Cybersecurity

RESEARCH

Open Access

Minimizing CNOT-count in quantum circuit of the extended Shor's algorithm for ECDLP

Xia Liu[1,2], Huan Yang[3] and Li Yang[1*]

Abstract

The elliptic curve discrete logarithm problem (ECDLP) is a popular choice for cryptosystems due to its high level of security. However, with the advent of the extended Shor's algorithm, there is concern that ECDLP may soon be vulnerable. While the algorithm does offer hope in solving ECDLP, it is still uncertain whether it can pose a real threat in practice. From the perspective of the quantum circuits of the algorithm, this paper analyzes the feasibility of cracking ECDLP using an ion trap quantum computer with improved quantum circuits for the extended Shor's algorithm. We give precise quantum circuits for extended Shor's algorithm to calculate discrete logarithms on elliptic curves over prime fields, including modular subtraction, three different modular multiplication, and modular inverse. Additionally, we incorporate and improve upon windowed arithmetic in the circuits to reduce the CNOT-counts. Whereas previous studies mostly focused on minimizing the number of qubits or the depth of the circuit, we focus on minimizing the number of CNOT gates in the circuit, which greatly affects the running time of the algorithm on an ion trap quantum computer. Specifically, we begin by presenting implementations of basic arithmetic operations with the lowest known CNOT-counts, along with improved constructions for modular inverse, point addition, and windowed arithmetic. Next, we precisely estimate that, to execute the extended Shor's algorithm with the improved circuits to factor an n-bit integer, the CNOT-count required is $1237n^3 / \log n + 2n^2 + n$. Finally, we analyze the running time and feasibility of the extended Shor's algorithm on an ion trap quantum computer.

Keywords Elliptic curve discrete logarithm problem, Extended Shor's algorithm, Quantum circuits, Ion trap quantum computer

Introduction

Elliptic curve cryptography (ECC) has attracted wide attention for its unique advantages since it was introduced in the 1980s (Miller 1985; Koblitz 1987). The safety of ECC relies on the elliptic curve discrete logarithm problem (ECDLP), which is the discrete logarithm problem (DLP) on the cyclic subgroup with a point on the elliptic curve as the generator. Although there are many attempts to solve DLP, the best-known classical algorithm for DLP is still exponentially complex (Miyaji 1992). Fortunately, with the development of quantum computing, the emergence of quantum algorithms offers hope for solving such problems. The most representative and compelling quantum algorithm is Shor's algorithm (Shor 1994, 1999), which can theoretically solve DLP over multiplicative groups for the prime fields in polynomial time (Shor 1994, 1999). This algorithm can be extended to elliptic curve groups (we call it extended Shor's algorithm in this paper), which makes ECDLP theoretically not difficult for a quantum computer, thus posing a threat to the cryptography system based on ECDLP. However, the gate number of a quantum algorithm's circuit determines the time to run the quantum algorithm on a quantum

*Correspondence:
Li Yang
yangli@iie.ac.cn
[1] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] Beijing Youzhuju Network Technology Co., Ltd., Beijing, China

computer and the exact quantum gates number of the extended Shor's algorithm has not been analyzed. Therefore, it is debatable whether the extended Shor's algorithm can pose a threat to ECC, which is exactly what we are trying to figure out. From the perspective of running time, for the extended Shor's algorithm to have a threat to ECC, it must run in a reasonable time. Therefore, our goal in this paper is to give a lower bound of the optimized circuits on the running time of the extended Shor's algorithm. A tight lower bound is difficult to obtain in theory. Instead, we give an implementation lower bound of optimized circuits from the view of circuit synthesis and optimization.

Quantum computers implement quantum computation by taking as input superposition quantum states representing all the different possible inputs and simultaneously evolving them into the corresponding outputs using a sequence of unitary transformations (Yao 1993; Nielsen and Chuang 2002; Chiribella et al. 2008; Dong et al. 2013; Nash et al. 2020; Kimura et al. 2021; Liu et al. 2021, 2022; Gao et al. 2022). Quantum computing can be described as a quantum circuit in which the unitary transformations are represented by quantum gates. The most basic quantum gates are the controlled-NOT (i.e., CNOT) gate and single-qubit gates. In an ion trap quantum computer, the operation time of the non-adjacent CNOT is much longer than that of single-qubit quantum gates, and the CNOT gates can only be executed in series (Yang and Zhou 2013). Therefore, the number of CNOT gates contained in the quantum circuit of a quantum algorithm largely determines the running time of this algorithm.

Since the advent of the first quantum algorithm to attack ECC in Boneh and Lipton (1995), research in this field has attracted extensive attention. Eicher and Opoku (1997), Proos and Zalka (2003), Kaye and Zalka (2004) proposed quantum algorithms that attack ECDLP defined on finite fields F_p and F_{2^m}. Roetteler et al. (2017) studied the extended Shor's algorithm to attack ECDLP on F_p and improved the algorithm of modular inverse in Proos and Zalka (2003). The resources needed in terms of the number of Toffoli gates were $448n^3\log_2(n) + 4090n^3$, but only rough results of $O(n^3)$ were obtained for the number of CNOT gates. Häner et al. (2020) improved the Kaliski algorithm in Roetteler et al. (2017). Fewer T gates were used in the circuit of modular inverse using windowed arithmetic introduced by Gidney (2019). In view of the size of the quantum computer, i.e., the number of qubits, a quantum circuit for calculating the discrete logarithm problem on a binary elliptic curve is optimized in Banegas et al. (2021).

Note that the resources required by the quantum circuit in previous papers did not analyze the CNOT-count in detail, but with the development of ion trap quantum computers, the running time of an algorithm is greatly affected by the CNOT-count (Knight et al. 1999). If even all the CNOTs in the circuits of an algorithm cannot be run within a reasonable time, it is unnecessary to consider other gates, such as T gates. Therefore, this paper analyzes the feasibility of the quantum algorithm to attack ECDLP by studying the CNOT-count of the circuit and discusses the applications of windowed arithmetic in detail. It is worth noting that based on the physical limitations of quantum computers, we consider whether a sufficiently large quantum computer in the future can complete the extended Shor's algorithm in a reasonable running time, so we do not focus on the number of qubits.

Our contributions

In this paper, we give precise quantum circuits for the extended Shor's algorithm to calculate discrete logarithms on elliptic curves over prime fields. More specifically, we have the following contributions.

1. We construct and improve the circuits of basic operations including modular subtraction, three different modular multiplication, modular inverse, and windowed arithmetic and further improve the quantum circuits of extended Shor's algorithm.
2. We combine the window technique with basic modular operations to reduce the CNOT-count, and further analyze the running time of the extended Shor's algorithm on ion trap quantum computers according to the CNOT-count we obtained.
3. We study the feasibility of the extended Shor's algorithm on ion quantum computers under the premise that the fault-tolerant quantum computer has enough space, further illustrating whether Shor's algorithm can really pose a threat to cryptosystems such as ECC.

Outline

The rest of the paper is organized as follows. Preliminaries section is the introduction to ECDLP and the elliptic curve group law. Quantum circuits for algebraic problems section introduces the basic circuits to compute scalar multiplication on the elliptic curve groups required by the algorithm, including modular multiplication, modular inverse, windowed arithmetic, etc. In Quantum circuits of point addition on elliptic curve groups section, we design a new method to calculate the point addition reversibly out-of-place (storing the results in a new register), which is different from the in-place method (replacing the input value by the sum) in Roetteler et al. (2017)

and reduces the CNOT-count. Discussion and conclusion section discusses the time required to attack ECDLP.

Preliminaries

In this section, we first give a brief description of DLP, and then show Shor's algorithm for solving DLP. Next, we elaborate on the algorithm for solving ECDLP, which we call extended Shor's algorithm.

Shor's quantum algorithm for solving the DLP

Discrete logarithms problem

Let g be a generator of a finite cyclic group G with the known order ord(g) = k, i.e. $g^k = 1$. The DLP over G is defined as, given an element $x \in G$, determining the unique $r \in [0, |G| - 1]$ such that $g^r = x$, then $r = \log_g x$. Consider the case when G is the additive group Z_N, where N is a positive integer and $\gcd(g, N) = 1$. Here the DLP is to find r satisfying $r \cdot g \equiv x \mod N$. The DLP over the Z_N can be solved by finding the multiplicative inverse of g modulo N with the extended Euclidean algorithm in polynomial time ($O(\log_2^2 N)$) (Proos and Zalka 2003). However, in the group $G = Z_p^*$ (i.e., the multiplicative group modulo p and $g^r \equiv x \mod p$), there was no classical algorithm to solve the DLP (i.e., calculate $r = \log_g x$) until Shor (1994, 1999) proposed a quantum algorithm that could theoretically solve this problem in polynomial time.

Shor's quantum algorithm

To be specific, Shor's algorithm uses three quantum registers to solve the DLP, each quantum register has n qubits and satisfies $p \le q = 2^n < 2p$. Shor's algorithm for DLP is shown as follows.

Algorithm 1 Shor's quantum algorithm for DLP

Require: g, x and p, such that $\gcd(g, p) = 1$.
Ensure: integer r such that $g^r = x \mod p$.

1: Prepare a $3(p-1)$-qubit initial state in three quantum registers $|0\rangle|0\rangle|0\rangle$.
2: Apply the Hadamard transform $H^{\otimes 2(p-1)}$ to the first two quantum registers to obtain

$$\frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a\rangle|b\rangle|0\rangle. \tag{1}$$

3: Perform a unitary transformation U_f such that $U_f|a\rangle|b\rangle|0\rangle \to |a\rangle|b\rangle|g^a x^{-b} \mod p\rangle$ to transform the initial state into

$$\frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}|a\rangle|b\rangle|g^a x^{-b} \mod p\rangle. \tag{2}$$

4: Perform quantum Fourier transform on the first two registers to get the state

$$\begin{aligned}&\frac{1}{(p-1)q}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2}\sum_{c=0}^{q-1}\sum_{d=0}^{q-1}\exp\left[\frac{2\pi i}{q}(ac+bd)\right]|c\rangle|d\rangle|g^a x^{-b} \mod p\rangle\\ =&\frac{1}{(p-1)q}\sum_{z=0}^{p-1}\sum_{c=0}^{q-1}\sum_{d=0}^{q-1}\sum_{g^{a-rb}=z \mod p}\exp\left[\frac{2\pi i}{q}(ac+bd)\right]|c\rangle|d\rangle|z\rangle.\end{aligned} \tag{3}$$

5: Measure and obtain the probability of $|c\rangle|d\rangle|z\rangle$ is:

$$P_{c,d,z} = \frac{1}{(p-1)^2 q^2}\left|\sum_{g^{a-rb}=z \mod p}\exp\left[\frac{2\pi i}{q}(ac+bd)\right]\right|^2, \tag{4}$$

determine r with high probability by classical post-processing on the measured results.

Using the above algorithm, Shor proved that r can be calculated with high probability in polynomial time. Based on Shor's algorithm to solve DLP, next we show the case of ECDLP.

Extended Shor's quantum algorithm for solving the ECDLP

Elliptic curve discrete logarithms problem

Let F_p be a field of characteristic $p \neq 2, 3$. An elliptic curve over F_p is the set of solutions $(x, y) \in F_p \times F_p$ to the equation

$$y^2 = x^3 + Ax + B, \tag{5}$$

where $A, B \in F_p$ satisfy $4A^3 + 27B^2 \neq 0$, together with the point O at infinity. The set of all the points on the elliptic curve is $E(F_p) = \{(x, y)|y^2 = x^3 + Ax + B; A, B \in F_p\} \cup \{\infty\}$. Then $E(F_p)$ forms the Abelian group with a point addition operation and O as the neutral element. Let $P \in E(F_p)$ be a generator of $\langle P\rangle$, which is a cyclic subgroup of $E(F_p)$ of known order $\mathrm{ord}(\mathrm{P}) = \mathrm{r}$, i.e., $rP = O$. Similar to DLP, the goal of ECDLP is to find the unique integer $m \in \{1, \ldots, r\}$ such that $mP = Q$, where $r, m \in F_p$ and Q is a given point in $\langle P\rangle$. Hasse (1936) pointed out that the number of all the points on the elliptic curve is $\#E(F_p) = p + 1 - t, |t| \leq 2\sqrt{p}$. Thus the order of $\langle P\rangle$ is no larger than p. Therefore, when analyzing ECDLP on $\langle P\rangle$, the order can be set to p, which has no effect on the results.

Extended Shor's quantum algorithm

Different from Shor's quantum algorithm, the extended Shor's algorithm uses two n-qubit and one $2n$-qubit registers with $n = \lceil \log_2 p \rceil$ to solve the ECDLP. The specific algorithm for ECDLP is shown as follows.

Algorithm 2 Extended Shor's quantum algorithm for ECDLP

Require: An elliptic curve $E(F_p)$ with two points P and Q.
Ensure: The smallest integer m such that $Q = mP$.

1: Prepare a $4n$-qubit initial state $|0\rangle^{\otimes n}|0\rangle^{\otimes n}|kP\rangle^{\otimes 2n}$ in three quantum registers.
2: Perform a Hadamard transform on the first two registers to transform the initial state into

$$\frac{1}{2^n}\sum_{a=0}^{2^n-1}\sum_{b=0}^{2^n-1}|a\rangle|b\rangle|kP\rangle. \tag{6}$$

3: Perform a unitary transformation U_f such that $U_f|a\rangle|b\rangle|kP\rangle \rightarrow |a\rangle|b\rangle|((a+k)P + bQ) \mod p\rangle$ to transform the initial state into

$$\frac{1}{2^n}\sum_{a=0}^{2^n-1}\sum_{b=0}^{2^n-1}|a\rangle|b\rangle|((a+k)P + bQ) \mod p\rangle. \tag{7}$$

4: Perform a quantum Fourier transform on the first two registers to obtain the state

$$\frac{1}{2^n}\sum_{a=0}^{2^n-1}\sum_{b=0}^{2^n-1}\sum_{c=0}^{2^n-1}\sum_{d=0}^{2^n-1}\exp[\frac{2\pi i}{2^n}(ac + bd)]|c\rangle|d\rangle|((a+k)P + bQ) \mod p\rangle. \tag{8}$$

5: Measure the first two registers and determine m with high probability by classical post-processing on the measured results.

The initial state of the third register is $|kP\rangle$ instead of $|0\rangle$ to satisfy the point addition rule on the elliptic curve. Whether we use $|0\rangle$ or $|kP\rangle$ has no effect on the result of measuring probability. The detailed proof can be seen in the Appendix.

Elliptic curve groups law

Before designing the circuits of the extended Shor's quantum algorithm, The elliptic curve group law on an affine Weierstrass curve we give the law on the group of elliptic curves.

Let $P(x_1, y_1) \neq O, Q(x_2, y_2), R(x_3, y_3) \in \langle P\rangle, P + Q = R$, the elliptic curve group law on the Eq. (5) can be computed as follows:

$$P + Q = R = \begin{cases} Q, & P = O, \\ O \quad (i.e \infty), & P = -Q = (x_2, -y_2), \\ (\lambda^2 - (x_1 + x_2), \lambda(x_1 - x_3) - y_1), & others, \end{cases} \tag{9}$$

where λ satisfies the following equation:

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & P \neq Q, \\ \frac{3x_1^2 + A}{2y_1}, & P = Q. \end{cases} \tag{10}$$

Thus we have

$$2P = (x', y') = \begin{cases} \left(\left(\frac{3x_1^2 + A}{2y_1} \right)^2 - 2x_1, \frac{3x_1^2 + A}{2y_1}(x_1 - x') - y_1 \right), & y_1 \neq O, \\ \infty, & y_1 = 0. \end{cases} \tag{11}$$

The detailed steps of how to transform coordinates from (x_1, y_1) to (x_3, y_3) can be found in Proos and Zalka (2003), Roetteler et al. (2017), Häner et al. (2020). Since the purpose of this paper is to minimize the CNOT-count of quantum circuit for the extended Shor's algorithm, we improve the previous circuits of the coordinate transformation in the Quantum circuits of point addition on elliptic curve groups section but at the cost of increasing the number of qubits, which is not the focus of this paper.

Quantum circuits for algebraic problems

In the implementation of the extended Shor's algorithm for ECDLP, the most important step is to design a quantum circuit to compute scalar multiplication on the elliptic curve groups, i.e., $((a + k)P + bQ) \mod p$, which includes a series of modular operations. In this section, we design the circuits of modular subtraction and direct modular multiplication operations. Meanwhile, we improve a series of basic operations as well as modular inverse and windowed arithmetic.

