Quantum Information Processing (2022) 21:181
https://doi.org/10.1007/s11128-022-03518-5

New quantum circuit implementations of SM4 and SM3

Jian Zou[1,2] · Liji Li[1,2] · Zihao Wei[3] · Yiyuan Luo[4] · Qian Liu[1,2] · Wenling Wu[5]

Received: 30 September 2021 / Accepted: 11 April 2022 / Published online: 13 May 2022

Abstract
In this paper, we propose some new quantum circuit implementations of SM4 block cipher and SM3 hash function, which are based on the following ideas. Firstly, we propose an improved classical circuit of SM4's S-box, which requires less AND gates than the previous works. Our improved classical circuit of SM4's S-box can be used for constructing a new quantum circuit of SM4's S-box. Secondly, we propose a new implementation of the Feistel-like structure of SM4 so as to reduce the number of qubits and T-depth simultaneously. Thirdly, we reduce the number of qubits in our quantum circuit of SM3 by making use of linear message expansion algorithm of SM3. Fourthly, we propose some in-place implementations of the linear permutations of SM4 and SM3. Based on our new techniques, our stand-alone memory-efficient quantum circuit implementation of SM4 only requires 384 qubits, seven ancilla qubits and 33,024 T-depth, while our depth-efficient quantum circuit of SM4 requires 384 qubits, 1080 ancilla qubits and 455 T-depth. Furthermore, we propose a stand-alone memory-efficient quantum circuit implementation of SM3 with 768 qubits, 33 ancilla qubits and 144,768 T-depth, while our depth-efficient quantum circuit of SM3 requires 768 qubits, 202 ancilla qubits, and 25,344 T-depth. Compared to the previous work, our new quantum circuits of SM3 requires less qubits and T-depth.

Keywords SM3 hash function · SM4 block cipher · Quantum circuit · Quantum resource estimation · Quantum gate

✉ Jian Zou
fzuzoujian15@163.com

1 College of Computer and Data Science, Fuzhou University, Fuzhou, China

2 Key Lab of Information Security of Network Systems, Fuzhou University, Fuzhou, China

3 Data Communication Science and Technology Research Institute, Beijing, China

4 School of Computer Science and Engineering, Huizhou University, Huizhou, China

5 Institute of Software, Chinese Academy of Sciences, Beijing, China

1 Introduction

Post-quantum cryptography arises a lot of security concerns in the current symmetric-key cryptographic community, which has been used to attack some symmetric cipher problem [20, 43, 44]. There are a lot of works on how to prevent quantum attacks, such as quantum key distribution [14, 26, 33], quantum digital signatures [24, 35, 42], quantum secret sharing [15, 17, 18], and quantum conference key agreement [12, 23, 34]. As pointed out by NIST first PQC standardization workshop, it is difficult to measure the complexity of quantum attacks. In order to solve this problem, some prior works proposed to use the time and memory cost of a quantum circuit as a measure of the complexity of a quantum attack. Therefore, how to construct an efficient quantum circuit is an important topic in the current cryptographic community. There are a lot of quantum circuit implementations of cipher algorithms, such as AES [5, 16, 21, 25], SHA-2 [3, 22], and ECC [6, 36].

Similar to AES [30] and SHA [31], it is necessary to estimate the security levels of Chinese commercial cryptography standard to quantum attacks, such as SM4 and SM3. SM4 block cipher [32] is used in Chinese WLAN Authentication and Privacy Infrastructure (WAPI) standard, which can provide data confidentiality. SM3 hash function [38] is the Chinese cryptographic hash function standard, which is similar to SHA-256. However, SM3 has stronger message dependency and more complex step function than SHA-256. As a result, it is worth of realizing the quantum circuits of SM4 and SM3 so as to obtain the precise resource estimate of the quantum attacks on SM4 and SM3.

In this paper, we focus on constructing some efficient quantum circuit implementations of SM4 and SM3. A quantum circuit can be constructed with the reversible and fault-tolerant logic gates set, such as the Hadamard gate, Phase gate, controlled-NOT gate (CNOT) and T gate. As shown in [2], the Hadamard gate, Phase gate, and CNOT gate are elements in Clifford set, while T gate does not belong to the Clifford set. In the current scenario, the cost of T gates is more expensive than the gates in Clifford set.

There are several works on constructing some optimal quantum circuits of block ciphers [3, 5, 16, 21, 22, 25, 45]. In 2016, Grassl et al. in [16] proposed the first quantum circuit implementation of AES. In 2019, Langenberg et al. in [25] presented an improved quantum circuit of AES. In details, Langenberg et al. observed that they could reduce the time and memory cost by converting the classical circuit implementation of AES's S-box [8] into a new quantum circuit implementation. In EUROCRYPT 2020, Jaques *et al.* [21] presented an improved quantum circuit of AES, which aimed to reduce the depth of their quantum circuit of AES. In Asiacrypt 2020, Zou *et al.* [45] proposed a more memory-efficient quantum circuit of AES. They observed that every operation in AES is invertible, which allows to uncompute a state from a later state.

The previous works [21, 25, 45] showed that an optimal classical circuit implementation could be converted into an optimal quantum circuit. As a result, we shall construct an optimal classical circuit of SM4's S-box so as to obtain an improved quantum circuit of SM4's S-box. There are several works on how to construct an efficient S-box in the classical setting. For interested readers, please refer to [7, 8, 10, 27, 40].

Apart from the above quantum circuits of AES, there are some works on how to construct an optimal quantum circuit of a Feistel-like structure. In [3], Amy et al. showed how to calculate the quantum costs of the preimage attacks on SHA-2 and SHA-3 with the number of qubits and quantum gates. In [22], Kim et al. proposed some optimal quantum circuits of SHA-2 by using the property of the message expansion function of SHA-2. In [37], Song et al. showed a new quantum circuit implementation of SM3. However, they did not utilize the property of the message expansion function of SM3 with 2721 qubits.

In this paper, we propose some memory-efficient and depth-efficient quantum circuit implementations of SM4 and SM3. Based on our quantum circuits of SM4 and SM3, we can obtain the precise resource estimation of the quantum attacks on SM4 and SM3.

1.1 Contribution

In this paper, we propose some new efficient quantum circuit implementations of SM4 block cipher and SM3 hash function. Firstly, we present an improved classical circuit of SM4's S-box by utilizing the tower technique. Secondly, we present some new techniques for converting our classical circuit of SM4's S-box into a new efficient quantum circuit of SM4's S-box. Thirdly, we propose a new implementation of the structure of SM4 to reduce the qubits number in our quantum circuit of SM4. Fourthly, we propose some new in-place implementations of SM3 and SM4. Fifthly, we reduce the qubits number in our quantum circuit of SM3 by utilizing the linear message expansion of SM3. To sum up, our contributions in each part can be detailed in the following.

We propose some improved classical and quantum circuit implementations of SM4's S-box. Firstly, we propose an efficient classical hardware implementation of SM4's S-box by utilizing the tower field technique. That is, our classical circuit implementation of SM4's S-box requires less classical AND gates than the previous works. Secondly, we observe the quantum circuit of SM4's S-box can be constructed with seven ancilla qubits by exploring the relationship between each parameters in our classical circuit of SM4's S-box. Thirdly, we propose some new rules to reduce the T-depth in our quantum circuit of SM4's S-box. Fourthly, we propose a memory-efficient quantum circuit implementation of SM4's round function as follows. That is, given an i-round transformation $X_{i+4} = X_i \oplus L(\tau((X_{i+1} \oplus X_{i+2} \oplus X_{i+3}, rk_i)))$, we can rewrite X_{i+4} as $X_{i+4} = L(L^{-1}(X_i) \oplus \tau((X_{i+1} \oplus X_{i+2} \oplus X_{i+3}, rk_i)))$, where L is the linear operation and τ is the nonlinear operation. Furthermore, we propose some new in-place implementation of the linear permutations L and L'. By adopting the above strategy, we can reduce the qubits number in our quantum circuit of SM4, because we do not need to introduce some new ancilla qubits for storing the output of SM4's S-box.

We propose an improved quantum circuit of SM3 hash function. Firstly, we propose a new memory-efficient quantum circuit of SM3's message expansion algorithm. That is, we just need to store any consecutive 16 words $\{W_i, W_{i+1}, \cdots, W_{i+15}\}$ (for $0 \leq i \leq 52$) instead of all message words. By using this observation, we can reduce the number of qubits in our quantum circuit of SM3. Secondly, we propose some new in-place implementations of the linear permutations P_0 and P_1 of SM3. Thirdly, we propose some new depth-efficient quantum circuit implementations of CH and MAJ Boolean functions. Note that our new quantum circuits implementations of the CH and MAJ functions can also be applied to SHA-1 and SHA-2.

In this work, we propose some stand-alone quantum circuits of SM4 and SM3, which shall uncompute some output qubits. If we want to design a quantum circuit of Grover on SM4/SM3, we do not need to uncompute the output qubits, because Grover algorithm can simply reverse the entire circuit. In [37], Song et al. showed a quantum circuit of Grover on SM3. We summarize the resources of the quantum implementations of SM4 and SM3 in Table 1. The parameter $T \cdot M$ is the multiplication of the T-depth (T) and the number of qubits (M). For a fairer comparison, we also propose some quantum circuits of Grover on SM4/SM3, which are labeled as "subroutine" in Table 1. Our stand-alone quantum circuits of SM4 and SM3 are labeled as "stand-alone" in Table 1. In addition, we replaced the Toffoli gates in [37] with the T gate design in the above quantum circuit of SM4 and SM3. That is, a Toffoli gate can be divided into seven T gates, seven CNOT gates and two H gates (see in [4]). Furthermore, we also convert QAND/QAND† to the gates in Clifford + T gates set. As shown in Table 1, the T-depth and qubits number in our depth-efficient quantum circuit of SM3 are both smaller than the quantum circuit proposed by Song *et al.* [37]. In addition, our memory-efficient quantum circuit of SM3 requires 801 qubits, which requires the least number of qubits so far.

1.2 Organization

This paper is organized as follows. In Sect. 2, we make a brief introduction to the quantum background. Sections 3 and 4 show the definition of SM4 block cipher and SM3 hash function. We also show an improved classical implementation of SM4's S-box in Sect. 3. In Sect. 5, we present some new implementation of SM4's S-box in the quantum hardware setting. Section 6 shows our improved quantum circuit implementations of SM4. We show our improved quantum circuit implementations of SM3 in Sect 7. Section 8 concludes this paper.

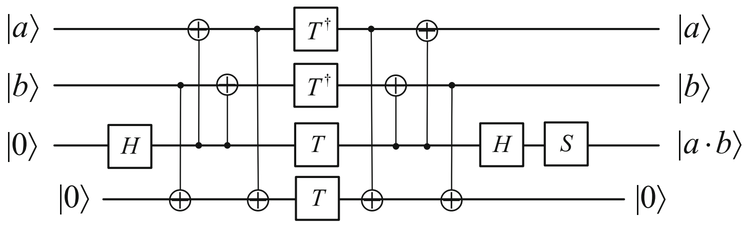
2 Quantum background

In this section, we make a brief introduction to qubits and quantum gates, which are the basic of the quantum computing. A qubit can be described in the superposition: $\alpha|0\rangle + \beta|1\rangle$, where $|\alpha|$ and $|\beta|$ form an orthonormal basis. In addition, $|\alpha|^2$ (or $|\beta|^2$) can be seen as the probability of observing $|0\rangle$ (or $|1\rangle$) after measurement, which satisfies $|\alpha|^2 + |\beta|^2 = 1$.

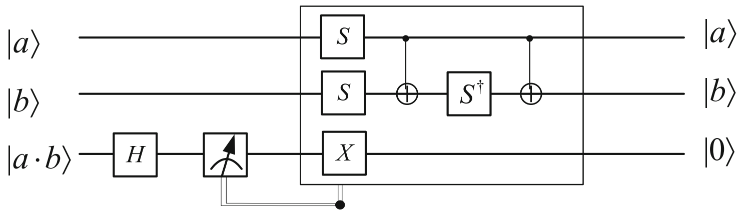
As shown in [2], any quantum circuit can be constructed with the Clifford + T gates set. The Clifford set consists of the Hadamard gate (H), Phase gate, NOT gate, and controlled-NOT gate (CNOT). Note that the T gate does not belong to Clifford set. A NOT gate takes a qubit $\alpha|0\rangle + \beta|1\rangle$ as input, and output $\alpha|1\rangle + \beta|0\rangle$. A CNOT gate can

Table 1 Summary of the resources of our quantum circuits of SM4 and SM3

Target	Qubits	T-Depth	T-gate	CNOT	1qCliff	$T \cdot M$	Type	Source
SM4	263	22,016	86,528	196,672	58,158	5,790,208	Subroutine	Section 6.1
	391	33,024	129,792	283,456	108,764	12,912,384	Stand-alone	Section 6.1
	284	5504	86,528	196,672	58,158	1,563,136	Subroutine	Section 6.1
	412	8256	129,792	283,456	108,764	3,401,472	Stand-alone	Section 6.1
	1336	231	32,768	183,872	68,654	28,056	Subroutine	Section 6.2
	1464	455	49,152	274,752	103,452	666,120	Stand-alone	Section 6.2
SM3	2721	Not reported	303,296	437,440	89,294	Not reported	Subroutine	[37]
	801	144,768	434,304	573,776	39,232	115,959,168	Subroutine	Section 7.1
	801	144,768	434,304	583,968	39,232	115,959,168	Stand-alone	Section 7.1
	970	25,344	1,187,072	1,370,432	420,160	24,583,680	Subroutine	Section 7.2
	970	25,344	1,187,072	1,380,624	420,160	24,583,680	Stand-alone	Section 7.2



(a) *QAND* gate



(b) *QAND*† gate

Fig. 1 A T-depth 1 circuit for an AND gate

be used to simulate a classical XOR gate, which outputs $U_{CNOT}|a\rangle|b\rangle = |a\rangle|a \oplus b\rangle$. The Toffoli gate can be used to simulate a classical AND gate, which takes a, b, c three qubits as input and outputs $U_{Toffoli}|a\rangle|b\rangle|c\rangle = |a\rangle|b\rangle|c \oplus ab\rangle$. Some prior works focused on decomposing a Toffoli gate into some T gates and the gates in Clifford set. In 2013, Amy et al. [4] observed a Toffoli gate could be decomposed into a circuit with the T-depth of 3, 7 T gates, seven CNOT gates and two H gates. In [21], Jaques et al. proposed a new depth-efficient quantum circuit of AES with a new quantum AND (QAND) gate with a T-depth of 1. A quantum AND (QAND) gate can be written as $U_{\text{QAND}}|a\rangle|b\rangle|0\rangle|0\rangle = |a\rangle|b\rangle|ab\rangle|0\rangle$ (see in Fig. 1). QAND† denotes the adjoint of QAND, which does not require the T-gate. As shown in [21], a QAND gate consists of four T gates, two H gates, one S gate and eight CNOT gates, while a QAND† just needs two CNOT gates, one X gate, one H gates and three S gate. Due to the space limitation, we just omit some details of qubits and quantum gates. For interested readers, please refer to [29].

Our quantum circuits can be constructed with CNOT gates, NOT gates, SWAP gates, the Hadamard gate (H), Toffoli gates [39], quantum AND gates (QAND) [21], and so on. Note that the cost of T gates are more expensive than the gates in Clifford group. Since only Toffoli gates and QAND gates contain T gates, the cost of the Toffoli gate and the QAND gate is both more expensive than the quantum gates in the Clifford group. In addition, we assume that the cost of a SWAP gate is free. Similar to [5, 16, 22, 25, 45], we define the time cost as the nonparallelizable of the T gate (i.e., T-depth), while the memory cost is the total number of logical qubits required to perform the quantum algorithm. In addition, we denote by $\#1qCliff$ the number of 1-qubit Clifford gates, which is similar to [21].

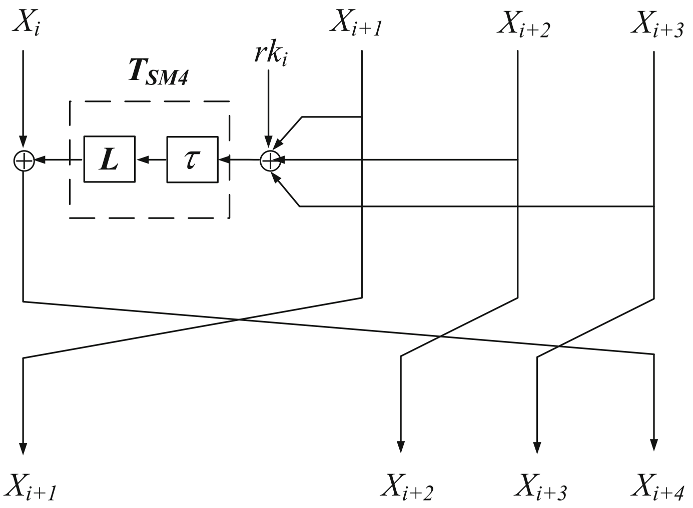


Fig. 2 Round function of SM4

Apart from the quantum gate, our quantum circuits of SM4 and SM3 need to deal with the following three different types of qubits to avoid the confusion caused by terminology. These three qubits are data qubits, ancilla qubits, and output qubits. Data qubits are used for storing input message, while output qubits are written with the output information. Ancilla qubits (or called work qubits) are recorded with unwanted information, which guarantees the reversibility of the whole line. Ancilla qubits shall be cleaned up in the end, while we do not need to clean up the output qubits. In order to reduce the cost of our quantum circuit, we try to apply T gates to the output qubits instead of ancilla qubits.

3 Specification of SM4 block cipher

The block length and the key length of SM4 block cipher are both of 128 bits. As shown in Fig. 2, SM4 adopts an unbalanced Feistel-SPN structure, which iterates a round function F 32 times. As shown in Fig. 2, the encryption procedure of SM4 can be described as follows.

1. Divide the 128-bit input plaintext into four 32-bit words (X_0, X_1, X_2, X_3).
2. Compute the ith round function of SM4 as below $(X_i, X_{i+1}, X_{i+2}, X_{i+3}) \rightarrow (X_{i+1}, X_{i+2}, X_{i+3}, X_{i+4})$, where $X_{i+4} = X_i \oplus T_{\text{SM4}}(X_{i+1} \oplus X_{i+2} \oplus X_{i+3}, rk_i)$ (for $0 \leq i \leq 31$).
3. After iterating SM4's round function 32 times, we can compute the 128-bit ciphertext with a switch permutation R as follows: $(C_1, C_2, C_3, C_4) = R(X_{32}, X_{33}, X_{34}, X_{35}) = (X_{35}, X_{34}, X_{33}, X_{32})$.

