Copyright ⓒ 2020 KSII

The Novel Efficient Dual-field FIPS Modular Multiplication

Tingting Zhang[1], Junru Zhu[1], Yang Liu[1], Fulong Chen[1,2,3*]
[1] School of Computer and Information, Anhui Normal University, Wuhu Anhui 241002, China
[e-mail: zhangtingting@ahnu.edu.cn]
[2] State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
[3] Anhui Provincial Key Lab of Network and Information Security, Wuhu Anhui 241002, China
[e-mail: long005@mail.ahnu.edu.cn]
*Corresponding author: Fulong Chen

Received June 3, 2019; revised August 2, 2019; accepted November 13, 2019; published February 29, 2020

Abstract

The modular multiplication is the key module of public-key cryptosystems such as RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography). However, the efficiency of the modular multiplication, especially the modular square, is very low. In order to reduce their operation cycles and power consumption, and improve the efficiency of the public-key cryptosystems, a dual-field efficient FIPS (Finely Integrated Product Scanning) modular multiplication algorithm is proposed. The algorithm makes a full use of the correlation of the data in the case of equal operands so as to avoid some redundant operations. The experimental results show that the operation speed of the modular square is increased by 23.8% compared to the traditional algorithm after the multiplication and addition operations are reduced about $(s^2 - s)/2$, and the read operations are reduced about $s^2 - s$, where $s = n/32$ for n-bit operands. In addition, since the algorithm supports the length scalable and dual-field modular multiplication, distinct applications focused on performance or cost could be satisfied by adjusting the relevant parameters.

Keywords: Montgomery algorithm, FIPS, Modular Multiplication, Modular Square, Dual-field.

This research was supported by National Natural Science Foundation of China (No. 61572036) and State Key Laboratory of Computer Architecture (CARCH201810).

http://doi.org/10.3837/tiis.2020.02.015 **ISSN : 1976-7277**

1. Introduction

Cryptography provides a strong support for data security with encryption and decryption. In particular, the public key cryptography represented by RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) is the core component of PKI/CA system. It solves the critical issues such as key distribution and identity recognition of symmetric cryptography, and ensures data confidentiality, integrity and authenticity of both parties in the e-commerce, e-government, e-bank and social networks applications [1]. Compared with RSA, ECC has a shorter key length at the same security strength, and has multiple advantages in a variety of wireless devices, smart cards and other resource-constrained device applications.

In our work, we will study on the design of a kind of efficient dual-field FIPS (Finely Integrated Product Scanning) modular multiplication which supports both RSA and SM2 cryptosystems in prime and binary fields. It takes full account of the modular multiplication of the public key cryptography algorithm under the condition of equal operands, reduces the multiplication, addition and reading operations in the case, improves the operation efficiency of modular multiplication, decreases the power consumption, and provides a unified dual-field modular multiplication algorithm for resource constrained devices to implement both RSA and SM2 algorithms. Moreover, by reorganizing the operation flow of dual-field modular square and optimizing its logic structure, with pipeline architecture which takes small area and low power consumption as the focus, using dual-field multiplier, dual-field adder and modular multiplication controller to control the data operation process in the key data path, an efficient modular exponentiation circuit is designed.

2. Related Works

RSA and ECC algorithms involve a large number of complex operations. Implemented using software alone, their efficiency is very low. In order to improve the efficiency and security of the public key cryptography algorithm, usually using FPGA (Field －Programmable Gate Array) or VLSI (Very Large Scale Integration) to implement the key operations involved in the algorithm [2], the cryptographic algorithm is integrated into the circuit design of related cryptographic coprocessor, cipher chip and so on. It focuses on complex operations such as modular multiplication, modular exponentiation and modular inversion.

In [3] and [4], the remainder system is used to improve the parallelism of modular multiplication and modular exponentiation, and the operation is accelerated by the efficient arithmetic unit architecture. Using the large number subtraction to optimize modular multiplication [6, 7], although the internal loop has been increased for one time, the period of modular multiplication and modular exponentiation is reduced. Li [8] and Chen [9] have made effective improvements in the design and implementation of dual finite field modular multiplication. Kadar [10] and Qi [11] use reversible logic gates to design modular multiplication and modular inverse circuits of cryptographic algorithms to prevent the loss of energy during computation. However, the security chip supporting both RSA and SM2 algorithms requires higher resource consumption. In recent years, with the development of chip manufacturing technology, hardware design of cryptographic systems emphasizes high speed and ignores the area, power and resource consumption. Therefore, more efficient

implementations are needed to overcome the environmental requirements in the application process.

Due to the efficiency and security of the algorithm, most of the key operations of RSA and ECC, especially the high complexity multiplication, are implemented in hardware. Using a simple addition, multiplication and shift operation to solve the tedious division problems in traditional algorithms, the Montgomery modular multiplication algorithm [12] is suitable for hardware implementation. In the recent study, the dependency graph and multiple process elements (PEs) are the research hotspots in the Montgomery modular multiplication algorithm. According to the algorithm, Lin et al [5, 13, 14] proposed a hardware architecture consisting of multiple PEs to work in parallel for reducing the delay and the memory bandwidth requirement, and achieving higher throughput. Renardy et al [15] designed an iterative modular architecture on FPGA and achieved better AT^2 (area delay). A novel iterative architecture [16] for the prime field $GF(p)$ is proposed lately by Morales-Sandoval to reduce the area. However, the above studies only consider the $GF(p)$ field or the binary field $GF(2^m)$ in a single field situation. Of the design of less dual-field models, Savas [17], Zheng [18] and Liao [19] et al have done much work on the design of modular multiplication hardware with significant contributions to scalability, speed, and the RSA/ECC coprocessors.

In general, on the one hand, the above research on the Montgomery modular algorithm is less supportive for the dual-field. On the other hand, these works focus on the pursuit of speed, and in terms of area, power consumption and other aspects, they are not suitable for the applications in wireless devices, smart cards and other resource-constrained devices. In order to meet the parameter requirements of resource-constrained devices, we re-design the Montgomery modular multiplication algorithm, and optimize the modular square based on the characteristics of the FIPS so as to increase the speed of the modular square, and reduce the operations of the operand access, multiplication and addition. Furthermore, the improved algorithm is extended to the $GF(2^m)$ field to support both the RSA and ECC cryptosystems, and the corresponding circuits are implemented on FPGA.

