FAST IMPLEMENTATION OF ELLIPTIC CURVE CRYPTOGRAPHIC ALGORITHM ON GF(3^M) BASED ON FPGA

Tan Yongliang, He Lesheng, Jin Haonan and Kong Qingyang

Information Institute, Yunnan University, Kunming, China

ABSTRACT

As quantum computing and the theory of bilinear pairings continue being studied in depth, elliptic curves on GF(3^m) are becoming of an increasing interest because they provide a higher security. What's more, because hardware encryption is more efficient and secure than software encryption in today's IoT security environment, this article implements a scalar multiplication algorithm for the elliptic curve on GF(3^m) on the FPGA device platform. The arithmetic in finite fields is quickly implemented by bit-oriented operations, and then the computation speed of point doubling and point addition is improved by a modified Jacobia projection coordinate system. The final experimental results demonstrate that the structure consumes a total of 7518 slices, which is capable of computing approximately 3000 scalar multiplications per second at 124 Mhz. It has relative advantages in terms of performance and resource consumption, which can be applied to specific confidential communication scenarios as an IP core.

KEYWORDS

GF(3^m), Elliptic Curve Cryptography, Scalar Multiplication, FPGA, IoT Security

1. INTRODUCTION

Ellipse Curve Cryptography (ECC) has advantages including a relatively short key size, a high security and the applicability to resource-constrained embedded products, which is widely used as an Advanced Encryption Standard (AES) in symmetric encryption algorithms[1] and has been a hot research topic in the application of cryptography in recent years. The current version 1.3 of the transport layer protocol highlights the growing importance of elliptic curve cryptographic algorithms[2], and with an increasing demand for security of data privacy in the IoT of modern network society, the key length of ECC required is getting longer and longer, which makes the implementation of traditional software more and more difficult while less and less efficient to achieve. Therefore, in today's IoT environment, software encryption is only applicable to general network security[3], and hardware encryption has undoubted advantages over software encryption in some areas that are extremely sensitive to security. According to the current status, the hardware implementation of the scalar multiplication algorithm on the binary and prime fields has been relatively well studied, while relatively little work has been done on GF(3^m). While Galbraith[4] experimentally pointed out that for Weil or Tate pairing-based cryptosystems, the security of GF(3^m) is higher in terms of bandwidth efficiency and security. Shen Shao[5] and other authors also proved that elliptic curves on GF(3^m) also had some properties similar to the fast computation of those on GF(2^m) in terms of computational efficiency. Other studies on GF(3^m) have only improved to reduce the computational complexity at the algorithm level[6,7]. Therefore, due to the lack of research on the hardware implementation of the scalar multiplication algorithm for elliptic curves on GF(3^m), the scalar multiplication algorithm for elliptic curves on GF(3^m) is

David C. Wyld et al. (Eds): COMIT, CRBL, BIOM, WiMNeT, SIP, AISO - 2021
pp. 91-100, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.111709

designed and implemented in this paper on the FPGA device platform. The arithmetic is quickly implemented by bit-oriented operation like binary fields, and a modified Jacobian projection coordinate system is used to increase the speed of point addition and point doubling. Meanwhile, point addition and point doubling operation are designed as a whole to improve the resource reuse rate. The finally implemented scalar multiplication structure has certain advantages over traditional fields in terms of resource consumption and computing speed, and is suitable for use as a secure cryptographic algorithm carrier in various communication scenarios. Next, this paper introduces the relevant theoretical knowledge in the second part, and the third part focuses on the arithmetic design and implementation over a finite field of characteristic 3. The fourth part focuses on the general design and implementation of the scalar product structure. And in the fifth part we find some related work to do the performance comparison analysis. At the end of this paper, we have made a summary of our work. It also gives directions for further improvement.