The above permutation T_{SM4} in round function consists of the following two operations:

1. The nonlinear permutation τ: The permutation τ consists of four S-box operations. Each S-box operation is applied to an 8-bit cell of the state. The 32-bit output word of τ is the input of the linear substitution L.
2. The linear permutation $L : F_2^{32} \to F_2^{32}$ is defined as $L(B) = B \oplus (B \lll 2) \oplus (B \lll 10) \oplus (B \lll 18) \oplus (B \lll 24)$, where $B \in F_2^{32}$ is the input of L. The inverse of L can be defined as follows: $L^{-1}(B) = B \oplus (B \lll 2) \oplus (B \lll 4) \oplus (B \lll 8) \oplus (B \lll 12) \oplus (B \lll 14) \oplus (B \lll 16) \oplus (B \lll 18) \oplus (B \lll 22) \oplus (B \lll 24) \oplus (B \lll 30)$.

The key schedule of SM4 is similar to the encryption function of SM4. Define (MK_0, MK_1, MK_2, MK_3) as the master key, we can calculate the round-key $(RK_0, RK_1, \cdots RK_{30}, RK_{31})$ as follows.

1. Obtain $(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3)$, where $FK_0 = 0xa3b1bac6$, $FK_1 = 0x56aa3350$, $FK_2 = 0x677d9197$, $FK_3 = 0xb27022dc$.
2. For $0 \leq i \leq 31$, compute $rk_i = K_{i+4} = K_i \oplus L' \circ \tau(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i)$, where $L'(B) = B \oplus (B \lll 13) \oplus (B \lll 23)$ and τ is the same nonlinear permutation used in SM4's round function. The parameter $CK_i = (CK_{i,0}, CK_{i,1}, CK_{i,2}, CK_{i,3})$ can be computed as $CK_{i,j} = (4i + j) \times 7(mod\ 256)$ with $i = 0, 1, 2, \cdots, 31$ and $j = 0, 1, 2, 3$, where $CK_{i,j} \in (F_2^8)$.
3. The inverse of L' can be defined as follows: $L'^{-1}(B) = B \oplus (B \lll 2) \oplus (B \lll 4) \oplus (B \lll 8) \oplus (B \lll 11) \oplus (B \lll 12) \oplus (B \lll 14) \oplus (B \lll 17) \oplus (B \lll 22) \oplus (B \lll 23) \oplus (B \lll 24) \oplus (B \lll 30) \oplus (B \lll 31)$.

The whole structure of the decryption algorithm of SM4 is same as the encryption algorithm of SM4, while we shall adopt the reverse order of the round keys (i.e., $(rk_{31}, rk_{30}, \cdots, rk_0)$) in the decryption algorithm. For a full description of SM4, please refer to [32].

3.1 The algebraic structure of SM4's S-box

The hardware implementation of SM4 is measured in terms of power consumption, area and throughput, which mainly depends upon the implementation of its S-box. In the official document of depiction of SM4 [32], its S-box is described as a lookup table. In other words, we shall store the 256 pre-computed values of SM4's S-box. Apart from the above method, the S-box of SM4 can be implemented by computing the multiplicative inverse of the 8-bit input followed by an affine transformation.

In this section, we propose an improved classical implementation of SM4's S-box by using the tower technique. The goal of our circuit is to identify a proper tower basis so as to minimize the overall gate numbers of SM4's S-box with AND gates, XOR gates and NOT gates. In details, we observe that SM4's S-box can be represented as follows: $S(b) = M_2 \cdot Inv(M_1 \cdot b \oplus C_1) \oplus C_2, b \in F_2^8$, where

$$M_1 = \begin{bmatrix} 1&1&0&1&0&0&1&1 \\ 1&1&1&0&1&0&0&1 \\ 1&1&1&1&0&1&0&0 \\ 0&1&1&1&1&0&1&0 \\ 0&0&1&1&1&1&0&1 \\ 1&0&0&1&1&1&1&0 \\ 0&1&0&0&1&1&1&1 \\ 1&0&1&0&0&1&1&1 \end{bmatrix}, C_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, M_2 = \begin{bmatrix} 1&1&0&1&0&0&1&1 \\ 1&1&1&0&1&0&0&1 \\ 1&1&1&1&0&1&0&0 \\ 0&1&1&1&1&0&1&0 \\ 0&0&1&1&1&1&0&1 \\ 1&0&0&1&1&1&1&0 \\ 0&1&0&0&1&1&1&1 \\ 1&0&1&0&0&1&1&1 \end{bmatrix}, C_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

By utilizing the tower field technique, the S-box of SM4 can be represented as $S_{\text{SM4}}(x) = B_S \cdot F_S(U_S(x))$. The details of U_S, F_S and B_S can be shown as follows. The matrix U_S takes $x_0, x_1, \cdots, x_7$ as input, and outputs $x_1, y_0, \cdots, y_{20}$.

$y_5 = \overline{x_0}$,	$y_{12} = x_6 \oplus x_2$,	$y_{10} = x_1 \oplus y_{12}$,	$y_{17} = \overline{x_6}$,
$t_0 = x_5 \oplus x_3$,	$y_1 = t_0 \oplus x_0$,	$y_3 = y_1 \oplus x_1$,	$y_9 = y_3 \oplus x_3$,
$y_4 = y_5 \oplus y_9$,	$y_{19} = t_0 \oplus x_6$,	$t_1 = x_7 \oplus x_4$,	$y_8 = x_5 \oplus t_1$,
$y_0 = y_1 \oplus y_8$,	$y_2 = y_{12} \oplus y_0$,	$y_{15} = y_3 \oplus y_2$,	$y_{16} = y_{17} \oplus y_{15}$,
$y_{11} = y_9 \oplus y_{16}$,	$t_2 = x_7 \oplus y_1$,	$y_7 = y_{11} \oplus t_2$,	$y_6 = y_{16} \oplus y_7$,
$y_{14} = y_5 \oplus y_7$,	$y_{13} = y_{11} \oplus y_{14}$,	$y_{18} = y_3 \oplus y_7$,	$t_3 = x_7 \oplus x_2$.
$y_{20} = t_0 \oplus t_3$.			

The F_s takes $x_1, y_1, \cdots, y_{21}$ as input and outputs $z_0, z_1, \cdots, z_{17}$.

$t_4 = y_7 \cdot y_3$,	$t_5 = y_{16} \cdot y_{15}$,	$t_6 = y_6 \cdot y_2$,	$t_7 = y_{13} \cdot y_{12}$,
$t_8 = y_{11} \cdot y_{10}$,	$t_9 = y_{14} \cdot x_1$,	$t_{10} = y_5 \cdot y_1$,	$t_{11} = y_9 \cdot y_8$,
$t_{12} = y_4 \cdot y_0$,	$t_{13} = t_{10} \oplus t_{12}$,	$t_{14} = t_{13} \oplus t_6$,	$t_{15} = t_{10} \oplus t_{11}$,
$t_{16} = t_{15} \oplus t_4$,	$t_{17} = t_{14} \oplus t_5$,	$t_{18} = t_{17} \oplus t_9$,	$t_{19} = t_{18} \oplus t_7$,
$t_{20} = t_{19} \oplus y_{20}$,	$t_{21} = t_{16} \oplus t_6$,	$t_{22} = t_{21} \oplus t_9$,	$t_{23} = t_{22} \oplus t_8$,
$t_{24} = t_{23} \oplus y_{19}$,	$t_{25} = t_{14} \oplus t_4$,	$t_{26} = t_{25} \oplus y_{18}$,	$t_{27} = t_{16} \oplus t_5$,
$t_{28} = t_{27} \oplus y_{17}$,			
$t_{29} = t_{20} \oplus t_{24}$,	$t_{30} = t_{24} \cdot t_{28}$,	$t_{31} = t_{26} \oplus t_{30}$,	$t_{32} = t_{29} \cdot t_{31}$,
$t_{33} = t_{20} \oplus t_{32}$,	$t_{34} = t_{26} \oplus t_{28}$,	$t_{35} = t_{20} \oplus t_{30}$,	$t_{36} = t_{34} \cdot t_{35}$,
$t_{37} = t_{26} \oplus t_{36}$,	$t_{38} = t_{28} \oplus t_{37}$,	$t_{39} = t_{31} \oplus t_{37}$,	$t_{40} = t_{26} \cdot t_{39}$,
$t_{41} = t_{40} \oplus t_{38}$,	$t_{42} = t_{31} \oplus t_{40}$,	$t_{43} = t_{33} \cdot t_{42}$,	$t_{44} = t_{43} \oplus t_{29}$,
$t_{45} = t_{37} \oplus t_{41}$,	$t_{46} = t_{37} \oplus t_{33}$,	$t_{47} = t_{41} \oplus t_{44}$,	$t_{48} = t_{46} \oplus t_{47}$,
$t_{49} = t_{33} \oplus t_{44}$,	$z_{17} = t_{37} \cdot y_3$,	$z_{16} = t_{45} \cdot y_{15}$,	$z_{15} = t_{41} \cdot y_2$,
$z_{14} = t_{48} \cdot y_{10}$,	$z_{13} = t_{47} \cdot y_{12}$,	$z_{12} = t_{46} \cdot x_1$,	$z_{11} = t_{33} \cdot y_1$,
$z_{10} = t_{49} \cdot y_8$,	$z_9 = t_{44} \cdot y_0$,	$z_8 = t_{37} \cdot y_7$,	$z_7 = t_{45} \cdot y_{16}$,
$z_6 = t_{41} \cdot y_6$,	$z_5 = t_{48} \cdot y_{11}$,	$z_4 = t_{47} \cdot y_{13}$,	$z_3 = t_{46} \cdot y_{14}$,
$z_2 = t_{33} \cdot y_5$,	$z_1 = t_{49} \cdot y_9$,	$z_0 = t_{44} \cdot y_4$.	

The matrix B_s takes $z_0, z_1, \cdots, z_{17}$ as input and outputs $s_0, s_1, \cdots, s_7$.

By using the tower field techniques, we can reduce the number of the AND gate in our classical circuit of SM4's S-box (see in Table 2). That is, our circuit of SM4's S-box only requires 32 AND gates. In [40], Wei et al. also proposed a circuit, which needed 32 AND gates and nine NOR gates. Since a NOR gate requires an AND gate, it means [40] needs more AND gates than our circuit.

$e_{11} = z_{17} \oplus z_{16}$,	$e_{10} = z_{15} \oplus z_{16}$,	$e_9 = z_{12} \oplus z_{13}$,	$e_8 = z_{12} \oplus z_{14}$,
$e_7 = z_{11} \oplus z_{10}$,	$e_6 = z_9 \oplus z_{10}$,	$e_5 = z_8 \oplus z_7$,	$e_4 = z_6 \oplus z_7$,
$e_3 = z_3 \oplus z_4$,	$e_2 = z_3 \oplus z_5$,	$e_1 = z_2 \oplus z_1$,	$e_0 = z_0 \oplus z_1$,
$s_4 = e_{10} \oplus e_8$,	$t_{50} = e_9 \oplus e_7$,	$t_{51} = \overline{e_1}$,	$t_{52} = e_3 \oplus t_{51}$,
$t_{53} = e_{11} \oplus e_9$,	$s_3 = t_{52} \oplus t_{53}$,	$t_{54} = e_4 \oplus e_0$,	$s_5 = t_{50} \oplus t_{54}$,
$s_7 = s_3 \oplus t_{54}$,	$t_{55} = e_5 \oplus t_{50}$,	$s_2 = e_3 \oplus t_{55}$,	$t_{56} = s_4 \oplus t_{52}$,
$s_6 = t_{50} \oplus t_{56}$,	$t_{57} = e_{10} \oplus e_6$,	$s_0 = \overline{t_{57}}$,	$t_{58} = e_4 \oplus e_2$,
$t_{59} = s_2 \oplus t_{56}$,	$s_1 = t_{58} \oplus t_{59}$,		

Table 2 Summary of the resources of our classical circuits of SM4's S-box

Algorithm	XOR/XNOR	NAND	AND	NOR	NOT	Source
SM4	99	0	58	0	11	[28]
	157	0	63	0	0	[9]
	134	0	36	0	0	[1]
	66	32	0	9	1	[40]
	83	0	32	0	4	Section 3.1

4 Description of SM3 hash function

SM3 adopts the Merkle-Damgård design, which outputs a 256-bit hash value. Given a message M, the hash output of SM3 can be computed as follows:

$$\begin{cases} CV_0 \leftarrow IV \\ CV_{i+1} \leftarrow CF(CV_i, M_i) \text{ for } i = 0, 1, \ldots, n-1, \end{cases}$$

where IV is the initial value, and CV_n is the 256-bit output of SM3.

Denote the length of input message M as len_M ($len_M < 2^{64} - 1$). Then, the padding procedure of SM3 can be explained as follows. Firstly, we put a single bit "1" and len_0 "0"s at the end of the message M, where $len_M + 1 + len_0 \equiv 448 \mod 512$. Secondly, we obtain the padded message M^* by putting 64 bits for the length of M at the end of M. Thirdly, we divide the padded message M^* into 512-bit blocks M_i ($i = 0, 1, \ldots, n-1$).

The compression function CF of SM3 takes 256-bit chain value and 512-bit input message M_i to output a new 256-bit chain value. In order to compute the CF, we shall compute the message expansion of SM3 in the first place. Secondly, we divide the 512-bit M_i into 16 32-bit words W_j ($j = 0, 1, \ldots, 15$). Thirdly, we can expand the 16 32-bit words to 68 32-bit words with the following formula $W_j \leftarrow P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15)) \oplus (W_{j-13} \lll 7) \oplus W_{j-6}$ for $16 \leq j \leq 67$. In addition, we still need to produce $W_i^{'}$ by $W_i^{'} = W_i \oplus W_{i+4}$ for $16 \leq i \leq 64$.

After the message expansion procession, we can compute the compression function $CF(CV_i, M_i)$ as follows:

1. Assign CV_i to S_0, where $S_j = (A_j, \ldots, H_j)$.

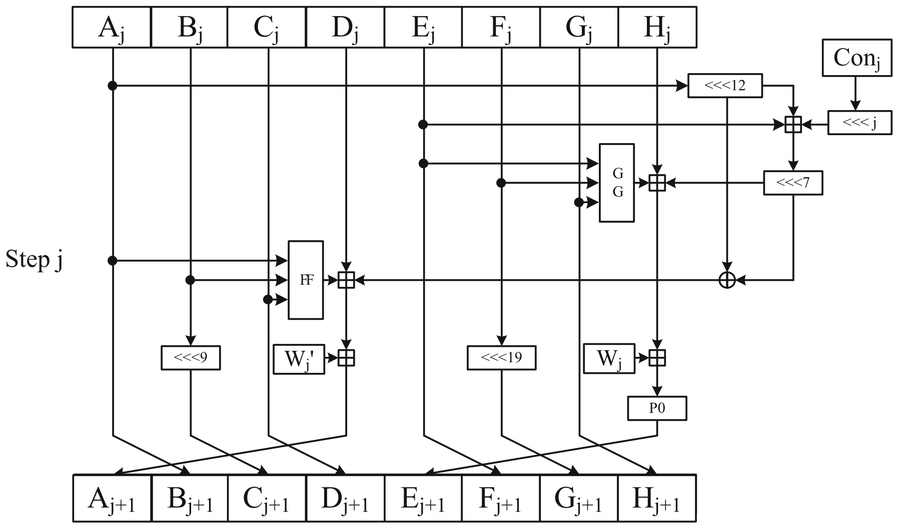

A_j B_j C_j D_j E_j F_j G_j H_j
Con_j
<<<12
<<<j
<<<7
GG
Step j
FF
<<<9
W_j'
<<<19
W_j
P0
A_{j+1} B_{j+1} C_{j+1} D_{j+1} E_{j+1} F_{j+1} G_{j+1} H_{j+1}

Fig. 3 jth step function of SM3

2. FOR $j = 0$ TO 63
   Update S_j by the step function f (see in Fig. 3).
   ENDFOR
3. Output $CV_{i+1} \leftarrow S_0 \oplus S_{64}$, i.e., $(A_0 \oplus A_{64}, \ldots, H_0 \oplus H_{64})$.

As shown in [38], the diffusion permutations are $P_0(X) = X \oplus (X \lll 9) \oplus (X \lll 17)$, and $P_1(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$. In addition, the Boolean functions FF_j, GG_j and constant Con_j employed in the step transformation f can be explained as follows:

$$FF_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z & 0 \le j \le 15 \\ (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z) & 16 \le j \le 63 \end{cases} \tag{1}$$

$$GG_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z & 0 \le j \le 15 \\ (X \wedge Y) \oplus (\neg X \wedge Z) & 16 \le j \le 63. \end{cases} \tag{2}$$

$$Con_j = \begin{cases} \text{0x79cc4519} & 0 \le j \le 15 \\ \text{0x7a879d8a} & 16 \le j \le 63. \end{cases} \tag{3}$$

5 Quantum circuit implementations of SM4's S-box

As pointed out in [3], the cost of the T gate is more expensive than the gates in Clifford set. Note that only the nonlinear function consists of the T gate. As a result, we focus on constructing some efficient quantum circuits of SM4's S-box in this section, which are based on our improved classical circuits of SM4's S-box. That is, we try to propose some depth-efficient and memory-efficient quantum circuit implementations of SM4's S-box. We present the details of our new quantum circuit of SM4's S-box in the following sections.

5.1 Some new memory-efficient quantum circuit implementations for the S-box of SM4

In this section, we propose a low-memory quantum circuit implementation of SM4's S-box. That is, we can reduce the number of Toffoli gates and qubits by exploring the linear relationship between each parameters in our classical circuit of SM4's S-box. Similar to [45], we propose the following two observations, which can be used to reduce the number of qubits in our quantum circuit of SM4's S-box.

Observation 1. As shown in Sect. 3.1, the 18 values of $z_0, \cdots, z_{17}$ can be obtained with the knowledge of $t_{33}, t_{37}, t_{41}, t_{44}, t_{45}, t_{46}, t_{47}, t_{48}, t_{49}$, and $x_1, y_0, y_1, \cdots, y_{20}$, where $y_0, \cdots, y_{20}$ are the linear combination of $x_0, x_1, \cdots, x_7$. In addition, $t_{45}, t_{46}, t_{47}, t_{48}, t_{49}$ can be obtained by the linear combination of $t_{33}, t_{37}, t_{41}, t_{44}$.

According to **Observation 1**, we can compute $z_0, \cdots, z_{17}$ only with the knowledge of $t_{33}, t_{37}, t_{41}, t_{44}$ and $x_0, x_1, \cdots, x_7$. Apart from **Observation 1**, we still need **Observation 2** to compute the SM4's S-box.