Modular subtraction

Modular subtraction with inputs $x, y < p$ computes $|(x - y) \mod p\rangle$, where p is the known n-bit constant. The computation can be done with the following steps:

1. Subtract y from $|x\rangle$ to obtain $|x - y\rangle$ using the reverse circuit of addition.
2. If the highest bit of $|x - y\rangle$ is 1, which corresponds to $x - y < 0$, then add p to $|x - y\rangle$. Otherwise, do nothing.
3. Compare the result of step 2 with $(p - y)$. Uncompute the auxiliary qubit and get $|(x - y) \mod p\rangle$.

If the input y is a known constant, then it can be ignored in quantum circuits and this operation is called constant modular subtraction. Otherwise, it is called quantum state modular subtraction.

Next, we give the details of the quantum circuit for performing addition and comparison.

(I). Quantum circuits for addition.

We use two circuits, 1-Add_y and 2-Add_y, to perform addition, i.e., $|x\rangle|y\rangle|0\rangle \rightarrow |(x + y)_{0,\ldots,n-1}\rangle|y\rangle|(x + y)_n\rangle$. The first two quantum registers both have n qubits and the third one has 1 qubit as the highest bit of the sum. The two circuits of addition are shown below.

1. First version of the quantum circuit for addition, 1-Add_y. Cuccaro et al. (2004) presented a way to calculate the addition as shown in Fig. 1, which we denote by 1-Add_y. It shows that each MAJ (i.e., compute the majority of three qubits in-place) and each UMA (i.e., UnMajority and Add) contain two CNOTs and one Toffoli gate. Since an n-qubit 1-Add_y has n MAJs and n UMAs, it consists of $4n + 1$ CNOTs and $2n$ Toffoli gates. At the same time, based on the standard decomposition of the Toffoli gate into the Clifford+T set, we obtain that one Toffoli gate contains six CNOTs (Nielsen and Chuang 2002).

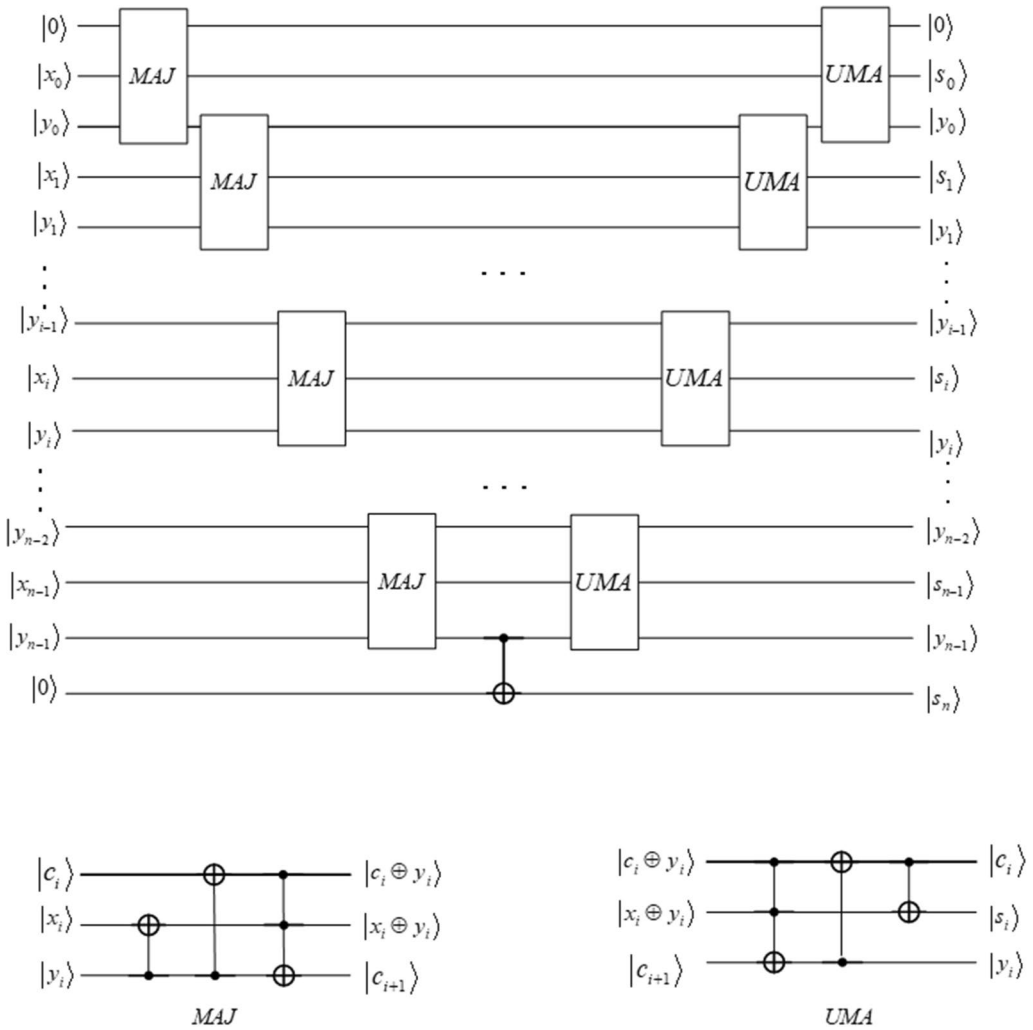

Fig. 1 The first quantum circuit of addition $1-\mathtt{Add}_y$ is constructed by MAJ blocks and UMA blocks. A MAJ block and a UMA block both have two CNOT gates and one Toffoli gate

Therefore, we conclude that the CNOT-count of an n-qubit $1-\mathtt{Add}_y$ is $16n+1$. According to $1-\mathtt{Add}_y$, we further design its controlled version in Fig. 2 with a CNOT-count of $26n+6$.

2 Second version of the quantum circuit for addition, $2-\mathtt{Add}_y$. Vedral et al. (1996) proposed another quantum circuit for calculating addition as shown in Fig. 3, where the blocks of CARRY and SUM are shown in Fig. 4 and the circuit of $\mathtt{CARRY}^{-1}$ consists of the same quantum gates in CARRY but in the reverse order. When the addend y is known, Markov and Saeedi (2012) modified the CARRY, SUM to the form shown in the last two rows of Fig. 4, that is, the y is omitted. At this point, one CARRY (or $\mathtt{CARRY}^{-1}$) contains on average 1 Toffoli gate and $\frac{1}{2}$ CNOT, and one SUM has on average 1 CNOT. Therefore, an n-qubit $2-\mathtt{Add}_y$ has a total of n CARRYs, n SUMs, $n-1$ $\mathtt{CARRY}^{-1}$s, and 1 additional CNOT. Combining with six CNOTs of one Toffoli, we conclude that the CNOT-count in $2-\mathtt{Add}_y$ is $14n-5.5$ when y is known. The left circuit in Fig. 5 is a common controlled version of $2-\mathtt{Add}_y$, while the right one proposed in Häner et al. (2020) gives a simpler controlled version: First, the control qubit ctrl uses NOT gates to control the known addend y to store in an n-qubit auxiliary register. Then $1-\mathtt{Add}$ is used because the addend y cannot be omitted. Finally, repeat the storage operation to restore the auxiliary qubits. Since encoding a known n-qubit addend y into the circuit requires an average of $\frac{n}{2}$ CNOTs, combined with the $1-\mathtt{Add}$, we conclude that $2-\mathtt{Add}$ requires $17n+1$ CNOTs.

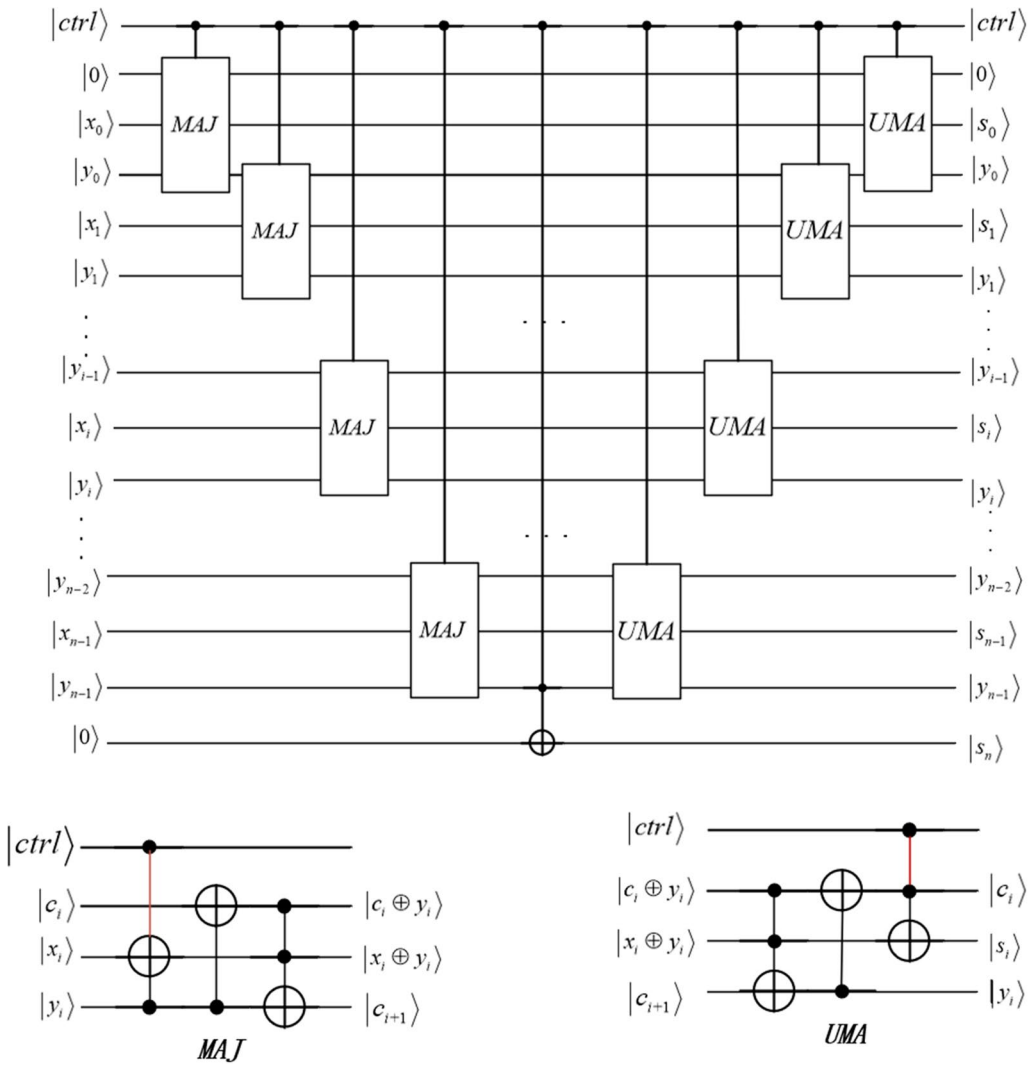

Fig. 2 The controlled version of $1-\mathtt{Add}_y$

(II). Quantum circuits for comparison

Now we also use two circuits, $1-\mathtt{Comp}_y$ and $2-\mathtt{Comp}_y$, to perform the comparison. Compare x and y by checking whether the highest qubit of $x-y$ is 0 or 1. When the highest qubit is 0, then $x-y>0$; otherwise, $x-y<0$. The difference between these two circuits is that $1-\mathtt{Comp}_y$ applies to the case where y is a known constant, while $2-\mathtt{Comp}_y$ can be used either for y known or for y unknown. We use $2-\mathtt{Comp}_y$ for all the comparisons covered in this paper and details of the two circuits are shown below.

1 First version of the quantum circuit for comparison, $1-\mathtt{Comp}_y$. $1-\mathtt{Comp}_y$ in Fig. 6 is obtained by modifying $1-\mathtt{Add}_y$ so that it outputs only the highest qubit of $|x-y\rangle$ (Markov and Saeedi 2012). But the premise is that the input is $-y+2^n$ instead of y, which means this way only works if y is a known constant instead of an unknown quantum state. When y is known, the MAJ can be simplified to Fig. 7, that is, one MAJ contains 1 Toffoli. Thus the number of CNOT in $1-\mathtt{Comp}_y$ is $12n+1$.

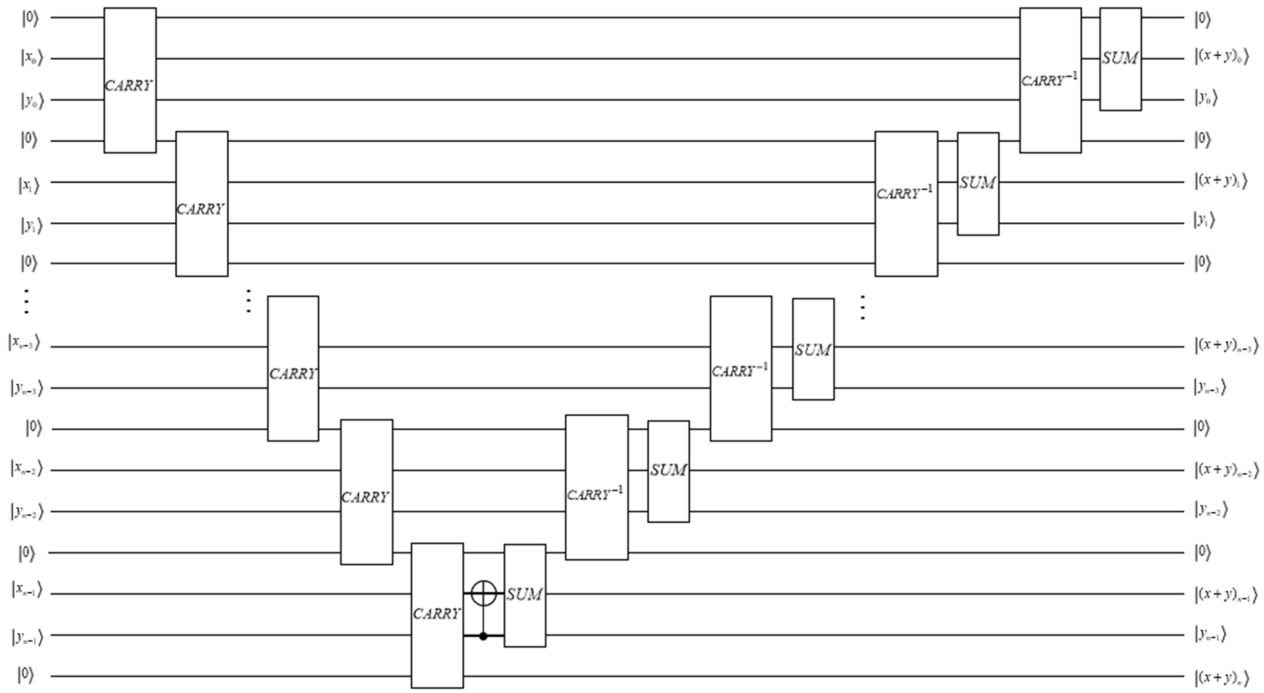

Fig. 3 The second quantum circuits of addition 2−Add, where the blocks of CARRY and SUM are shown in the first row of Fig. 4

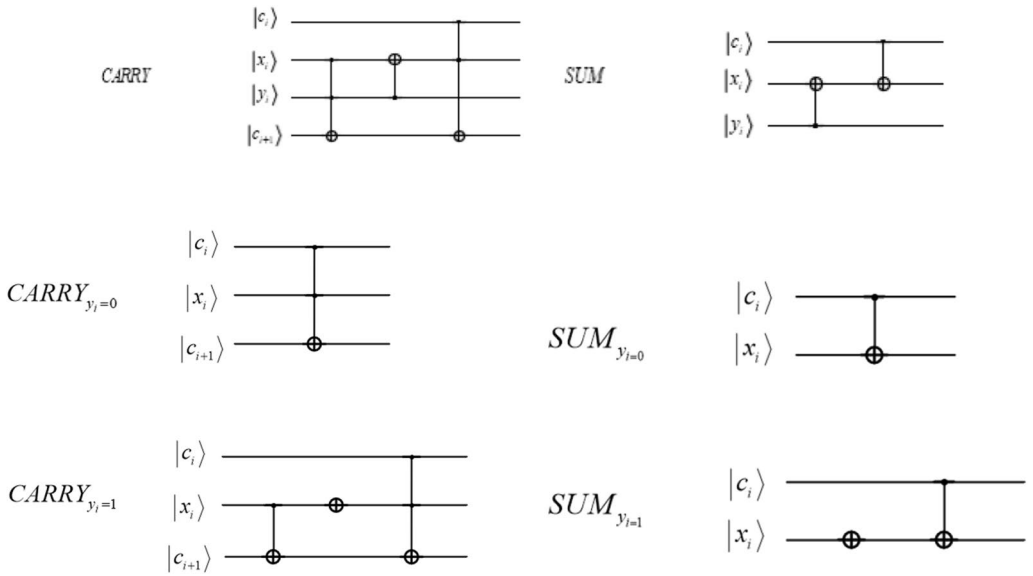

Fig. 4 The first row gives the versions of CARRY and SUM when *y* is an unknown quantum state. The last two rows show the versions of CARRY and SUM when *y* is known