2. RELATED KNOWLEDGE

The security of elliptic curve cryptosystems is based on the computational Diffie-Hellman problem in order-n subgroups (ECDLP security) and that in finite fields (MOV security)[8]. The Weierstrass equation[9] for an elliptic curve is an elliptic curve E(K) defined over a field K with the following expression:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_5 \quad (1)$$

When K = GF(3^m), we call E(K) an elliptic curve defined on GF(3^m). An abelian Group can be formed based on the set of all solutions on an elliptic curve E(K) plus one infinity point, and the algorithm in the exchange group formed should satisfy the basic theory of groups. The following describes the group operators on elliptic curves defined on field K and char (K) = 3. The expression of E(K) is given in equation (2):

$$E(K) = \{(x, y) \in K \times K \mid y^2 = x^3 + ax + b\} \bigcup \{o\} \quad (2)$$

Set $P = (x_1, y_1)$, $Q = (x_2, y_2)$ as two points on the elliptic curve, and according to the definition of group theory, there are $-\mathrm{O} = \mathrm{O}$, $-P = (x_1, -y_1)$ and $P + \mathrm{O} = \mathrm{O} + P = P$. Therefore, the expression of the third point obtained by adding two points on the elliptic curve on GF(3^m) has the following operating rules.

1) Point Addition

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = y_1 + y_2 - \lambda^3 \end{cases}, \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

2) Point Doubling

$$\begin{cases} x_3 = \lambda^2 + x_1 \\ y_3 = -\lambda^3 - y_1 \end{cases}, \quad \lambda = -\frac{a}{y_1}$$

3. DESIGN AND IMPLEMENTATION ALGORITHMS ON GF(3^M)

3.1. Addition and Subtraction

In binary fields, hardware implementation of addition and subtraction operation among polynomial elements is simply a matter of dissimilarity by corresponding bits. So, in order to improve the performance of the algorithm in the finite fields of characteristic there, the coefficients of each polynomial element in GF(3) are stored as two values q2 and q1, of which q2

stores the higher of the polynomial coefficients, while q1 stores the lower of them. q1 and q2 increase in size as the number of polynomials increases. By saving the high and low bits of each coefficient as two separate values, it is possible to perform arithmetic operation that is directly bit-oriented as arithmetic operation in the binary fields, and this basic logic is easy for FPGAs to implement, improving the efficiency of operation. For example, the polynomial element $2x^6+x^5+x^3+2x^2+2x+1$ is represented in Figure 1.

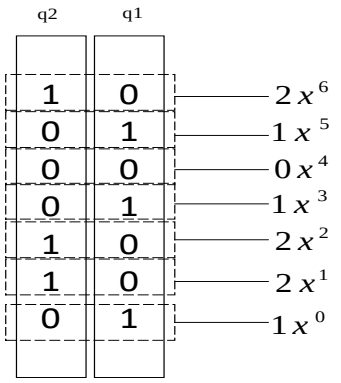

Figure 1. Polynomial element representation

The arithmetic operation of adding two polynomial elements with their coefficients is done in field GF(3), and the operation of adding two polynomials expressed through the above new method can be done using the basic operating logic. For example, the specific hardware flow implementation of adding polynomial A(x) to polynomial B(x) to obtain C(x) is shown in Figure 2.

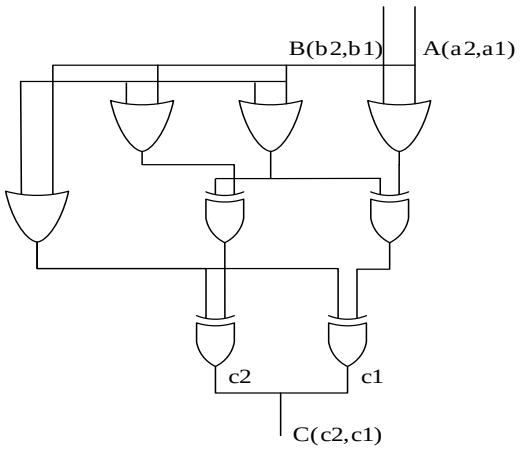

Figure 2. Hardware structure of addition

3.2. Multiplication

Multiplication in finite field is the most critical module in the entire design. Its operation is based on the principle of multiplying two polynomials and then taking the modulus P(x). P(x) is an irreducible polynomial defined in a finite field. A fully parallel modulo multiplier was initially considered to be implemented in this design, but the area and power consumption would be far more than expected, which would not be suitable for most practical cryptographic applications. So, an all-bit serial design was used to implement the multiplication operation. Repeatedly shifted