Observation 2. The $s_0, s_1, \cdots, s_7$ of SM4's S-box can be obtained by a linear combination of the 18 values of $z_0, \cdots, z_{17}$. That is, we can express the linear expression of s_i (for $0 \leq i \leq 7$) as follows, where $\bar{s}$ applies the NOT operation on s.

$$\begin{aligned}
s_0 &= \overline{z_9 \oplus z_{10} \oplus z_{15} \oplus z_{16}}, \\
s_1 &= \overline{z_1 \oplus z_2 \oplus z_3 \oplus z_5 \oplus z_6 \oplus z_8 \oplus z_{10} \oplus z_{11} \oplus z_{13} \oplus z_{14} \oplus z_{15} \oplus z_{16}}, \\
s_2 &= z_3 \oplus z_4 \oplus z_7 \oplus z_8 \oplus z_{10} \oplus z_{11} \oplus z_{12} \oplus z_{13}, \\
s_3 &= \overline{z_1 \oplus z_2 \oplus z_3 \oplus z_4 \oplus z_{12} \oplus z_{13} \oplus z_{16} \oplus z_{17}}, \\
s_4 &= z_{12} \oplus z_{14} \oplus z_{15} \oplus z_{16}, \\
s_5 &= z_0 \oplus z_1 \oplus z_6 \oplus z_7 \oplus z_{10} \oplus z_{11} \oplus z_{12} \oplus z_{13}, \\
s_6 &= \overline{z_1 \oplus z_2 \oplus z_3 \oplus z_4 \oplus z_{10} \oplus z_{11} \oplus z_{13} \oplus z_{14} \oplus z_{15} \oplus z_{16}}, \\
s_7 &= \overline{z_0 \oplus z_2 \oplus z_3 \oplus z_4 \oplus z_6 \oplus z_7 \oplus z_{12} \oplus z_{13} \oplus z_{16} \oplus z_{17}}.
\end{aligned}$$

Based on our **Observation 1** and **Observation 2**, we propose Algorithm 1 to compute the 8-bit output of SM4's S-box. The details of Algorithm 1 are shown as follows. Firstly, we need to introduce seven zero ancilla qubits in Algorithm 1, which are $T[j] = 0$ (for $0 \leq j \leq 5$) and $Z = 0$. Secondly, the 8-bit input of Algorithm 1 satisfies $U[j] = x_j$ (for $0 \leq j \leq 7$). As shown in the below, Algorithm 1 can compute the output of SM4's S-box and XOR it onto the output bits when they are nonzero. That is, we have no restrictions on $S[j]$ (for $0 \leq j \leq 7$). In details, we assume the output of S-box $S[j] = y_j$ (for $0 \leq j \leq 7$), where each y_j can be 0 or 1 (for $0 \leq j \leq 7$).

Algorithm 1 A New Memory-Efficient quantum circuit of SM4's S-box.

```
    Input, $Z = 0$
    Input, $U[j] = x_j$, for $0 \leq j \leq 7$;
    Input, $T[j] = 0$, for $0 \leq j \leq 5$;
    Input, $S[j] = y_j$, for $0 \leq j \leq 7$;
1: $U[0] = \overline{U[0]}$; \\ $U[0] = y_5$
2: $U[3] = \overline{U[3] \oplus U[0] \oplus U[5]}$; \\ $U[3] = y_1$
3: $U[2] = \overline{U[2] \oplus U[1] \oplus U[4] \oplus U[5] \oplus U[7]}$; \\ $U[2] = y_{16}$
4: $U[6] = \overline{U[6] \oplus U[2]}$; \\ $U[6] = y_{15}$
5: $U[5] = \overline{U[5] \oplus U[0] \oplus U[1] \oplus U[2] \oplus U[3] \oplus U[7]}$; \\ $U[5] = y_7$
6: $U[1] = U[1] \oplus U[3]$; \\ $U[1] = y_3$
7: $QAND(U[0], U[3], T[0], T[1])$; \\ $T[0] = t_{10}$
8: $QAND(U[5], U[1], T[2], T[4])$; \\ $T[2] = t_4$
9: $QAND(U[2], U[6], T[3], T[5])$; \\ $T[3] = t_5$
10: $T[1] = T[1] \oplus T[0]$; \\ $T[1] = t_{10}$
11: $U[1] = U[3] \oplus U[1]$; \\ $U[1] = x_1$
12: $U[2] = U[2] \oplus U[0] \oplus U[3] \oplus U[5] \oplus U[7]$; \\ $U[2] = y_4$
13: $U[3] = \overline{U[3] \oplus U[1] \oplus U[2] \oplus U[4] \oplus U[7]}$; \\ $U[3] = y_0$
14: $U[0] = U[0] \oplus U[2]$; \\ $U[0] = y_9$
15: $U[4] = \overline{U[4] \oplus U[1] \oplus U[2] \oplus U[7]}$; \\ $U[4] = y_8$
16: $U[7] = U[7] \oplus U[0] \oplus U[3] \oplus U[4]$; \\ $U[7] = y_6$
17: $U[6] = U[6] \oplus U[1] \oplus U[3] \oplus U[4]$; \\ $U[6] = y_2$
18: $T[0] = (U[2] \cdot U[3]) \oplus T[0]$; \\ $T[0] = t_{13}$
19: $T[1] = (U[0] \cdot U[4]) \oplus T[1]$; \\ $T[1] = t_{15}$
20: $T[0] = (U[7] \cdot U[6]) \oplus T[0]$; \\ $T[0] = t_{14}$
21: $T[1] = T[1] \oplus T[2]$; \\ $T[1] = t_{16}$
22: $T[2] = T[2] \oplus T[0]$; \\ $T[2] = t_{25}$
23: $T[0] = T[0] \oplus T[3]$; \\ $T[0] = t_{17}$
24: $T[3] = T[3] \oplus T[1]$; \\ $T[3] = t_{27}$
25: $T[1] = (U[7] \cdot U[6]) \oplus T[1]$; \\ $T[1] = t_{21}$
26: $U[3] = U[3] \oplus U[1] \oplus U[4] \oplus U[5]$; \\ $U[3] = y_{18}$
27: $T[2] = T[2] \oplus U[3]$; \\ $T[2] = t_{26}$
28: $U[2] = U[2] \oplus U[1] \oplus U[4] \oplus U[5]$; \\ $U[2] = y_{20}$
29: $T[0] = T[0] \oplus U[2]$; \\ $T[0] = t_{17} \oplus y_{20}$
30: $U[7] = U[7] \oplus U[3] \oplus U[6]$; \\ $U[7] = y_{17}$
31: $T[3] = T[3] \oplus U[7]$; \\ $T[3] = t_{28}$
32: $U[2] = U[2] \oplus U[0] \oplus U[3] \oplus U[4] \oplus U[7]$; \\ $U[2] = y_{19}$
33: $T[1] = T[1] \oplus U[2]$; \\ $T[1] = t_{21} \oplus y_{19}$
34: $U[2] = U[2] \oplus U[0] \oplus U[1] \oplus U[5] \oplus U[6]$; \\ $U[2] = y_{13}$
35: $U[3] = U[3] \oplus U[1] \oplus U[4] \oplus U[5] \oplus U[6]$; \\ $U[3] = y_{12}$
36: $U[1] = U[1] \oplus U[3]$; \\ $U[1] = y_{10}$
37: $U[7] = U[7] \oplus U[0] \oplus U[1] \oplus U[4]$; \\ $U[7] = y_{11}$
38: $T[0] = (U[2] \cdot U[3]) \oplus T[0]$; \\ $T[0] = t_{17} \oplus y_{20} \oplus t_7$
39: $T[1] = (U[1] \cdot U[7]) \oplus T[1]$; \\ $T[1] = t_{21} \oplus y_{19} \oplus t_8$
40: $U[1] = U[1] \oplus U[3]$; \\ $U[1] = x_1$
41: $U[7] = U[7] \oplus U[2]$; \\ $U[7] = y_{14}$
42: $T[0] = (U[1] \cdot U[7]) \oplus T[0]$; \\ $T[0] = t_{20}$
43: $T[1] = (U[1] \cdot U[7]) \oplus T[1]$; \\ $T[1] = t_{24}$
44: $QAND(T[1], T[3], T[5], T[4])$; \\ $T[5] = t_{30}$
45: $T[4] = T[4] \oplus T[2]$; \\ $T[4] = t_{26}$
46: $T[1] = T[1] \oplus T[0]$; \\ $T[1] = t_{29}$
47: $T[3] = T[3] \oplus T[2]$; \\ $T[3] = t_{34}$
48: $T[0] = T[0] \oplus T[5]$; \\ $T[0] = t_{35}$
49: $T[5] = T[5] \oplus T[2]$; \\ $T[5] = t_{31}$
50: $T[2] = (T[0] \cdot T[3]) \oplus T[2]$; \\ $T[2] = t_{37}$
51: $T[3] = T[3] \oplus T[4]$; \\ $T[3] = t_{28}$
52: $T[0] = (T[1] \cdot T[5]) \oplus T[0]$; \\ $T[0] = t_{35} \oplus t_{32}$
53: $T[5] = T[5] \oplus T[4]$; \\ $T[5] = t_{30}$
54: $T[0] = T[0] \oplus T[5]$; \\ $T[0] = t_{33}$
55: $T[5] = T[5] \oplus T[4]$; \\ $T[5] = t_{31}$
56: $T[5] = T[5] \oplus T[2]$; \\ $T[5] = t_{39}$
57: $T[3] = T[3] \oplus T[2]$; \\ $T[3] = t_{38}$
58: $T[3] = (T[4] \cdot T[5]) \oplus T[3]$; \\ $T[3] = t_{41}$
59: $T[2] = T[2] \oplus T[5]$; \\ $T[2] = t_{31}$
60: $T[2] = (T[4] \cdot T[5]) \oplus T[2]$; \\ $T[2] = t_{42}$
61: $T[1] = (T[0] \cdot T[2]) \oplus T[1]$; \\ $T[1] = t_{44}$
62: $T[2] = (T[4] \cdot T[5]) \oplus T[2]$; \\ $T[2] = t_{31}$
63: $T[2] = T[2] \oplus T[5]$; \\ $T[2] = t_{37}$
64: $U[1] = U[1] \oplus U[3] \oplus U[4] \oplus U[6]$;\\ $U[1] = y_3$
65: $Z = (T[2] \cdot U[1]) \oplus Z$;
66: $S[3] = S[3] \oplus Z$;
67: $S[7] = S[7] \oplus Z$;
68: $QAND^{\dagger}(T[2], U[1], Z)$;
69: $U[1] = U[1] \oplus U[3] \oplus U[4] \oplus U[6]$;\\ $U[1] = x_1$
70: $T[2] = T[2] \oplus T[3]$;\\ $T[2] = t_{45}$
71: $U[1] = U[1] \oplus U[3] \oplus U[4]$;\\ $U[1] = y_{15}$
72: $Z = (T[2] \cdot U[1]) \oplus Z$; \\ $Z = z_{16}$
73: $S[0] = S[0] \oplus Z$;
74: $S[1] = S[1] \oplus Z$;
75: $S[3] = S[3] \oplus Z$;
76: $S[4] = S[4] \oplus Z$;
77: $S[6] = S[6] \oplus Z$;
78: $S[7] = S[7] \oplus Z$;
79: $QAND^{\dagger}(T[2], U[1], Z)$;
80: $U[1] = U[1] \oplus U[3] \oplus U[4]$;\\ $U[1] = x_1$
81: $T[2] = T[2] \oplus T[3]$;\\ $T[2] = t_{37}$
82: $Z = (T[3] \cdot U[6]) \oplus Z$; \\ $Z = z_{15}$
83: $S[0] = S[0] \oplus Z$;
84: $S[1] = S[1] \oplus Z$;
85: $S[4] = S[4] \oplus Z$;
86: $S[6] = S[6] \oplus Z$;
87: $QAND^{\dagger}(T[3], U[6], Z)$;
88: $T[2] = T[2] \oplus T[0] \oplus T[1] \oplus T[3]$;\\ $T[2] = t_{48}$
89: $U[1] = U[1] \oplus U[3]$;\\ $U[1] = y_{10}$
90: $Z = (T[2] \cdot U[1]) \oplus Z$; \\ $Z = z_{14}$
91: $S[1] = S[1] \oplus Z$;
92: $S[4] = S[4] \oplus Z$;
93: $S[6] = S[6] \oplus Z$;
94: $QAND^{\dagger}(T[2], U[1], Z)$;
95: $T[2] = T[2] \oplus T[0] \oplus T[1] \oplus T[3]$;\\ $T[2] = t_{37}$
96: $U[1] = U[1] \oplus U[3]$;\\ $U[1] = x_1$
97: $T[1] = T[1] \oplus T[3]$;\\ $T[1] = t_{47}$
98: $Z = (T[1] \cdot U[3]) \oplus Z$; \\ $Z = z_{13}$
99: $S[1] = S[1] \oplus Z$;
100: $S[2] = S[2] \oplus Z$;
101: $S[3] = S[3] \oplus Z$;
102: $S[5] = S[5] \oplus Z$;
103: $S[6] = S[6] \oplus Z$;
104: $S[7] = S[7] \oplus Z$;
105: $QAND^{\dagger}(T[1], U[3], Z)$;
106: $T[1] = T[1] \oplus T[3]$;\\ $T[1] = t_{44}$
107: $T[0] = T[0] \oplus T[2]$;\\ $T[0] = t_{46}$
108: $Z = (T[0] \cdot U[1]) \oplus Z$; \\ $Z = z_{12}$
109: $S[2] = S[2] \oplus Z$;
110: $S[3] = S[3] \oplus Z$;
111: $S[4] = S[4] \oplus Z$;
112: $S[5] = S[5] \oplus Z$;
113: $S[7] = S[7] \oplus Z$;
114: $QAND^{\dagger}(T[0], U[1], Z)$;
115: $T[0] = T[0] \oplus T[2]$;\\ $T[0] = t_{33}$
116: $U[3] = U[3] \oplus U[4] \oplus U[6]$;\\ $U[3] = y_1$
117: $Z = (T[0] \cdot U[3]) \oplus Z$; \\ $Z = z_{11}$
118: $S[1] = S[1] \oplus Z$;
119: $S[2] = S[2] \oplus Z$;
120: $S[5] = S[5] \oplus Z$;
121: $S[6] = S[6] \oplus Z$;
122: $QAND^{\dagger}(T[0], U[3], Z)$;
123: $U[3] = U[3] \oplus U[4] \oplus U[6]$;\\ $U[3] = y_{12}$
124: $T[0] = T[0] \oplus T[1]$;\\ $T[0] = t_{49}$
125: $Z = (T[0] \cdot U[4]) \oplus Z$; \\ $Z = z_{10}$
126: $S[0] = S[0] \oplus Z$;
127: $S[1] = S[1] \oplus Z$;
128: $S[2] = S[2] \oplus Z$;
129: $S[5] = S[5] \oplus Z$;
130: $S[6] = S[6] \oplus Z$;
131: $QAND^{\dagger}(T[0], U[4], Z)$;
132: $T[0] = T[0] \oplus T[1]$;\\ $T[0] = t_{33}$
133: $U[3] = U[3] \oplus U[6]$;\\ $U[3] = y_0$
134: $S[0] = (T[1] \cdot U[3]) \oplus S[0]$;\\ $S[0] = S[0] \oplus z_9$
135: $U[3] = U[3] \oplus U[6]$;\\ $U[3] = y_{12}$
136: $Z = (T[2] \cdot U[5]) \oplus Z$; \\ $Z = z_8$
```

```
137: $S[1] = S[1] \oplus Z$;
138: $S[2] = S[2] \oplus Z$;
139: $QAND^{\dagger}(T[2], U[5], Z)$;
140: $T[2] = T[2] \oplus T[3]$;\\ $T[2] = t_{45}$
141: $U[0] = U[0] \oplus U[2] \oplus U[7]$;\\ $U[0] = y_{16}$
142: $Z = (T[2] \cdot U[0]) \oplus Z$; \\ $Z = z_7$
143: $S[2] = S[2] \oplus Z$;
144: $S[5] = S[5] \oplus Z$;
145: $S[7] = S[7] \oplus Z$;
146: $QAND^{\dagger}(T[2], U[0], Z)$;
147: $T[2] = T[2] \oplus T[3]$;\\ $T[2] = t_{37}$
148: $U[0] = U[0] \oplus U[2] \oplus U[7]$;\\ $U[0] = y_9$
149: $U[0] = U[0] \oplus U[2] \oplus U[5] \oplus U[7]$;\\ $U[0] = y_6$
150: $Z = (T[3] \cdot U[0]) \oplus Z$; \\ $Z = z_6$
151: $S[1] = S[1] \oplus Z$;
152: $S[5] = S[5] \oplus Z$;
153: $S[7] = S[7] \oplus Z$;
154: $QAND^{\dagger}(T[3], U[0], Z)$;
155: $U[0] = U[0] \oplus U[2] \oplus U[5] \oplus U[7]$;\\ $U[0] = y_9$
156: $T[2] = T[2] \oplus T[0] \oplus T[1] \oplus T[3]$;\\ $T[2] = t_{48}$
157: $U[7] = U[7] \oplus U[2]$;\\ $U[7] = y_{11}$
158: $S[1] = (T[2] \cdot U[7]) \oplus S[1]$;\\ $S[1] = S[1] \oplus z_5$
159: $U[7] = U[7] \oplus U[2]$;\\ $U[7] = y_{14}$
160: $T[2] = T[2] \oplus T[0] \oplus T[1] \oplus T[3]$;\\ $T[2] = t_{37}$
161: $T[1] = T[1] \oplus T[3]$;\\ $T[1] = t_{47}$
162: $Z = (T[1] \cdot U[2]) \oplus Z$; \\ $Z = z_4$
163: $S[2] = S[2] \oplus Z$;
164: $S[3] = S[3] \oplus Z$;
165: $S[6] = S[6] \oplus Z$;
166: $S[7] = S[7] \oplus Z$;
167: $QAND^{\dagger}(T[1], U[2], Z)$;
168: $T[1] = T[1] \oplus T[3]$;\\ $T[1] = t_{44}$
169: $T[0] = T[0] \oplus T[2]$;\\ $T[0] = t_{46}$
170: $Z = (T[0] \cdot U[7]) \oplus Z$; \\ $Z = z_3$
171: $S[1] = S[1] \oplus Z$;
172: $S[2] = S[2] \oplus Z$;
173: $S[3] = S[3] \oplus Z$;
174: $S[6] = S[6] \oplus Z$;
175: $S[7] = S[7] \oplus Z$;
176: $QAND^{\dagger}(T[0], U[7], Z)$;
177: $T[0] = T[0] \oplus T[2]$;\\ $T[0] = t_{33}$
178: $U[7] = U[7] \oplus U[5]$;\\ $U[7] = y_5$
179: $Z = (T[0] \cdot U[7]) \oplus Z$; \\ $Z = z_2$
180: $S[1] = S[1] \oplus Z$;
181: $S[3] = S[3] \oplus Z$;
182: $S[6] = S[6] \oplus Z$;
183: $S[7] = S[7] \oplus Z$;
184: $QAND^{\dagger}(T[0], U[7], Z)$;
185: $U[7] = U[7] \oplus U[5]$;\\ $U[7] = y_{14}$
186: $T[0] = T[0] \oplus T[1]$;\\ $T[0] = t_{49}$
187: $Z = (T[0] \cdot U[0]) \oplus Z$; \\ $Z = z_1$
188: $S[1] = S[1] \oplus Z$;
189: $S[3] = S[3] \oplus Z$;
190: $S[5] = S[5] \oplus Z$;
191: $S[6] = S[6] \oplus Z$;
192: $QAND^{\dagger}(T[0], U[0], Z)$;
193: $T[0] = T[0] \oplus T[1]$;\\ $T[0] = t_{33}$
194: $U[7] = U[7] \oplus U[0] \oplus U[5]$;\\ $U[7] = y_4$
195: $Z = (T[1] \cdot U[7]) \oplus Z$; \\ $Z = z_0$
196: $S[5] = S[5] \oplus Z$;
197: $S[7] = S[7] \oplus Z$;
198: $QAND^{\dagger}(T[1], U[7], Z)$;
199: $U[7] = U[7] \oplus U[0] \oplus U[5]$;\\ $U[7] = y_{14}$
200: $S[0] = \overline{S[0]}$; $S[1] = \overline{S[1]}$; $S[3] = \overline{S[3]}$; $S[6] = \overline{S[6]}$; $S[7] = \overline{S[7]}$;
201: $T[2] = T[2] \oplus T[5]$; \\ $T[2] = t_{37}$
202: $T[2] = (T[4] \cdot T[5]) \oplus T[2]$; \\ $T[2] = t_{31}$
203: $T[1] = (T[0] \cdot T[2]) \oplus T[1]$; \\ $T[1] = t_{44}$
204: $T[2] = (T[4] \cdot T[5]) \oplus T[2]$; \\ $T[2] = t_{42}$
205: $T[2] = T[2] \oplus T[5]$; \\ $T[2] = t_{31}$
206: $T[3] = (T[4] \cdot T[5]) \oplus T[3]$; \\ $T[3] = t_{41}$
207: $T[3] = T[3] \oplus T[2]$; \\ $T[3] = t_{38}$
208: $T[5] = T[5] \oplus T[2]$; \\ $T[5] = t_{39}$
209: $T[5] = T[5] \oplus T[4]$; \\ $T[5] = t_{31}$
210: $T[0] = T[0] \oplus T[5]$; \\ $T[0] = t_{33}$
211: $T[5] = T[5] \oplus T[4]$; \\ $T[5] = t_{30}$
212: $T[0] = (T[1] \cdot T[5]) \oplus T[0]$; \\ $T[0] = t_{35} \oplus t_{32}$
213: $T[3] = T[3] \oplus T[4]$; \\ $T[3] = t_{28}$
214: $T[2] = (T[0] \cdot T[3]) \oplus T[2]$; \\ $T[2] = t_{37}$
215: $T[5] = T[5] \oplus T[2]$; \\ $T[5] = t_{31}$
216: $T[0] = T[0] \oplus T[5]$; \\ $T[0] = t_{35}$
217: $T[3] = T[3] \oplus T[2]$; \\ $T[3] = t_{34}$
218: $T[1] = T[1] \oplus T[0]$; \\ $T[1] = t_{29}$
219: $T[4] = T[4] \oplus T[2]$; \\ $T[4] = t_{26}$
220: $QAND^{\dagger}(T[1], T[3], T[5])$; \\ $T[5] = t_{30}$
221: $T[1] = (U[1] \cdot U[7]) \oplus T[1]$; \\ $T[1] = t_{24}$
222: $T[0] = (U[1] \cdot U[7]) \oplus T[0]$; \\ $T[0] = t_{20}$
223: $U[7] = U[7] \oplus U[2]$; \\ $U[7] = y_{14}$
224: $U[1] = U[1] \oplus U[3]$; \\ $U[1] = x_1$
225: $T[1] = (U[1] \cdot U[7]) \oplus T[1]$; \\ $T[1] = t_{21} \oplus y_{19} \oplus t_8$
226: $T[0] = (U[2] \cdot U[3]) \oplus T[0]$; \\ $T[0] = t_{17} \oplus y_{20} \oplus t_7$
227: $U[7] = U[7] \oplus U[0] \oplus U[1] \oplus U[4]$; \\ $U[7] = y_{11}$
228: $U[1] = U[1] \oplus U[3]$; \\ $U[1] = y_{10}$
229: $U[3] = U[3] \oplus U[1] \oplus U[4] \oplus U[5] \oplus U[6]$; \\ $U[3] = y_{12}$
230: $U[2] = U[2] \oplus U[0] \oplus U[1] \oplus U[5] \oplus U[6]$; \\ $U[2] = y_{13}$
231: $T[1] = T[1] \oplus U[2]$; \\ $T[1] = t_{21} \oplus y_{19}$
232: $U[2] = U[2] \oplus U[0] \oplus U[3] \oplus U[4] \oplus U[7]$; \\ $U[2] = y_{19}$
233: $T[3] = T[3] \oplus U[7]$; \\ $T[3] = t_{28}$
234: $U[7] = U[7] \oplus U[3] \oplus U[6]$; \\ $U[7] = y_{17}$
235: $T[0] = T[0] \oplus U[2]$; \\ $T[0] = t_{17} \oplus y_{20}$
236: $U[2] = U[2] \oplus U[1] \oplus U[4] \oplus U[5]$; \\ $U[2] = y_{20}$
237: $T[2] = T[2] \oplus U[3]$; \\ $T[2] = t_{26}$
238: $U[3] = U[3] \oplus U[1] \oplus U[4] \oplus U[5]$; \\ $U[3] = y_{18}$
239: $T[1] = (U[7] \cdot U[6]) \oplus T[1]$; \\ $T[1] = t_{21}$
240: $T[3] = T[3] \oplus T[1]$; \\ $T[3] = t_{27}$
241: $T[0] = T[0] \oplus T[3]$; \\ $T[0] = t_{17}$
242: $T[2] = T[2] \oplus T[0]$; \\ $T[2] = t_{25}$
243: $T[1] = T[1] \oplus T[2]$; \\ $T[1] = t_{16}$
244: $T[0] = (U[7] \cdot U[6]) \oplus T[0]$; \\ $T[0] = t_{14}$
245: $T[1] = (U[0] \cdot U[4]) \oplus T[1]$; \\ $T[1] = t_{15}$
246: $T[0] = (U[2] \cdot U[3]) \oplus T[0]$; \\ $T[0] = t_{13}$
247: $U[6] = U[6] \oplus U[1] \oplus U[3] \oplus U[4]$; \\ $U[6] = y_2$
248: $U[7] = U[7] \oplus U[0] \oplus U[3] \oplus U[4]$; \\ $U[7] = y_6$
249: $U[4] = \overline{U[4] \oplus U[1] \oplus U[2] \oplus U[7]}$; \\ $U[4] = y_8$
250: $U[0] = U[0] \oplus U[2]$; \\ $U[0] = y_9$
251: $U[3] = \overline{U[3] \oplus U[1] \oplus U[2] \oplus U[4] \oplus U[7]}$; \\ $U[3] = y_0$
252: $U[2] = U[2] \oplus U[0] \oplus U[3] \oplus U[5] \oplus U[7]$; \\ $U[2] = y_4$
253: $U[1] = U[3] \oplus U[1]$; \\ $U[1] = x_1$
254: $T[1] = T[1] \oplus T[0]$; \\ $T[1] = t_{10}$
255: $QAND^{\dagger}(U[2], U[6], T[3])$; \\ $T[3] = t_5$
256: $QAND^{\dagger}(U[5], U[1], T[2])$; \\ $T[2] = t_4$
257: $QAND^{\dagger}(U[0], U[3], T[0])$; \\ $T[0] = t_{10}$
258: $U[5] = \overline{U[5] \oplus U[0] \oplus U[1] \oplus U[2] \oplus U[3] \oplus U[7]}$; \\ $U[5] =$
     $y_7$
259: $U[6] = \overline{U[6] \oplus U[2]}$; \\ $U[6] = y_{15}$
260: $U[2] = \overline{U[2] \oplus U[1] \oplus U[4] \oplus U[5] \oplus U[7]}$; \\ $U[2] = y_{16}$
261: $U[3] = \overline{U[3] \oplus U[0] \oplus U[5]}$; \\ $U[3] = y_1$
262: $U[0] = \overline{U[0]}$; \\ $U[0] = y_5$
  Output, $Z = 0$
  Output, $U[j] = x_j$, for $0 \le j \le 7$;
  Output, $T[j] = 0$, for $0 \le j \le 5$;
  Output, $S[j] = y_j \oplus s_j$, for $0 \le j \le 7$;
```