3. FIPS Algorithm Optimization

In the RSA cryptosystem, the modular exponentiation algorithm is implemented by the repeated modular multiplication. The modular multiplication is also the basic operation of the SM2 crytosystem. The modular multiplication is the lowest level operation in the encryption and decryption algorithm, and its performance determines the overall speed and efficiency. In the implementation of modular multiplication, Blakley [20], Barrett [21], Montgomery [12] and other algorithms are commonly used. Modular multiplication plays an important role in the basic operations in RSA and SM2. The probability of modular square operations is close to 50%. However, the computing speed of modular square is limited by modular multiplication. It is necessary to reduce the cycle of modular square operation so as to improve the efficiency of encryption and decryption.

3.1 Basic FIPS Algorithm

Through the analysis of different modular multiplication algorithms, the Montgomery modular multiplication algorithm is more suitable for hardware design, and the highest efficiency can be achieved. As a kind of efficient implementation, its FIPS modular multiplication algorithm is described as shown in **Algorithm 1**.

Algorithm 1. Basic FIPS modular multiplication algorithm

Input: N, A, B, n_0'

Output: $M \leftarrow A \times B \times R^{-1} \bmod N$

1. for i = 0 to s-1

2. $S \leftarrow S + \sum_{j=0}^{i-1} a_j b_{i-j} + \sum_{j=0}^{i-1} m_j n_{i-j}$;

3. $S \leftarrow S + a_i b_0$ **;**

4. $m_i \leftarrow S n_0' \bmod W$;

5. $S \leftarrow S + m_i n_0$;

6. $S \leftarrow S / W$;

7. end for

8. for i = s to 2s-1

9. $S \leftarrow S + \sum_{j=i-s+1}^{s-1} a_j b_{i-j} + \sum_{j=i-s+1}^{s-1} m_j n_{i-j}$;

10. $m_{i-s} \leftarrow S \bmod W$;

11. $S \leftarrow S / W$;

12. end for

13. $m_s \leftarrow S \bmod W$;

14. if $M \geq N$ then

15. $M \leftarrow M - N$;

16. endif

17. return M;

It can be seen that the FIPS modular multiplication algorithm based on product scanning mainly consists of two loops for calculating respectively the most s significant bits and least s significant bits of the final result. Taking the 6×6 word operands A and B for example, the calculation process of the FIPS modular multiplication algorithm is shown in **Fig. 1**. The red box shows i from 0 to 1, that is, m_i is computed from the right to the left by the product scanning mode. The result m_i of the current column i is used to calculate the operations that involve the ith row afterwards, as shown in the black solid line arrow. The low 6 bit m_i is replaced by a high 6 bit m_i update after the operation is completed.

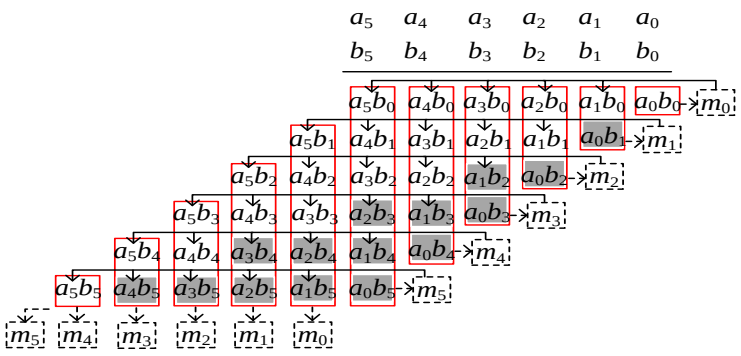

Fig. 1. FIPS Modular Multiplication

3.2 Optimized FIPS Algorithm

Assumed that only multiplication operations are considered, when $A \neq B$, it only needs to simply accumulate the partial products and takes s^2 word-based modular multiplication operations, as shown in **Algorithm 2**.

Algorithm 2. Optimized FIPS modular multiplication algorithm

Input: $A = (a_{s-1},...,a_1,a_0)_2, B = (b_{s-1},...,b_1,b_0)_2$

Output: $T = (t_{2s-1},...,t_1,t_0)_2 \leftarrow A \times B$

1. for i = 0 to s-1
2. $C \leftarrow 0$;
3. for j = 0 to s-1
4. $(C,S) \leftarrow t_{i+j} + a_j b_i + C$;
5. $t_{i+j} \leftarrow S$;
6. end for
7. $t_{i+s} \leftarrow S$;
8. end for
9. return T;

When $A = B$, the accumulated operations on the partial product can be simplified, as shown in **Algorithm 3**. Through the 2 times accumulation of $a_j b_i$ in the original algorithm, the calculation flow can be optimized, and ultimately, it only takes $(s^2 + s)/2$ word-based multiplication operations. As a result, the computation of $A = B$ compared with $A \neq B$ decreases $(s^2 + s)/2$ multiplication operations.

Algorithm 3. FIPS modular square algorithm

Input: $A = (a_{s-1},...,a_1,a_0)_2$

Output: $T = (t_{2s-1},...,t_1,t_0)_2 \leftarrow A \times A$

1. for i = 0 to s-1
2. $(C,S) \leftarrow t_{2i} + a_i^2$;
3. for j = i+1 to s-1
4. $(C,S) \leftarrow t_{i+j} + a_j a_i + C$;
5. $t_{i+j} \leftarrow S$;
6. end for
7. $t_{i+s} \leftarrow S$;
8. end for
9. return T;

With the modular square optimization, as shown in **Fig. 1**, the FIPS modular multiplication algorithm can be optimized. Since the partial product in the grey is symmetric to the current column, when accumulating the grey partial product, the symmetric partial product can be accumulated ahead of time, i.e., when $A = B$, $a_j b_{i-j} = a_{i-j} b_i$. The $a_j b_{i-j}$ accumulation of the

previous and posterior s columns can be written as $\sum_{j=0}^{i} a_j b_{i-j}$ and $\sum_{j=i-s+1}^{s-1} a_j b_{i-j}$ respectively. $\sum_{j=0}^{i} a_j b_{i-j}$ can be expanded as

$$\begin{cases} a_0b_1 + a_1b_0 = 2a_0b_1 & i=1 \\ a_2b_0 + a_1b_1 + a_0b_2 = 2a_0b_2 + a_1b_1 & i=2 \\ a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3 = 2a_0b_3 + 2a_1b_2 & i=3 \\ a_4b_0 + a_3b_1 + a_2b_2 + a_1b_3 + a_0b_4 = 2a_0b_4 + 2a_1b_3 + a_2b_2 & i=4 \\ a_5b_0 + a_4b_1 + a_3b_2 + a_2b_3 + a_1b_4 + a_0b_5 = 2a_0b_5 + 2a_1b_4 + 2a_2b_3 & i=5 \end{cases} \quad (1)$$