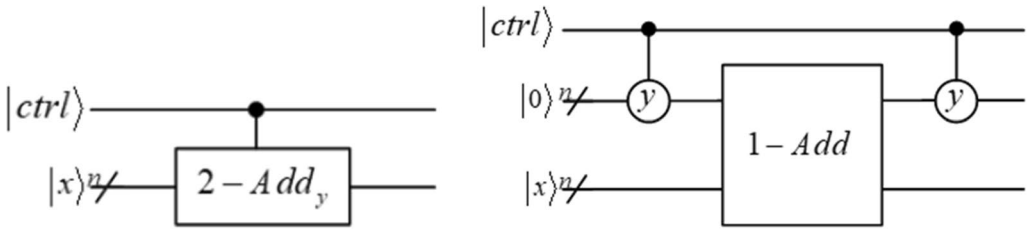

Fig. 5 The original controlled version of $2-\mathtt{Add}_y$ and the new controlled form of $2-\mathtt{Add}_y$ when y is known

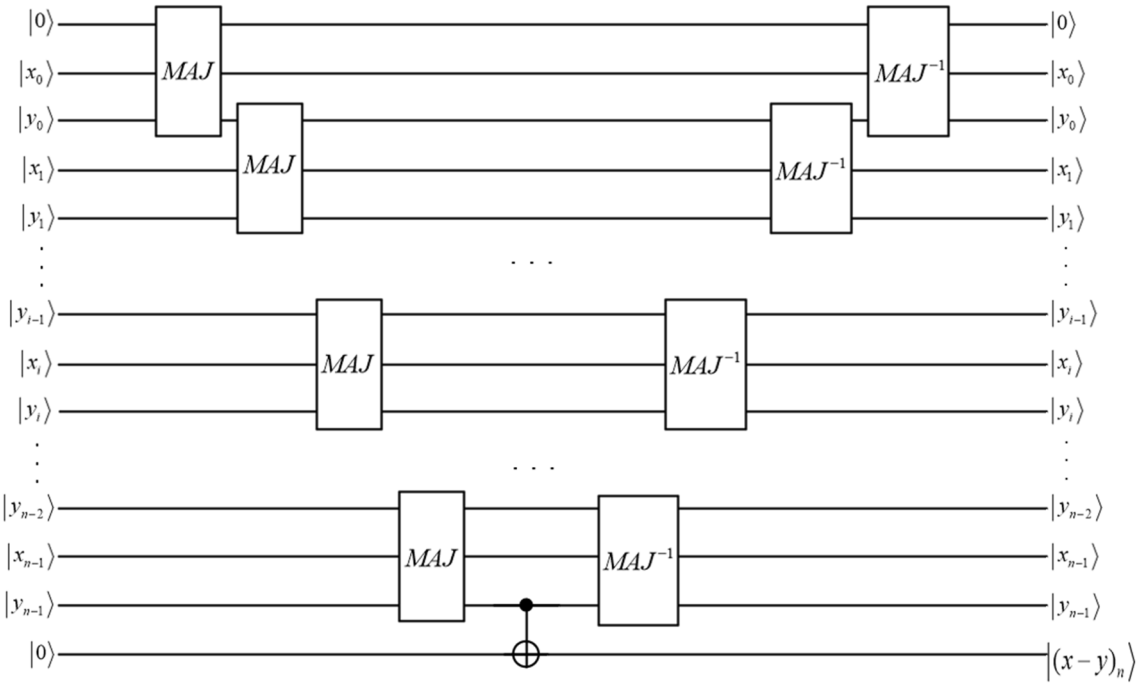

Fig. 6 $1-Comp_y$. The y is a known constant

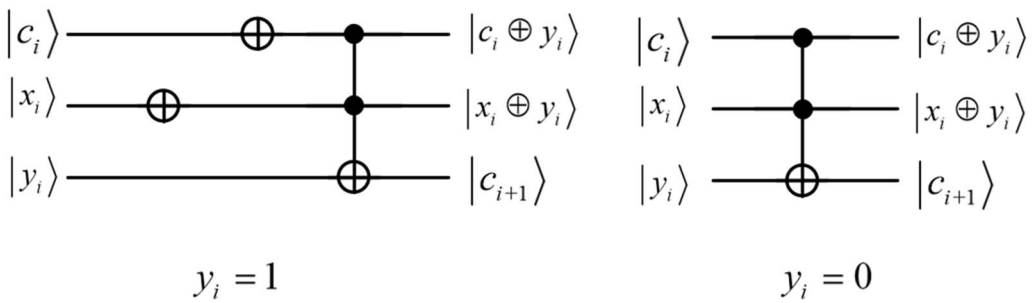

Fig. 7 The form of MAJ when y is known

2 Second version of quantum circuit for comparison, $2-\mathtt{Comp}_y$. Although the $1-\mathtt{Comp}_y$ does not work when the minus y is an unknown quantum state, it can be modified to not precompute $-y+2^n$. The specific steps are as follows. Firstly, input x, y and flip each of the x bits to get 2^n-1-x. Then use $1-\mathtt{Comp}_y$ to get the highest qubit of $2^n-1-x+y$. Finally, flip each of the x bits and the highest qubit of $2^n-1-x+y$ to recover x and get $(x-y)_n$, which represents the highest qubit of $x-y$. Following the

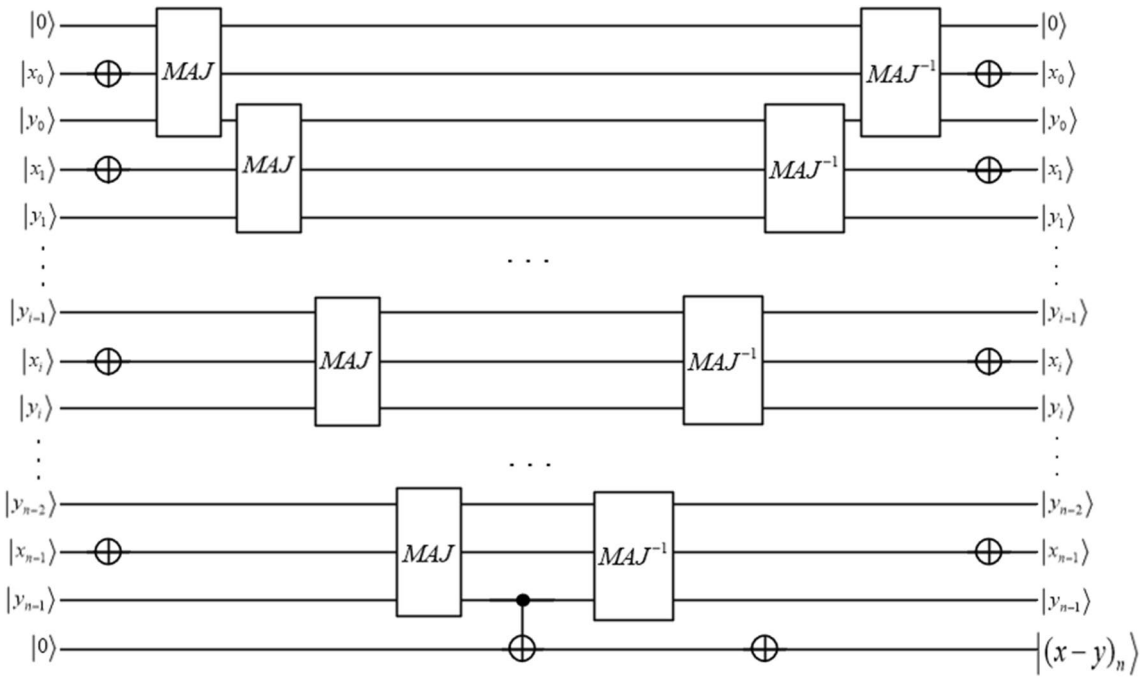

Fig. 8 $2-\mathrm{Comp}_y$. The y can be either known or unknown

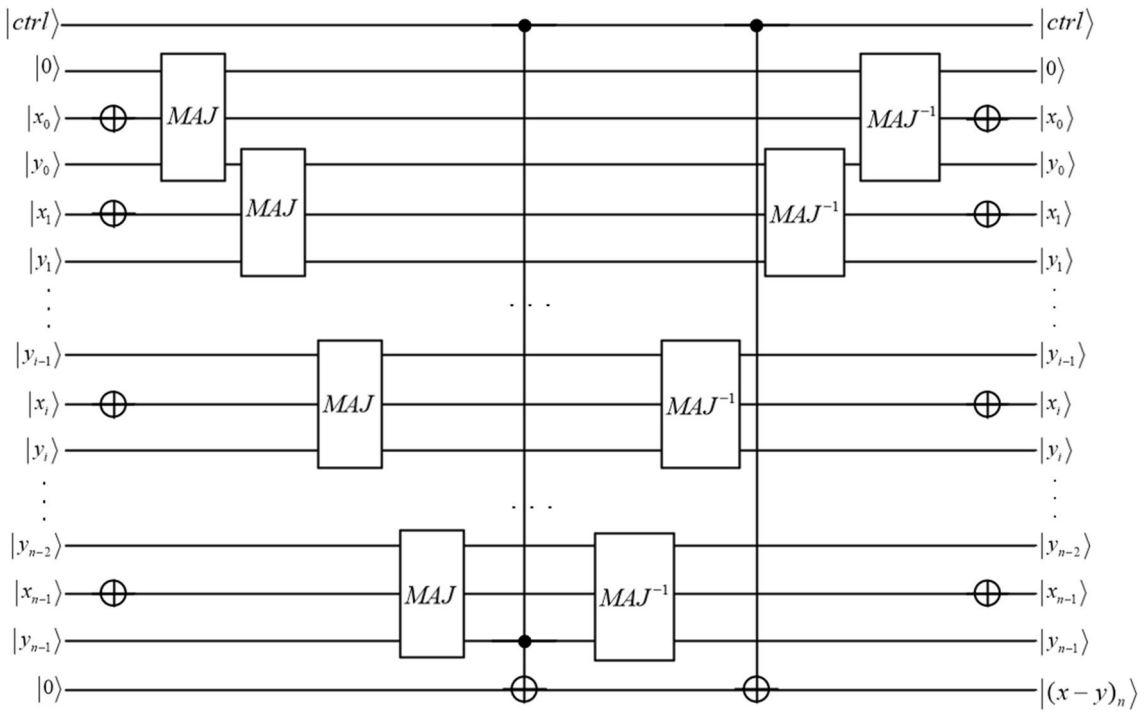

Fig. 9 The controlled version of $2-\mathrm{Comp}_y$

above steps we obtain the circuit $2-\mathtt{Comp}_y$ shown in Fig. 8. We see that $2-\mathtt{Comp}_y$ not only applies to where y is a known constant but also applies to an unknown quantum state. The CNOT-count in the former is $12n+1$, which is the same as $1-\mathtt{Comp}_y$, and the latter is $16n+1$. Figure 9 is the controlled version of $2-\mathtt{Comp}_y$. The corresponding CNOT-counts are $12n+7$ and $16n+7$, respectively.

After showing the circuits of addition and comparison in steps (I) and (II), we design the constant modular subtraction circuit $\mathtt{ModAdd}^{-1}(\cdot)$ in Fig. 10, which contains one $2-\mathtt{Add}_y^{-1}$, one CNOT, one $1-\mathtt{Add}$, one $2-\mathtt{Comp}_{p-y}$ with the known constant $p-y$, and two circuits of encoding p. Thus we conclude that the CNOT-count of $\mathtt{ModAdd}^{-1}(\cdot)$ is $43n-2.5$. Calculate $|(x+y) \mod p\rangle$ using the reverse circuit of $\mathtt{ModAdd}^{-1}(\cdot)$, which is denoted by $\mathtt{ModAdd}(\cdot)$.

The quantum state modular addition circuit $\mathtt{ModAdd}$ can be obtained in a similar way, which is shown in Fig. 11. Different from the constant modular addition, $\mathtt{ModAdd}$ contains one $1-\mathtt{Add}$, two $2-\mathtt{Comp}_y$ with the known constant y, two CNOTs, one $1-\mathtt{Add}^{-1}$, and two circuits of encoding p. Thus, we conclude that the CNOT-count of $\mathtt{ModAdd}$ is $61n+6$. Furthermore, using the reverse circuit of $\mathtt{ModAdd}$ we can calculate quantum state modular subtraction.

The controlled version of $\mathtt{ModAdd}^{-1}(\cdot)$ and $\mathtt{ModAdd}$ are shown in Figs. 12 and 13, respectively. The corresponding CNOT-counts are $46n+11$ and $71n+17$, respectively.

Negation

Given the value of $x \mod p$, it is easy to calculate $-x \mod p$ algebraically. However, performing this calculation using a quantum circuit is difficult. In order to solve this problem, Markov and Saeedi (2012) showed that it can be done by first flipping each of the bits x to get (2^n-1-x) and then subtracting (2^n-1-p) from $2-\mathtt{Add}^{-1}$ to get the result. According to these two steps, Fig. 14 shows the circuit of calculating $-x \mod p$ and Fig. 15 is its controlled version. The CNOT-count in $\mathtt{NegMod}$ is equal to that of $2-\mathtt{Add}^{-1}$, i.e. $14n-5.5$, while in the controlled circuit is $18n+1$.

