EURASIP Journal on Wireless Communications and Networking

RESEARCH

Open Access

Efficient scalar multiplication of ECC using SMBR and fast septuple formula for IoT

Chong Guo and Bei Gong*

*Correspondence:
gongbei@bjut.edu.cn
Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China

Abstract

In order to solve the problem between low power of Internet of Things devices and the high cost of cryptography, lightweight cryptography is required. The improvement of the scalar multiplication can effectively reduce the complexity of elliptic curve cryptography (ECC). In this paper, we propose a fast formula for point septupling on elliptic curves over binary fields using division polynomial and multiplexing of intermediate values to accelerate the computation by more than 14%. We also propose a scalar multiplication algorithm based on the step multi-base representation using point halving and the septuple formula we proposed, which significantly reduces the computational cost. The experimental results show that our method is more efficient over binary fields and contributes to reducing the complexity of ECC.

Keywords: Internet of things, Elliptic curve cryptography, Scalar multiplication, Step multi-base representation, Point septupling

1 Introduction

The Internet of Things (IoT) is developing rapidly in theory and application. With the development of narrow band Internet of Things (NB-IoT) and the accelerating popularity of 5G, the cost of connecting devices to IoT has been reduced, which greatly promotes its development. As a result, more and more devices are connected to the Internet of Things in various fields, from smart terminals to industrial facilities. Devices connected to the Internet of Things continue to generate and transmit large amounts of information. But the availability of wireless communication links to attackers means that data on the Internet of Things are vulnerable to various types of attacks, such as data eavesdropping and data tampering. Therefore, it is not safe to transmit information in plaintext form on the Internet of Things. Cryptography is the most widely used method to ensure the security of data transmission and sharing in the network. Encrypted data transmission on the Internet of Things has become the key to secure communication between devices. RSA public key cryptography has been well researched and widely used since the introduction of Rivest, Shamir and Adleman in 1975. The system relies on Integer Decomposition Problem (IFP), which typically uses a 1024-bit or more key. In 1976, Hellman proposed a public key exchange algorithm based on the Discrete Logarithm Problem (DLP) [1]. In 1986, Miller [2], Koblitz [3] independently proposed the Elliptic

Curve Cryptography (ECC). ECC is a kind of encryption technology based on DLP. It uses elliptic curve in finite domain to generate finite Abel group to implement public key cryptographic primitives. Unlike the RSA algorithm, which relies on the sub-exponential time algorithm to solve the integer decomposition problem, the best algorithm for solving the basic mathematical problems of ECC involves the Elliptic Curve Discrete Logarithm Problem (ECDLP). This leads to the infeasibility of solving the ECDLP algorithm, which increases rapidly with the size of the problem and is much higher than integer factorization and discrete logarithm problem. Therefore, ECC requires only smaller keys than public key cryptography (such as RSA and ElGamal), while providing the same level of security. For example, an ECC that provides the same level of security as RSA with a 1024-bit key size requires only 160 keys. Due to the high security of per key length, ECC is widely used for mobile devices and IoT. However, ECC operations are still very complex and costly for devices with poor computing power, limited energy reserves and intensive data transmission in IoT. This poses a challenge for the long-term stable function and real-time data transmission of IoT devices. Therefore, we need to improve the operation of ECC to make it lighter, thereby increasing the efficiency of cryptography and reducing costs.

The operation of the ECC works on a multiplication group over a finite field. The scalar multiplication of an elliptic curve is an operation that adds a point P on the curve k times.

$$Q = kP = P + P + \cdots + P,\ k\ times$$

where P is a point on an elliptic curve and k is a large positive integer. In any primitive implementations of ECC, scalar multiplication is the main computing operation. The key factor to improve the efficiency of ECC is how to realize fast scalar multiplication. Therefore, many researchers have proposed various studies on accelerated scalar multiplication. Morain et al. [4] proposed the non-adjacent form (NAF), which is a signed form of representation. This form ensures that at least one of any two adjacent terms is zero. Solinas et al. proposed the Joint Sparse Form (JSF) based on NAF [5]. JSF is the best signed binary representation of a pair of integers, which can generate more double-zero bits than NAF. Koblitz [6] and Solinas [7], respectively, proposed an anomalous binary curve on which Frobenius mapping can be used, and an effective scalar representation on that curve—reduced τ-adic non-adjacent form (RTNAF). The squaring on an anomalous binary curve is implemented by a displacement, which can be performed in a very short time. Under the RTNAF representation, the scalar multiplication τP is quickly obtained by squaring on the x and y coordinates of the point P. Cohen proposed a more efficient hybrid addition operation by combining projective coordinates and affine coordinates into mixed coordinates [8]. The introduction of Jacobian coordinates eliminates expensive inversion operation in scalar multiplication under affine coordinates. In recent years, the method of expressing a large integer k by double base and multi-base has attracted widespread attention. Dimitrov [9] first proposed the Double-Base Number System (DBNS) and applied it to speed up scalar multiplication, which effectively reduces the number of point additions in scalar multiplication, by taking advantage of the sparseness and the ternary nature of DBNS. But there are some repetitive computations in DBNS. In order to solve this problem, Dimitrov [10] proposed Double-Base

Chain (DBC) on the basis of DBNS. DBC performs computations in a nested form, so that the results of each part of the computation will be reused, reducing the occurrence of repeated computations. Mishra extended DBNS to Multi-Base Number Representation (MBNR), breaking the limitation that only two bases can be used to represent scalars, so as to bring higher redundancy in the representation of scalars [11]. But MBNS has some repetitive computations like DBNS.

In this paper, we propose an efficient formula for fast computation of the sevenfold of elliptic curve points over the binary fields, which can be used in DBNS and MBNR to compute the scalar multiplication of elliptic curves. This formula uses division polynomial and multiplexing of intermediate values in affine coordinates to increase the speed of computing the sevenfold point by more than 14%. We also proposed a scalar multiplication algorithm based on the Step Multi-Base Representation (SMBR). This algorithm uses the sevenfold point formula we proposed and replaces the traditional point doubling with the faster point halving. Experimental results show that our scalar multiplication algorithm is more efficient in affine coordinates over binary fields and contributes to reducing the cost of cryptography for devices in the IoT.

