Improvement of Inverse Algorithm of Modulus in Finite Field for Elliptic Curve Encryption System

Dandan Su

Foshan Polytechnic, Foshan, Guangdong 528137, China

Keywords: Elliptic curve, Gf(2m), Finite field, Inversion operation

Abstract: In this paper, we mainly study the improvement of the finite field inverse algorithm of elliptic curve encryption system. In this paper, by studying the arithmetic operation of elliptic curve finite field, the idea of parallel modular multiplication is put forward, and the traditional Euclidean inverse algorithm is improved, so that it can complete modular and inverse operation at the same time. Combining the inverse operation in the prime field and the binary field can greatly improve the operation efficiency of the algorithm, which is conducive to improving the decryption speed of ECC. In addition, NTL has obvious advantages in the selection of irreducible polynomials, which is more convenient than Crypto++, and the operation efficiency of modular multiplication, modular square and modular remainder is also very high. In this way, the fast operation on the base field accelerates the dot multiplication operation on the elliptic curve, so NTL algorithm library has great advantages when applied to elliptic cryptosystems on GF(2m) field.

1. Introduction

In 1985, N. Koblitz[1] and V. Miller[2] respectively introduced elliptic curve theory into cryptography. Since then, the study of elliptic curve cryptosystem (ECC) has gradually become an active branch of cryptography. The finite field inversion operation is the foundation of ECC and hyperelliptic curve cryptography (HECC). The key to realize ECC and HECC is to realize the inverse of finite field quickly. In the implementation of elliptic curve encryption system, modular inverse algorithm is often the bottleneck of algorithm implementation, because modular inverse operation involves a large number of division and subtraction operations, in which division operation takes up a lot of space and time. Point multiplication operation is the key to realize elliptic curve cryptography, which depends on the operation of the underlying finite field, so the effective implementation of arithmetic operation in the finite field is the prerequisite for the high efficiency of elliptic curve cryptography [3].Compared with RSA (rivest-Shamir-adleman public key crypto system), ECC does not need to generate large prime numbers and perform prime detection at first, and can provide the same security level with shorter key length. In fact, ECC is one of the most popular encryption algorithms that can provide the highest bit encryption strength. Therefore, it is considered to be the most interesting encryption algorithm, and it is a hot spot in the field of information security.

In this paper, the elliptic curve in $GF(2^m)$ is adopted as the hardware implementation scheme. By analyzing the algorithm of domain operation layer in binary domain, the arithmetic operation and structure design of the underlying finite domain are realized by designing parallel modular multiplication algorithm and improved modular inverse algorithm.

2. Elliptic Curve Encryption System

Elliptic curves used in ECC are defined on finite fields, and the type and size of finite fields determine the security and effectiveness of ECC. ECC usually chooses $GF(p)$ (p is a big prime number) and $GF(2^m)$ (finite field with characteristic 2) as its base field. It can be regarded as m -dimensional vector space on $GF(2)$. m elements on $GF(2^m)$ constitute a set $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$. If

Copyright © (2020) Francis Academic Press, UK DOI: 10.25236/mmmce.2020.040

these elements are linearly independent, they constitute a group of bases of $GF(2^m)$. Any element b on $GF(2^m)$ is uniquely expressed in the following form:

$$B = \sum_{i=0}^{m-1} b_i \alpha_i, \alpha_i \in GF \quad (1)$$

Therefore, B can also be expressed as a vector $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ composed of coefficients. Generally speaking, $GF(2^m)$ has many kinds of bases, and this paper mainly involves two kinds of bases, namely, polynomial base and normal base.

The Weierstrass equation on the binary finite field $GF(2^m)$ is shown in formula (2):

$$y^2 + xy = x^3 + cx^2 + d, c, d \in GF(2^m), d \neq 0 \quad (2)$$

When $d = 1$, it is called Koblitz curve. This kind of curve is the fastest in the implementation of elliptic curve cryptosystem [4]. The security problem of elliptic curve cryptosystem is the discrete logarithm problem based on elliptic curve, which is why it is superior to public key cryptosystems including RSA. For any $x, y \in GF(p), p(x, y) \in E(EF(p))$, if and only if the x and y coordinates of $p = (x, y)$ satisfy equation (2)

The elliptic curve cryptosystem based on the finite field is designed based on the elliptic curve discrete logarithm problem (ECDLP), that is, assuming that the two points P and Q on the elliptic curve are known and satisfy $Q = kP$, how to find k? This is very difficult. The difficulty is no less than the factorization of large integers.