The following conclusions can be drawn

$$\sum_{j=0}^{i} a_j b_{i-j} = \begin{cases} 2\sum_{j=0}^{i/2} a_j b_{i-j} & i\%2=1 \\ 2\sum_{j=0}^{i/2-1} a_j b_{i-j} + a_{i/2}b_{i/2} & i\%2=0 \,\&\, i\neq 0 \end{cases} \quad (2)$$

Similarly, for $\sum_{j=i-s+1}^{s-1} a_j b_{i-j}$, for the ith column with parity difference, there is a $j = i - j$ in the ith even columns, and the partial product has only one, e.g., $a_{i/2}b_{i/2}$. Therefore, the ith column is symmetric relative to the partial product, and the cumulative form is somewhat different because the difference of parity in the ith column. For $i \in [1, 2s-2]$, by this way, the computing of $a_j b_{i-j}$ can be reduced by nearly 50%. Moreover, in logic circuits, $2a_j b_{i-j}$ can be generated by only one left shift of $a_j b_{i-j}$.

4. Efficient Dual-field FIPS Modular Multiplication Algorithm

4.1 Algorithm improvement

Through the FIPS modular multiplication algorithm principle and operation characteristics analysis, we can see, for the logic optimization of FIPS modular multiplication algorithm, measures need to be taken in the critical computing process, as shown in **Fig. 2**. In addition, a dual-field modular multiplication unit that supports both the prime field $GF(p)$ and the binary field $GF(2^m)$ is needed. Therefore, the efficient dual-field FIPS algorithm is improved on the basis of modular multiplication, where field is used to control the selection of operation *fields*, and $equal \leftarrow A = B ? 0 : 1$ is used to control the selection of modular multiplication operations including modular multiplication or modular square.

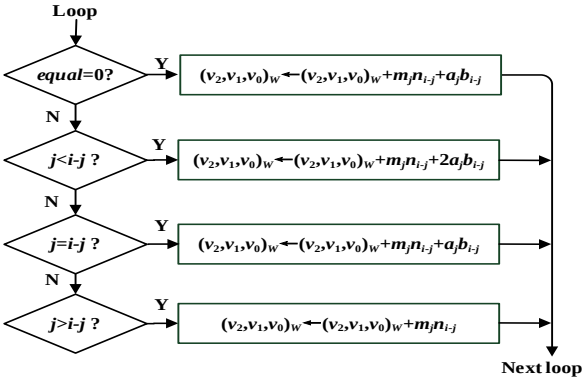

Fig. 2. Critical Computation Process

The improved FIPS modular multiplication algorithm is shown in **Algorithm 4**.

Algorithm 4. Efficient Dual-field FIPS modular multiplication algorithm

Input: N, A, B, n_0', *field*, *equal*

Output: $M \leftarrow A \times B \times R^{-1} \bmod N$

1. for i = 0 to s-1
2. for j = 0 to i-1
3. Calculate in accordance with key processes in **Fig. 2**;
4. end for
5. if i = 0 | *equal* = 0 then
6. $(v_2, v_1, v_0)_W \leftarrow (v_2, v_1, v_0)_W + a_i b_0$;
7. endif
8. $m_i \leftarrow v_0 n_0' \bmod W$
9. $(v_2, v_1, v_0)_W \leftarrow (v_2, v_1, v_0)_W + m_i n_0$;
10. $(v_2, v_1, v_0)_W \leftarrow (0, v_2, v_1)_W$;
11. end for
12. for i = s to 2s-1
13. for j = i–s+1 to s-1
14. Calculate in accordance with key processes in **Fig. 2**;
15. end for
16. $m_{i-s} \leftarrow v_0$;
17. $(v_2, v_1, v_0)_W \leftarrow (0, v_2, v_1)_W$;
18. end for
19. $m_s \leftarrow v_0$;
20. if $M \geq N$ && $field = 1$ then
21. $M \leftarrow M - N$;
22. endif
23. return M;

In the calculation process, two variables, j and $i - j$, are used to determine the flow of calculation, so as to avoid the increase of redundant variables.

- When $A = B$, i.e., $equal = 0$, similar to the traditional FIPS modular multiplication algorithm, each internal loop needs only a simple accumulation of $a_j b_{i-j}$ and $m_j n_{i-j}$. As shown in **Fig. 2**, the Formula 3 is executed until the algorithm is over.

$$(v_2, v_1, v_0)_W \leftarrow (v_2, v_1, v_0)_W + m_j n_{i-j} \tag{3}$$

- When $A \neq B$, i.e., $equal = 1$, as shown in **Fig. 1**, the symmetric partial product is accumulated in advance when accumulating the grey partial product $a_j b_{i-j}$, and after that, it just needs to accumulate $m_j n_{i-j}$. The algorithm controls the executed formula according to the relationship between j and $i - j$ each time. In the case of $j < i - j$, the partial product $a_j b_{i-j}$ is accumulated 2 times according to Formula 4. If $j = i - j$,

because there is no symmetric partial product, it will be accumulated according to Formula 5. For each time of the inner loop, both Formula 4 and Formula 5 need two multiplication operations and two addition operations. And when $j < i - j$, since the partial product $a_j b_{i-j}$ has accumulated ahead of time, and at this point only the $m_j n_{i-j}$ part is left, for each time of the inner loop, Formula 3 only needs one multiplication operation and one addition operation until the algorithm is over.

$$(v_2, v_1, v_0)_W \leftarrow (v_2, v_1, v_0)_W + 2a_j b_{i-j} + m_j n_{i-j} \tag{4}$$

$$(v_2, v_1, v_0)_W \leftarrow (v_2, v_1, v_0)_W + a_j b_{i-j} + m_j n_{i-j} \tag{5}$$

When calculating the modular square, the computational flow of traditional FIPS modular multiplication algorithm is the same as modular multiplication algorithm, and needs $2s^2 + s$ multiplication operations. The improved FIPS modular multiplication algorithm uses *equal*, j and $i - j$ signals to distinguish and control the modular multiplication and modular square operation. It reduces unnecessary operations, but also increases the difficulty of hardware implementation.