The organization of this paper is as follows. In Sect. 2, we briefly introduce the basics of elliptic curves, point halving, double-base chain and multi-base representations. In Sect. 3, we give an efficient formula for computing the sevenfold point of an elliptic curve over binary fields and provide proof of the formula and analysis of the computational cost. In Sect. 4, we propose a scalar multiplication algorithm based on SMBR and give the method of scalar k conversion to SMBR and the detailed steps of the scalar multiplication algorithm. In Sect. 5, we show experimental results, compare and analyze our algorithm with other research and demonstrate that our method is more efficient. Finally, in Sect. 6, we draw our conclusions.

2 Related work

In this section, we will review the concepts and research status of ECC, point halving, double-base chains and step multi-base representation.

2.1 Elliptic curve cryptography (ECC)

Definition 1 (*Elliptic curve cryptography*) An elliptic curve E over a finite field GF field K can be defined by the Weierstrass equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

where $a_1, a_2, a_3, a_4, a_6 \in K$, and $\Delta \neq 0$,where Δ is the discriminant of E.

In practice, adjusting the variables within the admissible range can greatly simplify the Weierstrass Eq. (1).

Over prime fields,$K = F_P$, if the characteristic of K is not equal to 2 and 3, then Eq. (1) can be simplified to

$$y^2 = x^3 + ax + b \tag{2}$$

where $a, b \in F_p, \Delta = 4a^3 + 27b^2 \neq 0$.

Over binary fields, $K = F_{2^m}$, the elliptic curve E is called the non-supersingular curve, and Eq. (1) can be rewritten as

$$y^2 + xy = x^3 + ax^2 + b \tag{3}$$

where $a, b \in F_{2^m}, \Delta = b \neq 0$.

The set $E(K)$ of rational points and the infinity point O defined on an elliptic curve E over a field K form an abelian group under the operation (usually denoted by addition) defined by the law of chord and tangent. If the points on an elliptic curve E are represented in affine coordinates, such as $P = (x, y)$ and $Q = (u, v)$, then both the point addition $(P + Q)$ and point doubling $(2P)$ require an expensive field inverse operation. We use $[i]$,$[s]$ and $[m]$ to represent the computational cost of one inversion I, one squaring S and one multiplication M, respectively. In order to facilitate the gauging of the computational cost of inversions, the $[i]/[m]$ ratio is defined according to the ratio of the cost between one inversion and one multiplication. It is generally assumed that $3 \leq [i]/[m] \leq 10$ for the binary fields [12], and $[i]/[m] \geq 30$ for the prime fields [13]. In addition, squaring is the least expensive of the three main operations. Over binary fields, squaring is a linear operation with negligible computational cost, and it is generally assumed that $[s] \leq 0.1[m]$ [12]. Over prime fields, $[s] = 0.8[m]$ is generally assumed, but in order to prevent side-channel attacks (SCA) from using side-channel atomicity [14], the same multiplier needs to be used to perform squarings and multiplications, then $[s] = [m]$.

2.2 Point halving

The point halving independently proposed by Knuden [15] and Schroeppel [16] is a reverse operation of point doubling. Assume that $P = (x, y)$ and $Q = (u, v)$ are two points defined on the elliptic curve E Over binary field and expressed in affine coordinates, satisfying $Q = 2P$. If we know the affine coordinates of the point P, the coordinates of point Q can be obtained by point doubling using the following equation:

$$\lambda = x + y/x \tag{4}$$

$$u = \lambda^2 + \lambda + a \tag{5}$$

$$v = x^2 + u(\lambda + 1) \tag{6}$$

Point halving is the completely opposite operation. When $Q = (u, v)$ is known, find $P = (x, y)$ so that $Q = 2P$, denoted as $P = \frac{1}{2}Q$. First, we need to solve $\lambda^2 + \lambda = u + a$ according to Eq. (5) to get λ, then solve $x^2 = v + u(\lambda + 1)$ from Eq. (6) to get x and finally calculate $y = \lambda x + x^2$ based on Eq. (4) to get y. The logic of the point halving is shown in Algorithm 1. The detailed analysis of the solving process and the calculation complexity about point halving are shown in [13].

Algorithm 1. Point Halving

Input: $Q = 2P = (u, v)$

Output: $P = (x, y)$

```
1.Solve $\lambda^2 + \lambda = u + a$ for $\lambda$
2.Find $t = v + u(\lambda + 1)$
3.If $Tr(t) = 0$
4.  then $\lambda \leftarrow \lambda, x \leftarrow \sqrt{t + u}$
5.Else
6.  $\lambda \leftarrow \lambda + 1, x \leftarrow \sqrt{t}$
7.Find $y = \lambda x + x^2$
8.Return $(x, y)$
```

Point halving is less expensive to compute than point doubling, and the advantage is more evident when the point P is unknown in advance and the $[i]/[m]$-ratio is small [13]. If the doubling required in the usual double-and-add operations is all replaced by point halving, the calculation speed can be accelerated by 50% [16].

2.3 Double-base chain (DBC)

The Double-Base Number System (DBNS) was originally proposed by Dimitrov [9] as a scheme for the representation of positive integers, where each positive integer k can be expressed as the sum or difference of a number of 2-integers (the product of the powers of two relatively prime positive integers). For example, the positive integer k represented by the application of {2,3}-integers (the sub-formats are 2^b3^t) are

$$k = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i}, \text{ with } s_i \in \{-1, 1\} \text{ and } b_i, t_i \geq 0 \tag{7}$$

In [9], Dimitrov described the attributes of DBNS and proved that DBNS have a high degree of redundancy. Obviously, there are no unique forms of DBNS representation of an integer. For example, there are 5 different DBNS representations of 10, 72 representations of 50 and 402 representations of 100, where only positive sums ($s_i = 1$,$i \in [1, m]$) are considered. The number of representations grows rapidly as the integer k increase.

Subsequently, Dimitrov [10] developed the Double-Base Chain (DBC) and specified that a DBNS can only be transformed into a DBC if the exponent of the base in the DBNS is a decreasing sequence, that is $k = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i}$ and satisfies $s_i \in \{-1, 1\}$,$b_1 \geq b_2 \geq \cdots \geq b_m \geq 0$ and $t_1 \geq t_2 \geq \cdots \geq t_m \geq 0$.