the multiplicative polynomial down to one place while shifting the multiplied polynomial up to the other to perform the multiplication operation. Then, a corresponding bit of the multiplied polynomial was added or subtracted from the output value in each iteration depending on whether the lowest significant bit in multiplicative polynomial q1 or q2 was set as 1. This is shown in Figure 3.

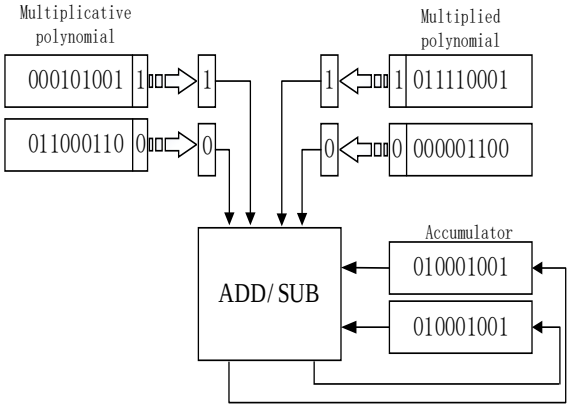

Figure 3. Multiplication of polynomial illustration

The advantage of this all-bit serial approach is that it does not require large intermediate storage, and a large amount of shift operation is more suitable for FPGAs to implement. Using a basic iterative structure and simple logic cells, neither a direct multiplier nor addition circuits are required, saving hardware resources. However, its disadvantage is that the calculation speed is relatively slow.

3.3. Inversion

Inverse is the most time and resource consuming operation in finite fields. Since the extended Euclidean algorithm effectively avoids the division operation, the use of this algorithm for inversion can effectively improve the calculation speed. The pseudocode for computing the inversion in hardware using the extended Euclidean algorithm is shown in Figure 4.

```
Algorithm 1

  Input: A(X)
  Output: A(X)^-1
1.  S:=P(x); R:=A(x); U:=1; V:=0; d:=0;
2.  q= S[`MOST]/R[`MOST]
3.  FOR i:=1 to 2m DO
4.  IF R[`MOST]=0
5.   THEN R:=x*R; U:=(x*u)modP(x); d:=d+1;
6.  ELSE IF d[0]=0
7.  THEN R:=x*(S-q*R); S:=R;
8.  U:=x*(V-qU)modP(x) V:=U; d:=d+1;
9.  ELSE IF d[0]!=0
10. THEN S:= x*(S-q *R); V:=V-q*U;
11. U:=(U/x)modP(x); d:=d-1;
12. END;
13. ( A(X)^-1=U/R[`MOST])
```

Figure 4. Algorithm for inversion in GF(3^m)

4. SCALAR MULTIPLICATION STRUCTURE DESIGN AND IMPLEMENTATION

The most central operation on elliptic curves is the calculation of C*P, which can be expressed as $c \cdot P = P + P + \cdots + P(c\,times)$. So, the scalar product is actually a multiplication of the same point on the elliptic curve, which can be implemented through the algorithm shown in Figure 5.

```
Algorithm 2

  Input: C,P
  Output: B
1.  B←O,  A←P;
2.  WHILE C>0 DO{
3:  IF c is odd THEN B←B+A
4:    A←A+A
5:  C ← floor(C/2)
6:  }
7: RETURN B
```

Figure 5. Scalar multiplication algorithm

Because the scalar multiplication algorithm calls point operation in every loop, in order to avoid the time-consuming inverse operation, the point doubling and point addition operation can be done using a projection coordinate system to increase the speed. At the beginning of the calculation, it is necessary to convert point (x, y) in the affine coordinate system to point (X, Y, Z) on the Jacobian projection coordinates, and the conversion process is as follows.