As shown in Algorithm 1, we can adopt the following strategies to reduce the width and depth cost of Algorithm 1. Firstly, we need to find an appropriate order to calculate t_i (for $0 \le t_i \le 68$) so as to optimize the Toffoli depth in Algorithm 1. Secondly, we find out that we can adopt QAND gates instead of Toffoli gates to compute some z_j

in Algorithm 1 when $S[i] = 0$ (for $0 \leq i \leq 7$). In other words, we can reduce the T depth, when some $S[i] = 0$. Thirdly, we can utilize $QAND^{\dagger}$ gates to reset Z to zero. Fourthly, as shown in Observation 2, z_5 (z_9) only appears in S_1 (S_0). As a result, we can store z_5 (z_9) in S_1 (S_0) without affecting the other output bits. At last, we can adopt some $QAND^{\dagger}$ gates in Algorithm 1 to uncompute quantum AND operations, because a $QAND^{\dagger}$ gate requires no T gates at all. To sum up, we can obtain the 8-bit output of SM4's S-box with seven ancilla qubits, 46 Toffoli gates, four QAND gates, 20 QAND†, 286 CNOT gates, and 19 NOT gates. We can rewrite the above cost as $338T$ gates, 680 CNOT gates, and $223\#1qCliff$ gates, while the T-depth of Algorithm 1 is 86. Anyone can verify the correctness of SM4's S-box via https://github.com/Asiacrypt2020submission370/SM4/blob/main/S-box.

5.2 A depth-efficient quantum circuit implementation for SM4's S-box

Different from Sect. 5.1, we propose a depth-efficient quantum circuit of SM4's S-box in this section. Our depth-efficient quantum circuit of SM4's S-box is similar to the depth-efficient quantum circuit of AES's S-box proposed in [21] (see in https://github.com/microsoft/grover-blocks). Jaques et al.'s depth-efficient quantum circuit of AES's S-box was based on Boyar and Peralta [8], while our quantum circuit is based on our classical circuit of SM4's S-box. In [21], Jaques et al. pointed out the circuit depth could be reduced by introducing ancilla qubits for storing each intermediate variable in Boyar et al.'s classical circuit, because they did not need to recompute each values again.

In this paper, we adopt some similar ideas to construct our depth-efficient quantum circuit of SM4's S-box. That is, we can adopt a quantum AND gate with T-depth 1 instead of Toffoli gate to reduce the T-depth. In the following, given a 8-bit input x, we propose a depth-efficient quantum circuit Algorithm 2 to compute $S(x)$ as follows. Firstly, the 8-bit input of Algorithm 2 satisfies $U[j] = x_j$ (for $0 \leq j \leq 7$). Secondly, we need to introduce 127 zero ancilla qubits to store the intermediate states in Algorithm 2. Since we store the intermediate states in ancilla qubits, according to observation 2, we can compute $S(x)$ by using some CNOT gates. Take $S[4]$ as an example (see step 102 in Algorithm 2). After computing and storing t_{51} and t_{53} in ancilla qubits $T[51]$ and $T[53]$, we can obtain s_4 by xoring $T[51]$ and $T[53]$. Even if $S[4] = 1$, we can still compute $S[4] = 1 \oplus s_4$ by computing $S[4] \oplus T[51] \oplus T[53]$. In other words, we have no restrictions on $S[j]$ (for $0 \leq j \leq 7$). In Algorithm 2, we assume the output of S-box $S[j] = y_j$ (for $0 \leq j \leq 7$), where the value of each y_j can be 0 or 1 (for $0 \leq j \leq 7$). Similar to Algorithm 1, Algorithm 2 can compute the output of SM4's S-box and XOR it onto the output bits when they are non-zero. Based on the above method, our depth-efficient quantum circuit of SM4's S-box can be implemented in an in-place way.