Definition 2 (*Double-base chain* (DBC)) Given a positive integer $k > 0$, a sequence $(C_n)_{n>0}$ of positive integers, if it satisfies:

$$C_1 = 1, C_{n+1} = 2^u 3^v C_n + s, \text{ with } s_i \in \{-1, 1\} \tag{8}$$

for some $u, v \geq 0$, such that $\exists m > 0$ satisfies $C_m = k$, the sequence C_m is called a double-base chain of k, and m is the length of this double-base chain. The length of the double base chain is equal to the number of $2^b 3^t$ subitems in DBNS.

Double base chain make all calculated values reusable by restricting the sequence (b_i) and (t_i) in DBNS to decrease monotonically (i.e.,$b_1 \geq b_2 \geq \cdots \geq b_m \geq 0$, $t_1 \geq t_2 \geq \cdots \geq t_m \geq 0$) and by applying recursive calculations. The double-base chain representations are highly redundant and can dramatically reduce the Hamming weights in scalar expansion. Algorithm 2 gives the process for transforming a large integer k into DBNS that conform to exponential constraints.

```
Algorithm 2.Conversion to DBNS with Restricted Exponents
Input: a large positive integer $k$; binary and ternary exponential limits $b_{\max}, t_{\max} > 0$ ,i.e.
$0 \leq (b_i)_{\max} \leq b_{\max}, 0 \leq (t_i)_{\max} \leq t_{\max}$
Output: the sequence $(s_i, b_i, t_i)_{i>0}$ such that $k = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_3}$ , with
$s_i \in \{-1,1\}, b_1 \geq b_2 \geq \ldots \geq b_m \geq 0, t_1 \geq t_2 \geq \ldots \geq t_m \geq 0$
1. $s \leftarrow 1$ and $i \leftarrow 0$
2. While $k > 0$ do
3.    Define $z = 2^b 3^t$, using the greedy algorithm to find the best approximation of $k$, and
      $0 \leq (b_i)_{\max} \leq b_{\max}, 0 \leq (t_i)_{\max} \leq t_{\max}$
4.    Print $(s, b, t)$
5.    $i \leftarrow i+1$
6.    $b_{\max} \leftarrow b, t_{\max} \leftarrow t$
7.    If $k < z$ then
8.        $s \leftarrow -s$
9.    $k \leftarrow |k - z|$
```

In Algorithm 2, the maximum exponential limit on the base x is generally set to $\log_x(k)$, i.e., $b_{\max} < \log_2(k) \leq n$, $t_{\max} < \log_3(k) \leq 0.65n$, where n is the binary bit size of the positive integer k. With a positive integer of 160-bit size, for example, $b_{\max} = 160$ and $t_{\max} = 103$ can be specified.

The ternary/binary method for fast computation of ECC scalar multiplication proposed by Ciet [17] was subsequently applied to double-base chain to decrease the number of inversions and the execution time by efficient point doubling ($2P$), point tripling ($3P$) and point quadrupling ($4P$).

2.4 Step multi-base representation (SMBR)

In [11], Mishra extends the Double-Base Number System to Multi-Base Number Representation. Given a large positive integer k and a set $B = \{b_1, b_2, \ldots, b_l\}$ of small positive integers, if there exists an expression for k which is the sum of the products of the powers of the elements in base B, i.e.,$k = \sum_{j=1}^{m} (s_j \prod_{i=1}^{l} b_i^{e_{ij}})$,with $s_j \in \{-1, 1\}$ and $e_{ij} \geq 0$, then the above equation is called the multi-base number representation of the integer k using base B. The integer m is the length of the MBNR and |B| denotes the number of elements in the set of B. If $|B| = 2$, the MBNR is simplified to a DBNS. DBNS already has a high degree of redundancy and a short expression. MBNR has a higher redundancy and shorter expressions than DBNS.

Definition 3 (*Step multi-base representation*) A step multi-base representation based on the set $B = \{b_1, b_2, \ldots, b_l\}$ is

$$k = \sum_{i=1}^{m} s_i b_1^{e_{1i}} b_2^{e_{2i}} \ldots b_l^{e_{li}} \quad \text{with } s_i \in \{-1, 1\} \tag{9}$$

where $\forall j \in \{1, 2, \ldots, l\}$, the sequence (e_j) is monotonously decreasing, then it is called the step multi-base representation of the integer k.

SMBR can be seen as a generalized extension of the DBC. If $|B| = 2$, the SMBR is simplified to a DBC. Same as the DBC, all intermediate calculated values can be reused in SMBR. Thus, SMBR has a higher redundancy and can further reduce the Hamming weights in scalar expansion. As the efficient formula for point quintupling was proposed in [11], the SMBR based on $B = \{2, 3, 5\}$, i.e., $k = \sum_{i=1}^{m} s_i 2^{b_i} 3^{t_i} 5^{q_i}$, is widely used and studied [18, 22]. In the scalar multiplication algorithm, the selection of the base of SMBR will determine the computational performance of the algorithm. Replacing the existing base with a more efficient multiple-point formulas can further increase the redundancy of the expression and reduce the computational cost of scalar multiplication algorithm. Therefore, in Chapter 3, we proposed an efficient septuple formulas, and in Sect. 4, we proposed a scalar multiplication algorithm based on the SMBR with $B = \{1/2, 3, 7\}$.

3 Septuple formula design

In this section, we give an efficient septuple formula for elliptic curve points over binary fields, a proof of the formula and an efficiency analysis.

3.1 Point septupling in elliptic curves over binary fields

In order to reduce inversions in multipoint operations, it is common practice in research work to convert points to other coordinate systems (e.g., the Jacobi coordinates). However, over binary fields, where the $[i]/[m]$-ratio is small, generally $3 \leq [i]/[m] \leq 10$, the elliptic curve group arithmetic in affine coordinates already has excellent performance.