4.2 Algorithm efficiency

In [22], the optimization for the modular square is also based on FIPS. Nevertheless some operations such operand accesses, multiplications and additions are more redundant in the algorithm, and the operations on the $GF(2^m)$ field are not supported. At the same time, no further analysis is made for the application of improved modular multiplier.

For the FIPS algorithm, the coordinate system is constructed with $i - j$ as the abscissa axis and j as the ordinate axis, as shown in **Fig. 3**. The FIPS algorithm uses the product scanning mode, indicated by the red dotted line where i increments from 0 to $2s-1$ and the dashed arrow points to the ascending order of the b_{i-j} subscript. When $A = B$, all $a_j b_{i-j}$ on the dot matrix is symmetric about the $j = i - j$ line, and the cumulative directions shown in the red dashed line are also symmetric about the $j = i - j$ line. This makes it easier to adopt Algorithm 3 to calculate $\sum_{j=0}^{i} a_j b_{i-j}$ of the front s columns and $\sum_{j=i-s+1}^{s-1} a_j b_{i-j}$ of the rear s columns without extra storage space.

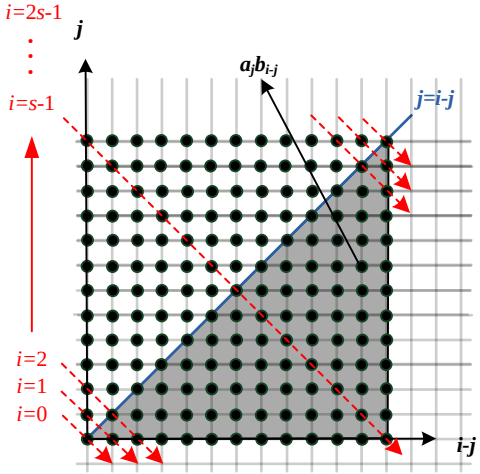

Fig. 3. FIPS Algorithm

According to **Fig. 3**, when $A = B$, the traditional FIPS algorithm needs to complete s^2 w-bit × w-bit product operations in the square matrix and accumulate them. Through the improvement of the FIPS algorithm, only $(3s^2 - s)/2$ multiplication operations in the shadow of the square matrix are needed, and $(3s^2 - s)/2$ multiplication operations are reduced. By Formula 3, 5 and 4, similarly, the read operations on multiplication operands and addition operations on product results are reduced. Their reductions are $s^2 - s$ and $(3s^2 - s)/2$ respectively. The number of operations decreases as the width of operands varies. **Table 1** shows the comparison of the cycles of modular square operations in different word widths, setting the frequency is 500MHz. In modular exponentiation operations of RSA and double point operations of SM2, it can make full advantage of the efficient dual-field FIPS modular multiplication algorithm.

Table 1. Comparison of modular square operation time in different word widths

Word Width	Traditional($\times 10^{-9}$s)	Proposed($\times 10^{-9}$s)	Improvement (%)
4	72	60	16.67
8	272	216	20.59
16	1056	816	22.73
24	2352	1800	23.47
32	4160	3168	23.85
40	6480	4920	24.07
46	9312	7056	24.23
56	12656	9576	24.34
64	16512	12480	24.42

In addition, the difference of algorithm design in dual-field is represented in the following two aspects:

- First, the addition operations in the prime field are carried with carries, and in the binary field, *XOR* operations are performed without carry. Similarly, the accumulation of the partial products in the multiplier is also different.
- Second, due to no carry on binary domain, the final result will not be greater than N, and the subtraction step is omitted.

5. Design of Efficient Dual-field FIPS Modular Multiplication

Although the dual-field efficient FIPS modular multiplication algorithm has a certain degree of efficiency improvement, the data path of the operation is more complex, increasing the difficulty of controller design. The modular multiplication circuit employs resource reuse technology, focuses on the area and power consumption, and at the same time, optimize synthetically the circuit design. It provides an important support for the applications of public key cryptography in resource constrained devices.

5.1 Logic structure

As shown in **Fig. 4**, the logic structure of the efficient dual-field FIPS modular multiplication mainly consists of the following modules.

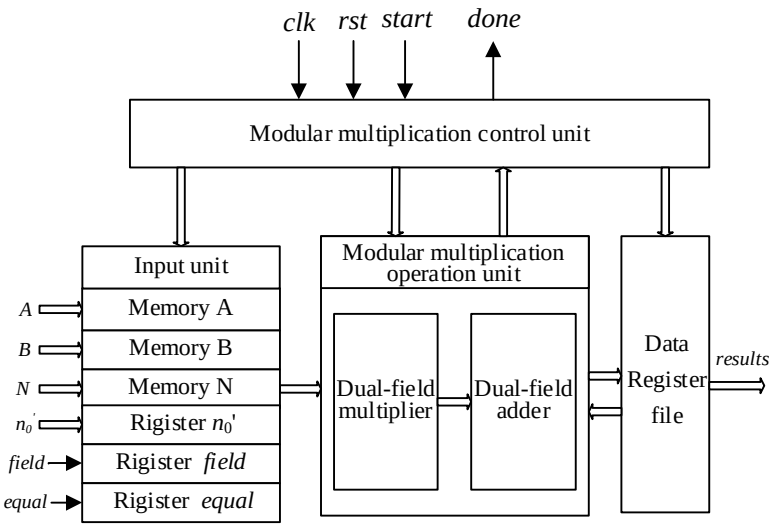

Fig. 4. Logic structure of efficient dual-field FIPS modular multiplication

- **Data input unit**

It uses 32 bit bus for in turn receiving operands including *A*, *B*, *N* and n_0', and storing them in the corresponding memory.

- **Modular multiplication operation unit**

The module is the main part of the operation, which is responsible for the calculation of the critical path of the modular multiplication algorithm. The operation in the dual domain is mainly embodied in the multiplier and adder of this part.

- **Data register file**

It is mainly responsible for the preservation of the intermediate results of the operation process, including the *field* and *equal* signals which are stored in registers. *field* is used to control the selection on the dual fields, i.e., if $field = 0$, the operation on the $GF(p)$ field will be executed; otherwise, the operation on the $GF(2^m)$ field will be executed. *equal* is used for modular square selection, if $equal = 0$, the operands are not equal and the modular multiplication operation will be executed; otherwise, the modular square operation will be executed.