Algorithm 2 A new depth-efficient quantum circuit for SM4's S-box

```
    Input, $U[j] = x_j$, for $0 \le j \le 7$;
    Input, $Y[j] = 0$, for $0 \le j \le 20$;
    Input, $T[j] = 0$, for $0 \le j \le 71$;
    Input, $z[j] = 0$, for $0 \le j \le 17$;
    Input, $EX[j] = 0$, for $0 \le j \le 15$;
    Input, $S[j] = y_j$, for $0 \le j \le 7$;
1: $Y[12] = Y[12] \oplus U[2] \oplus U[6]$;
2: $Y[10] = Y[10] \oplus Y[12] \oplus U[1]$;
3: $T[0] = T[0] \oplus U[3] \oplus U[5]$;
4: $Y[1] = Y[1] \oplus T[0] \oplus U[0]$;
5: $Y[3] = Y[3] \oplus U[1] \oplus Y[1]$;
6: $Y[9] = \overline{Y[9]} \oplus Y[3] \oplus U[3]$;
7: $U[0] = \overline{U[0]}$;
8: $Y[4] = Y[4] \oplus U[0] \oplus Y[9]$;
9: $Y[19] = Y[19] \oplus T[0] \oplus U[6]$;
10: $T[1] = T[1] \oplus U[4] \oplus U[7]$;
11: $Y[8] = \overline{Y[8]} \oplus T[1] \oplus U[5]$;
12: $U[6] = \overline{U[6]}$;
13: $Y[0] = Y[0] \oplus Y[8] \oplus Y[1]$;
14: $Y[2] = Y[2] \oplus Y[0] \oplus Y[12]$;
15: $Y[15] = Y[15] \oplus Y[2] \oplus Y[3]$;
16: $Y[16] = Y[16] \oplus Y[15] \oplus U[6]$;
17: $Y[11] = Y[11] \oplus Y[16] \oplus Y[9]$;
18: $T[2] = T[2] \oplus Y[1] \oplus U[7]$;
19: $Y[7] = Y[7] \oplus T[2] \oplus Y[11]$;
20: $Y[6] = Y[6] \oplus Y[7] \oplus Y[16]$;
21: $Y[14] = Y[14] \oplus Y[7] \oplus U[0]$;
22: $Y[13] = Y[13] \oplus Y[14] \oplus Y[11]$;
23: $Y[18] = Y[18] \oplus Y[7] \oplus Y[3]$;
24: $T[3] = T[3] \oplus U[2] \oplus U[7]$;
25: $Y[20] = Y[20] \oplus T[3] \oplus T[0]$;
26: $QAND(Y[7], Y[3], T[4], EX[0])$;
27: $QAND(Y[15], Y[16], T[5], EX[1])$;
28: $QAND(Y[6], Y[2], T[6], EX[2])$;
29: $QAND(Y[13], Y[12], T[7], EX[3])$;
30: $QAND(Y[11], Y[10], T[8], EX[4])$;
31: $QAND(Y[14], U[1], T[9], EX[5])$;
32: $QAND(U[0], Y[1], T[10], EX[6])$;
33: $QAND(Y[9], Y[8], T[11], EX[7])$;
34: $QAND(Y[4], Y[0], T[12], EX[8])$;
35: $T[13] = T[13] \oplus T[12] \oplus T[10]$;
36: $T[14] = T[14] \oplus T[13] \oplus T[6]$;
37: $T[15] = T[15] \oplus T[11] \oplus T[10]$;
38: $T[16] = T[16] \oplus T[4] \oplus T[15]$;
39: $T[17] = T[17] \oplus T[5] \oplus T[14]$;
40: $T[18] = T[18] \oplus T[9] \oplus T[17]$;
41: $T[19] = T[19] \oplus T[7] \oplus T[18]$;
42: $T[20] = T[20] \oplus Y[20] \oplus T[19]$;
43: $T[21] = T[21] \oplus T[6] \oplus T[16]$;
44: $T[22] = T[22] \oplus T[9] \oplus T[21]$;
45: $T[23] = T[23] \oplus T[8] \oplus T[22]$;
46: $T[24] = T[24] \oplus Y[19] \oplus T[23]$;
47: $T[25] = T[25] \oplus T[4] \oplus T[14]$;
48: $T[26] = T[26] \oplus Y[18] \oplus T[25]$;
49: $T[27] = T[27] \oplus T[5] \oplus T[16]$;
50: $T[28] = T[28] \oplus U[6] \oplus T[27]$;
51: $T[29] = T[29] \oplus T[24] \oplus T[20]$;
52: $QAND(T[24], T[28], T[30], EX[9])$;
53: $T[31] = T[31] \oplus T[30] \oplus T[26]$;
54: $QAND(T[29], T[31], T[32], EX[10])$;
55: $T[33] = T[33] \oplus T[32] \oplus T[20]$;
56: $T[34] = T[34] \oplus T[28] \oplus T[26]$;
57: $T[35] = T[35] \oplus T[30] \oplus T[20]$;
58: $QAND(T[34], T[35], T[36], EX[11])$;
59: $T[37] = T[37] \oplus T[36] \oplus T[26]$;
60: $T[38] = T[38] \oplus T[37] \oplus T[28]$;
61: $T[39] = T[39] \oplus T[37] \oplus T[31]$;
62: $QAND(T[26], T[39], T[40], EX[12])$;
63: $T[41] = T[41] \oplus T[38] \oplus T[40]$;
64: $T[42] = T[42] \oplus T[40] \oplus T[31]$;
65: $QAND(T[33], T[42], T[43], EX[13])$;
66: $T[44] = T[44] \oplus T[29] \oplus T[43]$;
67: $T[45] = T[45] \oplus T[41] \oplus T[37]$;
68: $T[46] = T[46] \oplus T[33] \oplus T[37]$;
69: $T[47] = T[47] \oplus T[44] \oplus T[41]$;
70: $T[48] = T[48] \oplus T[47] \oplus T[46]$;
71: $T[49] = T[49] \oplus T[44] \oplus T[33]$;
72: $QAND(T[37], Y[3], z[17], EX[0])$;
73: $QAND(T[45], Y[15], z[16], EX[1])$;
74: $QAND(T[41], Y[2], z[15], EX[2])$;
75: $QAND(T[48], Y[10], z[14], EX[3])$;
76: $QAND(T[47], Y[12], z[13], EX[4])$;
77: $QAND(T[46], U[1], z[12], EX[5])$;
78: $QAND(T[33], Y[1], z[11], EX[6])$;
79: $QAND(T[49], Y[8], z[10], EX[7])$;
80: $QAND(T[44], Y[0], z[9], EX[8])$;
81: $QAND(T[37], Y[7], z[8], EX[9])$;
82: $QAND(T[45], U[6], z[7], EX[10])$;
83: $QAND(T[41], Y[6], z[6], EX[11])$;
84: $QAND(T[48], Y[11], z[5], EX[12])$;
85: $QAND(T[47], Y[13], z[4], EX[13])$;
86: $QAND(T[46], Y[15], z[3], EX[14])$;
87: $QAND(T[33], U[0], z[2], EX[15])$;
88: $QAND(T[49], Y[9], z[1], Y[5])$;
89: $QAND(T[44], Y[4], z[0], Y[17])$;
90: $T[50] = T[50] \oplus z[16] \oplus z[17]$;
91: $T[51] = T[51] \oplus z[16] \oplus z[15]$;
92: $T[52] = T[52] \oplus z[13] \oplus z[12]$;
93: $T[53] = T[53] \oplus z[14] \oplus z[12]$;
94: $T[54] = T[54] \oplus z[10] \oplus z[11]$;
95: $T[55] = T[55] \oplus z[10] \oplus z[9]$;
96: $T[56] = T[56] \oplus z[7] \oplus z[8]$;
97: $T[57] = T[57] \oplus z[7] \oplus z[6]$;
98: $T[58] = T[58] \oplus z[4] \oplus z[3]$;
99: $T[59] = T[59] \oplus z[5] \oplus z[3]$;
100: $T[60] = T[60] \oplus z[1] \oplus z[2]$;
101: $T[61] = T[61] \oplus z[1] \oplus z[0]$;
102: $S[4] = S[4] \oplus T[53] \oplus T[51]$;
103: $T[62] = \overline{T[62]} \oplus T[51] \oplus T[54]$;
104: $T[63] = \overline{T[60]}$;
105: $T[64] = T[64] \oplus T[63] \oplus T[58]$;
106: $T[65] = T[65] \oplus T[52] \oplus T[50]$;
107: $S[3] = S[3] \oplus T[65] \oplus T[64]$;
108: $T[66] = T[66] \oplus T[61] \oplus T[57]$;
109: $S[5] = S[5] \oplus T[66] \oplus T[62]$;
110: $S[7] = S[7] \oplus T[66] \oplus S[3]$;
111: $T[67] = T[67] \oplus T[62] \oplus T[56]$;
112: $S[2] = S[2] \oplus T[67] \oplus T[58]$;
113: $T[68] = T[68] \oplus T[64] \oplus S[4]$;
114: $S[6] = S[6] \oplus T[68] \oplus T[62]$;
115: $T[69] = \overline{T[69]} \oplus T[55] \oplus T[51]$;
116: $S[0] = \overline{T[69]}$;
117: $T[70] = T[70] \oplus T[59] \oplus T[57]$;
118: $T[71] = T[71] \oplus T[68] \oplus S[2]$;
119: $S[1] = S[1] \oplus T[71] \oplus T[70]$;
120: $T[71] = T[71] \oplus T[68] \oplus S[2]$;
121: $T[70] = T[70] \oplus T[59] \oplus T[57]$;
122: $T[69] = T[69] \oplus T[55] \oplus T[51]$;
```

```
123: $T[68] = T[68] \oplus T[64] \oplus S[4]$;
124: $T[67] = T[67] \oplus T[62] \oplus T[56]$;
125: $T[66] = T[66] \oplus T[61] \oplus T[57]$;
126: $T[65] = T[65] \oplus T[52] \oplus T[50]$;
127: $T[64] = T[64] \oplus T[63] \oplus T[58]$;
128: $T[63] = \overline{T[60]}$;
129: $T[62] = T[62] \oplus T[51] \oplus T[54]$;
130: $T[61] = T[61] \oplus z[1] \oplus z[0]$;
131: $T[60] = T[60] \oplus z[1] \oplus z[2]$;
132: $T[59] = T[59] \oplus z[5] \oplus z[3]$;
133: $T[58] = T[58] \oplus z[4] \oplus z[3]$;
134: $T[57] = T[57] \oplus z[7] \oplus z[6]$;
135: $T[56] = T[56] \oplus z[7] \oplus z[8]$;
136: $T[55] = T[55] \oplus z[10] \oplus z[9]$;
137: $T[54] = T[54] \oplus z[10] \oplus z[11]$;
138: $T[53] = T[53] \oplus z[14] \oplus z[12]$;
139: $T[52] = T[52] \oplus z[13] \oplus z[12]$;
140: $T[51] = T[51] \oplus z[16] \oplus z[15]$;
141: $T[50] = T[50] \oplus z[16] \oplus z[17]$;
142:   $QAND^{\dagger}(T[37], Y[3], z[17])$;
143:   $QAND^{\dagger}(T[45], Y[15], z[16])$;
144:   $QAND^{\dagger}(T[41], Y[2], z[15])$;
145:   $QAND^{\dagger}(T[48], Y[10], z[14])$;
146:   $QAND^{\dagger}(T[47], Y[12], z[13])$;
147:   $QAND^{\dagger}(T[46], U[1], z[12])$;
148:   $QAND^{\dagger}(T[33], Y[1], z[11])$;
149:   $QAND^{\dagger}(T[49], Y[8], z[10])$;
150:   $QAND^{\dagger}(T[44], Y[0], z[9])$;
151:   $QAND^{\dagger}(T[37], Y[7], z[8])$;
152:   $QAND^{\dagger}(T[45], U[6], z[7])$;
153:   $QAND^{\dagger}(T[41], Y[6], z[6])$;
154:   $QAND^{\dagger}(T[48], Y[11], z[5])$;
155:   $QAND^{\dagger}(T[47], Y[13], z[4])$;
156:   $QAND^{\dagger}(T[46], Y[15], z[3])$;
157:   $QAND^{\dagger}(T[33], U[0], z[2])$;
158:   $QAND^{\dagger}(T[49], Y[9], z[1])$;
159:   $QAND^{\dagger}(T[44], Y[4], z[0])$;
160: $T[49] = T[49] \oplus T[44] \oplus T[33]$;
161: $T[48] = T[48] \oplus T[47] \oplus T[46]$;
162: $T[47] = T[47] \oplus T[44] \oplus T[41]$;
163: $T[46] = T[46] \oplus T[33] \oplus T[37]$;
164: $T[45] = T[45] \oplus T[41] \oplus T[37]$;
165: $T[44] = T[44] \oplus T[29] \oplus T[43]$;
166:   $QAND^{\dagger}(T[33], T[42], T[43])$;
167: $T[42] = T[42] \oplus T[40] \oplus T[31]$;
168: $T[41] = T[41] \oplus T[38] \oplus T[40]$;
169:   $QAND^{\dagger}(T[26], T[39], T[40])$;
170: $T[39] = T[39] \oplus T[37] \oplus T[31]$;
171: $T[38] = T[38] \oplus T[37] \oplus T[28]$;
172: $T[37] = T[37] \oplus T[36] \oplus T[26]$;
173:   $QAND^{\dagger}(T[34], T[35], T[36])$;
174: $T[35] = T[35] \oplus T[30] \oplus T[20]$;
175: $T[34] = T[34] \oplus T[28] \oplus T[26]$;
176: $T[33] = T[33] \oplus T[32] \oplus T[20]$;
177:   $QAND^{\dagger}(T[29], T[31], T[32])$;
178: $T[31] = T[31] \oplus T[30] \oplus T[26]$;
179:   $QAND^{\dagger}(T[24], T[28], T[30])$;
180: $T[29] = T[29] \oplus T[24] \oplus T[20]$;
181: $T[28] = T[28] \oplus U[6] \oplus T[27]$;
182: $T[27] = T[27] \oplus T[5] \oplus T[16]$;
183: $T[26] = T[26] \oplus Y[18] \oplus T[25]$;
184: $T[25] = T[25] \oplus T[4] \oplus T[14]$;
185: $T[24] = T[24] \oplus Y[19] \oplus T[23]$;
186: $T[23] = T[23] \oplus T[8] \oplus T[22]$;
187: $T[22] = T[22] \oplus T[9] \oplus T[21]$;
188: $T[21] = T[21] \oplus T[6] \oplus T[16]$;
189: $T[20] = T[20] \oplus Y[20] \oplus T[19]$;
190: $T[19] = T[19] \oplus T[7] \oplus T[18]$;
191: $T[18] = T[18] \oplus T[9] \oplus T[17]$;
192: $T[17] = T[17] \oplus T[5] \oplus T[14]$;
193: $T[16] = T[16] \oplus T[4] \oplus T[15]$;
194: $T[15] = T[15] \oplus T[11] \oplus T[10]$;
195: $T[14] = T[14] \oplus T[13] \oplus T[6]$;
196: $T[13] = T[13] \oplus T[12] \oplus T[10]$;
197:   $QAND^{\dagger}(Y[4], Y[0], T[12])$;
198:   $QAND^{\dagger}(Y[9], Y[8], T[11])$;
199:   $QAND^{\dagger}(U[0], Y[1], T[10])$;
200:   $QAND^{\dagger}(Y[14], U[1], T[9])$;
201:   $QAND^{\dagger}(Y[11], Y[10], T[8])$;
202:   $QAND^{\dagger}(Y[13], Y[12], T[7])$;
203:   $QAND^{\dagger}(Y[6], Y[2], T[6])$;
204:   $QAND^{\dagger}(Y[15], Y[16], T[5])$;
205:   $QAND^{\dagger}(Y[7], Y[3], T[4])$;
206: $Y[20] = Y[20] \oplus T[3] \oplus T[0]$;
207: $T[3] = T[3] \oplus U[2] \oplus U[7]$;
208: $Y[18] = Y[18] \oplus Y[7] \oplus Y[3]$;
209: $Y[13] = Y[13] \oplus Y[14] \oplus Y[11]$;
210: $Y[14] = Y[14] \oplus Y[7] \oplus U[0]$;
211: $Y[6] = Y[6] \oplus Y[7] \oplus Y[16]$;
212: $Y[7] = Y[7] \oplus T[2] \oplus Y[11]$;
213: $T[2] = T[2] \oplus Y[1] \oplus U[7]$;
214: $Y[11] = Y[11] \oplus Y[16] \oplus Y[9]$;
215: $Y[16] = Y[16] \oplus Y[15] \oplus U[6]$;
216: $Y[15] = Y[15] \oplus Y[2] \oplus Y[3]$;
217: $Y[2] = Y[2] \oplus Y[0] \oplus Y[12]$;
218: $Y[0] = Y[0] \oplus Y[8] \oplus Y[1]$;
219: $U[6] = \overline{U[6]}$;
220: $Y[8] = Y[8] \oplus T[1] \oplus U[5]$;
221: $T[1] = T[1] \oplus U[4] \oplus U[7]$;
222: $Y[19] = Y[19] \oplus T[0] \oplus U[6]$;
223: $Y[4] = Y[4] \oplus U[0] \oplus Y[9]$;
224: $U[0] = \overline{U[0]}$;
225: $Y[9] = Y[9] \oplus Y[3] \oplus U[3]$;
226: $Y[3] = Y[3] \oplus U[1] \oplus Y[1]$;
227: $Y[1] = Y[1] \oplus T[0] \oplus U[0]$;
228: $T[0] = T[0] \oplus U[3] \oplus U[5]$;
229: $Y[10] = Y[10] \oplus Y[12] \oplus U[1]$;
230: $Y[12] = Y[12] \oplus U[2] \oplus U[6]$;
     Output, $U[j] = x_j$, for $0 \le j \le 7$;
     Output, $Y[j] = 0$, for $0 \le j \le 20$;
     Output, $T[j] = 0$, for $0 \le j \le 71$;
     Output, $z[j] = 0$, for $0 \le j \le 17$;
     Output, $EX[j] = 0$, for $0 \le j \le 15$;
     Output, $S[j] = s_j \oplus y_j$, for $0 \le j \le 7$;
```

As shown in Algorithm 2, given a 8-bit input x, we can adopt Algorithm 2 to compute the $S(x)$. We can express the above circuit of SM4's S-box as $|x\rangle^8|y\rangle^8|0\rangle^{127} \xrightarrow{Alg.\ 2} |x\rangle^8|S(x) \oplus y\rangle^8|0\rangle^{127}$. Different from our memory-efficient quantum circuit of SM4's S-box, we shall introduce more zero ancilla qubits to store the intermediate state, which can reduce the T-depth. We write some steps in Algorithm 2 like $T[i] = T[i] \oplus T[j] \oplus z[k]$, which can be implemented with two CNOT gates as follows. The first CNOT gate has $T[j]$ as its control, while the other has $z[k]$ as its control. Note that the $T[i]$ is the target. The T-depth of Algorithm 2 is 7, because we can compute some QAND gates in parallel as follows. Firstly, we can compute $T[i]$ (for $4 \leq i \leq 12$) in parallel. Secondly, we can compute $T[30]$, $T[32]$, $T[36]$, $T[40]$ and $T[43]$ in a serial mode. Thirdly, we can compute all $z[i]$ (for $0 \leq i \leq 17$) in parallel. To sum up, the width and depth cost of Algorithm 2 is 16 qubits, 127 ancilla qubits, 128 T gates, 638 CNOT gates, and $264\#1qCliff$ gates. Compared with our memory-efficient quantum circuit of SM4's S-box, our depth-efficient quantum circuit of SM4's S-box can reduce the T-depth greatly.

6 Quantum circuit implementations of SM4

In this section, we propose some memory-efficient and depth-efficient quantum circuit implementations of SM4 block cipher.

6.1 A memory-efficient quantum circuit implementation of SM4

In this subsection, we propose a memory-efficient quantum circuit of SM4, which requires only several hundred qubits. As shown in [19], the qubits number in a real quantum computer increases very slowly. In other words, the number of qubits is an important parameter in designing a quantum circuit. As a result, our memory-efficient quantum circuit of SM4 can be implemented on a real quantum computer as soon as possible. We present the details of our memory-efficient quantum circuit of SM4 as follows.

In order to construct our memory-efficient quantum circuit of SM4, we need to construct our quantum circuit of the round function F of SM4 in the first place. As shown in Sect. 3, the permutation T_{SM4} consists of the nonlinear transformation τ and the linear transformation L. We observe that τ can be implemented with our quantum circuit of SM4's S-box, while L and L^{-1} can be implemented with the method proposed by Xiang et al. [41]. The details of our in-place implementation of L can be explained in Algorithm 3. Given a 32-bit input $x[i]$ (for $0 \leq i \leq 31$), it requires 83 CNOT gates to compute $L(x)$. In addition, L^{-1} requires the same cost as L, which is also pointed out in [41].

In the following, we can rewrite F as $F(X_i, X_{i+1}, X_{i+2}, X_{i+3}) = L(L^{-1}(X_i) \oplus \tau(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i))$. By using this strategy, we do not need to introduce some new ancilla qubits to store the output of τ. Since we do not need to compute τ twice to remove the output of τ, we can reduce the T-depth and qubits number simultaneously. To sum up, the cost of the round function F can be computed as follows. Firstly, we need 96 CNOT gates to obtain $X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i$, which is the input to τ. Secondly, we shall adopt our memory-efficient quantum circuit of SM4's S-box (in Sect. 5.1) four times to compute the output of τ, which requires 28 ancilla qubits, $338 \times 4 = 1352$ T gates, $680 \times 4 = 2720$ CNOT gates, and $223 \times 4 = 892\ \#1qCliff$. Since we can implement the 4 S-box in τ in parallel, the T-depth of τ is 86. However, if the ancilla qubits are limited to 7, the 4 S-box operations shall be implemented in a serial mode. That is, the T depth of τ is increased to 344 in this case.

After computing the output to τ, we need to compute $L^{-1}(X_i)$ and $L^{-1}(X_i) \oplus \tau(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i)$, which requires 83 CNOT gates and 32 CNOT gates, respectively. Given the value of $L^{-1}(X_i) \oplus \tau(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i)$, we still need to 83 CNOT gates to compute $L(L^{-1}(X_i) \oplus \tau(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i))$. Since we can store $X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i$ in the qubits of X_{i+1}, we shall recompute X_{i+1} by computing $X_{i+1} = X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i$ with 96 CNOT gates.