$$X = x,\ \ Y = y,\ \ Z = 1$$

Thus, at the end of the scalar multiplication operation, we only need to convert the projection coordinates to affine coordinates, which requires only one inverse operation, as follows.

$$x = X / Z^2, y = Y / Z^3$$

In this paper, a modified Jacobia projection coordinate system[10] is used for point addition and point doubling calculation, in which a quadratic representation of (x, y) transformed into (X,Y,Z,aZ^4) is used. Because the modified Jacobia projection coordinate system can further improve the overall arithmetic performance. Figure 6 and 7 show the specific computation process and data flow of the designed point addition and point doubling operation as well as the data dependencies among each operation. The quadrilateral represents cubic operation, the ellipse represents multiplication operation, and the rectangle represents addition and subtraction operation. We can see that in this way there is no inverse operation in each loop of the scalar multiplication, and only one inverse operation is needed at the end of the calculation to transfer the points back to the affine coordinate system. On the other hand benefits from the fact that the cubic operation on GF(3^m) is much faster than the multiplication operation, largely reducing the computational complexity compared to the paper[11] where the cubic result is computed by two multiplication steps, thus making the overall scalar multiplication algorithm run much faster.

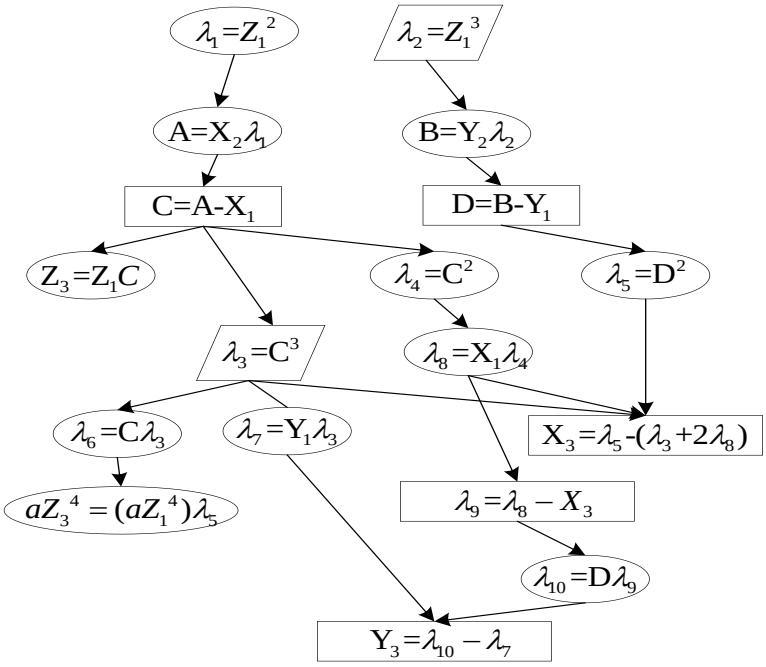

Figure 6. Point addition steps and data flow

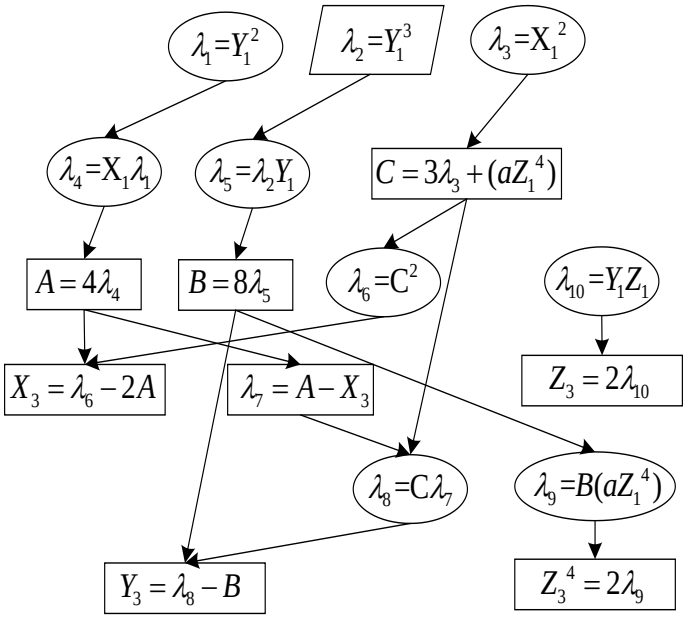

Figure 7. Point doubling steps and data flow

