ S_{merge} = 176 \\ S_{set} = 175 \\ S_{clear} = 12 \end{cases} \tag{9}$$

The space complexity which means the materials used in this algorithm is acceptable because the high storage capacity is one of the major advantages of DNA computing. The time complexity analysis demonstrates the high parallelism in data processing of DNA computing. In other computing platform, we have to consider 2^{325} different cases of all possible inputs and therefore 2^{325} times of parallel reduction over GF(2^{163}) needs to be executed to get the solution space of this operation. Obviously, the time complexity of parallel reduction is also very small while turning to other four types of finite field GF(2^n) with the proposed molecule algorithm.

5. Related Works

Several parallel reductions as part of elliptic curve cryptosystem over this specific finite field GF(2^n) have been proposed recently. Differing from our algorithm designed for DNA computing, these researches impose their parallel implementation on other computing platforms. We introduce some researches in the last few years to contrast our work.

In [12], a fast reconfigurable method for elliptic curve cryptography acceleration in GF(2^n) was implemented on an Intel P4 2.8 Ghz processor. This paper determined that systematic reduction was the best choice for finite fields defined with trinomials orpentanomials. In[13], a scalable reduction unit was designed as part of a flexible-low resources elliptic curve cryptography processor over binary finite fields GF(2^n) on FPGA platforms. The implementation of the flexible ECC processor consumes only 1278 slices for 32-bit data path on Spartan III XC3S200. An elliptic curve cryptosystem (ECC) over GF(2^n) was implemented in [14] on sensor motes using small word size for low-powered microprocessors. After that, these authors implemented another elliptic curve cryptosystem with Koblitz curve on 16-bit word on 16-bit Tmote Sky mote in [15]. Research in [16] introduced a low resources scalable elliptic curve cryptography processor over binary finite fields GF(2^n) in which reduction over GF(2^n) was designed as scalable reduction unit. This scalable ECC processor implemented on FPGA platform consumes only 1127 slices for 32-bit data path on Spartan III XC3S200.

From above discussion, it can be seen that other studies imposed on reducing computing unit, accelerating computing speed or lowering power consumption. Our works are mainly focused on figuring out the solution space within as less steps as possible.

6. Conclusion

This paper aims to provide a DNA sticker algorithm to compute parallel reduction over binary finite field $GF(2^n)$ and we introduced a concrete example over $GF(2^{163})$ to describe the details of our algorithm. Only 3059 DNA operations are needed to obtain the 2^{235} solution space. The algorithm could be applied to other binary finite fields introduced in section 2.

The described DNA algorithm is totally designed theoretically based on the condition that all DNA operations such as *Merge*, *Separate*, *Set* and *Clear* are ideal. There would be still a long way to go to overcome many complicate factors of DNA experiment such as purity of separation process and amplification of answer sequences. However, the experiments presented in [17] proved that the molecular computer is possible to perform parallel computation in the future.

The DNA algorithm proposed in this paper only provides a method of parallel reduction for some special types of binary finite field. We plan to design DNA algorithm of computing parallel modular-multiplication over binary finite field, by combining with the algorithm proposed in this paper the elliptic curves might be proved insecure. The future design will be introduced in later publication.

Acknowledgements

This study is sponsored by the fund of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2012ZX-06, the Hi-tech Research and Development Program of China (863 Program) under Grant No. 2011AA01A205, Beijing Natural Science Foundation under Grant No. 4122042 and the National Natural Science Foundation of China under Grant No. 61232009.