Therefore, we propose a new point septupling formula in affine coordinates. Let $P = (x, y)$ be a known point on an elliptic curve shown in Eq. (3) over a binary field. Assume that the sevenfold point of P is expressed as $7P = (u, v)$. Then, we can obtain u and v by the following formula:

Let us define a set of polynomials as follows: Then,

$$\begin{aligned}
A &= x^4 + x^3 + a \\
B &= x(A + x^3) \\
C &= A^3 + x^4 B \\
D &= A(B^2 + C) \\
E &= A^6 + x^4 B(A^3 + B^2) \\
F &= B(A^2 D + C^2)
\end{aligned}$$

$$\begin{cases} u = x + xD\left(\frac{xF}{E^2}\right) \\ v = y + u + \left(\frac{xF}{E^2}\right)\left[\frac{CF}{E} + (x^2 + y)D\right] \end{cases} \tag{10}$$

3.2 Proof of septuple formula over binary fields

For non-supersingular curves over the binary fields, $K = F_{2^m}$, there are division polynomials as follow:

$$\psi_1 = 1$$

$$\psi_2 = x$$

$$\psi_3 = x^4 + x^3 + a$$

$$\psi_4 = x^6 + ax^2 = x^2(x^4 + a)$$

The higher degree division polynomials can be deduced from the following recursive relations:

$$\psi_{2n+1} = \psi_{n+2}\psi_n^3 - \psi_{n-1}\psi_{n+1}^3 \tag{11}$$

$$\psi_2\psi_{2n} = \psi_{n+2}\psi_n\psi_{n-1}^2 - \psi_{n-2}\psi_n\psi_{n+1}^2 \tag{12}$$

Applying these recursive relations, by sequentially assigning $n = 2$ in Eq. (11), $n = 3$ in Eq. (12), $n = 3$ in Eq. (11) and $n = 4$ in Eq. (12), we can obtain:

$$\psi_5 = \psi_4\psi_2^3 - \psi_1\psi_3^3$$

$$\psi_6 = \left(\psi_5\psi_3\psi_2^2 - \psi_1\psi_3\psi_4^2\right) \Big/ \psi_2$$

$$\psi_7 = \psi_5\psi_3^3 - \psi_2\psi_4^3$$

$$\psi_8 = \left(\psi_6\psi_4\psi_3^2 - \psi_2\psi_4\psi_5^2\right)\Big/\psi_2$$

Depending on the polynomials $\psi_1\psi_2\psi_3\psi_4$ and the recurrence relation (11), (12), for any point $P = (x, y)$ on a non-supersingular curve, the n-fold of this point can be derived from the formula:

$$[n]P = \left(x + \frac{\psi_{n+1}\psi_{n-1}}{\psi_n^2}, y + \psi_2 on + \frac{\psi_{n+1}^2\psi_{n-2}}{\psi_2\psi_n^3} + h_4\frac{\psi_{n+1}\psi_{n-1}}{\psi_2\psi_n^2}\right) \tag{13}$$

where,

$$\psi_2 on = x + \frac{\psi_{n+1}\psi_{n-1}}{\psi_n^2}$$

and

$$h_4 = \left(x^2 + y\right)$$

Assuming that the affine coordinates of $7P$ is (u, v), we can calculate directly from Eq. (14):

$$u = x + \frac{\psi_8\psi_6}{\psi_7^2}$$

$$v = y + \psi_2 on + \frac{\psi_8^2\psi_5}{\psi_2\psi_7^3} + h_4\frac{\psi_8\psi_6}{\psi_2\psi_7^2} = y + u + \frac{\psi_8^2\psi_5}{\psi_2\psi_7^3} + \left(x^2 + y\right)\left(\frac{\psi_8\psi_6}{\psi_2\psi_7^2}\right)$$

However, calculating (u, v) directly using the Formulae above is not the most suitable method. In the process of calculating $\psi_1\psi_2\ldots\psi_8$ and (u, v), many intermediate values are generated. The calculation can be accelerated by transforming the formula forms and multiplexing the intermediate values. We define polynomials: $A = x^4 + x^3 + a$, $B = x(A + x^3)$,$C = A^3 + x^4B$, $D = A(B^2 + C)$, $E = A^6 + x^4B(A^3 + B^2)$ and $F = B(A^2D + C^2)$, then the transformed forms of formulae $\psi_1\psi_2\ldots\psi_8$ and (u, v) are as follows:

$$\psi_1 = 1$$

$$\psi_2 = x$$

$$\psi_3 = x^4 + x^3 + a = A$$

$$\psi_4 = x^6 + ax^2 = x^2(x^4 + a) = x^2(A + x^3) = x\left[x(A + x^3)\right] = xB$$

$$\psi_5 = \psi_4\psi_2^3 - \psi_1\psi_3^3 = x^4B + A^3 = C$$

$$\psi_6 = \left(\psi_5\psi_3\psi_2^2 - \psi_1\psi_3\psi_4^2\right)\Big/\psi_2 = \left(x^2AC - x^2AB^2\right)\Big/x = x\left[A(B^2 + C)\right] = xD$$

$$\psi_7 = \psi_5\psi_3^3 - \psi_2\psi_4^3 = A^3C - x^4B^3 = A^6 + x^4B(A^3 + B^2) = E$$

$$\psi_8 = \left(\psi_6\psi_4\psi_3^2 - \psi_2\psi_4\psi_5^2\right)\Big/\psi_2 = \left(x^2A^2BD + x^2BC^2\right)\Big/x = x\left[B(A^2D + C^2)\right] = xF$$

Substituting these values into formula (13) and then transforming, we can get septuple formula (10). The specific transformation process is as follows:

$$u = x + \frac{\psi_8\psi_6}{\psi_7^2} = x + \frac{x^2DF}{E^2} = x + xD\left(\frac{xF}{E^2}\right)$$

$$v = y + \psi_2 on + \frac{\psi_8^2\psi_5}{\psi_2\psi_7^3} + h_4\frac{\psi_8\psi_6}{\psi_2\psi_7^2} = y + u + \frac{xCF^2}{E^3} + \left(x^2 + y\right)\left(\frac{xDF}{E^2}\right)$$
$$= y + u + \left(\frac{xF}{E^2}\right)\left[\frac{CF}{E} + (x^2 + y)D\right]$$