Algorithm 3 A new in-place implementation of L in SM4.

```
Input, $x[i]$, for $0 \le i \le 31$;
1: $x[30] = x[30] \oplus x[22]$;
2: $x[20] = x[20] \oplus x[4]$;
3: $x[22] = x[22] \oplus x[6]$;
4: $x[11] = x[11] \oplus x[27]$;
5: $x[27] = x[27] \oplus x[3]$;
6: $x[4] = x[4] \oplus x[28]$;
7: $x[15] = x[15] \oplus x[7]$;
8: $x[29] = x[29] \oplus x[5]$;
9: $x[26] = x[26] \oplus x[10]$;
10: $x[24] = x[24] \oplus x[0]$;
11: $x[7] = x[7] \oplus x[23]$;
12: $x[10] = x[10] \oplus x[18]$;
13: $x[0] = x[0] \oplus x[8]$;
14: $x[13] = x[13] \oplus x[21]$;
15: $x[9] = x[9] \oplus x[25]$;
16: $x[25] = x[25] \oplus x[17]$;
17: $x[8] = x[8] \oplus x[16]$;
18: $x[16] = x[16] \oplus x[10]$;
19: $x[10] = x[10] \oplus x[20]$;
20: $x[17] = x[17] \oplus x[1]$;
21: $x[20] = x[20] \oplus x[12]$;
22: $x[1] = x[1] \oplus x[27]$;
23: $x[27] = x[27] \oplus x[5]$;
24: $x[5] = x[5] \oplus x[23]$;
25: $x[12] = x[12] \oplus x[6]$;
26: $x[6] = x[6] \oplus x[30]$;
27: $x[28] = x[28] \oplus x[20]$;
28: $x[5] = x[5] \oplus x[31]$;
29: $x[31] = x[31] \oplus x[25]$;
30: $x[18] = x[18] \oplus x[2]$;
31: $x[2] = x[2] \oplus x[4]$;
32: $x[23] = x[23] \oplus x[25]$;
33: $x[4] = x[4] \oplus x[22]$;
34: $x[20] = x[20] \oplus x[30]$; \\ $y[22] = x[20]$;
35: $x[25] = x[25] \oplus x[11]$;
36: $x[5] = x[5] \oplus x[13]$; \\ $y[23] = x[5]$;
37: $x[22] = x[22] \oplus x[24]$;
38: $x[27] = x[27] \oplus x[21]$;
39: $x[11] = x[11] \oplus x[19]$;
40: $x[12] = x[12] \oplus x[14]$;
41: $x[19] = x[19] \oplus x[13]$;
42: $x[30] = x[30] \oplus x[14]$;
43: $x[21] = x[21] \oplus x[29]$;
44: $x[3] = x[3] \oplus x[11]$;
45: $x[13] = x[13] \oplus x[7]$;
46: $x[11] = x[11] \oplus x[29]$; \\ $y[29] = x[11]$;
47: $x[29] = x[29] \oplus x[7]$;
48: $x[7] = x[7] \oplus x[9]$;
49: $x[24] = x[24] \oplus x[16]$;
50: $x[14] = x[14] \oplus x[22]$; \\ $y[24] = x[14]$;
51: $x[30] = x[30] \oplus x[0]$; \\ $y[0] = x[30]$;
52: $x[9] = x[9] \oplus x[1]$;
53: $x[22] = x[22] \oplus x[8]$;
54: $x[6] = x[6] \oplus x[8]$; \\ $y[8] = x[6]$;
55: $x[7] = x[7] \oplus x[31]$; \\ $y[9] = x[7]$;
56: $x[0] = x[0] \oplus x[26]$;
57: $x[21] = x[21] \oplus x[15]$; \\ $y[7] = x[21]$;
58: $x[8] = x[8] \oplus x[24]$; \\ $y[10] = x[8]$;
59: $x[31] = x[31] \oplus x[15]$; \\ $y[17] = x[31]$;
60: $x[24] = x[24] \oplus x[18]$; \\ $y[2] = x[24]$;
61: $x[26] = x[26] \oplus x[2]$;
62: $x[18] = x[18] \oplus x[26]$; \\ $y[28] = x[18]$;
63: $x[26] = x[26] \oplus x[28]$; \\ $y[12] = x[26]$;
64: $x[1] = x[1] \oplus x[25]$; \\ $y[3] = x[1]$;
65: $x[28] = x[28] \oplus x[12]$; \\ $y[6] = x[28]$;
66: $x[12] = x[12] \oplus x[4]$; \\ $y[14] = x[12]$;
67: $x[15] = x[15] \oplus x[17]$;
68: $x[17] = x[17] \oplus x[9]$; \\ $y[27] = x[17]$;
69: $x[9] = x[9] \oplus x[3]$; \\ $y[11] = x[9]$;
70: $x[16] = x[16] \oplus x[0]$; \\ $y[18] = x[16]$;
71: $x[23] = x[23] \oplus x[15]$; \\ $y[25] = x[23]$;
72: $x[3] = x[3] \oplus x[19]$; \\ $y[13] = x[3]$;
73: $x[2] = x[2] \oplus x[10]$; \\ $y[20] = x[2]$;
74: $x[15] = x[15] \oplus x[7]$; \\ $y[1] = x[15]$;
75: $x[13] = x[13] \oplus x[21]$; \\ $y[15] = x[13]$;
76: $x[19] = x[19] \oplus x[27]$; \\ $y[5] = x[19]$;
77: $x[27] = x[27] \oplus x[11]$; \\ $y[21] = x[27]$;
78: $x[10] = x[10] \oplus x[26]$; \\ $y[4] = x[10]$;
79: $x[0] = x[0] \oplus x[24]$; \\ $y[26] = x[0]$;
80: $x[4] = x[4] \oplus x[20]$; \\ $y[30] = x[4]$;
81: $x[22] = x[22] \oplus x[30]$; \\ $y[16] = x[22]$;
82: $x[25] = x[25] \oplus x[9]$; \\ $y[19] = x[25]$;
83: $x[29] = x[29] \oplus x[5]$; \\ $y[31] = x[29]$;
Output, $x[i]$ for $0 \le i \le 31$;
```

After implementing F, we can implement the key schedule of SM4 in a similar way. Firstly, we can rewrite rk_i as $L'(L'^{-1}(K_i) \oplus \tau(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i))$, which does not need to introduce new ancilla qubits to store τ. Secondly, we propose a new in-place implementation of L' by using the method proposed by Xiang et al. [41]. Our new in-place implementation of L' can be explained in Algorithm 4. Given a 32-bit input $x[i]$ (for $0 \leq i \leq 31$), it requires 78 CNOT gates to compute $L'(x)$. Note that L'^{-1} require the same cost of L'. Since CK_i is a constant, we only require some NOT gates to implement the Constant-XOR operation of $\oplus CK_i$. That is, we just need 64 CNOT gates and some NOT gates to obtain the input $K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i)$ to τ. Since the hamming weight of all CK_i (for $0 \leq i \leq 31$) is 567, we just require 567 NOT gates to compute $\oplus CK_i$ (for $0 \leq i \leq 31$) in the key schedule of SM4. Similar to F, after obtaining the input $K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i$ of τ, we can compute the output of τ with 86 T-depth, 28 ancilla qubits, 1352 T gates, 2720 CNOT gates, and 892 $\#1qCliff$. In addition, we can compute $L'^{-1}(K_i)$ and $L'^{-1}(K_i) \oplus \tau(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i)$ with 78 CNOT gates and 32 CNOT gates, respectively. Given the value of $L'^{-1}(K_i) \oplus \tau(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i)$, we still need 78 CNOT gates to compute $L'(L'^{-1}(K_i) \oplus \tau(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i))$. Since we store $K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i$ in the qubits of K_{i+3}, we shall recompute K_{i+3} by compute $K_{i+3} = K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i$ again, which requires some CNOT gates and NOT gates.

Algorithm 4 A new in-place implementation of L' in SM4.

```
Input, $x[i]$ for $0 \leq i \leq 31$;
1: $x[22] = x[22] \oplus x[0]$;
2: $x[3] = x[3] \oplus x[22]$;
3: $x[15] = x[15] \oplus x[28]$;
4: $x[16] = x[16] \oplus x[3]$;
5: $x[28] = x[28] \oplus x[6]$;
6: $x[6] = x[6] \oplus x[12]$;
7: $x[9] = x[9] \oplus x[0]$;
8: $x[0] = x[0] \oplus x[10]$;
9: $x[29] = x[29] \oplus x[20]$;
10: $x[28] = x[28] \oplus x[19]$; \\ $y[19] = x[28]$;
11: $x[29] = x[29] \oplus x[19]$;
12: $x[2] = x[2] \oplus x[21]$;
13: $x[9] = x[9] \oplus x[19]$; \\ $y[0] = x[9]$;
14: $x[10] = x[10] \oplus x[20]$;
15: $x[21] = x[21] \oplus x[31]$;
16: $x[21] = x[21] \oplus x[12]$; \\ $y[12] = x[21]$;
17: $x[12] = x[12] \oplus x[20]$;
18: $x[20] = x[20] \oplus x[30]$;
19: $x[30] = x[30] \oplus x[17]$;
20: $x[30] = x[30] \oplus x[7]$; \\ $y[30] = x[30]$;
21: $x[7] = x[7] \oplus x[16]$;
22: $x[16] = x[16] \oplus x[19]$;
23: $x[19] = x[19] \oplus x[25]$;
24: $x[25] = x[25] \oplus x[31]$;
25: $x[31] = x[31] \oplus x[18]$;
26: $x[31] = x[31] \oplus x[8]$; \\ $y[31] = x[31]$;
27: $x[8] = x[8] \oplus x[17]$;
28: $x[17] = x[17] \oplus x[26]$;
29: $x[26] = x[26] \oplus x[3]$;
30: $x[17] = x[17] \oplus x[4]$; \\ $y[17] = x[17]$;
31: $x[3] = x[3] \oplus x[12]$;
32: $x[8] = x[8] \oplus x[27]$; \\ $y[8] = x[8]$;
33: $x[12] = x[12] \oplus x[18]$;
34: $x[18] = x[18] \oplus x[27]$;
35: $x[27] = x[27] \oplus x[14]$;
36: $x[3] = x[3] \oplus x[6]$;
37: $x[27] = x[27] \oplus x[4]$; \\ $y[27] = x[27]$;
38: $x[4] = x[4] \oplus x[23]$;
39: $x[4] = x[4] \oplus x[13]$; \\ $y[4] = x[4]$;
40: $x[18] = x[18] \oplus x[5]$; \\ $y[18] = x[18]$;
41: $x[13] = x[13] \oplus x[22]$; \\ $y[13] = x[13]$;
42: $x[22] = x[22] \oplus x[19]$;
43: $x[19] = x[19] \oplus x[15]$;
44: $x[15] = x[15] \oplus x[5]$; \\ $y[28] = x[15]$;
45: $x[5] = x[5] \oplus x[14]$;
46: $x[14] = x[14] \oplus x[23]$;
47: $x[14] = x[14] \oplus x[1]$; \\ $y[14] = x[14]$;
48: $x[23] = x[23] \oplus x[0]$; \\ $y[23] = x[23]$;
49: $x[0] = x[0] \oplus x[6]$;
50: $x[5] = x[5] \oplus x[24]$; \\ $y[5] = x[5]$;
51: $x[6] = x[6] \oplus x[24]$;
52: $x[24] = x[24] \oplus x[1]$;
53: $x[1] = x[1] \oplus x[10]$; \\ $y[1] = x[1]$;
54: $x[0] = x[0] \oplus x[10]$;
55: $x[10] = x[10] \oplus x[29]$; \\ $y[10] = x[10]$;
56: $x[29] = x[29] \oplus x[16]$;
57: $x[24] = x[24] \oplus x[11]$; \\ $y[24] = x[24]$;
58: $x[16] = x[16] \oplus x[22]$; \\ $y[16] = x[16]$;
59: $x[22] = x[22] \oplus x[25]$;
60: $x[25] = x[25] \oplus x[2]$;
61: $x[2] = x[2] \oplus x[11]$; \\ $y[2] = x[2]$;
62: $x[11] = x[11] \oplus x[20]$; \\ $y[11] = x[11]$;
63: $x[20] = x[20] \oplus x[21]$;
64: $x[20] = x[20] \oplus x[12]$;
65: $x[12] = x[12] \oplus x[9]$;
66: $x[12] = x[12] \oplus x[28]$;
67: $x[25] = x[25] \oplus x[21]$; \\ $y[25] = x[25]$;
68: $x[12] = x[12] \oplus x[0]$; \\ $y[9] = x[12]$;
69: $x[0] = x[0] \oplus x[3]$; \\ $y[3] = x[0]$;
70: $x[3] = x[3] \oplus x[29]$; \\ $y[29] = x[3]$;
71: $x[29] = x[29] \oplus x[7]$; \\ $y[20] = x[29]$;
72: $x[22] = x[22] \oplus x[9]$; \\ $y[22] = x[22]$;
73: $x[7] = x[7] \oplus x[26]$; \\ $y[7] = x[7]$;
74: $x[19] = x[19] \oplus x[28]$; \\ $y[6] = x[19]$;
75: $x[26] = x[26] \oplus x[13]$; \\ $y[26] = x[26]$;
76: $x[6] = x[6] \oplus x[19]$;
77: $x[6] = x[6] \oplus x[25]$; \\ $y[15] = x[6]$;
78: $x[20] = x[20] \oplus x[31]$; \\ $y[21] = x[20]$;
Output, $x[i]$ for $0 \leq i \leq 31$;
```

Given our quantum circuit of the key schedule and the round function F of SM4, we can construct a low-memory quantum circuit of SM4 as follows. In the first place, we show the time and memory cost of Round 1 as follows (see in Fig. 4). Firstly, we can generate K_0, K_1, K_2 and K_3 by computing $(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3)$. Here, (FK_0, FK_1, FK_2, FK_3) are constant, which contains 64 bits "1." In other words, we can obtain (K_0, K_1, K_2, K_3) with 64 NOT gates. Secondly, given the value of (K_0, K_1, K_2, K_3), we can compute $K_3 = K_3 \oplus K_1 \oplus K_2 \oplus CK_0$ so as to obtain the input to the nonlinear operation τ. Since CK_0 contains 9 bits "1," this operation requires 64 CNOT gates and 9 NOT gates. Note that we can compute and store K_4 in the qubits for K_0, because K_0 is only used in Round 0. According to Algorithm 1, we can compute the $\tau(K_3)$ with 86 T-depth, 28 ancilla qubits, 1352 T gates, 2720 CNOT gates, and 892 $\#1qCliff$. Thirdly, given the value of $\tau(K_3)$, we can compute $K_4 = L'(L'^{-1}(K_0) \oplus \tau(K_3))$ with 156+32=188 CNOT gates. Here, L' and L'^{-1} can be implemented with 156 CNOT gates, while the XOR operation requires 32 CNOT gates. After computing K_4 (or called rk_0), we still need 64 CNOT gates and 9 NOT gates to recompute K_3 with $K_3 = K_3 \oplus K_1 \oplus K_2 \oplus CK_0$. To sum up, we need 86 T-depth, 28 ancilla qubits, 1352 T gates, $64+2720+188+64 = 3036$ CNOT gates, and $64+9+892+9 = 974$ $\#1qCliff$ to compute K_4.

After generating K_4, we can generate X_4 with the round function F of SM4 as follows. Firstly, we need to compute $X_1 = X_1 \oplus X_2 \oplus X_3 \oplus K_4$ with $32 \times 3 = 96$ CNOT gates, which is the input to τ. Secondly, we shall adopt Algorithm 1 to compute $\tau(X_1)$, which requires 86 T-depth, 28 ancilla qubits, 1352 T gates, 2720 CNOT gates, and 892 $\#1qCliff$. Thirdly, after computing $\tau(X_1)$, we can compute $X_4 = L(L^{-1}(X_0) \oplus \tau(X_1))$ with 198 CNOT gates. Here, L and L^{-1} can be implemented with 166 CNOT gates, while the XOR operation requires 32 CNOT gates. At last, we need to regenerate X_1 by calculating $X_1 = X_1 \oplus X_2 \oplus X_3 \oplus K_4$ with $32 \times 3 = 96$ CNOT gates. That is, we need 86 T-depth, 28 ancilla qubits, 1352 T gates, $96 + 2720 + 198 + 96 = 3110$ CNOT gates, and 892 $\#1qCliff$. Note that we just need to compute K_4 and X_4 in Round 1. The quantum resource of Round 1 in SM4 is 28 ancilla qubits, 86+86=172 T-depth, $1352 \times 2 = 2704$ T gates, $3036 + 3110 = 6146$ CNOT gates, and $974 + 892 = 1866$ $\#1qCliff$ gates. When the number of ancilla qubits is limited to 7, we shall implement the four Sbox in τ in a serial mode. As a result, the time and memory cost of Round 1 is $86 \times 8 = 688$ T-depth, 2704 T gates, 6146 CNOT gates, and 1866 $\#1qCliff$ gates in this case.

After implementing Round 1, we can compute the time and memory cost of the left 31 rounds in a similar way. Due to the space limitation, we just show the result and omit the details. As shown in Fig. 4, our memory-efficient quantum circuit of SM4 requires 256 qubits, 28 ancilla qubits, $172 \times 32 = 5504$ T-depth, $2704 \times 32 = 86528$ T gates, $5146 \times 32 = 196672$ CNOT gates, and 58158 $\#1qCliff$ gates. Note that the constant values in different rounds are different, which means the number of NOT gates in different rounds are different. When the number of ancilla qubits is limited to 7, the above memory-efficient quantum circuit of SM4 requires 256 qubits, seven ancilla qubits, $5504 \times 4 = 22016$ T-depth, 86528 T gates, 196672 CNOT gates, and 58158 $\#1qCliff$ gates. We call the above quantum circuit MEQ_{SM4} in the following.

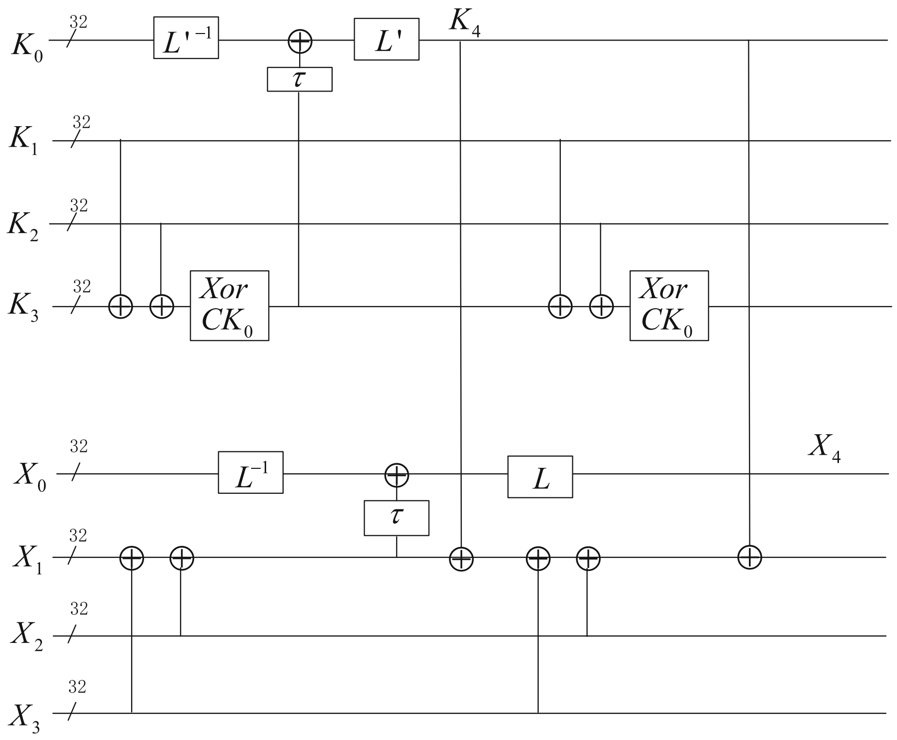


Fig. 4 Implementation of the first step function of SM4

After obtaining the output ciphertext of SM4, the qubits for storing the round keys of SM4 contain the value of $(rk_{28}, rk_{29}, rk_{30}, rk_{31})$. According to the property of quantum circuit, we shall restore the original input key. As a result, the above circuit MEQ_{SM4} is not a stand-alone quantum circuit of SM4. However, we can apply MEQ_{SM4} as a subroutine in the Grover algorithm on SM4, because we can simply reverse the entire circuit to obtain the original input key (K_0, K_1, K_2, K_3) and input message (X_0, X_1, X_2, X_3).