- **Modular multiplication control unit**

The control part of the modular multiplication algorithm is implemented by a state machine, which mainly controls the data flow of the modular multiplication operation circuit.

5.2 Modular multiplication operation unit

The design of the dual field modular multiplication operation unit is mainly for multipliers and adders, e.g. 32×32 bit multiplier, which compresses the partial product in the way of the Wallace tree [24], and obtains respectively the products of $GF(p)$ field and $GF(2^m)$ field. The structure of the dual-field multiplier is designed as shown in **Fig. 5**. For the dual-field adder, *n* dual-field adder units are cascaded in **Fig. 6** to complete the addition operations of two *n*-bit operands in dual fields.

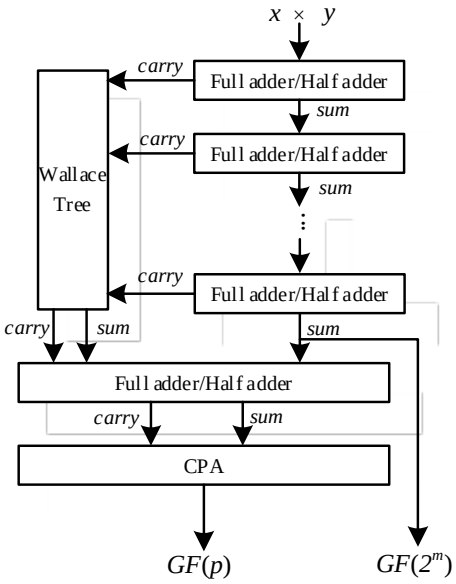

Fig. 5. Dual-field multiplier

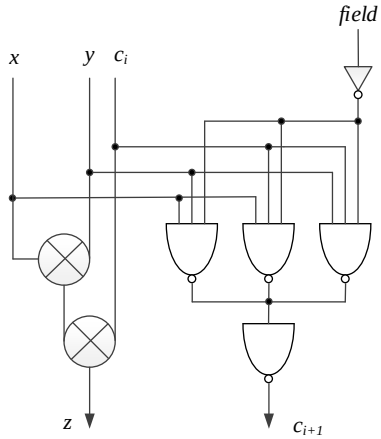

Fig. 6. Dual-field adder

Considering the different requirements of speed, power consumption and area, the critical path of the pipelined modular multiplication circuit is implemented by two schemes. In the case that the application environment requires higher resources, a 32-bit dual-field multiplier and a 96-bit and 65-bit dual-field adders can be used, as shown in **Fig. 7**, in which the data stream and controller are relatively simple. A higher performance implementation is to use two 32-bit dual-field multipliers and two dual-field adders, as shown in **Fig. 8** in which the area is relatively increased and the controller is more complex.

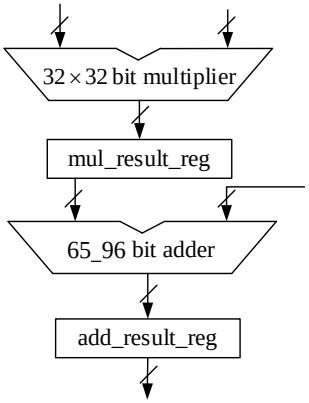

Fig. 7. Data Path of Single-multiplier

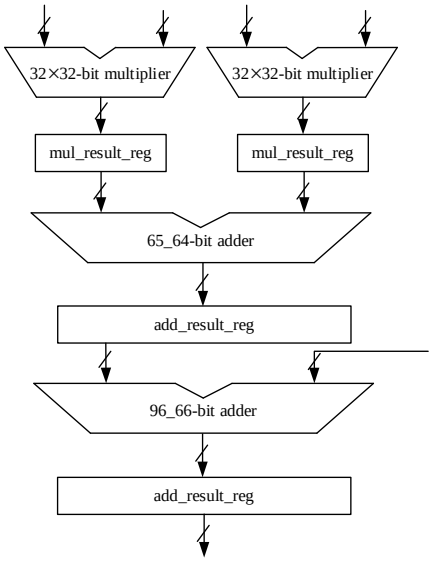

Fig. 8. Data Path of Double-multiplier

The main problem of the efficient dual-field FIPS modular multiplication circuit based on the critical path lies in the following two aspects. One is the operand width which determines the speed, area and power consumption of the circuit. Considering the application of resource-constrained, the multiplier in small width has the advantages of small size and low power consumption, but low speed. The other is the multiplier number. Double multipliers can increase the speed but it will double the area and increase the complexity of the controller.

5.3 Modular multiplication control unit

The modular multiplication control unit is implemented by a state machine, and mainly used to send the address to the RAM for reading a data, control the data flow on the data path, and send the address to the RAM for writing a data, as shown in **Fig. 9**, which includes nine states as follows.

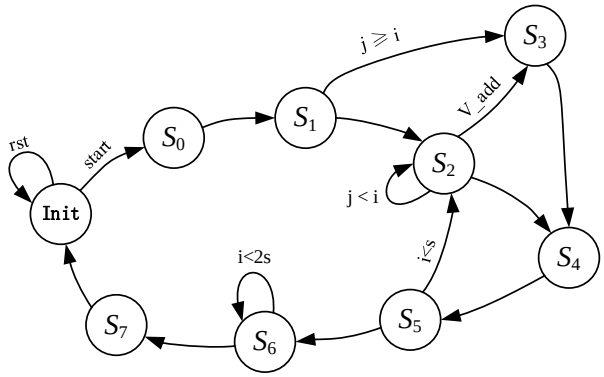

Fig. 9. State Transition of the Single Multiplier

- *Init*: *Initial* state, waiting for data to be transferred to the RAM.
- *S0*: The *start* signal becomes true when the data transmission is over, and then the reading signal is issued to start multiplying.
- *S1*: a_0 and b_0 are read.
- *S2*: Save the result to the register heap after multiplying and adding.
- *S3*: $(v_2, v_1, v_0)_W \leftarrow (v_2, v_1, v_0)_W + a_i b_0$.
- *S4*: $m_i \leftarrow v_0 n_0^{'} \bmod W$.
- *S5*: $(v_2, v_1, v_0)_W \leftarrow (v_2, v_1, v_0)_W + m_i n_0$, and then right shift.
- *S6*: Internal calculation of the second cycle.
- *S7*: Send signal *done* for denoting that the modular multiplication is over.