3.3 Cost of septuple formula over binary fields

Given a point $P = (x, y)$ on a non-supersingular curve (Eq. 3), let us check the sub-expressions and costs required to compute $7P$ by applying Eq. (10) without any pre-computation. Table 1 lists the sub-expressions, intermediate values and costs for computing $7P$. Next, we analyze the efficiency of the septuple formula. Table 2 lists the costs of various field operations over binary fields. A number of methods for computing $7P$ have been proposed in previous research. In [19], the authors proposed several septuple formulae over prime fields, one with $15[s] + 14[m]$ costs in Jacobian coordinates, and the other with $9[s] + 13[m]$ costs in Jquartic coordinates. In [20], the author uses

Table 1 Cost of septuple formula for elliptic curves over binary fields

Sub-expression	Intermediate value	Cost
A	x^2, x^3, x^4	$2[s] + 1[m]$
B		$1[m]$
C	A^2, A^3, x^4B	$1[s] + 2[m]$
D	$B^2, A(B^2 + C)$	$1[s] + 1[m]$
E	$A^6, x^4B(A^3 + B^2)$	$1[s] + 1[m]$
F	C^2, A^2D	$1[s] + 2[m]$
$\frac{1}{E}$		$1[i]$
$\frac{xF}{E^2}$	$\frac{1}{E^2}, xF$	$1[s] + 2[m]$
u	$xD, xD\left(\frac{xF}{E^2}\right)$	$2[m]$
v	$CF, \frac{CF}{E}, (x^2 + y)D, \left(\frac{xF}{E^2}\right)\left[\frac{CF}{E} + (x^2 + y)D\right]$	$4[m]$
Total:$1[i] + 7[s] + 16[m]$		

Table 2 Costs of various field operations for elliptic curves over binary fields in affine coordinates

Operation	Propose	Cost	Operation	Propose	Cost
$P+Q$		$1[i]+1[s]+2[m]$	$4P+Q$	[17]	$2[i]+6[s]+10[m]$
$2P$		$1[i]+1[s]+2[m]$	$4P+Q$	[10]	$3[i]+3[s]+5[m]$
$2P+Q$	[17]	$1[i]+2[s]+9[m]$	$5P$	[11]	$1[i]+5[s]+13[m]$
$3P$	[17]	$1[i]+4[s]+7[m]$	$7P$	[20]	$3[i]+7[s]+18[m]$
$3P+Q$	[17]	$2[i]+3[s]+9[m]$	$7P$	[21]	$2[i]+7[s]+14[m]$
$4P$	[17]	$1[i]+5[s]+8[m]$	$7P$	[21]	$1[i]+6[s]+20[m]$
$4P$	[10]	$2[i]+3[s]+3[m]$	$7P$	This paper	$1[i]+7[s]+16[m]$

the expensive sevenfold point operation over binary fields, which requires three inversion, and the total computational cost is as high as $3[i]+7[s]+18[m]$. Later, in [21], the authors proposed two fast methods for point septupling over binary fields with computational costs of $2[i]+7[s]+14[m]$ (method 1) and $1[i]+6[s]+20[m]$ (method 2), and the break-even ratio between the two methods is $[i]/[m]=6$.

The costs of septuple formula we proposed is $1[i]+7[s]+16[m]$. Compared with the method in [20], the formula proposed in this paper reduces two inversion and two multiplication, resulting in a speed up of 33% at the ratio of $[i]/[m]=4$ and 43% at the ratio of $[i]/[m]=8$, with more speed up at higher $[i]/[m]$-ratio. Compared with the method 2 in [21], our formula reduces four multiplication and adds one squaring (the cost of squaring can be ignored over binary fields), and the speed is increased by 17% at the ratio of $[i]/[m]=4$ and by 14% at the ratio of $[i]/[m]=8$, which is more significant when the $[i]/[m]$-ratio is smaller. Compared with the method 1 in [21], our formula reduces one inversion and adds two multiplication. The break-even ratio between our formula and method 1 in [21] is $[i]/[m]=2$. Our formula is faster when $[i]/[m]>2$, with up to 17% faster at the ratio of $[i]/[m]=4$ and 20% faster at the ratio of $[i]/[m]=8$. Over binary fields, the $[i]/[m]$-ratio is generally not less than 3 and is commonly assumed to be $[i]/[m]=8$. Therefore, the septuple formula we proposed is more efficient than the methods in [20, 21].

4 Methods

In order to improve the computation speed of elliptic curve scalar multiplication, we modify the MBNR based on $B=\{2,3,5\}$ proposed in [11] and propose a SMBR based on $B=\{1/2,3,7\}$, denoted $k=\sum_{i=1}^{m} s_i(\frac{1}{2})^{b_i}3^{t_i}7^{q_i}$. We retain the original point tripling, replace the point doubling and quadrupling with faster point halving, replace the point quintupling with the point septupling we proposed in Sect. 3 and restrict the exponential sequences $(b_i)(t_i)(q_i)$ of 1/2, 3 and 5 to decreasing monotonically, respectively. The modified SMBR is defined as follows.

Definition 4 ($\{1/2,3,7\}$-*step multi-base representation*) A multi-base number representation based on the set of $B=\{1/2,3,7\}$.

$$k=\sum_{i=1}^{m} s_i\left(\frac{1}{2}\right)^{b_i}3^{t_i}7^{q_i}, \text{ with } s_i\in\{-1,1\} \tag{14}$$

where the sequence of $(b_i)(t_i)(q_i)$ is each monotonically decreasing. Then, the representation is called a $\{1/2, 3, 7\}$-step multi-base representation of the integer k.