The general structure of the scalar multiplication module is given in Figure 8. The master controller is designed to control the final scalar multiplication operation by judging whether the input signal is valid or not. All intermediate variables are stored in the register heap, and because they will be read frequently, dual-port registers are used for the maximum reuse of hardware resources. The data after operation is stored in the data unit.

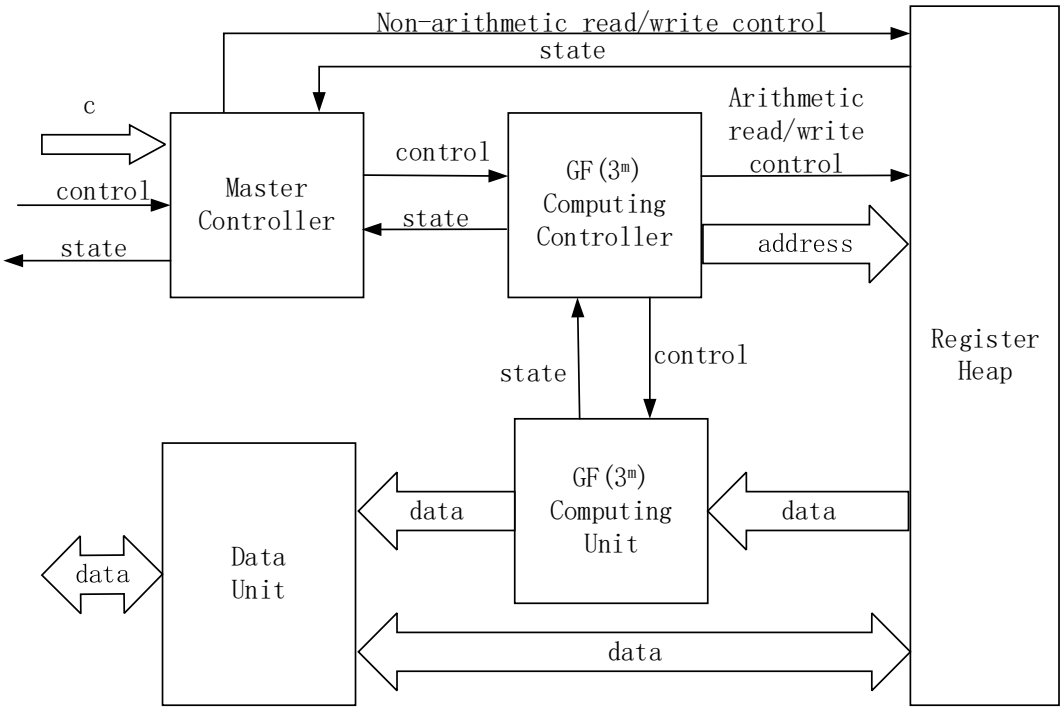

Figure 8. General framework of scalar multiplication

5. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

5.1. Experimental Results

The choice of the hyperelliptic curve $y^2=x^3-x+1$ is defined on GF(3^m). Compared with the conventional elliptic curve cryptosystem, the Hyperelliptic Curve Cryptosystem (HCC)[12] has a higher security in the context of the general trend of quantum computer development and quantum attacks. Simulate and verify the elliptic curve with base field on GF(3^{97}) on the target device. The results of the inputs and outputs associated with the scalar multiplication module at a system clock frequency of 124Mhz are shown in Figure 9, where c is selected as the parameter of the order of this curve. To ensure the accuracy of the experimental results, we implemented the arithmetic on GF(3^{97}) through the Linux platform with reference to the software of literature[13], and did comparison as well as verification of the results with the same elliptic curve and input parameters, and finally the results obtained from the simulation verification in VIVADO 2018.3 were consistent with those obtained from Linux. The hardware platform we use is the XILINX-Xc7z020 device. Finally, we know through simulation that it takes only 0.335ms to calculate a scalar multiplication.