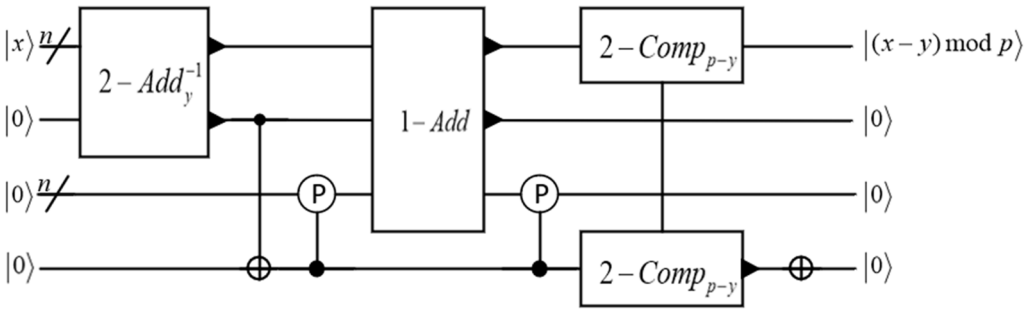

Fig. 10 Circuit of the constant modular subtraction $\mathtt{ModAdd}^{-1}(\cdot)$. Since addition is the inverse operation of subtraction and y is a known constant, we can use the reverse circuit of $2-\mathtt{Add}_y$ to compute $|x-y\rangle$ and denote it $2-\mathtt{Add}_y^{-1}$. The black triangle symbols in this figure as well as all the other figures in this paper indicate that the corresponding qubit registers are modified and hold the results of the computation

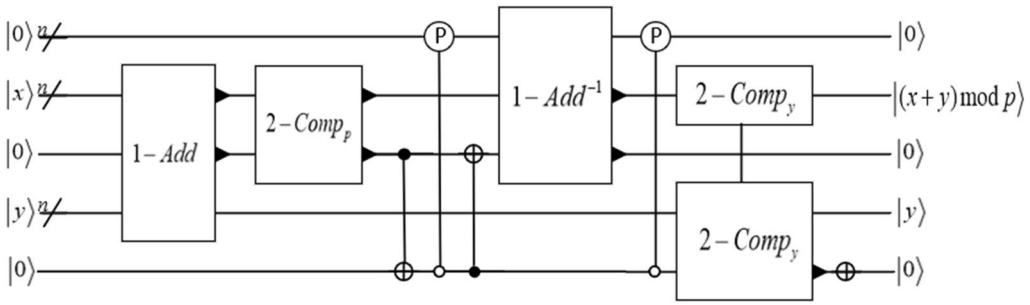

Fig. 11 Circuit of the quantum state modular addition $\mathtt{ModAdd}$. Since addition is the inverse operation of subtraction, we can use the reverse circuit of $1-\mathtt{Add}$ to compute $|x-y\rangle$ and denote it $1-\mathtt{Add}^{-1}$

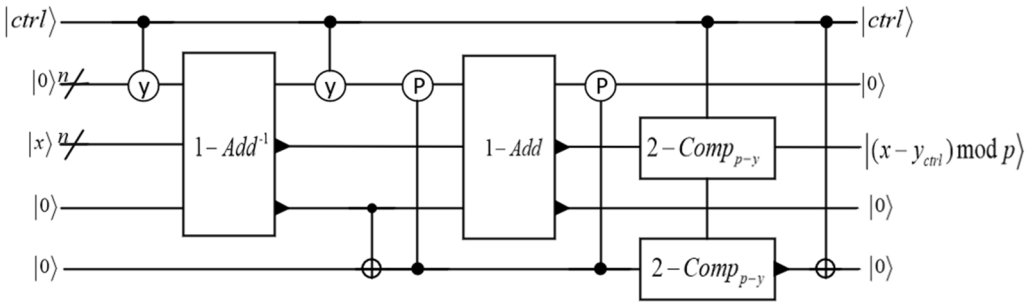

Fig. 12 The controlled version of $\texttt{ModAdd}^{-1}(\cdot)$

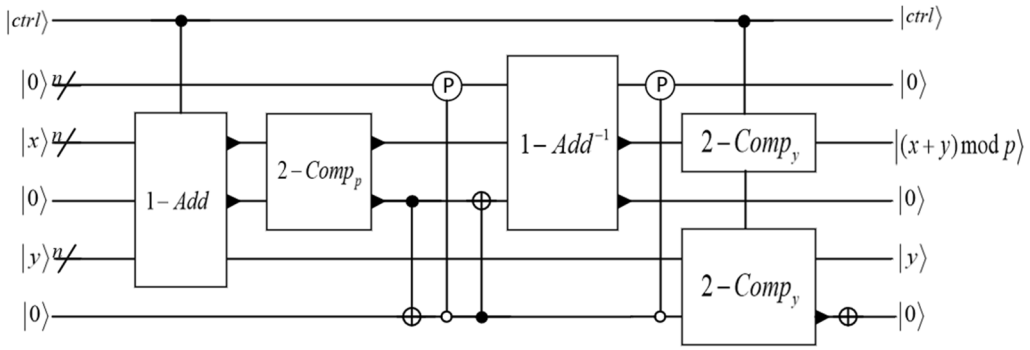

Fig. 13 The controlled version of ModAdd

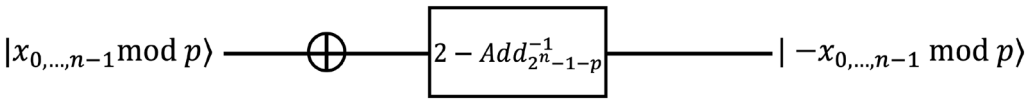

Fig. 14 The circuit of negation NegMod

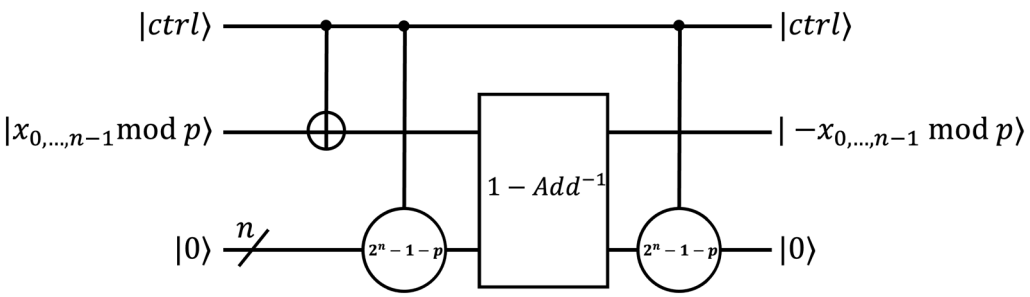

Fig. 15 The controlled version of NegMod

Modular shift

For constructing the circuit of modular shift, i.e., $|x \mod p\rangle \to |2x \mod p\rangle$, we first show the circuits of the binary shift. The functions of the binary shift are as follows.

Left shift $\texttt{l-shift}: |0x_{n-1}\cdots x_1x_0\rangle \longrightarrow |x_{n-1}\cdots x_1x_00\rangle$;
Right shift $\texttt{r-shift}: |x_{n-1}\cdots x_1x_00\rangle \longrightarrow |0x_{n-1}\cdots x_1x_0\rangle$.

The original method, as shown in Fig. 16, uses SWAP gates to implement. However, we note that there is no

need to swap two qubits with a SWAP operation if a qubit is known to be in the state of $|0\rangle$. Hence, we reconstruct the modular shift circuit for an n-qubit quantum register to reduce the CNOT-count, which is shown in Fig. 17.

1 The original shift method shown in Fig. 16 requires $3n$ CNOTs. The controlled version of the second method needs to use one qubit to control the middle CNOT in each SWAP gate. Then the circuit requires $2n$ CNOTs and n Toffoli gates in total, that is, $8n$ CNOTs.

2 Our shift method shown in Fig. 17 requires $2n$ CNOTs and the controlled version uses one control qubit to control each CNOT, which needs $2n$ Toffoli gates, that is, $12n$ CNOTs.

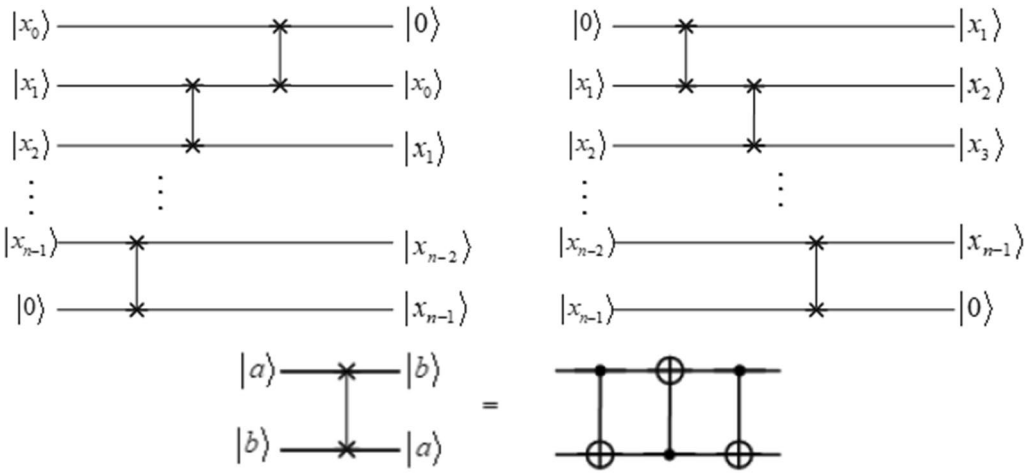

Fig. 16 First method to perform binary shift, `l-shift` and `r-shift`, respectively

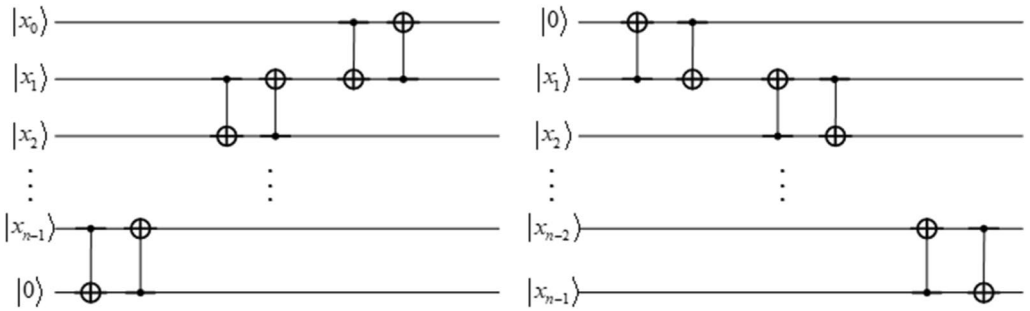

Fig. 17 Second method to perform binary shift, `l-shift` and `r-shift`, respectively

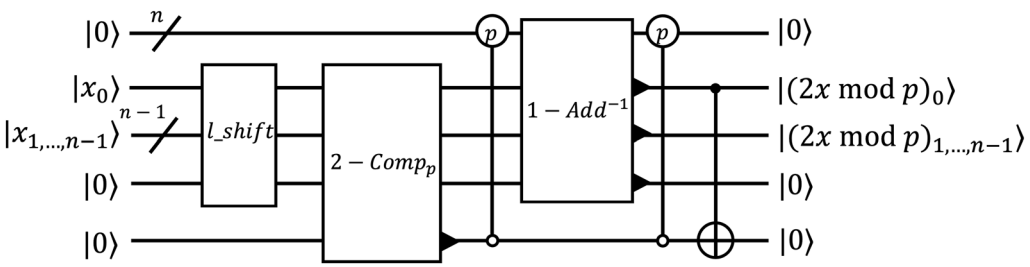

Fig. 18 Circuit of the modular shift, `ShiftMod`

Based on the above two methods to perform modular shift, we can choose an appropriate circuit to minimize the CNOT-count, that is, choose the original when a controlled mode is involved, and choose our method otherwise.

As shown in Fig. 18, we improve the modular shift by replacing the subtraction of the constant p with a comparison of the constant p. The CNOT-count of our modular shift is $31n + 15$ by selecting the appropriate binary shift method.

Modular multiplication

There are three kinds of modular multiplication methods: fast modular multiplication, Montgomery modular multiplication, and direct modular multiplication. The first way is to compute by repeating modular and conditional modular additions. The second way is often the most efficient choice for modular multiplication when modular p is not close to a power of 2. The last method is to calculate it in the most direct way, that is, first do the binary multiplication and then subtract multiples of p.

Fast modular multiplication

Proos and Zalka (2003), fast modular multiplication is used to calculate the modular multiplication, and the circuit of this method is designed in detail in section 3.3 of Ref. Roetteler et al. (2017), which requires $104n^2 - 86.5n - 11.5$ CNOTs. Furthermore, modular addition and modular shift in the fast modular multiplication apply to the circuits mentioned earlier in this paper.

Montgomery modular multiplication

According to the Montgomery algorithm (Kaliski 1995), inputting x and y, we can obtain $(x \cdot y \cdot 2^{-n} \mod p)$, where $2^{n-1} < p < 2^n$ and Roetteler et al. (2017) gave a specific quantum circuit. With the circuits of basic arithmetic operations improved earlier in this paper, we obtain the Montgomery modular quantum circuit in Fig. 19. The result in `M-Mul` is $(x \cdot y \cdot 2^{-n} \mod p)$, where `Add` is `1-Add`, and `Add`$^{-1}$ is the constant subtraction `2-Add`$^{-1}$. The reverse operation of `M-Mul`, which is denoted by `M-Mul`$^{-1}$, is used to restore the auxiliary bits. The entire quantum circuit of Montgomery modular multiplication is a combination of `M-Mul` and `M-Mul`$^{-1}$ with a CNOT-count of $90n^2 + 78n - 9$. Actually, to obtain the value $(x \cdot y \cdot 2^{-n} \mod p)$, we still need to set n CNOTs to encode the value into extra n-qubit auxiliary qubits before performing `M-Mul`$^{-1}$.

Direct modular multiplication

Now we give a method to construct the circuit of modular multiplication according to its calculation. Observe that $x \cdot y = kp + (x \cdot y \mod p)$, where $k = \lfloor \frac{x \cdot y}{p} \rfloor$ and $1 < x, y < p$. Thus we can rewrite

$$\begin{aligned} x \cdot y \mod p &= x \cdot y - kp \\ &= \sum_{i=0}^{n-1} 2^i x_i \cdot y - \sum_{i=0}^{n-1} 2^i k_i p, \quad k_i \in \{0, 1\}, \end{aligned}$$

where the second equality follows from $x \cdot y < p^2 < 2^n p$. The target result is then obtained by comparing the sizes of $x \cdot y$ and $2^i p$. Since this method is constructed directly according to the calculation, we call it direct modular multiplication. More specifically, this method is divided into the following three steps.

1 Calculate the value of $x \cdot y$.
2 For each i from $n - 1$ to 0, calculate the value of $x \cdot y - 2^i p$, i.e., $(xy)_i \cdots (xy)_{n+i} - p$. If the highest qubit of the result is 1, then add p to the result.
3 The circuit for steps 1 and step 2 is shown in Fig. 20. Run the reverse of this circuit to recover the auxiliary qubits.

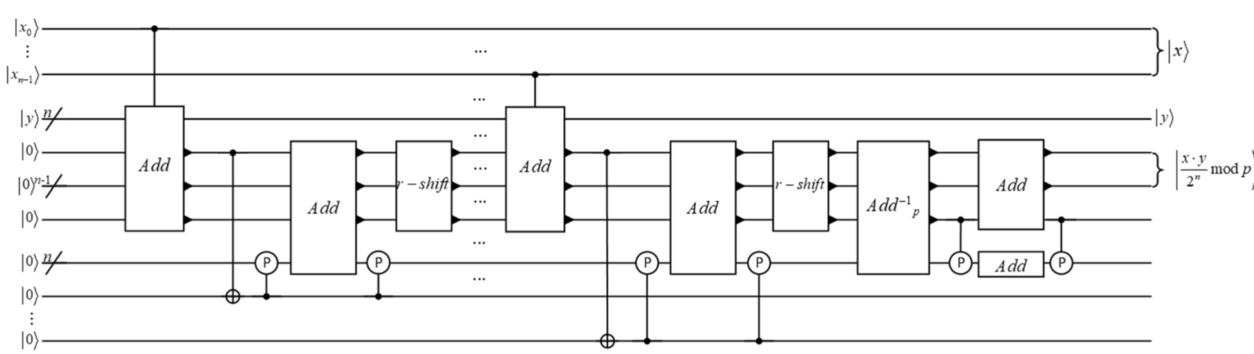

Fig. 19 The partial quantum circuit of Montgomery modular multiplication `M-Mul` : $|x\rangle|y\rangle|0\rangle \rightarrow |x\rangle|y\rangle|x \cdot y \cdot 2^{-1} \mod p\rangle$

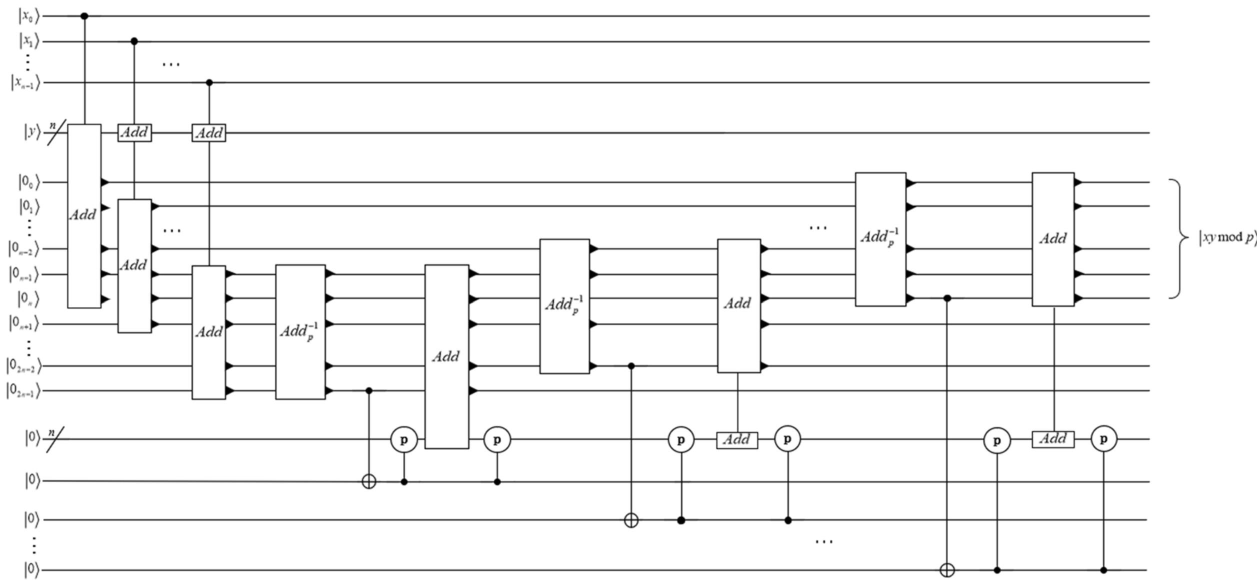

Fig. 20 The partial quantum circuit of direct modular multiplication D−Mul : $|x\rangle|y\rangle|0\rangle \rightarrow |x\rangle|y\rangle|x \cdot y \mod p\rangle$

According to the first two steps, we can obtain the following partial quantum circuit `D−Mul`. The circuit of step 3 to restore the auxiliary qubits is denoted by `D−Mul`$^{-1}$, i.e. the reverse of `D−Mul`, where the `Add` and `Add`$_p$ are `1−Add` and `2−Add` respectively in Fig. 20. Thus the whole quantum circuit of direct modular multiplication needs $114n^2 + 5n$ CNOTs. Similar to the Montgomery modular multiplication, to obtain the value $(x \cdot y \mod p)$ we still need to apply n CNOTs to encode the value into extra n-qubit auxiliary qubits before performing `D−Mul`$^{-1}$.