3. Improvement and Analysis of Modular Inverse Algorithm in $GF(2^m)$ Domain

Two time-consuming operations in finite field $GF(2^m)$ are modular multiplication and modular inverse. Generally speaking, the latter requires more computation time and more complex circuits. There are two commonly used methods for inverse operation of elements in $GF(2^m)$ domain: (1) Inverse method based on Fermat's small theorem; (2) Method based on extended Euclidean algorithm and its variation [5-6].

I/O privacy is based on the typical indistinguishability theory, which can guarantee that no useful information about I/O will be revealed. More intuitively, if the encryption algorithm Encrypt can output indistinguishable results for different inputs, then this computing outsourcing algorithm is input privacy. Then, the definition of input privacy is given through an experiment.

$Exp_B^{Inprivacy}[x, m]$

Requests and responses:

$For\ i = 1, \cdots, n$

$-y_i \leftarrow B(y_0, \alpha_{x_0}, \cdots, y_{i-1}, \alpha_{x_{i-1}})$

$-\gamma_i \leftarrow KeyGen(1^m), \alpha_{x_i} \leftarrow Encrypt(\gamma_i, y_i)$

Challenge:

$-(y^{(0)}, y^{(1)}) \leftarrow B(y_0, \alpha_{x_0}, \cdots, y_z, \alpha_{x_z})$

$-d \leftarrow R\{0,1\}, \gamma^{(d)} \leftarrow KeyGen(1^m), \alpha_{x^{(d)}} \leftarrow Encrypt(\gamma^{(d)}, y^{(d)})$

$-\hat{d} \leftarrow B(y_0, \gamma_{x_0}, \cdots, y_z, \gamma_{x_z}, \delta_z \gamma_{x^{(d)}})$

Output 1 if $\hat{d} = d$, otherwise, output 0.

In the query and response stage, the adversary is given access to *KeyGen* and *Encrypt* prediction machines. In the challenge stage, if adversary B has the ability to distinguish the output result of encryption algorithm, namely $\hat{d} = d$, it is considered that adversary won the

experiment.

It can be seen from the product formula that the process of modular multiplication includes two processes: multiplication and modulo taking. Traditional multiplication is to multiply two m-bit operands and then $f(x)$-modulo them. Multiplication in binary domain has many forms, including serial structure, parallel structure and digital parallel structure combining serial and parallel. In order to improve the efficiency of modular multiplication algorithm, refer to the typical shift serial algorithm introduced in reference [7] and further optimize it, among which the algorithm in reference [8] is as follows:

$INPUT: B(x) = b_{m-1}x^{m-1} + \cdots + b_1 x + b_0, C(x) = c_{m-1}x^{m-1} + \cdots + c_1 x + c_0$

Reduce polynomial $f(x)$.

$OUTPUT: B(x) \cdot C(x) \bmod f(x)$

$if b_0 = 1, then\ D = C; else\ D = 0$

$i\ \ from\ 1\ to\ n-1, do:$

$c = c(<<1) \bmod f(x)$

$if\ \ b_i = 1, then\ d = d \bmod c$

$return\ d$.

In this algorithm, a clock is shifted once, and then the XOR operation is carried out in the next step. Then, m clocks are needed to complete the m-bit operation, which takes a long time. According to the characteristics of binary modular subtraction operation, it is actually the highest order value to judge the modulus.

In order to get c/b, we can modify the algorithm and replace $r_1(x) \leftarrow z$ with $r_1(x) \leftarrow c$. When $m = 1$, the algorithm stops and gets $r_1(x) = cb^{-1}$.

Because $r_1(x)$ in the algorithm is located at the standard length in the iteration step, that is to say, the initial value of $r_1(x)$ has little influence on the execution time of the whole algorithm, so the cost of the improved division algorithm is the same as that of the algorithm.

Let's assume that points on elliptic curves are represented by affine coordinates. If the inversion operation is carried out, one inversion operation (denoted by "T") and two multiplication operations (denoted by "U") are needed to calculate the addition of two different points, that is, the required calculation amount is $T + 2U$; If the division operation is carried out, calculating the addition of two different points requires an inversion operation (denoted by "T") and a multiplication operation (denoted by "U"), that is, the required calculation amount is $T + M$. When the value of T/U is small, it is meaningful to make such improvement. For example, if $T/U = 3$, then

$$\frac{(T+2U)-(T+U)}{T+2U} = \frac{U}{5U} = 0.2 \quad (3)$$

Obviously, if the algorithm is used for division, it will reduce the overhead of point addition by 21%. However, when $T/U > 6$,

$$\frac{(T+2U)-(T+U)}{T+2U} < \frac{U}{8U} = 0.125 \quad (4)$$

That is, the reduction of calculation overhead will not exceed 12.4%. Therefore, if the ratio of T/U is not small, it is possible to use the strategy of reducing the number of inversion operations, so that the point addition operation in the elliptic curve number multiplication operation involves the inversion of relatively few fields, so as to offset the overhead savings obtained by the division operation.