This machine is triggered by three input signals such as *clk*, *rst* and *start* and will generate an output signal *V_add*.

- *clk*. *clk* is a clock signal.
- *rst*. *rst* is used to initialize the modular multiplication controller and registers.
- *start*. When *start* becomes true, it denotes that the data is loaded and the modular multiplication operation need to be executed.
- *V_add*. In the state S_2, if $i = 0 \,|\, equal = 0$, then $V\_add = 1$, otherwise $V\_add = 0$.
- *done*. The *done* signal is initialized to 0 at the time of *rst* reset or in the *Init* state, and it will be set to be 1 when the operation in the state S_7 is completed.
- *V_Straight* and *V_Shift*. The design employs three 32-bit registers such as V_2, V_1 and V_0 as the additions and registers in the efficient dual-field FIPS modular multiplication, in which V_2 keeps the 32 bits on the MSB (Most Significant Bit) so as to facilitate the shift operations as shown in Algorithm 4. The machine controls the changes of the register value through *V_Straight* and *V_Shift*: a) when $V\_Straight = 1$, the sum value *add_z* of the dual-field adder is saved into the register; b) on the falling edge of *clk* in the S_5 state, i.e., the accumulation of $m_i n_0$ has been completed, $V\_Shift = 1$, the registers will execute the shift operation.
- P_0 - P_3. They correspond to the 4 judgment conditions in **Fig. 2**. They also determine the read-write signal and enable signal of the memory *A*, *B* and *N*, and the bit extension in the dual-field multiplier. When the accumulation of each column in the front *s* columns is completed, P_4 is used to control whether to add $a_i b_0$ to the current column.

- i and j. These two registers and *equal* together decide the value of P_0 - P_3. They also determine the beginning and end of each inner loop and each outer loop.

Under the combined action of the registers and the control signals, the specific pipeline of the dual-field multiplier and adder is as follows.

- The register i is initialized to 0 at the start of the operation. As described in Algorithm 2, this condition does not satisfy the condition of the inner loop of the first outer loop. Thus, the accumulation of $a_i b_0$ can be carried out directly. This is known beforehand before all modular multiplication operations start. Therefore, in the first cycle after the *start* signal is valid, the operands a_0 and b_0 are read directly.
- In the second cycle, the multiply-add operation will be carried out. And, according to *field*, *equal* and P_0 - P_4 and other related signals, the next operands to be calculated will be read.
- According to **Fig. 2**, when Formula 4 and 5 are executed, after each time a_j and b_{i-j} are read, in the next time m_j and n_{i-j} must be read. This facilitates the control of read/write and enable signals of memory.
- When $j < i - j$ and $p_{23} = 1$, $a_j b_{i-j}$ is shifted left.
- When $j > i - j$, only the operand m_j and n_{i-j} are read. In each subsequent cycle, the operations including multiply-add, read, and storing the result of the calculation in the corresponding register stack are executed repeatedly.
- When $A \neq B$, S_6 is always executed. When $A = B$, S_6 is executed only when the first column is computed.
- In the state S_7, the signal $done = 1$ so as to generate the completeness signal of the modular multiplication operation.

In order to perform respectively 1024-bit modular multiplication operation and modular square operation with the pipeline organization,in traditional FIPS algorithms, 2084 and 1044 clock cycles are required, and only 1588 and 796 clock cycles are required in the improved FIPS algorithm.

6. Experiment and Simulation

In the case of single multiplier and double multiplier setting frequency of 500MHz, the cycle number and the optimization rate of modular multiplication and modular square operation under different word width are shown in **Fig. 10**. The number of cycles is more than 4 cycles compared to the theoretical cycles shown in **Table 1**, and the optimization rate is equal to the ratio of the number of reduced cycles to the number of cycles of the modular multiplication. For n-bit operands, the number of cycles of modular square is reduced by $(s^2 - s)/2$, where $s = n/32$. The optimization rate increases as the number of bits of operands increase. When s tends to infinity, the optimization rate approaches 25%.

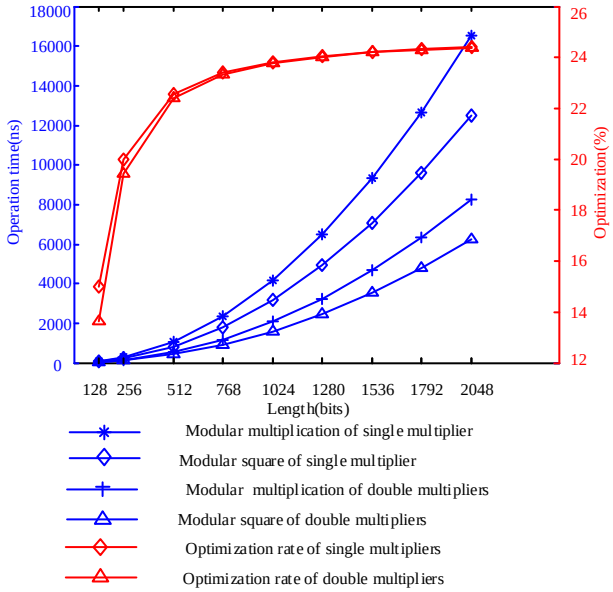

Fig. 10. Cycles comparison of modular multiplication and modular square in different word widths

When the number of 1 in the key is half of the width in different word width, the number of cycles between L_R and ML is shown in **Fig. 11**, setting the frequency is 500MHz. L_R guarantees the maximum utilization of the modular square circuit. The speed of L_R is 1.25 times of ML which needs two modular multipliers.