Due to the inclusion of point halving, the $\{1/2, 3, 7\}$-SMBR of the large integer k has to be derived indirectly. Assuming that the size of the binary field is p, firstly we find a large exponent 2^r with base 2 that the value approximating p. Then, we multiply the original scalar k by 2^r and then model p and denote the result as k', as shown in Eq. (15).

$$k' = 2^r k \mod p \tag{15}$$

This allows us to transform finding a $\{1/2, 3, 7\}$-SMBR of k into finding a $\{2, 3, 7\}$-MBNR with exponential restrictions of k'. Thus, the form of the representation of k can become

$$k = \frac{k'}{2^r} = \frac{\sum_{i=1}^{m} s_i 2^{b'_i} 3^{t_i} 7^{q_i}}{2^r} = \sum_{i=1}^{m} s_i \left(\frac{1}{2}\right)^{(r-b_i)} 3^{t_i} 7^{q_i} \mod p, \text{ with } k' = 2^r k \mod p \tag{16}$$

where $s_i \in \{-1, 1\}$, $0 \le b'_1 \le b'_2 \le \cdots \le b'_m$, $t_1 \ge t_2 \ge \cdots \ge t_m \ge 0$ and $q_1 \ge q_2 \ge \cdots \ge q_m \ge 0$.

Subsequently, we find MBNR of k' with base $\{2, 3, 7\}$ and restrict the exponential sequences of 3 and 7 monotonic decrease, but the exponential sequence of 2 monotonic increases. We obtain this MBNR in an iterative way. First, we find n such that $k = 0 \mod n$, where the trial order of n is $\{42, 36, 32, 28, 27, 24, 21, 18, 16, 14, 12, 9, 8, 7, 6, 4, 3, 2\}$. If $k = 0 \mod 42$, then return $2 \cdot 3 \cdot 7\left(\frac{k}{42}\right)$. If $k = 0 \mod 36$, then return $2^2 \cdot 3^2\left(\frac{k}{36}\right)$. If $k = 0 \mod 32$, then return $2^5\left(\frac{k}{32}\right)$. And so on, if $k = 0 \mod n$, then return $2^{n_1} \cdot 3^{n_2} \cdot 7^{n_3}\left(\frac{k}{n}\right)$, where $2^{n_1} \cdot 3^{n_2} \cdot 7^{n_3} = n$. If no suitable match is found for all trials, we find a power of 2 that is closest to k denoted as k_c and return the absolute value $|k - k_c|$ of the difference between k and k_c. We chose the power of 2 as the approximation for k because point doubling will become point halving afterwards (and may also constitute half-and-add) with less cost than point tripling and point septupling. As the return value of $|k - k_c|$ becomes smaller and smaller, it can always be approximated in the next round by a lower power of 2. So in this MBNR, the exponents of 2 are keep monotonically decreasing. Therefore, triple-and-add are rarely required in this scalar multiplication. The iterations do not stop until k is equal to 1 or the power of 2, 3 and 7, which means that for any non-negative integer b, t and q, $2^b 3^t 7^q$ can represent a positive integer. The return terms of this iterative algorithm form a MBNR of k' and are ordered from the highest exponent of 2 multiplied by the lowest power of 3 and 7 to the lowest exponent of 2 multiplied by the highest power of 3 and 7. Then, we reverse the order of the sub-terms of the MBNR, so that the exponents of 3 and 7 decrease, and the exponents of 2 increase. Finally, by dividing the MBNR by 2^r, so that all exponents of 2 are negative, the exponents of 1/2 are monotonically decreasing. From this, we can obtain an $\{1/2, 3, 7\}$-SMBR of k as shown in Eq. (16).

Based on this representation, we can propose a scalar multiplication for elliptic curves over binary fields using $\{1/2, 3, 7\}$-SMBR, as described in Algorithm 3. The number of additions (including half-and-add and triple-and-add) is equal to the number of items in the presentation minus one. Half-and-add is used instead of addition as long as the exponent of 1/2 is not zero. If the exponent of 1/2 is zero, but the exponent of 3 is not zero, then triple-and-add is used instead of addition. Since there is no formula for quadruple-and-add, we can only use the typical addition operation if the exponents of 1/2 and 3 are zero at the same time. A total of b_1 point halving (including half-and-add), t_1 point tripling (including triple-and-add) and t_1 point septupling are required in the execution of Algorithm 3.

Algorithm 3. Scalar multiplication for elliptic curves using SMBR over binary fields

Input: The SMBR of a positive integer $k=\sum_{i=1}^{m} s_i(\frac{1}{2})^{b_i}3^{t_i}7^{q_i}$, with $s_i \in \{-1,1\}$, $b_1 \geq b_2 \geq \ldots \geq b_m \geq 0$, $t_1 \geq t_2 \geq \ldots \geq t_m \geq 0$, $q_1 \geq q_2 \geq \ldots \geq q_m \geq 0$; a point $P \in E\left(F_{2^m}\right)$.

Output: The point $kP \in E\left(F_{2^m}\right)$

```
1. $Z \leftarrow s_1 P$
2. For $i = 1,\ldots,m-1$ do
3.   $u \leftarrow b_i - b_{i+1}$
4.   $v \leftarrow t_i - t_{i+1}$
5.   $w \leftarrow q_i - q_{i+1}$
6.   If $u = 0$ then
7.     $Z \leftarrow 7^w Z$
8.     If $v \neq 0$ then
9.       $Z \leftarrow 3\left(3^{v-1}Z\right) + s_{i+1}P$ //(Triple-and-Add)
10.    Else
11.      $Z \leftarrow Z + s_{i+1}P$
12.  Else
13.    $Z \leftarrow 7^w Z$
14.    $Z \leftarrow 5^v Z$
15.    $Z \leftarrow \left(\frac{1}{2}\right)^{u-1} Z$
16.    $Z \leftarrow \left(\frac{1}{2}\right)Z + s_{i+1}P$ //(Half-and-Add)
17.Return Z
```

5 Results and discussion

The experiments were performed on the elliptic curve recommended by the National Institute of Standards and Technology (NIST). The experiments are divided into three test groups in total, it including the elliptic curves NIST B-163, NIST B-233 and NIST B-283, and the size of the binary field was selected as 160-bit, 233-bit and 283-bit, respectively. In order to analyze the performance of the scalar multiplication algorithm proposed in this paper more visually, we compare our algorithm with the NAF proposed in [4], the DBC proposed in [10], the MBNR proposed in [11] and the MMBR proposed in [22]. For each test group, 1000 large integer scalar quantities k are selected at random, scalar multiplication is performed with each algorithm in turn without using any precomputation and pre-storage points, the average of the number of field operations of each algorithm is counted, and the number of inverse, square and multiplication are expressed in terms of I, S and M, respectively. The experimental environment is: the hardware environment is Intel (R) Core (TM) i7 CPU @ 2.20 GHz, the installed memory is 16 GB, the software environment is LINUX operating system, and the algorithms are implemented in C/C++ with the Multiprecision Integer and Rational Arithmetic C/C++ Library.