If we want to design a stand-alone memory-efficient quantum circuit of SM4, we shall not only restore $(rk_{28}, rk_{29}, rk_{30}, rk_{31})$ to (MK_0, MK_1, MK_2, MK_3), but also need to keep the value of the input message (X_0, X_1, X_2, X_3) in our quantum circuit of SM4. In the first place, we shall introduce 128 new zero qubits for storing the 128-bit state (X_0, X_1, X_2, X_3). That is, we can compute $|X_0||X_1||X_2||X_3\rangle|0\rangle^{128} \xrightarrow{CNOT} |X_0||X_1||X_2||X_3\rangle|X_0||X_1||X_2||X_3\rangle$ with 128 CNOT gates. Then, we can adopt the above memory-efficient quantum circuit to compute the ciphertext of SM4 on these new calculated 128 qubits of $|X_0||X_1||X_2||X_3\rangle$, while the original input (X_0, X_1, X_2, X_3) are kept unchanged.

Apart from the above operation, we still need to design a new circuit MRK_{SM4} to reverse the key schedule of SM4. In the following, we will show how to restore rk_{27} in the first place. Then, we can restore the other round key in a similar way. According to SM4's key schedule of SM4, we can rewrite rk_{27} (or called K_{31}) as $rk_{27} = L'(L'^{-1}(rk_{31}) \oplus \tau(rk_{28} \oplus rk_{29} \oplus rk_{30}))$ with $(rk_{28}, rk_{29}, rk_{30}, rk_{31})$. In other words, we can compute and store rk_{27} in rk_{31}. Similar to rk_{27}, given $(rk_{27}, rk_{28}, rk_{29}, rk_{30})$, we can compute rk_{26} and store rk_{26} in rk_{30}. Obviously, we can repeat the above operation 32 times to obtain (K_0, K_1, K_2, K_3), where $K_j = rk_{j-4}$ for $0 \leq j \leq 31$. At last, given (K_0, K_1, K_2, K_3), we require 64 NOT gates to obtain (MK_0, MK_1, MK_2, MK_3) by $(MK_0, MK_1, MK_2, MK_3) = (K_0 \oplus FK_0, K_1 \oplus FK_1, K_2 \oplus FK_2, K_3 \oplus FK_3)$. When the number of ancilla qubits is 28, MRK_{SM4} can be implemented with $86 \times 32 = 2752$ T-depth, $1352 \times 32 = 43264$ T gates, $2704 \times 32 + 128 = 86656$ CNOT gates, and 50542 $\#1qCliff$ gates. We also require 384 qubits in the above operation, because we need to store the 128-bit input key, 128-bit plaintext, and 128-bit ciphertext in these qubits.

A stand-alone quantum circuit of SM4 can be constructed by combining MRK_{SM4} with MEQ_{SM4}. To sum up, a stand-alone quantum circuit of SM4 requires 384 qubits, 28 ancilla qubits, $5504 + 2752 = 8256$ T-depth, $86528 + 43264 = 129792$ T gates, $196672 + 86656 + 128 = 283456$ CNOT gates, and 58158+50542+64=108764 $\#1qCliff$ gates. Similarly, when the number of ancilla qubits is limited to 7, we also can design a stand-alone quantum circuit of SM4 with 384 qubits, seven ancilla qubits, 33024 T-depth, 129792 T gates, 283456 CNOT gates, and 108764 $\#1qCliff$ gates.

6.2 A depth-efficient quantum circuit implementations of SM4

Apart from the number of qubits, the T-depth is also an important cost metric in a quantum circuit implementation. In this subsection, we propose a new depth-efficient quantum circuit of SM4, which is similar to [21]. Firstly, we need to make the computation as parallel as possible, which can be achieved by introducing some new ancilla qubits. Secondly, the T-depth of the QAND/QAND† gate is smaller than a Toffoli gate. That is, we shall adopt the QAND/QAND† gate instead of the Toffoli gate in our depth-efficient quantum circuit of SM4. In Sect. 6.1, we express F as $F(X_i, X_{i+1}, X_{i+2}, X_{i+3}) = L(L^{-1}(X_i) \oplus \tau(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i))$. According to Algorithm 2, we have no restrictions on $S[j]$ (for $0 \leq j \leq 7$). As a result, we can also adopt the above strategy to construct our depth-efficient quantum circuit of SM4. We show the details of our depth-efficient quantum circuit of SM4 as follows.

Firstly, we need to generate K_0, K_1, K_2 and K_3 by using $(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3)$. As pointed out in Sect. 6.1, the parameter (FK_0, FK_1, FK_2, FK_3) is constant, which contains 64 bits "1." That is, we can compute $(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3)$ with 64 NOT gates. Secondly, we shall compute $K_3 = K_3 \oplus K_1 \oplus K_2 \oplus CK_0$ to obtain the input to τ. Since CK_0 contains 9 bits "1," this operation requires $32 \times 2 = 64$ CNOT gates and 9 NOT gates to compute K_3. Thirdly, we can compute $L'^{-1}(K_0)$ by using an in-place implementation with 78 CNOT gates. Fourthly, given $L'^{-1}(K_0)$, we can adopt Algorithm 2 4 times in parallel to compute the value of $L'^{-1}(K_0) \oplus \tau(K_3)$, which requires 64 qubits, 508 ancilla qubits, 7 T-depth, 512 T gates, 2552 CNOT gates, and 1056 $\#1qCliff$ gates. After computing $L'^{-1}(K_0) \oplus \tau(K_3)$, we can compute $K_4 = L'(L'^{-1}(K_0) \oplus \tau(K_3))$ with 78

CNOT gates. Fifthly, we still need 64 CNOT gates and 9 NOT gates to recompute K_3 with $K_3 = K_3 \oplus K_1 \oplus K_2 \oplus CK_0$. To sum up, the quantum resource to generate K_4 is 508 ancilla qubits, 7 T-depth, 512 T gates, $64 + 78 + 78 + 638 \times 4 + 64 = 2836$ CNOT gates, and $64 + 9 + 264 \times 4 + 9 = 1138$ #1$qCliff$ gates.

After generating K_4 (or called rk_0), we can generate X_4 and K_5 (or called rk_1) as follows. Firstly, we can compute $X_1 = X_1 \oplus X_2 \oplus X_3 \oplus K_4$ with $32 \times 3 = 96$ CNOT gates to obtain the input to τ in the round function of SM4. Secondly, we can compute $K_4 = K_4 \oplus K_3 \oplus K_2 \oplus CK_1$ to obtain the input to τ in the key schedule of SM4. Since CK_1 contains 12 bits "1," we can implement the above operation with 64 CNOT gates and 12 NOT gates. Thirdly, we can compute $L^{-1}(X_0)$ and $L'^{-1}(K_1)$ by using an in-place implementation with 83 CNOT and 78 CNOT gates, respectively. Fourthly, we can adopt Algorithm 2 8 times to compute $L^{-1}(X_0) \oplus \tau(X_1)$ and $L'^{-1}(K_1) \oplus \tau(K_4)$ in parallel. After computing $L^{-1}(X_0) \oplus \tau(X_1)$ and $L'^{-1}(K_1) \oplus \tau(K_4)$, we can compute $X_4 = L(L^{-1}(X_0) \oplus \tau(X_1))$ and $K_5 = L'(L'^{-1}(K_1) \oplus \tau(K_4))$ with 83+78=161 CNOT gates. Fifthly, we shall regenerate X_1 and K_4 by calculating $X_1 = X_1 \oplus X_2 \oplus X_3 \oplus K_4$ and $K_4 = K_4 \oplus K_3 \oplus K_2 \oplus CK_1$ with $96 + 64 = 160$ CNOT gates and 12 NOT gates. To sum up, the quantum resource to generate X_4 and K_5 is $1016 + 64 = 1080$ ancilla qubits, 7 T-depth, $128 \times 8 = 1024$ T gates, $96 + 64 + 83 + 78 + 638 \times 8 + 161 + 160 = 5746$ CNOT gates, and $12 + 264 \times 8 + 12 = 2136$ #1$qCliff$ gates.

We can generate X_5 and K_6 in a similar way as X_4 and K_5. Since we can implement τ with an in-place way, we do not need to introduce some extra ancilla qubits to generate X_5 and K_6. Due to the space limitation, we just give the result and omit the irrelevant details. Obviously, we can repeat the above operations 31 times to compute $(X_{31}, \ldots, X_{34})$ and $(K_{32}, \ldots, K_{35})$, where $K_{j+4} = rk_j$ for $0 \le j \le 31$.

After the above operation, we still need to calculate X_{35} so as to obtain the ciphertext of SM4. Firstly, we can compute $X_{32} = X_{32} \oplus X_{33} \oplus X_{34} \oplus K_{35}$ with $32 \times 3 = 96$ CNOT gates to obtain the input to τ. Secondly, we can compute $L^{-1}(X_{31})$ by using an in-place implementation with 83 CNOT gates. Thirdly, we can adopt Algorithm 2 4 times to compute $\tau(X_{32})$ in the 32 zero ancilla qubits in parallel, but also need to xor $\tau(X_{32})$ to $L^{-1}(X_{31})$ with 32 CNOT gates. Fourthly, after computing $L^{-1}(X_{31}) \oplus \tau(X_{32})$, we can compute $X_{35} = L(L^{-1}(X_{31}) \oplus \tau(X_{32}))$ with 83 CNOT gates. Fifthly, we shall regenerate X_{32} by calculating $X_{32} = X_{32} \oplus X_{33} \oplus X_{34} \oplus K_{35}$ again, which needs $32 \times 3 = 96$ CNOT gates. To sum up, the quantum resource to generate X_{35} is $1016+64 = 1080$ ancilla qubits, 7 T-depth, $128 \times 4 = 512$ T gates, $96+83+638 \times 4+83 + 96 = 2910$ CNOT gates, and 264 #1$qCliff$ gates. Note that X_{32}, X_{33}, X_{34} and X_{35} are ciphertext. We call the above circuit as DEQ_{SM4} in the following. After the above operation, we can compute the width and depth cost of DEQ_{SM4} as follows. That is, it requires 1080 ancilla qubits, $7+7 \times 31+7 = 231$ T-depth, $128 \times 8 \times 32 = 32768$ T gates, $2836 + 5746 \times 31 + 2910 = 183872$ CNOT gates, 68654 #1$qCliff$ gates.

The above depth-efficient quantum circuit of SM4 does not uncompute the value of the round key $(rk_{28}, \ldots, rk_{31})$. Similar to our memory-efficient quantum circuit of SM4, we can reverse the entire circuit to remove the round keys of $(rk_{28}, \ldots, rk_{31})$ in the circuit of the Grover algorithm. In other words, DEQ_{SM4} can only be used as a subprogram in the Grover algorithm on SM4.

In the following, we design a new circuit DRK_{SM4} to remove the round keys of $(rk_{28}, \ldots, rk_{31})$. Then, a stand-alone depth-efficient quantum circuit of SM4 can be constructed by combining DEQ_{SM4} with DRK_{SM4}. In the first place, we shall introduce 128 new zero qubits for storing the 128-bit state (X_0, X_1, X_2, X_3), which requires 128 CNOT gates. In addition, DRK_{SM4} shall remove the round keys of $(rk_{28}, \ldots, rk_{31})$. In details, we take $rk_{31} = rk_{27} \oplus L'((\tau(rk_{28} \oplus rk_{29} \oplus rk_{30}))$ as an example to show how to construct DRK_{SM4} as follows.

Firstly, since $CK_{31} = 0x646B7279$, we can obtain $rk_{28} = rk_{28} \oplus rk_{29} \oplus rk_{30} \oplus CK_{31}$ with 64 CNOT gates and 17 NOT gates. Secondly, we can get the value of $L'^{-1}(rk_{27}) \oplus \tau(rk_{28})$ by adopting L'^{-1} on rk_{31}, which requires 78 CNOT gates. Thirdly, given $L'^{-1}(rk_{27}) \oplus \tau(rk_{28})$, we can adopt Algorithm 2 4 times to obtain $L'^{-1}(rk_{27})$ in parallel. After the above operation, we can obtain rk_{27} by applying L' on $L'^{-1}(rk_{27})$, which needs 78 CNOT gates. At last, we shall regenerate rk_{28} by computing $rk_{28} = rk_{28} \oplus rk_{29} \oplus rk_{30} \oplus CK_{31}$ again, which requires 64 CNOT gates and 17 NOT gates. To sum up, we can remove rk_{31} with the cost of 540 ancilla qubits, 7 T-depth, $128 \times 4 = 512\,T$ gates, $64 + 78 + 638 \times 4 + 78 + 64 = 2836$ CNOT gates, $17 + 264 \times 4 + 17 = 1,090\,\#1qCliff$ gates.

Similar to rk_{31}, we can remove rk_j (for $0 \le j \le 30$) in a serial mode. In other words, we can remove $(rk_0, \ldots, rk_{30})$ by repeating the above operation 31 times. The parameters $(Ck_0, \ldots, Ck_{30})$ contains 486 bits "1," which are used to compute the number of NOT gates to remove $(rk_0, \ldots, rk_{30})$. That is, we can remove $(rk_0, \ldots, rk_{31})$ with 540 ancilla qubits, $7 \times 32 = 224\,T$-depth, $128 \times 4 \times 32 = 16384\,T$ gates, $2836 \times 32 = 90752$ CNOT gates, $264 \times 4 \times 32 + 503 \times 2 = 34798\,\#1qCliff$ gates.

We can design a stand-alone depth-efficient quantum circuit of SM4 by combining DEQ_{SM4} with DRK_{SM4}. That is, we require 384 qubits, 1080 ancilla qubits, 224+231=455 T-depth, $16384 + 32768 = 49152\,T$ gates, $90752 + 183872 + 128 = 274752$ CNOT gates, and $34798 + 68654 = 103,452$ NOT gates.

7 Our quantum circuit implementations of SM3 hash function

In this section, we try to propose some improved quantum circuit implementations of SM3 hash function. In [37], Song et al. showed a quantum circuit of SM3, which requires 2721 qubits, 43328 Toffoli gates, 134144 CNOT gates, 2638 NOT gates. In details, their quantum circuit of the message schedule of SM3 required 2176 qubits for storing the extended message W_j (for $0 \le j \le 67$). For a fairer comparison, we can rewrite the quantum gates in [37] as $43328 \times 7 = 303,296\,T$ gates, $43328 \times 7 + 134144 = 437,440$ CNOT gates, $43328 \times 2 + 2638 = 89,294\,\#1qCliff$ gates.

Since SM3 is similar to SHA-256, we may apply the previous quantum circuit of SHA-256 to construct our quantum circuit of SM3. In [4], Amy et al. proposed a quantum circuit of SHA-256 with 2402 qubits, 57184 Toffoli gates, 534272 CNOT gates. Firstly, they proposed some new quantum circuits of MAJ and CH. Secondly, they adopted a poly-depth ADDER in their memory-efficient quantum circuit of SHA-256 to minimize the number of qubits. To sum up, their quantum circuit of SHA-256 not only required 2048 qubits for storing the extended message W_i (for $0 \le i \le 63$), but also needed 66 ancilla qubits for storing intermediate state value and the ADDER operation (modular addition).

In [22], Kim et al. proposed some improved quantum circuit implementations of SHA-256. Their memory-efficient quantum circuit of SHA-256 requires 801 qubits and 36368 Toffoli depth, while their depth-efficient quantum circuits of SHA-256 requires 938 qubits and 10112 Toffoli depth. Their new ideas of memory-efficient and depth-efficient quantum circuits of SHA-256 can be explained as follows. Firstly, they observed that all the expanded message words of SHA-256 can be determined by any consecutive 16 words. Based on this new observation, their quantum circuit of SHA-256 only required 512 qubits for the message schedule of SHA-256. Secondly, they could reduce the number of ancilla qubits by computing each operations of the round function in serial mode. Thirdly, they adopted two different versions of ADDER operations in their two quantum circuits of SHA-256. In details, their memory-efficient quantum circuits of SHA-256 chose a poly-depth ADDER operation [11], while their depth-efficient quantum circuits of SHA-256 chose a log-depth ADDER operation [13].

We can utilize some techniques used in the previous quantum circuits of SHA-256 to construct our new quantum circuit of SM3. However, due to the following reasons, we cannot apply the previous quantum circuit of SHA-256 to construct our quantum circuit of SM3 directly. Firstly, SM3 employs different Boolean functions $X \oplus Y \oplus Z$, MAJ and CH in its round function for different round, while SHA-256 only adopts MAJ and CH in its round function. Secondly, SM3 has stronger message dependency than SHA-256, it needs to process W_i and W_{i+1} in the ith step, while SHA-256 just need W_i in the ith step. Thirdly, the step function of SM3 is more complex than the step function of SHA-256. As a result, we shall utilize some new strategies to construct a new quantum circuit of SM3.

7.1 Our memory-efficient quantum circuit implementations of SM3

In this subsection, we propose an improved quantum circuits of SM3, which tries to minimize the number of qubits as low as possible. That is, we shall minimize the number of qubits in our quantum circuit of SM3. In order to achieve this goal, we shall propose some memory-efficient quantum circuits of the round function and the message expansion function of SM3.

Our memory-efficient quantum circuits of the round function of SM3 can be constructed as follows. Firstly, we shall propose some memory-efficient quantum circuits of the basic operations CH, MAJ, ADDER (modular addition), XOR, P_0 and P_1 in the round function of SM3. Firstly, we can adopt the memory-efficient quantum circuits of CH and MAJ Boolean functions (see in Figs. 4 and 5 in [3]) proposed by Amy *et al.* in our quantum circuit of SM3. That is, $ab \oplus \neg ac$ can be rewritten as $a(b \oplus c) \oplus c$, which requires a single Toffoli gate and a single ancilla qubit. Besides, the MAJ function can be computed with a CNOT gate, two Toffoli gates, and a single ancilla qubit. In other words, we can implement FF_i with 32 Toffoli gates and 64 CNOT gates, while GG_i can be implemented with 64 Toffoli gates and 32 CNOT gates (for $16 \leq i \leq 63$). Secondly, the modular addition can be implemented with a poly-depth ADDER operation [11], which requires 61 Toffoli-depth, 61 Toffoli gates, 157 CNOT gates, 58 NOT gates, and 1 ancilla qubit. Thirdly, we propose some improved quantum circuit of P_0 and P_1 by using the method proposed by Xiang *et al.* [41]. Assume $x[0], x[1], \cdots, x[31]$ are the 32-bit input to P_0, our new in-place implementation of P_0 can compute $P_0(x)$ with 68 CNOT gates. Similar to P_0, our in-place implementation of P_1 can compute $P_1(x)$ with 68 CNOT gates. We show the details of P_0 and P_1 in Algorithm 5 and Algorithm 6. Note that Song et al. in [37] also proposed a new way to implement P_0 and P_1, which needed to use the CNOT gates repeatedly to recompute some original values. For example, they required 19 XOR gates to obtain $x[16] = x[16] \oplus x[7] \oplus x[31]$ in P_0. By using our new implementations of P_0 and P_1, we need less CNOT gates in P_0 and P_1 than Song *et al.*'s idea in [37].