Modular inverse

The most common method of the modular inverse is the extended Euclidean algorithm (EEA). Proos and Zalka (2003) described the idea of using EEA to calculate modular inverse and it required $O(n)$ times of division in total and each step was performed $O(n^2)$ times. However, implementing the EEA in a quantum circuit is greatly complicated. Thus we consider using the Montgomery inversion algorithm described in detail in Roetteler et al. (2017). The algorithm repeats the Montgomery-Kaliski round function $2n$ times to get $x^{-1}R \mod p$. Subsequently, Häner et al. (2020) improved this algorithm. The improved circuit uses fewer CNOTs, but the modular inverse part is the same. In this paper, we choose the improved algorithm in Häner et al. (2020) as the round function and redesign a simpler circuit to calculate the modular inverse.

For inputs x, p, and n such that $p > x > 0$ and $2^{n-1} < x < 2^n$, the Montgomery-Kaliski algorithm consists of two steps. First, calculate $\gcd(x, p)$ and $x^{-1} \cdot 2^k \mod p$. Second, calculate $x^{-1} \cdot 2^n \mod p$. When the input quantum state is a superposition state, the number of iterations k in the first step is related to the integer x corresponding to a certain ground state. Considering all possible ground states in the superposition state, the first step requires $2n$ rounds of iteration. However, before each round, it is necessary to judge whether the iteration process in the corresponding ground state has ended by determining whether v is 0, so as to determine whether this round is really iterated. Due to $k > n$, all ground states of the input superposition state need to go through the first n rounds of iteration and only need to judge whether v is 0 before the iteration of the last n rounds. In the second step, the intermediate result $x^{-1} \cdot 2^k \mod p$ is shifted to the right by $k - n$ qubits. In the last n-round iteration of the first step, the results of the subsequent `ShiftMod` of the second step are stored in the auxiliary qubit and $x^{-1} \cdot 2^n \mod p$ is obtained.

Combining the round function circuit of Fig. 6b in Häner et al. (2020) with the above algorithm steps, the quantum circuit of the modular inverse `Inv` in Fig. 21 is obtained. The quantum circuit for restoring the auxiliary bits is `Inv`$^{-1}$, i.e. the reverse of `Inv`, and the complete quantum circuit is a combination of `Inv` and `Inv`$^{-1}$.

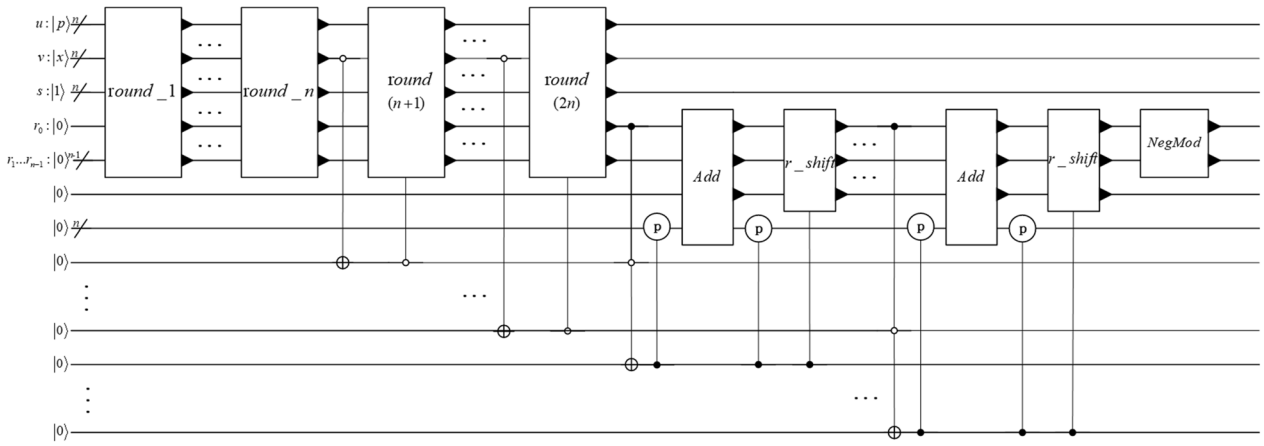

Fig. 21 The partial quantum circuit of modular inverse $\texttt{Inv} : |x \mod p\rangle|0\rangle \rightarrow |x\rangle|x^{-1} \cdot 2^n \mod p\rangle$

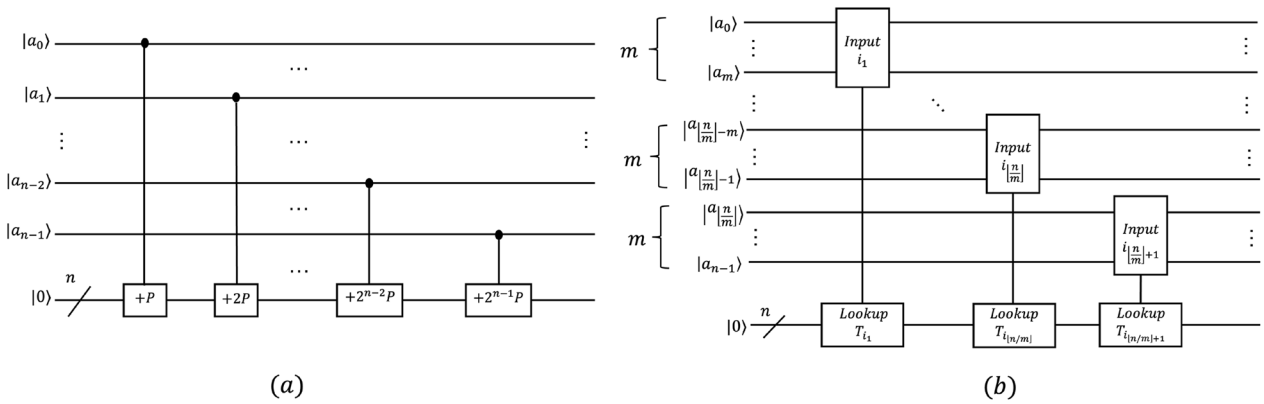

Fig. 22 **a** is a general method for calculating aP using the quantum circuit. **b** is the quantum circuit using windowed arithmetic to calculate aP, where m is the size of the window and T_i is 2^m pre-calculated values. Here is just a simple schematic diagram of the window technique to show its principle. We omit the process of point operation and recovery of auxiliary qubits. For more specific circuits see Fig. 33

According to Inv, we conclude that the whole quantum circuit of modular inverse needs $578n^2 + 283n - 13$ CNOTs.

Windowed arithmetic

In this section, we use the window form described in Gidney (2019)to design quantum circuits that attack ECDLP, reducing the CNOT-count N from $O(n^3)$ to $O(n^2) < N < O(n^3)$.

The general method to calculate aP by a quantum circuit is to express a in its binary expansion and control the operation of P by using each bit of a respectively, i.e.,

$$\begin{aligned} aP &= \left(2^{n-1}a_{n-1} + 2^{n-2}a_{n-2} + \cdots + 2a_1 + a_0\right)P \\ &= 2^{n-1}a_{n-1}P + 2^{n-2}a_{n-2}P + \cdots + 2a_1P + a_0P. \end{aligned}$$

The circuit is shown in Fig. 22a.

It is pointed out that m different a_i can be selected first and the 2^m cases, $a'P$ represented by m a_i, can be calculated and stored in an n-qubit register, where $a' = \sum_{j=1}^{m} 2^{i_j} a_{i_j}$ (Häner et al. 2020). Then $a'P$ is used to perform the point addition operation on the group of elliptic curves. This method is called windowed arithmetic as shown in Fig. 22b, and m is the size of the window. The left circuit in Fig. 23 shows the situation of $m = 2$, where T_i represents each of the four cases of a'. Only the abscissa of point $P(x, y)$ is shown in the figure and $\frac{n}{2}$ CNOTs are required on average. Therefore, it is estimated that a total of 8 Toffoli and $4n$ CNOTs are needed for the calculation point $P(x, y)$, i.e., $(4n + 48)$ CNOTs in all.

For general m, 2^{m+1} m-controlled CNOTs (i.e., $2^{m+1}(2m - 3)$ Toffoli gates) and $2^m n$ CNOTs are required in the circuit, so a total of $(24m + n - 36) \cdot 2^m$ CNOTs

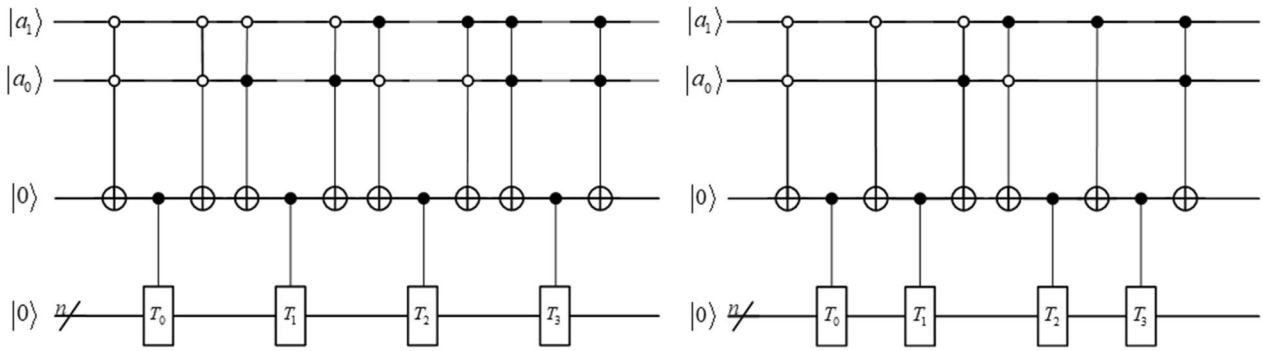

Fig. 23 Quantum circuits of windowed arithmetic at $m = 2$. Circuit of right one is the simplified of left one

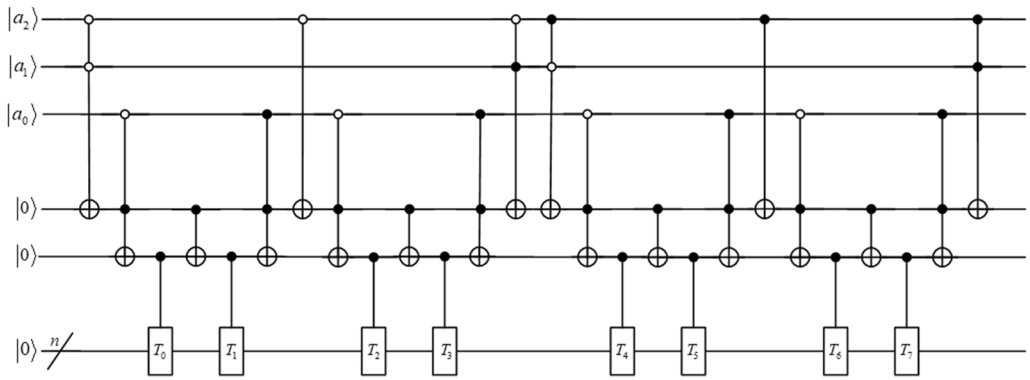

Fig. 24 The simplified circuit at $m = 3$

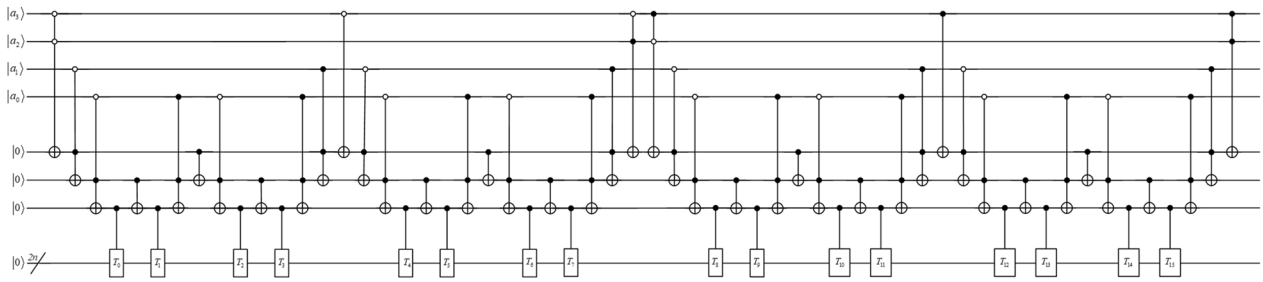

Fig. 25 The simplified circuit at $m = 4$

are required. However, we improve the circuit above to the right one in Fig. 23. Specifically, taking $m = 2$ as an example, we combine the second and the third Toffoli gates in the original method into one CNOT. At the same time, the sixth and the seventh Toffoli gates are merged into one CNOT. Thus we just need $(4n + 26)$ CNOTs when $m = 2$.

Figures 24 and 25 give the improved circuits of situations for $m = 3$ and $m = 4$, respectively, in a similar way to the method of combining Toffoli gates at $m = 2$. According to the recursive formula, $(2^{m+1} - 4)$ Toffoli and $[(n + 1) \cdot 2^m - 2]$ are required for m with $m \geq 3$. Thus the CNOT-count is reduced to $[(n + 13) \cdot 2^m - 26]$.

Quantum circuits of point addition on elliptic curve groups

Above, we describe the construction of basic arithmetic operations used in point addition on the elliptic curve groups. In this section, we design a new algorithm to calculate point addition reversibly out-of-place (storing the results in a new register), which reduces the CNOT-counts of modular inverse and

modular multiplication compared to the in-place method (replacing the input value by the sum) given by Roetteler et al. (2017), while using $O(n^2)$ qubits. Based on the new approach for point addition, this section gives the schematic circuit of the overall extended Shor's algorithm for ECDLP and then applies windowed arithmetic (Gidney 2019) to obtain the windowed scalar multiplication of the given point on elliptic curves.

Controlled point addition

The algorithm of controlled point addition on an elliptic curve operates on a quantum register holding the point $P_1 = (x_1, y_1) \neq \emptyset$, a control qubit *ctrl*, and ten auxiliary qubits c_i. The second point $P_2 = (x_2, y_2) \neq \emptyset, P_2 \neq \pm P_1$ is assumed to be a precalculated classical constant. If $ctrl = 1$, the algorithm correctly calculates $c_9 \leftarrow x_1 + x_2, c_{10} \leftarrow y_1 + y_2$; if $ctrl = 0, c_9 \leftarrow x_1, c_{10} \leftarrow y_1$.

Tables 1 and 2 describe the process of calculating $P_1 + P_2$ and restoring auxiliary bits, respectively. Figures 26 and 27 show quantum circuits corresponding to Tables 1 and 2. The quantum registers all consist of n logical qubits, whereas $|ctrl\rangle$ is a single logical qubit. Thus the CNOT-count is $896n^2 + 1064n + 14$.

After each calculation of $P_1 + P_2$, the result will be used as the next input P_1 for a new calculation and then will be restored as an auxiliary qubit. However, the result of the last calculation should be kept in the auxiliary register without any need to be restored. Thus, the circuit of the last calculation is modified as shown in Fig. 28. And the CNOT-count is $886n^2 + 783.5n - 18.5$.