In this section, the performance of our proposed algorithm is described in detail. In order to clearly compare the total computational cost of the different algorithms, we select $[i]/[m] = 8$ and ignore $[s]$ over binary fields. The comparison of the total computational cost of different scalar multiplication algorithms is shown in Fig. 1, the cost of all algorithms rises as the length of the scalar k grows. Since our algorithm and MMBR both use point halving operation, the computational cost is significantly lower than other algorithms. Compared with MMBR, our algorithm has better performance, and the greater the length of the scalar k, the more obvious the advantage of our algorithm.

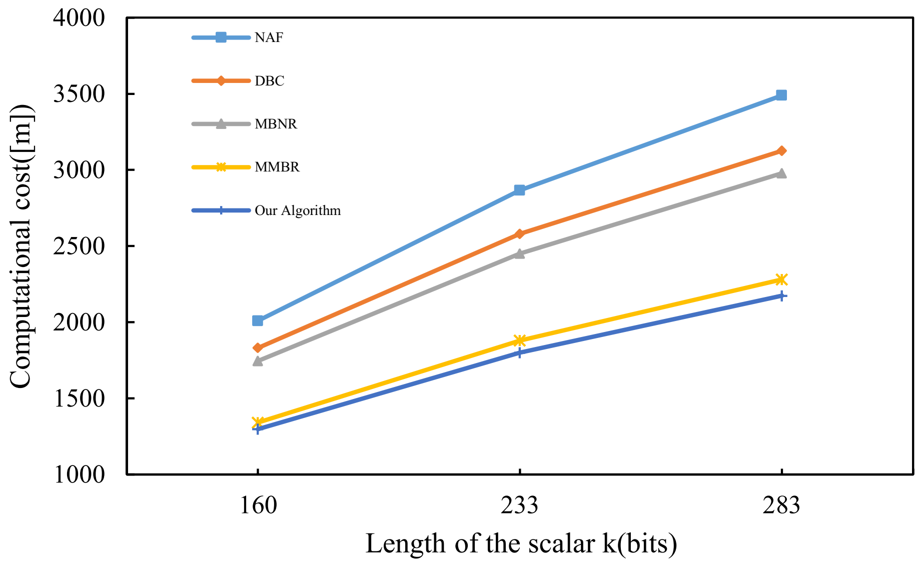

Fig. 1 Comparison of the total computational cost of different scalar multiplication algorithms. Compares the performance of our method with the NAF method in [4], the DBC method in [10], the MBNR method in [11] and the MMBR method in [22]

This is because our algorithm uses new septuple formulas we proposed, which is more effective in reducing the number of expensive field inverse operation than the quintuple formulas.

Table 3 shows that detailed data of computational cost of different scalar multiplication algorithms. On the curve NIST B-163 ($k = 160$ bit), the cost of our algorithm is 35% lower than the NAF, 29% lower than the DBC, 25% lower than the MBNR and 3% lower than the MMBR. On the curve NIST B-233($k = 233$ bit), the cost of our algorithm is 37% lower than the NAF, 30% lower than the DBC, 26% lower than the MBNR and 4% lower than the MMBR. On the curve NIST B-283($k = 283$ bit), the cost of our algorithm is 38% lower than the NAF, 30% lower than the DBC, 27%lower than the MBNR and 4% lower than the MMBR. Based on the above results, we can see that the cost of the algorithm in this paper is significantly reduced compared with NAF, DBC, and MBNR and has a small enhancement compared with MMBR, and the improvement effect becomes more obvious with the increase in the length of the scalar k. As a result, in the case of the same elliptic curve over binary fields and the same field size, the scalar multiplication algorithm we propose is more efficient.

6 Conclusions

ECC is a crucial method to ensure secure communication between devices in the IoT. Scalar multiplication is one of the major operations in ECC. How to reduce the computational complexity of ECC scalar multiplication is key to maintaining the long-term stable functioning of IoT devices. There has been much research proposing schemes to speed up scalar multiplication. In this paper, we present an efficient formula for point septupling on elliptic curves over binary fields in affine coordinates and a scalar multiplication algorithm based on the step multi-base representation. The septuple formula uses division polynomial and multiplexing of intermediate values in affine coordinates to speed up computations by more than 14%. The scalar multiplication algorithm is based

Table 3 Detailed data of computational cost of different scalar multiplication algorithms with $[i]/[m]= 8$

Curve	Algorithm	Propose	I	S	M	Cost ($\approx [m]$)
NIST B-163 ($k = 160$ bit)	NAF	[4]	163	217	705	2009
	DBC	[10]	137	366	735	1831
	MBNR	[11]	114	337	833	1745
	MMBR	[22]	77	162	725	1341
	Our algorithm	This paper	74	168	706	1298
NIST B-233 ($k = 233$ bit)	NAF	[4]	233	349	1003	2867
	DBC	[10]	202	465	964	2580
	MBNR	[11]	158	420	1187	2451
	MMBR	[22]	103	227	1048	1880
	Our algorithm	This paper	98	215	1017	1801
NIST B-233 ($k = 233$ bit)	NAF	[4]	283	437	1225	3489
	DBC	[10]	241	589	1198	3126
	MBNR	[11]	194	535	1426	2978
	MMBR	[22]	124	273	1289	2281
	Our algorithm	This paper	116	271	1246	2174

on the SMBR with $B = \{1/2, 3, 7\}$, which drastically reduces the computational cost by using point halving and the point septupling we propose. The experimental results indicate that our method can effectively reduce the cost of scalar multiplication for elliptic curves over the binary fields and contribute to the lightweight of ECC. In addition, the elliptic curve scalar multiplication method studied in this paper is still at the theoretical research stage, and future research needs to further consider the newer iterations of IoT terminal devices.