References

[1] L. M. Adleman, "Molecular computation of solutions to combinatorial problems", Science, vol. 266, no. 5187, **(1994**), pp. 1021-1024.
[2] W. L. Chang, S. C. Huang, K. W. Lin and M. Ho, "Fast parallel DNA-based algorithms for molecular computation: discrete logarithm", Journal Of Supercomputing, vol. 56, no. 2, **(2011**), pp. 129-163.
[3] W. Chang, K. W. Lin, J. Chen, C. Wang, L. C. Lu, M. Guo and M. S. Ho, "Molecular solutions of the RSA public-key cryptosystem on a DNA-based computer", Journal Of Supercomputing, **(2011**), pp. 1-31.
[4] X. Geng, L. Pan and J. Xu, "A DNA sticker algorithm for bit-substitution in a block cipher", Journal Of Parallel And Distributed Computing, vol. 68, no. 9, **(2008**), pp. 1201-1206.
[5] E. A. Roweis, "A Sticker-Based Model for DNA Computation", Journal Of Computational Biology, vol. 5, **(1998**), pp. 615-629.
[6] S. Roweis, E. Winfree, R. Burgoyne, N. Chelyapov, M. Goodman, P. Rothemund and L. Adleman, "A sticker based architecture for DNA computation", Proceedings of Second annual meeting on dna based computers**, (1996**) June 10-12, Providence.
[7] G. Paun and G. Rozenberg, "Sticker systems", Theoretical Computer Science, vol. 204, no. 12, (**1998**), pp. 183-203.
[8] G. P˘aun, G. Rozenberg and A. Salomaa, "DNA Computing: New Computing Paradigms", Springer-Verlag, (**1998**).
[9] C. N. Yang and C. B. Yang, "A DNA solution of SAT problem by a modified sticker model", Biosystems, vol. 81, no. 1, **(2005**), pp. 1-9.
[10] National Institute of Standards and Technology, "Digital signature standard", FIPS Publication 186-2. http://csrc.nist.gov/publications/PubsFIPSArch.html, **(2000)**.

[11] D. Hankerson, A. Menezes and S. Vanstone, "Guide to elliptic curve cryptography", Springer, **(2004)**.
[12] A. E. Cohen and K. K. Parhi, "Fast reconfigurable elliptic curve cryptography acceleration for GF(2 m) on 32 bit processors", Journal of Signal Processing Systems, vol. 60, no. 1, **(2010**), pp. 31-45.
[13] M. N. Hassan, M. Benaissa and A. Kanakis, "Flexible hardware/software co-design for scalable elliptic curve cryptography for low-resource applications", Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors, **(2010)** July 7-9; Rennes, France.
[14] S. C. Seo, D. Han, H. C. Kim and S. Hong, "TinyECCK: Efficient elliptic curve cryptography implementation over GF(2m) on 8-bit micaz mote", Ieice Transactions on Information And Systems, vol. E91-D, no. 5, **(2008)**, pp. 1338-1347.
[15] S. C. Seo, D. Han and S. Hong, "TinyECCK16: An efficient field multiplication algorithm on 16-bit environment and its application to Tmote Sky sensor motes", Ieice Transactions on Information And Systems, vol. E92-D, no. 5, **(2009)**, pp. 918-928.
[16] M. N. Hassan and M. Benaissa, "A scalable hardware/software co-design for elliptic curve cryptography on picoblaze microcontroller", Proceedings of 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems, (**2010**) May 30 - June 2; Paris, France.
[17] Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh and E. Shapiro, "Programmable and autonomous computing machine made of biomolecules", Nature, vol. 414, no. 6862, **(2001**), pp. 430-434.

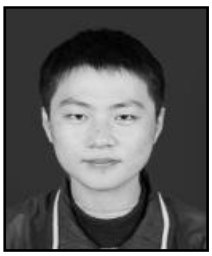
Authors

Yongnan Li is a Ph.D. student at School of Computer Science and Engineering, Beihang University. His main research areas are computer architecture, cloud computing, parallel computing and information security.

Limin Xiao is a professor of the Institute of Computer Architecture, Beihang University. He is also a senior membership of China Computer Federation. His main research areas are computer architecture, computer system software, high performance computing, virtualization and cloud computing.

Ruan Li is a lecturer of the Institute of Computer Architecture, Beihang University. She is also a senior membership of China Computer Federation. Her main research areas are virtualization and cloud computing, computer system software, high performance computer.

Guangchao Yao is a graduate student in Beihang University. His main research areas are computer architecture, computer system software, virtualization and cloud computing.



Ke Xie is a graduate student in Beihang University. His main research areas are computer architecture, parallel file system, computer system software, virtualization and cloud computing.