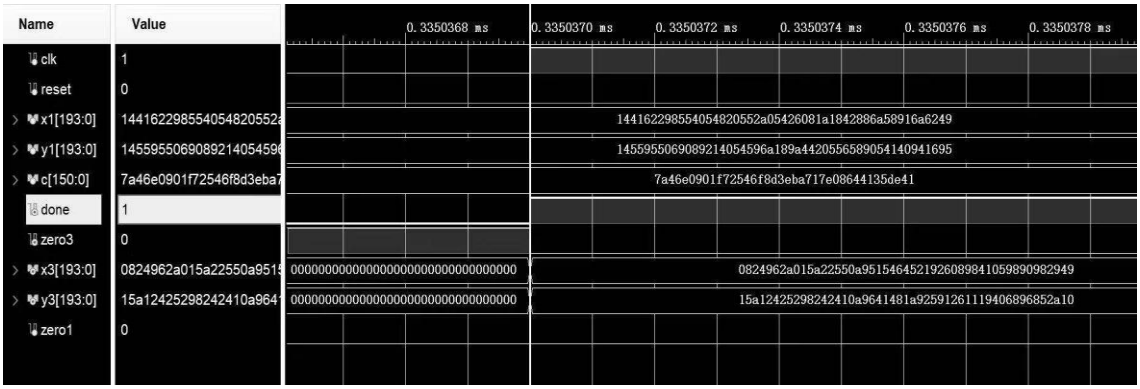

Figure 9. Simulation results.

The main arithmetic in the scalar multiplication structure implemented is then simulated separately, the specific performance at a maximum frequency of 124Mhz and the resources occupied are shown in Table 1.

Table 1. GF(3^{97})- Major arithmetic performance and resource usage

GF(3^{97})-Arithmetic	Time/us	Resources/slices
multiplication	0.225	811
inversion	1.785	1373
Point addition	2.785	3201
Point doubling	2.325	2578

5.2. Performance Analysis and Comparison

Finally, in order to demonstrate that the scalar multiplication structure implemented in this paper has certain advantages in terms of both resource consumption and operational performance, the performance of scalar multiplication structures implemented in related conventional fields hardware is compared in this paper. Direct comparisons are difficult due to different platforms used, different algorithms defined and the variability of bit widths. Therefore, to ensure the

relative fairness of the comparison, some conventional fields with similar bit widths are selected for it.

Table 2. Performance comparison with conventional fields of similar bit width

Design	Hardware platform	Bit width/ fields	Resources (Slices)	Frequency (MHz)	Time (Ms)	Number of Cycles(K)
[14]	Virtex-4	192	7080	21.55	15.87	342
[15]	Virtex-4	192	8590	48	2.3	110
[16]	XC4VLX	GF(2^{163})	9308	12.5	195.09	2438
[17]	Virtex-5	GF(2^{163})	9670	147.5	0.283	41.7
This paper	Xc7z020	194	7518	124	0.335	41.5

From Table 2, it can be seen that the scalar product structure designed and implemented in this paper consumes the least number of clock beats to compute one scalar result [14-16], which is slightly inferior to the computing performance of the design by Anissa[17], but saves about 22% in terms of resources. The comparison fully illustrates that the scalar product structure designed in this paper does have some advantages. However, the scalar multiplication structure implemented in this paper also has some shortcomings. If triple point arithmetic is used to implement the scalar multiplication algorithm, the performance will be better, but it will consume more resources, so how to improve the computing performance to a large extent without consuming too many resources is worthy of further study.