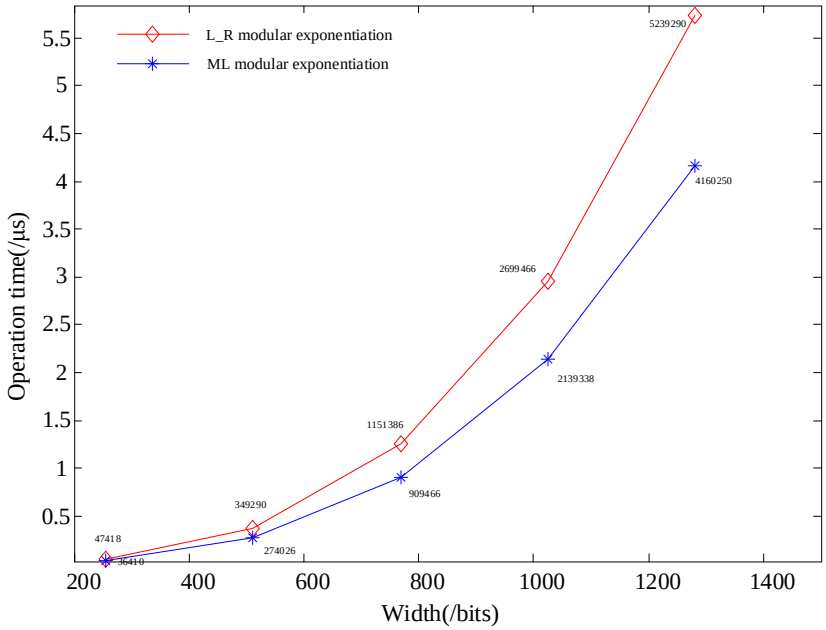

Fig. 11. Cycles comparison of L_R and ML in different word widths

The synthesis results of 1024-bit modular multiplication on the Xilinx Artix-7^{TM} FPGA device are compared as shown in **Table 2**. Due to the different FPGA devices with different maximal clock frequency, in order to evaluate more objectively, we mainly list the area (LUTs)

and the latency (cycles), and compare them with the throughput AT^2 [15], in which A is the area (LUTs) and T is the latency (cycles). Different from [31] and [32], the proposed scheme supports dual-field operations. The AT^2 in the proposed design is slightly larger than Sudhakar [33], but the number of LUTs is smaller. Compared with other modular multiplication circuits, the proposed method has good AT^2 throughput with less power consumption.

Table 2. Implementation comparison of modular multiplication circuits

Design	Area(LUTs)	Latency(μs)	AT^2(LUTs×μs^2)
Ours	2557	3.18	25857.41
Verma [27]	5718	2.68	41068.96
Erdem [28]	34904	0.91	28904.00
Dai [29]	6047	4.00	96752.00
Wu [30]	11361	2.89	94888.21
Wang [34]	5430	4.05	89065.58

7. Conclusions and Future Works

Through optimizing the logic of the modular square based on the FIPS Montgomery modular multiplication algorithms, we can effectively improves the efficiency of the public key cryptography algorithm and reduces the power consumption of redundant operations, so as to make it suitable for use in resource constrained devices. Experimental results show that the proposed circuit increases 23.8% speed compared to the traditional FIPS on 1024-bit modular square. In addition, the circuit has strong expansibility, and the supported modular multiplication length can be increased according to the actual demand.

The dual-field FIPS modular multiplication supports both $GF(p)$ and $GF(2^m)$ fields, and provides the basic modular operation for the RSA and SM2 cryptosystems. Based on the dual-field FIPS modular multiplication, some modular operations such as modular square and modular exponentiation can be implemented for the RSA cryptosystem. However, some high-level operations over SM2 modular arithmetic layer including the single point operation layer and the multiple point operation layer are not addressed. It is a worthwhile direction to employ the efficient dual-field FIPS modular multiplication module to perform point operations so as to realize the hardware support of the dual-field modular multiplication for both RSA and SM2 cryptosystems. In addition, the operations on dual-filed include multiplication and addition. An arithmetic unit supporting both multiplication and addition needs be be further optimized in logic design in order to reduce logic resource and power consumption.

References

[1] F. Chen, Y. Luo, J. Zhang, J. Zhu, Z. Zhang, C. Zhao and T. Wang, “An infrastructure framework for privacy protection of community medical internet of things-Transmission protection, Storage Protection and Access Control,” *World Wide Web*, vol. 21, no. 1, pp. 33-57, 2018. Article (CrossRef Link)

[2] N. Rajitha and R. Sridevi, “Implementations of Reconfigurable Cryptoprocessor A Survey,” in *Proc. of Third International Conference of Information Systems Design and Intelligent Applications*, pp. 11-19, 2016. Article (CrossRef Link)

[3] F. Gandino, F. Lamberti, G. Paravati, et al., “An Algorithmic and Architectural Study on Montgomery Exponentiation in RNS,” *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1071-1083, 2012. Article (CrossRef Link)