Table 2 The steps to restore the ancillary bits. Symbols $|\cdot\rangle_1$ and $|\cdot\rangle_0$ respectively represent the state when the control bit is 1 and 0. The states in the table represent the change of the quantum registers corresponding to each step and the unwritten states are the same as the states in the previous step

	Process	The change in value
13.2	ctrl-CNOTc_9, c_{10}, c_1, c_2, ctrl	$c_9 \leftarrow [x_3]_1, [0]_0; c_{10} \leftarrow [y_3]_1, [0]_0$
12.2	ModAdd c_{10}, c_4	$c_{10} \leftarrow [\lambda(x_1 - x_3)]_1, [0]_0$
11.2	D−Mul^{-1} c_{10}, c_3, c_7	$c_{10} \leftarrow 0$
10.2	ModAdd c_3, c_4	$c_3 \leftarrow [x_1]_1, [0]_0$
9.2	ctrl − ModAdd(·) c_9, x_2, ctrl	$c_9 \leftarrow [\lambda^2 - x_1]_1, [0]_0$
8.2	ModAdd c_9, c_3	$c_9 \leftarrow [\lambda^2]_1, [0]_0$
7.2	ctrl-CNOTc_9, c_8, ctrl	$c_9 \leftarrow 0$
6.2	D−Mul^{-1} c_8, c_6, c_7	$c_7 \leftarrow 0; c_8 \leftarrow 0$
5.2	M−Mul^{-1} c_6, y_1, c_5	$c_6 \leftarrow 0$
4.2	Inv^{-1} c_5, x_1	$c_5 \leftarrow 0$
3.2	ModAdd(·) x_1, y_1, x_2, y_2	$x_1 \leftarrow x_1; y_1 \leftarrow y_1$
2.2	ctrl-CNOTc_3, c_4, x_1, y_1, ctrl	$c_3 \leftarrow 0; c_4 \leftarrow 0$
1.2	CNOTc_1, c_2, x_1, y_1	$c_1 \leftarrow 0; c_2 \leftarrow 0$

Therefore, the schematic quantum circuit yin Fig. 29 of the overall extended Shor's algorithm for ECDLP can be obtained by combining Figs. 26, 27 and 28.

Windowed point addition

The algorithm of windowed point addition on the elliptic curve operates on a quantum register holding the point $P_1 = (x_1, y_1) \neq \emptyset$, $P_2(x_2, y_2) \neq \emptyset, P_2 \neq \pm P_1$, and eight auxiliary qubits. In this form, the second point P_2 is stored in the quantum register as a quantum state and cannot be precomputed as a classical constant.

Table 1 The steps from (x_1, y_1) to (x_3, y_3) by point addition. Symbols $|\cdot\rangle_1$ and $|\cdot\rangle_0$ respectively represent the state when the control bit is 1 and 0. The states in the table represent the change of the quantum registers corresponding to each step and the unwritten states are the same as the states in the previous step

	Process	The change in value
1.1	CNOTc_1, c_2, x_1, y_1	$c_1 \leftarrow x_1; c_2 \leftarrow y_1$
2.1	ctrl-CNOTc_3, c_4, x_1, y_1, ctrl	$c_3 \leftarrow [x_1]_1, [0]_0; c_4 \leftarrow [y_1]_1, [0]_0$
3.1	ModAdd^{-1}(·) x_1, y_1, x_2, y_2	$x_1 \leftarrow x_1 - x_2; y_1 \leftarrow y_1 - y_2$
4.1	Inv c_5, x_1	$c_5 \leftarrow \frac{1}{x_1 - x_2}$
5.1	M−Mul c_6, y_1, c_5	$c_6 \leftarrow \lambda$
6.1	D−Mul c_8, c_6, c_7	$c_7 \leftarrow \lambda; c_8 \leftarrow \lambda^2$
7.1	ctrl-CNOTc_9, c_8, ctrl	$c_9 \leftarrow [\lambda^2]_1, [0]_0$
8.1	ModAdd^{-1} c_9, c_3	$c_9 \leftarrow [\lambda^2 - x_1]_1, [0]_0$
9.1	ctrl − ModAdd^{-1}(·) c_9, x_2, ctrl	$c_9 \leftarrow [\lambda^2 - x_1 - x_2 = x_3]_1, [0]_0$
10.1	ModAdd^{-1} c_3, c_4	$c_3 \leftarrow [x_1 - x_3]_1, [0]_0$
11.1	D−Mul c_{10}, c_3, c_7	$c_{10} \leftarrow [\lambda(x_1 - x_3)]_1, [0]_0$
12.1	ModAdd^{-1} c_{10}, c_4	$c_{10} \leftarrow [\lambda(x_1 - x_3) - y_1 = y_3]_1, [0]_0$
13.1	ctrl-CNOTc_9, c_{10}, c_1, c_2, ctrl	$c_9 \leftarrow [x_3]_1, [x_1]_0; c_{10} \leftarrow [y_3]_1, [y_1]_0$

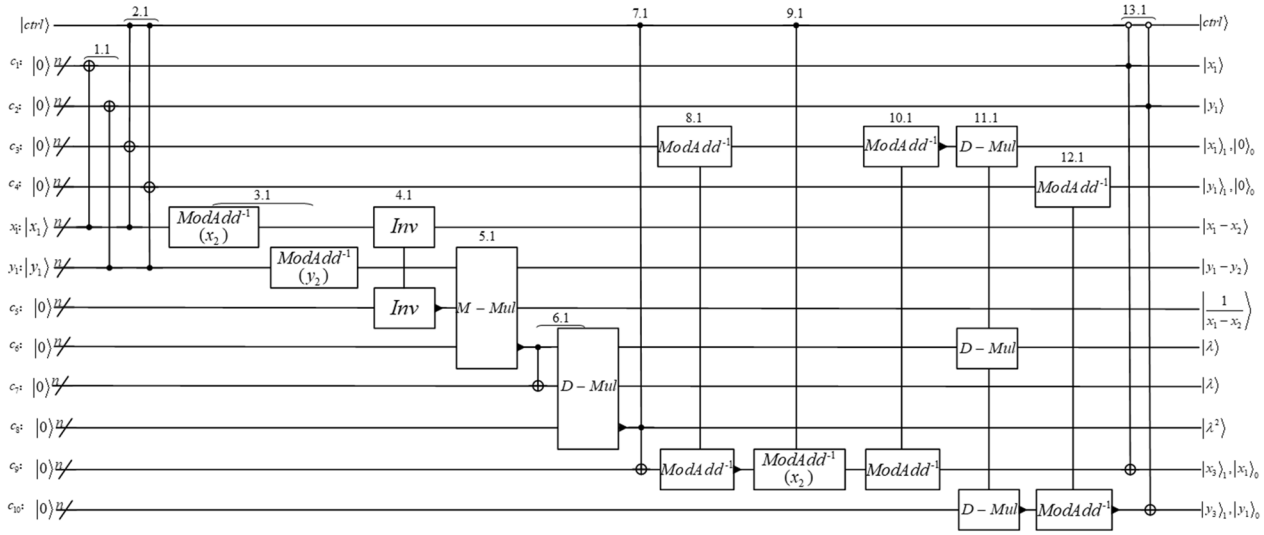

Fig. 26 The circuit for calculating $(P_1 + P_2) \mod p$ in controlled point addition

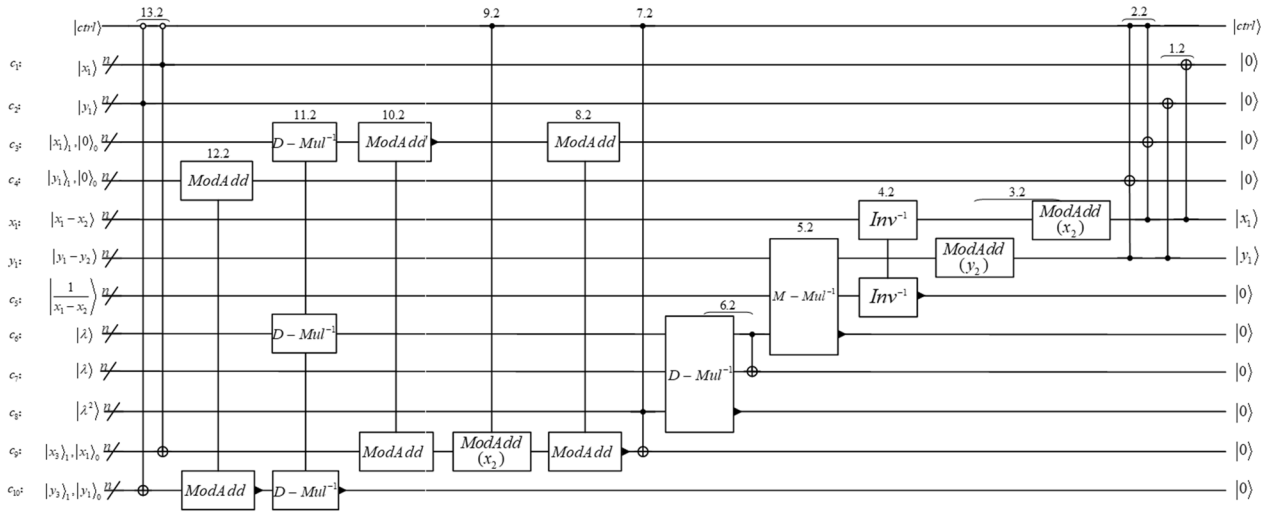

Fig. 27 The circuit for restoring the auxiliary bits in controlled point addition

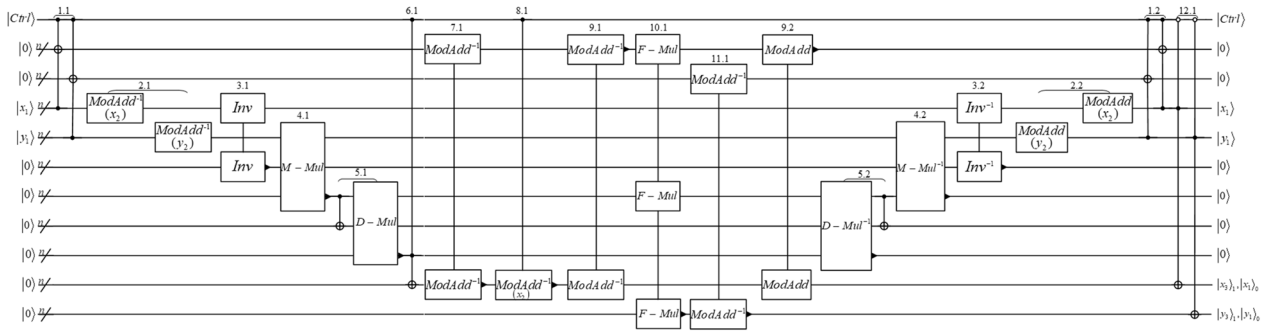

Fig. 28 The circuit of the $PointAdd_{last}$ block in Fig. 29

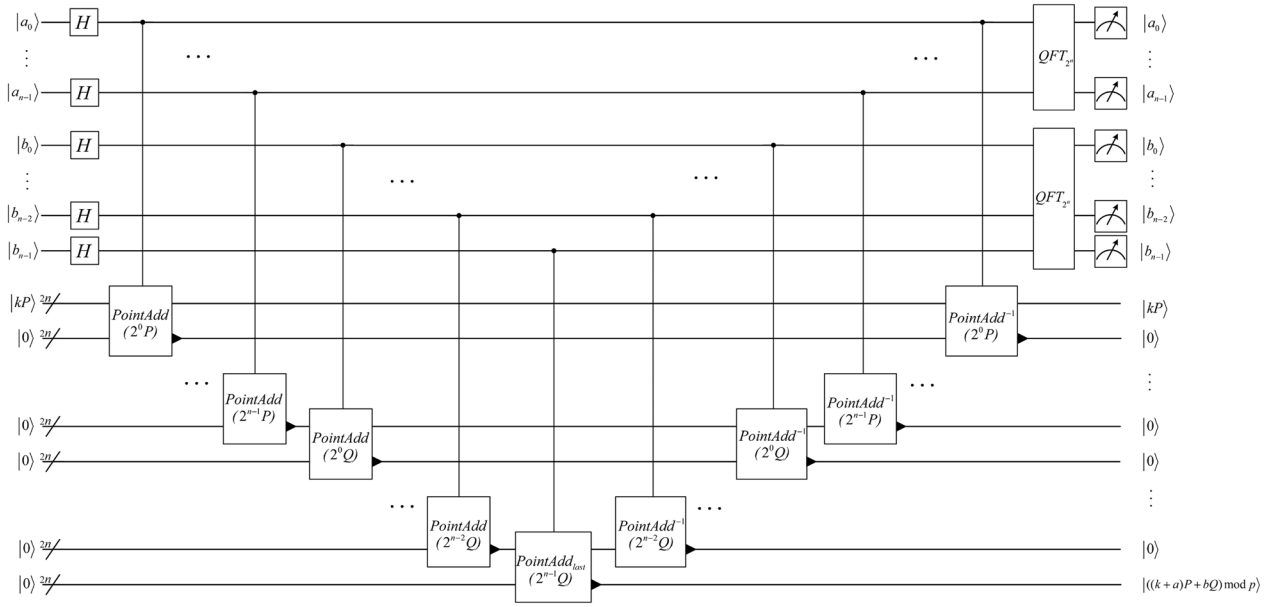

Fig. 29 Schematic quantum circuit of overall extended Shor's algorithm for ECDLP

Table 3 The steps from (x_1, y_1) to (x_3, y_3) using windowed arithmetic by point addition. The states in the table represent the change of the quantum registers corresponding to each step and the unwritten states are the same as the states in the previous step

	Process	The change in value
1.1	CNOTw_1, w_2, x_1, y_1	$w_1 \leftarrow x_1, w_2 \leftarrow y_1$
2.1	ModAdd^{-1} x_1, y_1, x_2, y_2	$x_1 \leftarrow x_1 - x_2, y_1 \leftarrow y_1 - y_2$
3.1	Inv w_3, x_1	$w_3 \leftarrow \frac{1}{x_1 - x_2}$
4.1	M-Mul w_4, y_1, w_3	$w_4 \leftarrow \lambda$
5.1	D-Mul w_6, w_4, w_5	$w_6 \leftarrow \lambda^2$
6.1	CNOTw_7, w_6	$w_7 \leftarrow \lambda^2$
7.1	ModAdd$^{-1}w_7, w_1$	$w_7 \leftarrow \lambda^2 - x_1$
8.1	ModAdd^{-1} w_7, x_2	$w_7 \leftarrow \lambda^2 - x_1 - x_2 = x_3$
9.1	ModAdd^{-1} w_1, w_7	$w_1 \leftarrow x_1 - x_3$
10.1	D-Mul w_8, w_1, w_4	$w_8 \leftarrow \lambda(x_1 - x_3)$
11.1	ModAdd^{-1} w_8, w_2	$w_8 \leftarrow \lambda(x_1 - x_3) - y_1 = y_3$

Table 4 The steps to restore the auxiliary bits. The states in the table represent the change of the quantum registers corresponding to each step and the unwritten states are the same as the states in the previous step

	Process	The change in value
11.2	ModAdd w_8, w_2	$w_8 \leftarrow \lambda(x_1 - x_3)$
10.2	D-Mul^{-1} w_8, w_1, w_4	$w_8 \leftarrow 0$
9.2	ModAdd w_1, w_7	$w_1 \leftarrow x_1$
8.2	ModAdd w_7, x_2	$w_7 \leftarrow \lambda^2 - x_1$
7.2	ModAdd w_7, w_1	$w_7 \leftarrow \lambda^2$
6.2	CNOTw_7, w_6	$w_7 \leftarrow 0$
5.2	D-Mul^{-1} w_6, w_4, w_5	$w_6 \leftarrow 0$
4.2	M-Mul^{-1} w_4, y_1, w_3	$w_4 \leftarrow 0$
3.2	Inv^{-1} w_3, x_1	$w_3 \leftarrow 0$
2.2	ModAdd x_1, y_1, x_2, y_2	$x_1 \leftarrow x_1, y_1 \leftarrow y_1$
1.2	CNOTw_1, w_2, x_1, y_1	$w_1 \leftarrow 0, w_2 \leftarrow 0$

Tables 3 and 4 describe the process of calculating $P_1 + P_2$ by windowed arithmetic and restoring auxiliary bits, respectively.