Abbreviations
IoT: Internet of things; ECC: Elliptic curve cryptography; SMBR: Step multi-base representation; NB-IoT: Narrowband Internet of Things; IFP: Integer factorisation problems; DLP: Discrete logarithm problem; ECDLP: Elliptic curve discrete logarithm problem; NAF: Non-adjacent form; JSF: Joint sparse form; RTNAF: τ-Adic non-adjacent form; DBNS: Double-base number system; DBC: Double-base chain; MBNR: Multi-base number representation; SCA: Side-channel attacks; NIST: National institute of standards and technology.

Acknowledgements
This work was supported by the National Natural Science Foundation of China (61971014), National Key Research and Development Project of China (2019YFB2102303).

Authors' contributions
CG contributed in investigation, the overall design, design of the septuple formula and the scalar multiplication algorithm, result analysis, and draft manuscript writing. BG contributed in methodology, reviewing and editing the manuscript, and funding acquisitions. All the authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China (61971014), National Key Research and Development Project of China (2019YFB2102303).

Availability of data and materials
The data used to support the findings of this study is included within the article.

Declarations

Competing interests
No conflict of interest exits in the submission of this manuscript, and manuscript is approved by all authors for publication. I would like to declare on behalf of my co-authors that the work described was original research that has not been published previously, and not under consideration for publication elsewhere, in whole or in part. All the authors listed have approved the manuscript that is enclosed.

Received: 29 December 2020 Accepted: 24 March 2021
Published online: 07 April 2021

References
1. W. Diffie, M.E. Hellman, New directions in cryptography. IEEE Trans. Inf. Theory **22**(6), 644–654 (1976)
2. V.S. Miller, Use of elliptic curves in cryptography, in *Lecture Notes in Computer Sciences 218 on Advances in Cryptology—CRYPTO 85* (1986), pp. 417–426.
3. N. Koblitz, Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)
4. F. Morain, J. Olivos, Speeding up the computations on an elliptic curve using addition-subtraction chains. Inf. Theory Appl. **24**, 531–543 (1989)
5. J.A. Solinas, Low-weight binary representations for pairs of integers. Center for Applied Cryptographic Research, University of Water-loo, Waterloo. Research Report CORR 2001–41.
6. N. Koblitz, CM curves with good cryptographic properties, in *Advances in Cryptology—Crypto'91* (1992), pp. 279–287.
7. J.A. Solinas, Efficient arithmetic on Koblitz curves. Des. Codes Cryptogr. **19**(2), 195–249 (2000)
8. H. Cohen, A. Miyaji, T. Ono, Efficient elliptic curve exponentiation using mixed coordinates, in *International Conference on the Theory and Application of Cryptology and Information Security* (1998), pp. 51–65.
9. V.S. Dimitrov, G. Jullien, W. Miller, Theory and applications of the double-base number system. IEEE Trans. Comput. **48**(10), 1098–1106 (1999)
10. V.S. Dimitrov, L. Imbert, P.K. Mishra, Efficient and secure elliptic curve point multiplication using double-base chains, in *11th International Conference on the Theory and Application of Cryptology and Information Security* (2005), pp. 59–78.
11. P.K. Mishra, V.S. Dimitrov, Efficient quintuple formulas for elliptic curves and efficient scalar multiplication using multibase number representation, in *10th International Conference on Information Security* (2007), pp. 390–406.
12. D. Hankerson, J. Lopez Hernandez, A. Menezes, Software implementation of elliptic curve cryptography over binary fields, in *2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000)* (2000), pp. 1–24.

13. K. Fong, D. Hankerson, J. Lopez, A. Menezes, Field inversion and point halving revisited. IEEE Trans. Comput. **53**(8), 1047–1059 (2004)
14. B. Chevalier-Mames, M. Ciet, M. Joye, Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Trans. Comput. **53**(6), 760–768 (2004)
15. E.W. Knudsen, Elliptic scalar multiplication using point halving, in *5th Annual International Conference on the Theory and Application of Cryptology and Information Security* (1999), pp. 135–149.
16. R. Schroeppel, *Elliptic Curve Point Ambiguity Resolution Apparatus and Method*, U. S. Patent 7200225.
17. M. Ciet, M. Joye, K. Lauter, P.L. Montgomery, Trading inversions for multiplications in elliptic curve cryptography. Des. Codes Cryptogr **39**(2), 189–206 (2006)
18. S.M. Cho, S.G. Gwak, C.H. Kim, S. Hong, Faster elliptic curve arithmetic for triple-base chain by reordering sequences of field operations. Multimed. Tools Appl. **75**(22), 1–13 (2016)
19. D. Bernstein, P. Birkner, T. Lange, C. Peters, Optimizing double-base ellipticcurve single-scalar multiplication, in *Progress in Cryptology—INDOCRYPT 2007. Lecture Notes in Computer Science*, vol. 4859 (2007), pp. 167–182.
20. G.N. Purohit, A.S. Rawat, K. Manoj, Elliptic curve point multiplication using MBNR and point halving. Int. J. Adv. Netw. Appl. **3**(5), 1329–1337 (2012)
21. L. Zhongxi, Z. Tiejun, D. Dongya, Algorithm for directly computing 7p elliptic curves and its application. J. Comput. Appl. **33**(7), 1870–1874 (2013)
22. M.I. Abdulwahed, R. Mohamad et al., An algorithm to enhance elliptic curves scalar multiplication combining MBNR with point halving. Appl. Math. Sci. **4**(26), 1259–1272 (2010)

Publisher's Note

Chong Guo received his B.S. degree from Beijing Forestry University in 2018. He is currently pursuing the M.S. degree in Computer science and technology from Beijing University of Technology. His research interests include the security of Internet of Things (IoT), cryptography and trusted computing.

Bei Gong received his B.S. degree from Shandong University in 2005, and the Ph.D. degree from the Beijing University of Technology in 2012. He participates in six National invention patent and one monograph textbooks. In the past five years, he has published more than 30 papers in the first-class SCI/EI and other international famous journals and top international conferences in relevant research fields. His research interests include trusted computing, Internet of things security, mobile Internet of things, and mobile edge computing. He has presided over 8 national projects such as the National Natural Science Foundation and 6 provincial and ministerial projects such as the general science and technology program of Beijing Municipal Education Commission.