[4] J. Wei, W. Guo, H. Liu, et al, "A Unified Cryptographic Processor for RSA and ECC in RNS," in *Proc. of CCF National Conference on Compujter Engineering and Technology*, pp. 19-32, 2013. Article (CrossRef Link)
[5] W. C. Lin, J. H. Ye and M. D. Shieh, "Scalable Montgomery Modular Multiplication Architecture with Low-Latency and Low-Memory Bandwidth Requirement," *IEEE Transactions on Computers*, vol. 63, no. 2, pp. 475-483, 2014. Article (CrossRef Link)
[6] G. Hachez and J. J. Quisquater, "Montgomery Exponentiation with no Final Subtractions: Improved Results," in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 293-301, 2000. Article (CrossRef Link)
[7] J. Shao, L. Wu and X. Zhang, "Design and Implementation of Long Integer Modular Exponentiation Unit of Asymmetric Encryption in Smart Card," *Microelectronics & Computer*, vol. 32, no. 2, pp. 37-41, 2015. Article (CrossRef Link)
[8] M. Li, D. Wu, K. Dai and X. Zou, "Research and Design of a High-Performance Scalable Public-Key Cipher Coprocessor," *Acta Electronica Sinica*, vol. 39, no. 3, pp. 665-670, 2011. Article (CrossRef Link)
[9] G. Chen,J. Zhu, M. Liu and W. Zeng, "Dual-field Modular Multiplication Algorithm and Modular Inversion Algorithm with VLSI Implementation," *Journal of Electronics & Information Technology*, vol. 32, no. 9, pp. 2095-2100, 2010. Article (CrossRef Link)
[10] M. M. A. Kadar and A. V. Ananthalakshmi, "An energy efficient Montgomery modular multiplier for security systems using reversible gates," in *Proc. of IEEE International Conference on Communications and Signal Processing*, pp. 0071-0074, 2015. Article (CrossRef Link)
[11] X. Qi, Q. Tang, F. Chen, et al, "Design of Modular Inversion Circuits Using Reversible Logic on Galois Field," *Journal of Frontiers of Computer Science & Technology*, vol. 9, no. 5, pp. 555-564, 2015. Article (CrossRef Link)
[12] P. L., "Montgomery Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519-521, 1985. Article (CrossRef Link)
[13] G. Wu, X. Xie, D. Wu, et al, "Design and implementation of high radix Montgomery modular multiplication array structures," *Computer Engineering and Science*, vol. 36, no. 2, pp. 201-205, 2014. Article (CrossRef Link)
[14] J. H. Ye, T. W. Hung and M. D. Shieh, "Energy-efficient architecture for word-based Montgomery modular multiplication algorithm," in *Proc. of International Symposium on VLSI Design, Automation and Test*, pp. 1-4, 2013. Article (CrossRef Link)
[15] A. P. Renardy, N. Ahmadi, A. A. Fadila, et al, "Hardware implementation of montgomery modular multiplication algorithm using iterative architecture," *International Seminar on Intelligent Technology and ITS Applications*, pp. 99-102, 2015. Article (CrossRef Link)
[16] M. Morales-Sandoval and A. Diaz-Perez, "Scalable GF(p) Montgomery multiplier based on a digit–digit, computation approach," *Iet Computers & Digital Techniques*, vol. 10, no. 3, pp. 102-109, 2016. Article (CrossRef Link)
[17] E. Savas and A. F. Tenca, "A Scalable and Unified Multiplier Architecture for Finite Fields GF(p) and GF(2m)," in *Proc. of International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 277-292, 2000. Article (CrossRef Link)
[18] Z. Zheng, Y. Zi, Y. Tian, et al, "Design and Application of High Speed Dual-Field Multiplier," *Microelectronics and Computer*, vol. 33, no. 5, pp. 1-5, 2016. Article (CrossRef Link)
[19] W. Liao, M. Wan, K. Dai, et al, "Design and research of dual-field scalable modular multiplier," *Huazhong Univ. of Sci. and Tech. (Natural Science Edition)*, vol. 43, no. 9, pp. 51-54, 2015. Article (CrossRef Link)
[20] G. R. Blakely, "A computer algorithm for calculating the product AB modulo M," *IEEE Transactions on Computers*, vol. 32, no. 5, pp. 497-500, 1983. Article (CrossRef Link)
[21] P. Barrett, "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor," *Proceedings of Advances in Cryptology*, pp. 311-323, 1986. Article (CrossRef Link)
[22] Q. Shao, "Improvment of RSA Crypitography Algorithm and Implementation of Its IP Core," *Shanghai Jiao Tong University*, 2014. Article (CrossRef Link)

[23] M. Joye and S. M. Yen, "The Montgomery powering ladder," in *Proc. of 4th International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 2523, pp. 291-302, 2002. Article (CrossRef Link)
[24] H. Bansal, K. G. Sharma and T. Sharma, "Wallace Tree Multiplier Designs: A Performance Comparison Review," *Innovative Systems Design & Engineering*, vol. 5, no. 5, pp. 60-67, 2014. Article (CrossRef Link)
[25] L. Chen, W. Sun, X. Chen, et al, "Montgomery Modular Inversion Algorithm Based on Signed Digit System and Hardware Implementation," *Acta Electronica Sinica*, vol. 40, no. 3, pp. 489-494, 2012. Article (CrossRef Link)
[26] E. A. Kuzu and A. Tangel, "A new style CPA attack on the ML implementation of RSA," in *Proc. of IEEE Computer Science and Engineering Conference*, pp. 323-328, 2014. Article (CrossRef Link)
[27] Verma R., Dutta M. and Vig R., "RSA Cryptosystem Based on Early Word Based Montgomery Modular Multiplication," *SERVICES 2018 in Computer Science, Springer*, vol. 10975, pp. 33-47, 2018. Article (CrossRef Link)
[28] S.S. Erdem, T. Yanık and A. Çelebi, "A general digit-serial architecture for montgomery modular multiplication," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 25, no. 5, pp.1658-1668, 2017. Article (CrossRef Link)
[29] W. Dai, D. D. Chen, R. C. C. Cheung and Ç. K. Koç, "Area-Time Efficient Architecture of FFT-Based Montgomery Multiplication," *IEEE Transactions on Computers*, vol. 66, no. 3, pp. 375-388, 1 March 2017. Article (CrossRef Link)
[30] T. Wu, "Improving radix-4 feedforward scalable montgomery modular multiplier by precomputation and double booth-encodings," in *Proc. of 2013 3rd International Conference on Computer Science and Network Technology*, pp. 596-600, 2013. Article (CrossRef Link)
[31] M.-D. Shieh, J.-H. Chen, H.-H. Wu, and W.-C. Lin, "A New Modular Exponentiation Architecture for Efficient Design of RSA Cryptosystem," *IEEE Transactions on VLSI Systems*, vol. 16, no. 9, pp. 1151-1161, 2008. Article (CrossRef Link)
[32] S. S. Erdem, T. Yanik and A. Celebi, "A General Digit-Serial Architecture for Montgomery Modular Multiplication," *IEEE Transactions on VLSI Systems*, vol. 25, no. 5, pp. 1658-1668, 2017. Article (CrossRef Link)
[33] M. Sudhakar, R.V. Kamala and M.B. Srinivas, "A bit-sliced, scalable and unified Montgomery multiplier architecture for RSA and ECC," in *Proc. of IFIP International Conference on Very Large Scale Integration*, pp. 252-257, 2007. Article (CrossRef Link)
[34] S. Wang, W. Lin, J. Ye and M. Shieh, "Fast scalable radix-4 Montgomery modular multiplier," in *Proc. of 2012 IEEE International Symposium on Circuits and Systems*, pp. 3049-3052, 2012. Article (CrossRef Link)

Tingting Zhang received her bachelor's degree from Anhui Normal University, China, in 2018. She is currently a graduate student at School of Computer and Information, Anhui Normal University. Her research interest is cryptography.

Junru Zhu received her master degree from Anhui Normal University, China, in 2017. Her research interest is digital systems.

Yang Liu received his bachelor's degree from Anhui Normal University, China, in 2018. He is currently a graduate student at School of Computer and Information, Anhui Normal University. His research interest is cryptography.

Fulong Chen received the Master degree in computer application technology from China West University in 2005 and recieved his Ph.D. degree in Computer Science and Technology from Northwestern Polytechnical University in 2011. He is currently a Professor at School of Computer and Information, Anhui Normal University, China. His research interests are embedded and pervasive computing, cyber-physical systems, high-performance computer architecture, security of the Internet of things.