Figures 30 and 31 show quantum circuits corresponding to Tables 3 and 4. The quantum registers all consist of n logical qubits. Thus the CNOT-count is $896n^2 + 1108n + 36$.

After each calculation of $P_1 + P_2$, the result will be used as the next input P_1 for a new calculation and then will be restored as an auxiliary qubit. However, the result of the last calculation should be kept in the auxiliary register without any need to be restored and the coefficients of P and Q are different in the extended Shor's quantum algorithm. Therefore, the window arithmetic cannot be used in the last calculation, and the circuit is modified as shown in Fig. 32 with $886n^2 + 833.5n + 9.5$ CNOTs.

Figure 33 is the schematic quantum circuit to calculate ECDLP by the extended Shor's algorithm using

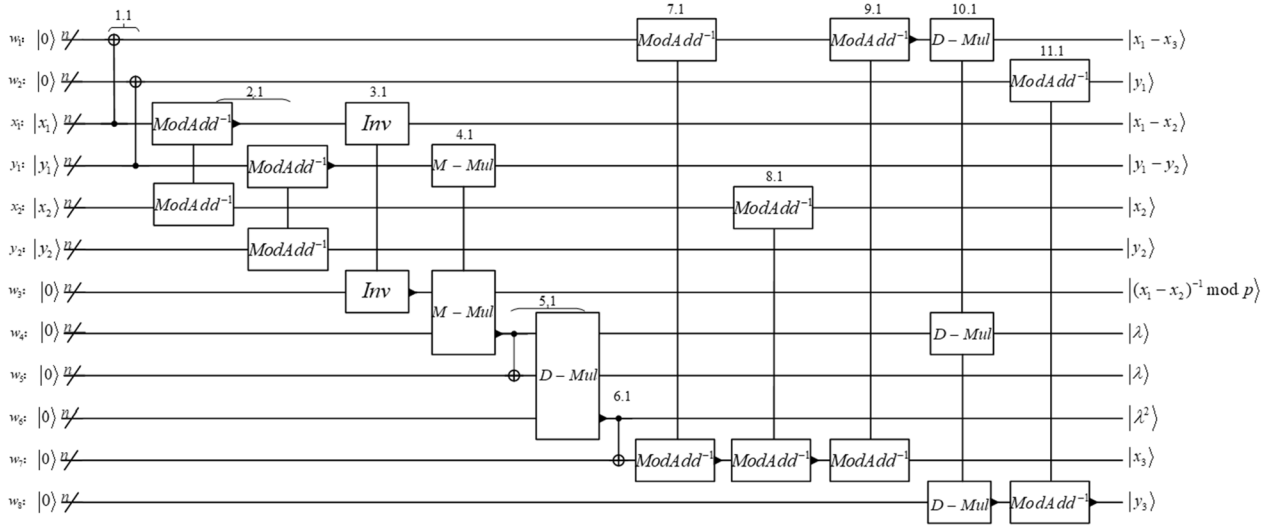

Fig. 30 Windowed point addition

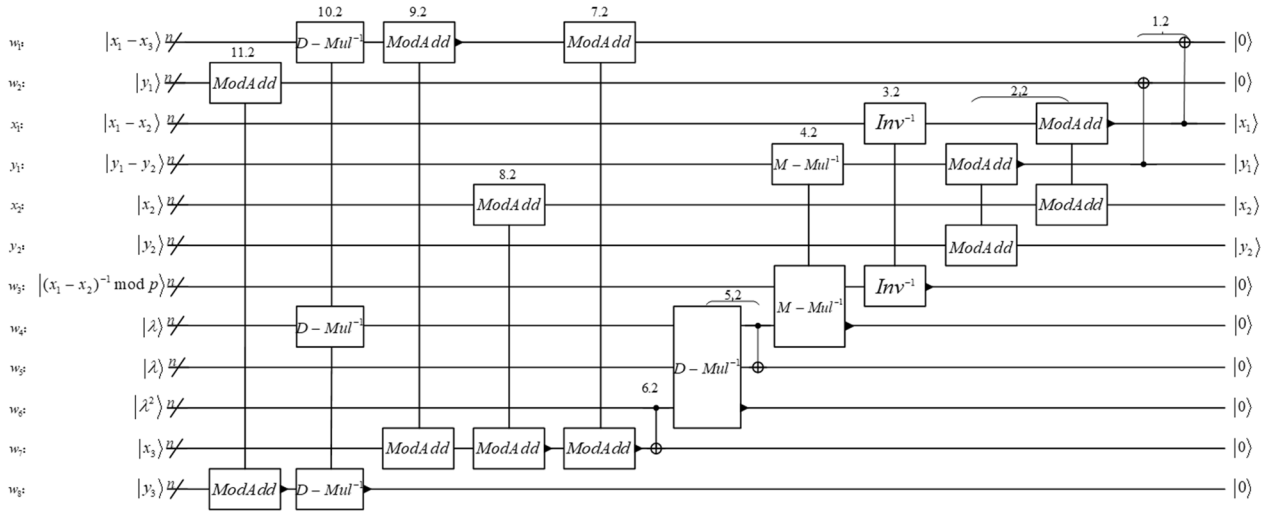

Fig. 31 The inverse operation of windowed point addition to restore the auxiliary bits

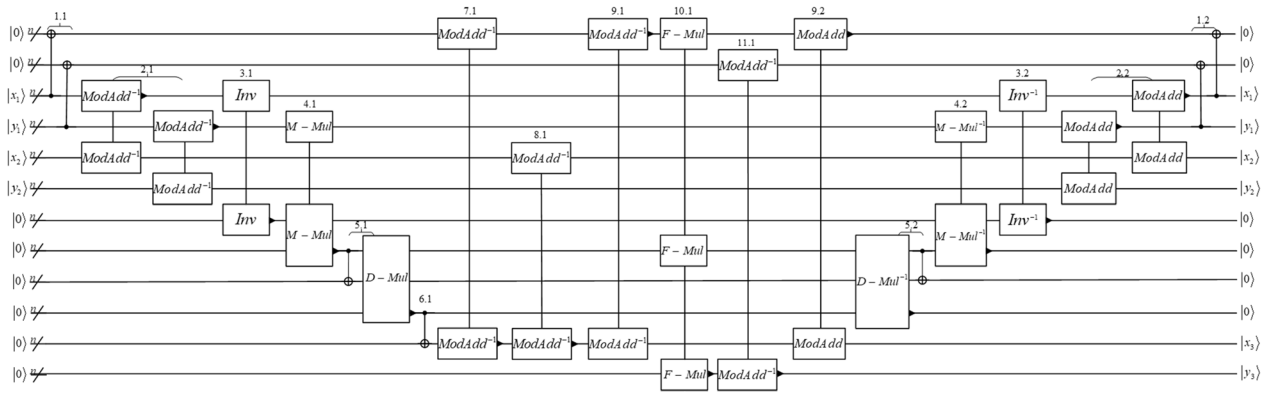

Fig. 32 The last full round of windowed point addition

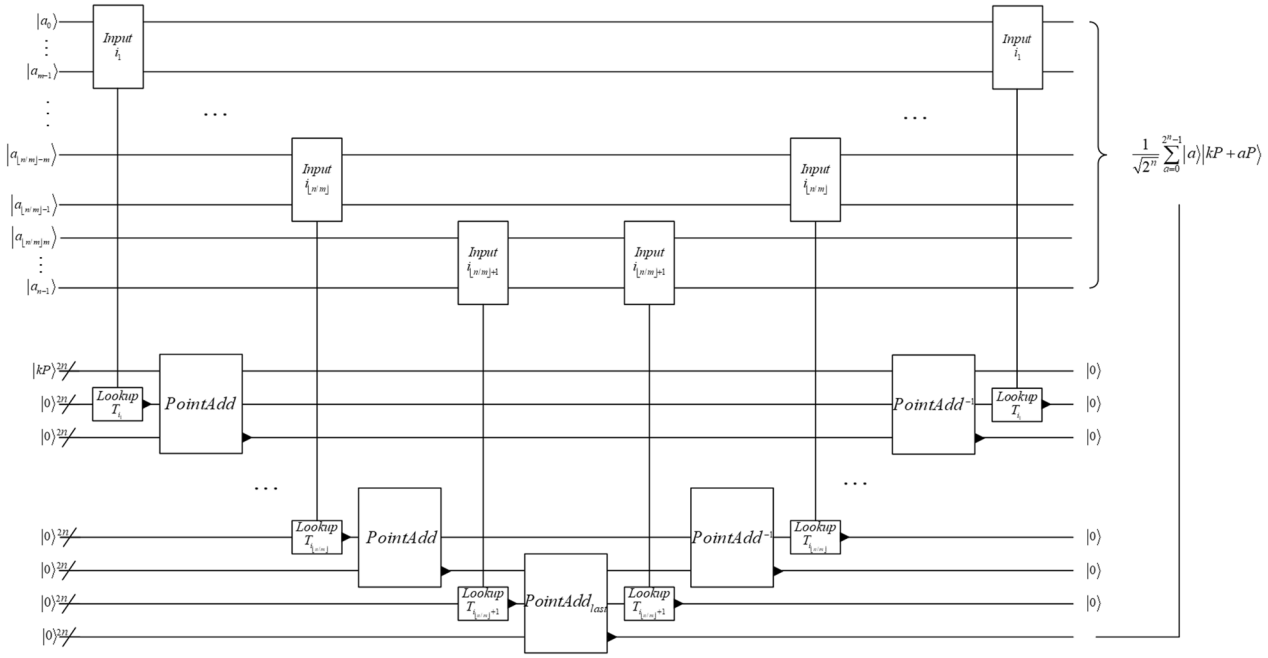

Fig. 33 Schematic quantum circuit of overall extended Shor's algorithm for ECDLP using windowed arithmetic

windowed arithmetic, where the `Lookup` is a situation where several controlled operations can be merged into a single operation acting on a value produced by a small QROM lookup (Gidney 2019) and the point addition is the circuit introduced in Figs. 30, 31 and 32.

According to the process of calculating point addition, the CNOT-count of the first $2n-1$ point addition is $896n^2+1064n+14$ (including the circuits for recovering auxiliary qubits) and the $2n$-th has $886n^3+783.5n-18.5$ CNOTs. Therefore, the CNOT-count to calculate the point addition of ECDLP using controlled point addition is $1792n^3+2118n^2-252.5n-32.5$. When using the windowed version, the modular subtraction of a constant is changed to $\texttt{ModAdd}^{-1}$ and the CNOT-count increases from $43n-2.5$ to $61n+6$. At the same time, the CNOT-count of the controlled circuit increases from $46n+11$ to $71n+17$. Thus the CNOT-count of the first $n-1$ point addition is $896n^2+1108n+36$ and the n-th has $886n^3+833.5n+9.5$ CNOTs in calculating $(a+k)P \bmod p$. Hence the CNOT-count to calculate the point addition of ECDLP using the windowed version with window size m is $N(n,m)=2\lceil\frac{n}{m}\rceil[(n+13)\cdot 2^{m+1}+896n^2+1108n+4]$ $-20n^2-549n-53$.

Now we analyze the whole circuit of the extended Shor's algorithm to obtain a specific CNOT-count. In order to minimize the CNOT-count, we find that the growth rate of the CNOT-count is polynomial with n only when $m=O(\log n)$, while the other cases are exponential. So we further fit $N(n, m)$ to find a suitable value of m to make the CNOT-count as low as possible. To be specific, we calculate $\frac{\partial N(n,m)}{\partial m}$, and for each $n_i \in (128, 521)$, we use Matlab to approximate the zero m_i of $\frac{\partial N(n_i,m)}{\partial m}$ to obtain a pair (n_i, m_i). Because m should be an integer, we round each m_i up and down to get m_i' and m_i'', respectively. Then letting $N_{i_{\min}}=\min(N(n_i,m_i'),N(n_i,m_i''))$ for each i and fitting N with respect to n based on all the pairs $(n_i, N_{i_{\min}})$, we obtain $N=1237n^3/\log n$. Plus the $2n^2+n$ CNOTs used for two QFT_n, the total CNOT-count of the extended Shor's algorithm for ECDLP is $N=1237n^3/\log n+2n^2+n$. The lower limit of time for executing a CNOT gate on an ion trap quantum computer is about $2.85\times 10^{-4}s$ (Yang and Zhou 2013). Combined with the CNOT-count to run the extend Shor's algorithm, the time to break 512-bit ECDLP is at least 51 years after three levels of coding.

Discussion and conclusion

Although there have been many attempts to improve the qubit number or the circuit depth of the extended Shor's algorithm for ECDLP, their focus has not been on optimizing the CNOT-count, which greatly affects the time to run the algorithm on an ion trap quantum computer. In this paper, we improve the quantum circuits of basic arithmetic operations, including modular subtraction, three different modular multiplication, modular

Table 5 The number of Toffoli gates and CNOT gates for Clifford+Toffoli implementations

The basic arithmetic operations	#Toffoli	#CNOTs for Clifford+Toffoli
1−Add$_y$ (unknown state y)	$2n$	$4n+1$
ctrl-1−Add$_y$ (unknown state y)	$4n+1$	$2n$
2−Add$_y$ (known constant y)	$2n-1$	$2n+0.5$
ctrl-2−Add$_y$	$2n$	$5n+1$
1−Comp	$2n$	1
2−Comp$_y$ (known constant y)	$2n$	1
ctrl-2−Comp$_y$ (known constant y)	$2n+1$	1
2−Comp$_y$ (unknown state y)	$2n$	$4n+1$
ctrl-2−Comp$_y$ (unknown state y)	$2n+1$	$4n+1$
ModSub$_y$ or ModAdd$_y$ (known constant y)	$6n-1$	$7n+3.5$
ctrl-ModSub$_y$ (known constant y)	$6n+1$	$10n+5$
ModAdd$_y$ (unknown state y)	$8n$	$13n+6$
ctrl-ModAdd$_y$ (unknown state y)	$10n+2$	$11n+5$
Neg	$2n-1$	$2n-0.5$
ctrl-Neg	$2n$	$6n+1$
1−Shift	-	$2n$
ctrl-1−Shift	$2n$	-
2−Shift	-	$3n$
ctrl-2−Shift	n	$2n$
ShiftMod	$4n$	$7n+3$
M−Mul (half)	$6n^2+5n-1$	$9n^2+9n+0.5$
D−Mul (half)	$8n^2$	$9n^2+1.5$

Table 6 The number of T gates and CNOT gates for Clifford+T implementations

The basic arithmetic operations	#T gates	#CNOTs for Clifford+T
1−Add$_y$ (unknown state y)	$14n$	$16n+1$
ctrl-1−Add$_y$ (unknown state y)	$28n+7$	$26n+6$
2−Add$_y$ (known constant y)	$14n-7$	$14n-5.5$
ctrl-2−Add$_y$	$14n$	$17n+1$
1−Comp	$14n$	$12n+1$
2−Comp$_y$ (known constant y)	$14n$	$12n+1$
ctrl-2−Comp$_y$ (known constant y)	$14n+7$	$12n+7$
2−Comp$_y$ (unknown state y)	$14n$	$16n+1$
ctrl-2−Comp$_y$ (unknown state y)	$14n+7$	$16n+7$
ModSub$_y$ or ModAdd$_y$ (known constant y)	$42n-7$	$43n-2.5$
ctrl-ModSub$_y$ (known constant y)	$42n+7$	$46n+11$
ModAdd$_y$ (unknown state y)	$56n$	$61n+6$
ctrl-ModAdd$_y$ (unknown state y)	$70n+14$	$71n+17$
Neg	$14n-7$	$14n-5.5$
ctrl-Neg	$14n$	$18n+1$
ctrl-1−Shift	$14n$	$12n$
ctrl-2−Shift	$7n$	$8n$
ShiftMod	$28n$	$31n+15$
M−Mul (half)	$42n^2+35n-7$	$45n^2+39n-4.5$
D−Mul (half)	$56n^2$	$57n^2+2.5n$

inverse, and windowed arithmetic. Table 5 summarizes the CNOT-counts of basic arithmetic operations using the Clifford+Toffoli gate set, while Table 6 summarizes the CNOT-counts of basic arithmetic operations using the Clifford+T gate set. These improvements lead to a reduced CNOT-count of the quantum circuit of the extended Shor's algorithm. We further reduce the CNOT-count by choosing a suitable window size m with the help of numerical fitting, lowering the CNOT-count from $O(n^3)$ to $O(n^3/\log n)$. The time required by the extended Shor's algorithm to attack 512-bit ECDLP is estimated to be 51 years, which means it is hard to attack ECDLP using an ion trap quantum computer in a reasonable time. However, this estimated time does not take into account the fault tolerance of the circuit, which we will study in the future.