6. CONCLUSIONS

In this paper, we propose and implement a scalar product structure for elliptic curves on a base field of GF(3^m) based on the theory of elliptic curve cryptography methods. It is demonstrated by experimental comparison that the arithmetic performance of the implementation is sufficiently comparable to that of the conventional fields that have been extensively studied so far. This structure can be used as an IP core in some communication fields with higher security requirements. For finite fields of other bit widths, this can be achieved simply by modifying the design parameters, so it also has some generality. How to improve the inverse operation on finite field and scalar multiplication algorithm to increase the calculation speed is worthy of further study.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (No. U1631121). Thanks to all the contributors who have worked on this paper.

REFERENCES

[1] Jiang, J. Hou, J., Huang, H., Zhao, Y., Feng, X., & Science, S. O. (2019) Research on area-efficient low-entropy masking scheme for AES. Journal on Communications.
[2] RESCORLA E, MOZILLA. (2020) The transport layer security (TLS) proto-col version 1.3: RFC8446[S]. IETF, (2018-08).
[3] Yao T, Cao BW. (2018) An overview of cyberspace security[J]. China New Communications, 03(v.20): 170-170.
[4] Galbraith S D. (2001) Super singular Curves in Cryptography[C]. International Conference on the Theory and Application of Cryptology and Information Security.
[5] Shen Shao. (2015) Research on Fast Algorithms for ScalarMultiplication of Elliptic CurveCryptography over GF (3n)[J]. Computer Science & Application, 04(4): 390-399.

[6] Yeniaras, E., & Cenk, M. (2020). Faster Characteristic Three Polynomial Multiplication and Its Application to NTRU Prime Decapsulation.
[7] Kim, K. H., Kim, S. I., & Ju, S. C. (2007). New Fast Algorithms for Arithmetic on Elliptic Curves over Fields of Characteristic Three.
[8] Koblitz, N. (1987) Elliptic curve cryptosystems. Mathematics of Computation, 48(177), 203-209.
[9] El-Tantawy, S. A., Salas, A. H., Alharthi, M. R., & Engineering, M. (2021) On the analytical solutions of the forced damping duffing equation in the form of weierstrass elliptic function and its applications. Mathematical Problems in Engineering.
[10] Cohen H. (1998) Efficient Elliptic Curve Exponentiation Using Mixed Coordinates[C]. ASIACRYPTO'98.
[11] Li Fan, Li Yunfeng, Weng Tianheng, Zhang Junjie, (2020) The Rapid Parallel Realization of SM2 Point Operation Based on FPGA [J]. Electronic Measurement Technology,43(15):105-111.
[12] Salam, T., & Hossen, M. S. (2020). HECC (Hyperelliptic Curve Cryptography).
[13] Duanmu QF, Wang YB, Zhang KZ, (2009) GF(3^m)-ECC algorithm and its software implementation[J]. Computer Engineering, (14): 7-9.
[14] Hu, X., Zheng, X., Zhang, S., Cai, S., & Xiong, X. (2018). A low hardware consumption elliptic curve cryptographic architecture over gf(p) in embedded application. Electronics, 7(7), 104-.
[15] Javeed K, Wang X. (2016) FPGA Based High Speed SPA Resistant Elliptic Curve Scalar Multiplier Architecture[J]. International Journal of Reconfigurable Computing, (2016-7-10), 2016, 2016: 2.
[16] Imran, M., Kashif, M., & Rashid, M. (2018). Hardware Design and Implementation of Scalar Multiplication in Elliptic Curve Cryptography (ECC) over GF(2^163) on FPGA.
[17] Anissa, S., Medien, Z., Chiraz, M., & Mohsen, M. (2017). Design and implementation of low area/power elliptic curve digital signature hardware core. Electronics, 6(2), 46-.

AUTHORS

Tan Yongliang, Yunnan University, Main research on IoT security and cryptography

He Lesheng, Yunnan University, Associate Professor, Research on IoT security and embedded systems

Jin Haonan, Yunnan University, Main research on digital signal processing

Kong Qingyang, Yunnan University, China Main research on network security