4. Algorithm Performance Analysis

Using VHDL as a design tool, the improved modular inverse algorithms in finite fields $GF(2^{152}), GF(2^{237})$ and $GF(2^{296})$ are implemented quickly. The functional simulation and testing of the module are divided into two steps: (1) The functional simulation of the realization results is carried out by using ModelSim software platform to ensure the correctness of the logic design. (2) using dual-port RAM technology to connect the module with the microcontroller, using Quartus II software platform, selecting the device environment as CycloneIIEP2C35F672C6 to complete the logic synthesis and timing simulation of the module, and downloading it to FPGA development environment for testing. The simulation and test all get correct results.

We compare the speed of modular inverse operation between NTL5_3_2 and Crypto++5.1 in finite field $GF(2^m)$ by programming. The result shows that NTL is 5-10 times faster than Crypto++5 .1. Table 1 gives some test results.

Table 1 Comparison of 10^4 Modular Inverse Operations (Unit: Second)

Algorithms library	$GF(2^{152})$	$GF(2^{237})$	$GF(2^{296})$
Crypto++5.1	4.03	5.66	7.41
NTL5_3_2	0.27	0.42	0.46

From the comparison results in Table 1, the operation efficiency of NTL on the finite field $GF(2^m)$ is still higher than that of Crypto++. In fact, the operation of Crypto++ to deal with the finite field $GF(2^m)$ is also oriented to the binary stream, and it also uses 32bit and 64bit words for block processing, but because the algorithm processing is not as good as NTL, the operation speed is less than NTL

5. Conclusion

By analyzing one of the general modular inverse algorithms in finite field $GF(2^m)$, this paper points out the difficulties in hardware implementation of this algorithm, and proposes an improved form. In order to improve the execution efficiency of inversion, an algorithm of outsourcing inversion to cloud server is proposed. It is proved that the algorithm satisfies verifiability, input privacy and output privacy. Compared with other algorithms, this design greatly improves the operation speed, consumes less resources, achieves a good compromise between area and speed, and has a higher neutral price in published literature. The effective implementation of ECC depends on the quick implementation of dot multiplication and dot addition, while the effective implementation of dot addition and dot multiplication depends on the finite field arithmetic operation that defines the curve. In this way, the fast operation on the base field accelerates the dot multiplication operation on the elliptic curve, so NTL algorithm library has great advantages when applied to elliptic cryptosystems on $GF(2^m)$ field.

References

[1] Zhang Qiang, Qu Yingjie. Research on VLSI implementation method of GF(2^m) domain elliptic curve finite domain. Information Technology, No. 12, pp. 123-128, 2017.

[2] Yu Wei, Li Bao, Wang Kunpeng, et al. Co-Z Montgomery algorithm for elliptic curves on the finite field of feature 3. Journal of Computer Science, Vol. 40, No. 005, pp. 1121-1133, 2017.

[3] Zhang Qiang, Qu Yingjie. Research and implementation of finite field over GF (2m)elliptic curve based on VLSI%GF(2^m). Information Technology, No. 012, pp. 115-120, 2017.

[4] Liu Haifeng, Lu Kaiyi, Liang Xingliang. Encryption algorithm of elliptic curve lattice group on finite field GF(2~8) based on normal basis representation. Journal of Wuhan University of Science

and Technology, Vol. 41, No. 05, pp. 78-83, 2018.

[5] Li Chao, Zhang Qiang, Qu Yingjie. Research on VLSI Implementation Method of Domain Elliptic Curve Point Multiplication. Computer Measurement and Control, No. 12, pp. 232-236, 2017.

[6] Liu Zengfang, Wei Xingjia, Lu Dianjun. Data simulation of elliptic curve addition. Journal of Shaoyang University (Natural Science Edition), Vol. 015, No. 003, pp. 8-13, 2018.

[7] Yang Bo, Meng Lilin, Tao Qiong. Design and Implementation of Modular Multiplication and Inverse of F_P Domain Based on FPGA. Microelectronics and Computers, Vol. 34, No. 005, pp. 54-58, 2017.

[8] Liu Li, Xu Ming. Design and implementation of an anti-side channel attack elliptic curve scalar multiplication algorithm. Computer Application Research, No. 2, pp. 508-513, 2017.