According to the results of the CNOT-count, we can consider the lower bound of the CNOT-count required by the extended Shor's algorithm. If we assume that the time required to run extended Shor's algorithm is T, the time required to execute a CNOT is t, and the lower bound of the number of CNOTs is N, which is a function of the number of qubits n. Then, the lower bound of the running time of the extended Shor's algorithm can be expressed as $T = N(n)t$. The modular inverse can be constructed using basic arithmetic operations, such as modular addition. Therefore, the CNOT-count of modular inverse must be greater than that required for modular addition. Because the quantum circuit of modular addition is a modular operation, the CNOT-count of modular addition must be larger than that of the addition circuit. For two n qubits x, y, we have that $c_{i+1} = x_i + (x_i + y_i)(x_i + c_i), s_i = x_i + y_i + c_i$, where x_i and y_i are the i-th bits of the binary representation of x, y, c_{i+1} is the i-th carry. Therefore, each qubit addition requires at least one Toffoli gate and three CNOTs. Thus, the addition of n qubits requires at least $9n$ CNOTs. Here we just give a lower bound of the circuit of an addition operation. Although the whole circuit of the extended Shor's algorithm consists of many addition operations, we have not obtained a tighter lower bound to run this algorithm, which we plan to derive in our future work.

Appendix

We first prove in detail that whether the input state of the third quantum register is $|0\rangle$ or $|1\rangle$ has no effect on the measurement probability. Then, we give the specific derivation process of the number of CNOT gates in the n-controlled-NOT.

The value of the input state has no effect on the result

Proof (1) When the input state is $|0\rangle$, we have

$$\begin{aligned}|0\rangle|0\rangle|0\rangle \rightarrow & \frac{1}{p-1}\sum_{a_1=0}^{p-2}\sum_{b_1=0}^{p-2}|a_1\rangle|b_1\rangle|0\rangle \\ \rightarrow & \frac{1}{p-1}\sum_{a_1=0}^{p-2}\sum_{b_1=0}^{p-2}|a_1\rangle|b_1\rangle|(a_1P+b_1Q) \mod p\rangle \\ \rightarrow & \frac{1}{(p-1)q}\sum_{a_1,b_1=0}^{p-2}\sum_{c_1,d_1=0}^{q-1}\exp\left[\frac{2\pi i}{q}(a_1c_1+b_1d_1)\right] \\ & |c_1\rangle|d_1\rangle|(a_1P+b_1Q) \mod p\rangle \\ = & \frac{1}{(p-1)q}\sum_{a_1,b_1=0}^{p-2}\sum_{c_1,d_1=0}^{q-1}\exp\left[\frac{2\pi i}{q}(a_1c_1+b_1d_1)\right] \\ & |c_1\rangle|d_1\rangle|(a_1+b_1m)P \mod p\rangle,\end{aligned}$$

where $a_1+b_1m=l_1 \mod (p-1)$, so $a_1=l_1-b_1m-(p-1)\lfloor\frac{l_1-b_1m}{p-1}\rfloor$, thus the probability of getting a $|c\rangle|d\rangle|y\rangle$ is

$$\left|\frac{1}{(p-1)1}\sum_{a_1,b_1=0}^{p-2}\exp\left[\frac{2\pi i}{q}(a_1c_1+b_1d_1)\right]\right|^2 = \left|\frac{1}{(p-1)q}\sum_{b_1=0}^{p-2}\exp\left[\frac{2\pi i}{q}\left(l_1c_1-b_1c_1m+b_1d_1-(p-1)c_1\left\lfloor\frac{l_1-b_1m}{p-1}\right\rfloor\right)\right]\right|^2$$

so when taking all of $|y\rangle$, the probability of $|c\rangle|d\rangle$ is

$$\sum_{l_1=0}^{p-2}\left|\frac{1}{(p-1)q}\sum_{b_1=0}^{p-2}\exp\left[\frac{2\pi i}{q}(l_1c_1-b_1c_1m+b_1d_1-(p-1)c_1\left\lfloor\frac{l_1-b_1m}{p-1}\right\rfloor\right)\right]\right|^2 \tag{12}$$

(2) When the input state is $|kP\rangle$, we have

$$\begin{aligned}|0\rangle|0\rangle|kP\rangle \rightarrow & \frac{1}{p-1}\sum_{a_2=0}^{p-2}\sum_{b_2=0}^{p-2}|a_2\rangle|b_2\rangle|kP\rangle \\ \rightarrow & \frac{1}{p-1}\sum_{a_2=0}^{p-2}\sum_{b_2=0}^{p-2}|a_2\rangle|b_2\rangle|((a_2+k)P+b_2Q) \mod p\rangle \\ \rightarrow & \frac{1}{(p-1)q}\sum_{a_2,b_2=0}^{p-2}\sum_{c_2,d_2=0}^{q-1}\exp\left[\frac{2\pi i}{q}(a_2c_2+b_2d_2)\right] \\ & |c_2\rangle|d_2\rangle|((a_2+k)P+b_2Q) \mod p\rangle \\ = & \frac{1}{(p-1)q}\sum_{a_2,b_2=0}^{p-2}\sum_{c_2,d_2=0}^{q-1}\exp\left[\frac{2\pi i}{q}(a_2c_2+b_2d_2)\right] \\ & |c_2\rangle|d_2\rangle|((a_2+k)+b_2m)P \mod p\rangle,\end{aligned}$$

where $a_2+k+b_2m=l_2 \mod (p-1)$, so $a_2=l_2-k-b_2m-(p-1)\lfloor\frac{l_2-k-b_2m}{p-1}\rfloor$, thus the probability of getting a $|c\rangle|d\rangle|y\rangle$ is

$$\left|\frac{1}{(p-1)q}\sum_{a_2,b_2=0}^{p-2}\exp\left[\frac{2\pi i}{q}(a_2c_2+b_2d_2)\right]\right|^2 = \left|\frac{1}{(p-1)q}\sum_{b_2=0}^{p-2}\exp\left[\frac{2\pi i}{q}(l_2c_2-b_2c_2m-kc_2+b_2d_2-(p-1)c_2\left\lfloor\frac{l_2-k-b_2m}{p-1}\right\rfloor\right)\right]\right|^2$$

so when taking all of $|y\rangle$, the probability of $|c\rangle|d\rangle$ is

$$\sum_{l_2=0}^{p-2}\left|\frac{1}{(p-1)q}\sum_{b_2=0}^{p-2}\exp\left[\frac{2\pi i}{q}(l_2c_2-b_2c_2m-kc_2+b_2d_2-(p-1)c_2\left\lfloor\frac{l_2-k-b_2m}{p-1}\right\rfloor\right)\right]\right|^2 \tag{13}$$

(3) To prove that it doesn't matter whether the input is $|0\rangle$ or $|kP\rangle$, just prove that (5) = (6) There has

$$\begin{aligned}
&\sum_{l_1=0}^{p-2}\left|\frac{1}{(p-1)q}\sum_{b_1=0}^{p-2}\exp\left[\frac{2\pi i}{q}\left(l_1c_1-b_1c_1m+b_1d_1-(p-1)c_1\left\lfloor\frac{l_1-b_1m}{p-1}\right\rfloor\right)\right]\right|^2\\
&\quad=\sum_{l_1=0}^{p-2-k}\left|\frac{1}{(p-1)q}\right.\\
&\qquad\left.\sum_{b_1=0}^{p-2}\exp\left[\frac{2\pi i}{q}\left(l_1c_1-b_1c_1m+b_1d_1-(p-1)c_1\left\lfloor\frac{l_1-b_1m}{p-1}\right\rfloor\right)\right]\right|^2\\
&\qquad+\sum_{l_1=p-1-k}^{p-2}\left|\frac{1}{(p-1)q}\right.\\
&\qquad\left.\sum_{b_1=0}^{p-2}\exp\left[\frac{2\pi i}{q}\left(l_1c_1-b_1c_1m+b_1d_1-(p-1)c_1\left\lfloor\frac{l_1-b_1m}{p-1}\right\rfloor\right)\right]\right|^2\\
&\quad=①+②;\\
&\sum_{l_2=0}^{p-2}\left|\frac{1}{(p-1)q}\right.\\
&\qquad\left.\sum_{b_2=0}^{p-2}\exp\left[\frac{2\pi i}{q}\left(l_2c_2-b_2c_2m-kc_2+b_2d_2-(p-1)c_2\left\lfloor\frac{l_2-k-b_2m}{p-1}\right\rfloor\right)\right]\right|^2\\
&\quad=\sum_{l_2=0}^{k-1}\left|\frac{1}{(p-1)q}\right.\\
&\qquad\left.\sum_{b_2=0}^{p-2}\exp\left[\frac{2\pi i}{q}\left(l_2c_2-b_2c_2m-kc_2+b_2d_2-(p-1)c_2\left\lfloor\frac{l_2-k-b_2m}{p-1}\right\rfloor\right)\right]\right|^2\\
&\qquad+\sum_{l_2=k}^{p-2}\left|\frac{1}{(p-1)q}\right.\\
&\qquad\left.\sum_{b_2=0}^{p-2}\exp\left[\frac{2\pi i}{q}\left(l_2c_2-b_2c_2m-kc_2+b_2d_2-(p-1)c_2\left\lfloor\frac{l_2-k-b_2m}{p-1}\right\rfloor\right)\right]\right|^2\\
&\quad=③+④.
\end{aligned}$$

Because $c_1=c_2, d_1=d_2$, so we have ① = ④, ② = ③, then (5) = (6). □

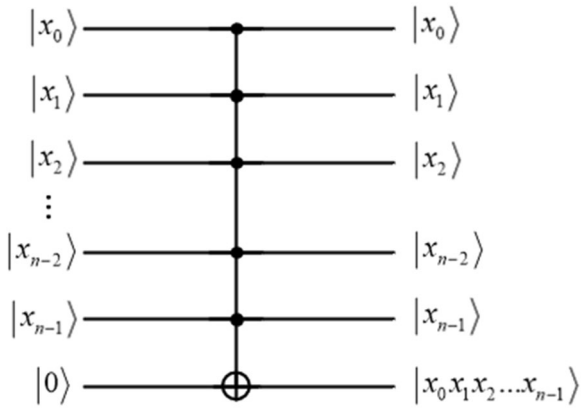

Fig. 34 n-controlled-NOT

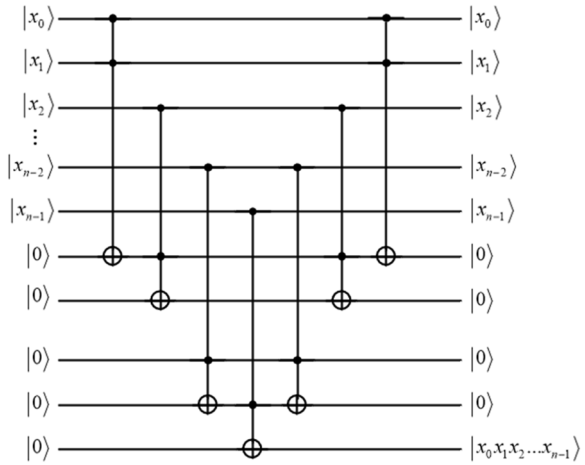

Fig. 35 The equivalent form of n-controlled-NOT

The number of CNOT gates in n-controlled-NOT

The Fig. 34 is the quantum circuit of n-controlled-NOT, which can be contrusted by $n-2$ auxiliary qubits and $2n-3$ Toffoli. The quantum circuit is shown in Fig. 35.

Acknowledgements
We thank all reviewers for their comments and suggestions.

Author contributions
This work is completed by Xia Liu and Huan Yang under the supervision of Prod. Li Yang. Xia Liu and Huan Yang discuss this problem in detail and consider its application in cryptanalysis in this paper. Li Yang put forward many meaningful ideas to help the work to be completed. Finally, the author(s) read and approved the final manuscript. Because the funding contains relevant author information, we also listed the fund here.

Funding
This work was supported by National Natural Science Foundation of China (Grant No. 61672517) and National Natural Science Foundation of China (Key Program, Grant No. 61732021).

Availability of data and materials
All data and materials are included in this paper.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 18 May 2023 Accepted: 21 July 2023
Published online: 06 December 2023

References

Banegas G, Bernstein DJ, van Hoof I, Lange T (2021) Concrete quantum cryptanalysis of binary elliptic curves. In: IACR transactions on cryptographic hardware and embedded systems 451–472
Boneh D, Lipton RJ (1995) Quantum cryptanalysis of hidden linear functions. In: Annual international cryptology conference. Springer, pp 424–437
Chiribella G, D'Ariano GM, Perinotti P (2008) Quantum circuit architecture. Phys Rev Lett 101(6):060401
Cuccaro SA, Draper TG, Kutin SA, Moulton DP (2004) A new quantum ripple-carry addition circuit. arXiv:quant-ph/0410184
Dong D, Chen C, Jiang M, Wang L-C (2013) Quantum control and quantum information technology. Hindawi
Eicher J, Opoku Y (1997) Using the quantum computer to break elliptic curve cryptosystems
Gao W, Yang L, Zhang D, Liu X (2022) Quantum identity-based encryption from the learning with errors problem. Cryptography 6(1):9
Gidney C (2019) Windowed quantum arithmetic. arXiv:1905.07682
Häner T, Jaques S, Naehrig M, Roetteler M, Soeken M (2020) Improved quantum circuits for elliptic curve discrete logarithms. In: International conference on post-quantum cryptography. Springer, pp 425–444
Hasse H (1936) Zur theorie der abstrakten elliptischen funktionenkörper iii. die struktur des meromorphismenrings. die riemannsche vermutung
Kaliski BS (1995) The Montgomery inverse and its applications. IEEE Trans Comput 44(8):1064–1065
Kaye P, Zalka C (2004) Optimized quantum implementation of elliptic curve arithmetic over binary fields. arXiv:quant-ph/0407095
Kimura T, Shiba K, Chen C-C, Sogabe M, Sakamoto K, Sogabe T (2021) Variational quantum circuit-based reinforcement learning for POMDP and experimental implementation. Math Probl Eng 2021:1–11
Knight P, Murao M, Plenio M, Vedral V (1999) Ion trap quantum gates, decoherence and error correction
Koblitz N (1987) Elliptic curve cryptosystems. Math Comput 48(177):203–209
Liu X, Liu G, Huang J, Wang X (2022) Mitigating barren plateaus of variational quantum eigensolvers. arXiv:2205.13539
Liu X, Yang H, Yang L (2021) CNOT-count optimized quantum circuit of the Shor's algorithm. arXiv:2112.11358
Markov IL, Saeedi M (2012) Constant-optimized quantum circuits for modular multiplication and exponentiation. arXiv:1202.6614
Miller VS (1985) Use of elliptic curves in cryptography. In: Conference on the theory and application of cryptographic techniques. Springer, pp 417–426
Miyaji A (1992) Elliptic curves over f p suitable for cryptosystems. In: International workshop on the theory and application of cryptographic techniques. Springer, pp 477–491
Nash B, Gheorghiu V, Mosca M (2020) Quantum circuit optimizations for NISQ architectures. Quantum Sci Technol 5(2):025010
Nielsen MA, Chuang I (2002) Quantum computation and quantum information. American Association of Physics Teachers
Proos J, Zalka C (2003) Shor's discrete logarithm quantum algorithm for elliptic curves. arXiv:quant-ph/0301141
Roetteler M, Naehrig M, Svore K.M, Lauter K (2017) Quantum resource estimates for computing elliptic curve discrete logarithms. In: International conference on the theory and application of cryptology and information security. Springer, pp 241–270

Shor PW (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th annual symposium on foundations of computer science. IEEE, pp 124–134
Shor PW (1999) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev 41(2):303–332
Vedral V, Barenco A, Ekert A (1996) Quantum networks for elementary arithmetic operations. Phys Rev A 54(1):147
Yang L, Zhou R-R (2013) On the post-quantum security of encrypted key exchange protocols. arXiv:1305.5640
Yao AC-C (1993) Quantum circuit complexity. In: Proceedings of 1993 IEEE 34th annual foundations of computer science. IEEE, pp 352–361

Publisher's Note