Algorithm 5 A new in-place implementation of P_0.

```
    Input, $x[i]$ for $0 \le i \le 31$;
1: $x[8] = x[8] \oplus x[0]$;
2: $x[16] = x[16] \oplus x[24]$;
3: $x[10] = x[10] \oplus x[18]$;
4: $x[25] = x[25] \oplus x[9]$;
5: $x[2] = x[2] \oplus x[26]$;
6: $x[26] = x[26] \oplus x[18]$;
7: $x[18] = x[18] \oplus x[9]$;
8: $x[9] = x[9] \oplus x[0]$;
9: $x[0] = x[0] \oplus x[23]$;
10: $x[9] = x[9] \oplus x[24]$; \\ $y[9]$;
11: $x[24] = x[24] \oplus x[7]$;
12: $x[23] = x[23] \oplus x[14]$;
13: $x[14] = x[14] \oplus x[5]$;
14: $x[7] = x[7] \oplus x[22]$;
15: $x[23] = x[23] \oplus x[6]$; \\ $y[23]$;
16: $x[7] = x[7] \oplus x[30]$; \\ $y[7]$;
17: $x[22] = x[22] \oplus x[5]$;
18: $x[30] = x[30] \oplus x[21]$;
19: $x[5] = x[5] \oplus x[20]$;
20: $x[20] = x[20] \oplus x[3]$;
21: $x[6] = x[6] \oplus x[21]$;
22: $x[21] = x[21] \oplus x[4]$;
23: $x[4] = x[4] \oplus x[19]$;
24: $x[5] = x[5] \oplus x[28]$; \\ $y[5]$;
25: $x[21] = x[21] \oplus x[12]$; \\ $y[21]$;
26: $x[12] = x[12] \oplus x[3]$;
27: $x[3] = x[3] \oplus x[26]$; \\ $y[3]$;
28: $x[28] = x[28] \oplus x[19]$;
29: $x[19] = x[19] \oplus x[26]$;
30: $x[26] = x[26] \oplus x[17]$;
31: $x[17] = x[17] \oplus x[8]$; \\ $y[17]$;
32: $x[24] = x[24] \oplus x[15]$; \\ $y[24]$;
33: $x[26] = x[26] \oplus x[18]$; \\ $y[26]$;
34: $x[18] = x[18] \oplus x[1]$; \\ $y[18]$;
35: $x[1] = x[1] \oplus x[16]$; \\ $y[1]$;
36: $x[16] = x[16] \oplus x[8]$;
37: $x[8] = x[8] \oplus x[15]$;
38: $x[8] = x[8] \oplus x[31]$;
39: $x[0] = x[0] \oplus x[15]$; \\ $y[0]$;
40: $x[15] = x[15] \oplus x[30]$;
41: $x[30] = x[30] \oplus x[13]$; \\ $y[30]$;
42: $x[31] = x[31] \oplus x[22]$;
43: $x[22] = x[22] \oplus x[13]$; \\ $y[22]$;
44: $x[13] = x[13] \oplus x[4]$;
45: $x[13] = x[13] \oplus x[28]$; \\ $y[13]$;
46: $x[15] = x[15] \oplus x[6]$; \\ $y[15]$;
47: $x[31] = x[31] \oplus x[14]$; \\ $y[31]$;
48: $x[14] = x[14] \oplus x[29]$; \\ $y[14]$;
49: $x[6] = x[6] \oplus x[29]$; \\ $y[6]$;
50: $x[29] = x[29] \oplus x[12]$;
51: $x[4] = x[4] \oplus x[27]$; \\ $y[4]$;
52: $x[12] = x[12] \oplus x[27]$; \\ $y[12]$;
53: $x[27] = x[27] \oplus x[10]$; \\ $y[27]$;
54: $x[29] = x[29] \oplus x[20]$; \\ $y[29]$;
55: $x[20] = x[20] \oplus x[11]$; \\ $y[20]$;
56: $x[28] = x[28] \oplus x[11]$; \\ $y[28]$;
57: $x[11] = x[11] \oplus x[2]$; \\ $y[11]$;
58: $x[19] = x[19] \oplus x[2]$;
59: $x[19] = x[19] \oplus x[10]$; \\ $y[19]$;
60: $x[10] = x[10] \oplus x[25]$;
61: $x[2] = x[2] \oplus x[25]$;
62: $x[25] = x[25] \oplus x[16]$;
63: $x[16] = x[16] \oplus x[24]$;
64: $x[16] = x[16] \oplus x[8]$; \\ $y[16]$;
65: $x[2] = x[2] \oplus x[26]$; \\ $y[2]$;
66: $x[10] = x[10] \oplus x[18]$; \\ $y[10]$;
67: $x[25] = x[25] \oplus x[9]$; \\ $y[25]$;
68: $x[8] = x[8] \oplus x[0]$; \\ $y[8]$;
    Output, $x[i]$ for $0 \le i \le 31$;
```

Algorithm 6 A new in-place implementation of P_1.

```
    Input, $x[i]$, for $0 \le i \le 31$;
1: $x[25] = x[25] \oplus x[1]$;
2: $x[11] = x[11] \oplus x[3]$;
3: $x[19] = x[19] \oplus x[27]$;
4: $x[17] = x[17] \oplus x[9]$;
5: $x[2] = x[2] \oplus x[18]$;
6: $x[9] = x[9] \oplus x[1]$;
7: $x[1] = x[1] \oplus x[18]$;
8: $x[18] = x[18] \oplus x[27]$;
9: $x[27] = x[27] \oplus x[4]$;
10: $x[4] = x[4] \oplus x[13]$;
11: $x[4] = x[4] \oplus x[21]$; \\ $y[4]$;
12: $x[13] = x[13] \oplus x[22]$;
13: $x[18] = x[18] \oplus x[3]$; \\ $y[18]$;
14: $x[3] = x[3] \oplus x[20]$;
15: $x[20] = x[20] \oplus x[5]$;
```

```
16: $x[5] = x[5] \oplus x[22]$;
17: $x[20] = x[20] \oplus x[29]$; \\ $y[20]$;
18: $x[29] = x[29] \oplus x[21]$;
19: $x[21] = x[21] \oplus x[6]$;
20: $x[6] = x[6] \oplus x[15]$;
21: $x[15] = x[15] \oplus x[24]$;
22: $x[22] = x[22] \oplus x[31]$;
23: $x[22] = x[22] \oplus x[7]$; \\ $y[22]$;
24: $x[6] = x[6] \oplus x[23]$; \\ $y[6]$;
25: $x[7] = x[7] \oplus x[24]$;
26: $x[31] = x[31] \oplus x[23]$;
27: $x[23] = x[23] \oplus x[8]$;
28: $x[24] = x[24] \oplus x[9]$; \\ $y[24]$;
29: $x[8] = x[8] \oplus x[9]$;
30: $x[9] = x[9] \oplus x[26]$;
31: $x[26] = x[26] \oplus x[11]$; \\ $y[26]$;
32: $x[9] = x[9] \oplus x[1]$; \\ $y[9]$;
33: $x[1] = x[1] \oplus x[10]$; \\ $y[1]$;
34: $x[10] = x[10] \oplus x[19]$; \\ $y[10]$;
35: $x[19] = x[19] \oplus x[11]$;
36: $x[11] = x[11] \oplus x[12]$;
37: $x[3] = x[3] \oplus x[12]$; \\ $y[3]$;
38: $x[27] = x[27] \oplus x[12]$; \\ $y[27]$;
39: $x[11] = x[11] \oplus x[28]$;
40: $x[12] = x[12] \oplus x[29]$; \\ $y[12]$;
41: $x[29] = x[29] \oplus x[21]$;
42: $x[21] = x[21] \oplus x[30]$; \\ $y[21]$;
43: $x[29] = x[29] \oplus x[14]$; \\ $y[29]$;
44: $x[28] = x[28] \oplus x[13]$;
45: $x[13] = x[13] \oplus x[30]$; \\ $y[13]$;
46: $x[28] = x[28] \oplus x[5]$; \\ $y[28]$;
47: $x[5] = x[5] \oplus x[14]$; \\ $y[5]$;
48: $x[30] = x[30] \oplus x[15]$;
49: $x[14] = x[14] \oplus x[31]$; \\ $y[14]$;
50: $x[31] = x[31] \oplus x[23]$;
51: $x[30] = x[30] \oplus x[7]$; \\ $y[30]$;
52: $x[23] = x[23] \oplus x[0]$; \\ $y[23]$;
53: $x[15] = x[15] \oplus x[0]$; \\ $y[15]$;
54: $x[0] = x[0] \oplus x[17]$; \\ $y[0]$;
55: $x[7] = x[7] \oplus x[16]$; \\ $y[7]$;
56: $x[31] = x[31] \oplus x[16]$; \\ $y[31]$;
57: $x[16] = x[16] \oplus x[25]$; \\ $y[16]$;
58: $x[17] = x[17] \oplus x[25]$;
59: $x[25] = x[25] \oplus x[2]$;
60: $x[2] = x[2] \oplus x[18]$;
61: $x[2] = x[2] \oplus x[19]$; \\ $y[2]$;
62: $x[19] = x[19] \oplus x[11]$;
63: $x[19] = x[19] \oplus x[27]$; \\ $y[19]$;
64: $x[11] = x[11] \oplus x[3]$; \\ $y[11]$;
65: $x[8] = x[8] \oplus x[17]$; \\ $y[8]$;
66: $x[17] = x[17] \oplus x[25]$;
67: $x[25] = x[25] \oplus x[1]$; \\ $y[25]$;
68: $x[17] = x[17] \oplus x[9]$; \\ $y[17]$;
    Output, $x[i]$ for $0 \le i \le 31$;
```

After constructing a memory-efficient quantum circuit of the round function of SM3, we also need to minimize the number of qubits in the linear message expansion function of SM3. According to the message schedule of SM3, all expanded message words of SM3 can be determined by any consecutive 16 words $\{W_i, W_{i+1}, \cdots W_{i+15}\}$ (for $0 \le i \le 52$). Besides, we need to compute W_i and $W'_i = W_i \oplus W_{i+4}$ in SM3's ith round function. In this paper, we propose a newly quantum circuit of the message expansion function of SM3 with 512 qubits, which can be explained as follows. Firstly, given the 512-bit input message M, we can obtain and store $\{W_0, W_1, \cdots, W_{15}\}$ in the 512 qubits quantum memory. Secondly, we can compute the round function of SM3 iteratively until 11th round with the value of $\{W_0, W_1, \cdots, W_{15}\}$. In the 12th round, we need the value of W_{16} to compute W'_{12}, which can be obtained by using $W_{16} \leftarrow P_1(W_0 \oplus W_7 \oplus (W_{13} \lll 15)) \oplus (W_3 \lll 7) \oplus W_{10}$. Since we do not need W_0 after the 12th round, we can store W_{16} in W_0. Similar to W_{16}, we can compute and store W_j in W_{j-16} so as to obtain W'_{j-4} in the $j-4$th round (for $16 \le j \le 67$), by using $W_j \leftarrow P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15)) \oplus (W_{j-13} \lll 7) \oplus W_{j-6}$.

We show the time and memory cost of SM3's 12th round function as follows (also see in Fig. 5). We assume that the cost of the Swap operation and the rotation operation are free.

1. According to the message expansion algorithm of SM3, we need to compute $W_{16} = P_1(W_0 \oplus W_7 \oplus (W_{13} \lll 15)) \oplus (W_3 \lll 7) \oplus W_{10}$, which contains 4 times 32-bit XOR operation and a P_1 permutation. Since P_1 needs 68 CNOT gates, it requires $68 + 32 \times 4 = 196$ CNOT gates. Since we do not need W_0 after the 12th round, we can store W_{16} in W_0. In other words, we do not need to introduce new qubits for storing W_{16}.
2. After computing W_{16}, we shall compute and store FF_{12} in the work qubits $|0\rangle^{32}$. Firstly, it requires 64 CNOT gates to calculate FF_{12}. Secondly, we need 32 CNOT

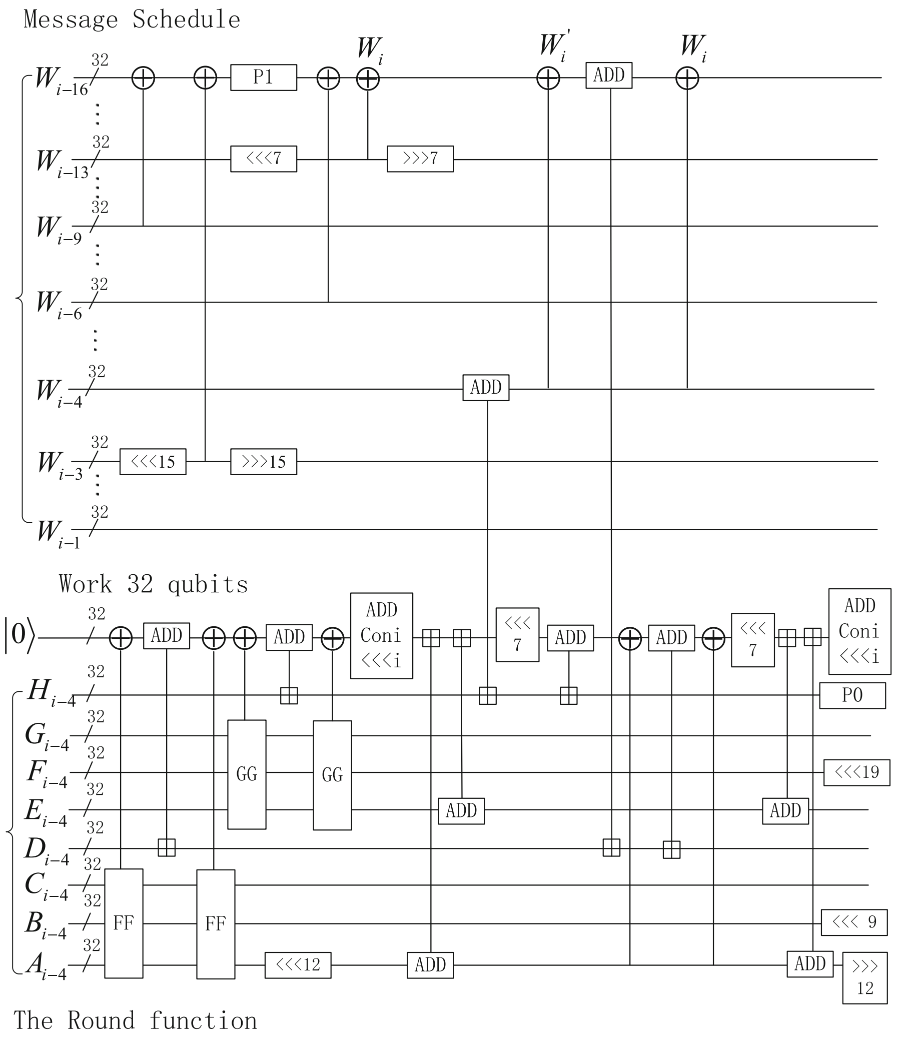


Fig. 5 *i*th round of SM3 hash function

gates to xor FF_{12} to the work qubits $|0\rangle^{32}$. After obtaining FF_{12}, we can compute $D_{12} = D_{12} + FF_{12}$ with a poly-depth ADDER. After the above operation, we do not need FF_{12} any more. As a result, we shall resume the 32 work qubits to $|0\rangle^{32}$ with 96 CNOT gates by xoring FF_{12} again.

3. Similar to FF_{12}, we need 96 CNOT gates to compute and store GG_{12} in the work qubits $|0\rangle^{32}$. Given[4.] GG_{12}, we can compute $H_{12} = H_{12} + GG_{12}$ with a poly-depth ADDER. After computing the new value of H_{12}, we do not need GG_{12} any more. As a result, we can resume the 32 work qubits to $|0\rangle^{32}$ with 96 CNOT gates.
5. After the above operation, we shall copy $Con_{12} \lll 12$ to the work qubits $|0\rangle^{32}$. Here $Con_{12} \lll 12$ is constant, which contains 15 bits “1.” As a result, we can

implement the above operation with 15 NOT gates. In addition, we can compute $((Con_{12} \lll 12) + (A_{12} \lll 12) + E_{12}) \lll 7$ and $H_{12} = W_{12} + H_{12}$ with three poly-depth ADDER operations. Furthermore, we can calculate $H_{12} = H_{12} + ((Con_{12} \lll 12) + (A_{12} \lll 12) + E_{12}) \lll 7$ with a poly-depth ADDER operation.

6. Since we need W'_{12} in the 12th round, we can compute $W'_{12} = W_{12} \oplus W_{16}$ with 32 CNOT gates. Note that W'_{12} is stored in W_{16}. Then, we can compute $D_{12} = D_{12} + W'_{12}$ with a poly-depth ADDER operation. Since we do not need W'_{12} after $D_{12} = D_{12} \oplus W'_{12}$, we shall recompute $W_{16} = W'_{12} \oplus W_{16}$ to compute W_{17} in the 17th round. This operations require 32 CNOT gates. In order to complete the round function of SM3, we still need to xor $A_{12} \lll 12$ to the 32 work qubits with 32 CNOT gates. Then, we can compute $D_{12} = D_{12} + ((((Con_{12} \lll 12) + (A_{12} \lll 12) + E_{12}) \lll 7) \oplus (A_{12} \lll 12))$.
7. Since we do not need $((((Con_{12} \lll 12) + (A_{12} \lll 12) + E_{12}) \lll 7) \oplus (A_{12} \lll 12))$ after the above operation, we shall clean up the 32 work qubits as follows. Firstly, we can xor $A_{12} \lll 12$ to the 32 work qubits again to obtain $((Con_{12} \lll 12) + (A_{12} \lll 12) + E_{12}) \lll 7$. Secondly, we can resume the 32 work qubits to $(Con_{12} \lll 12) + (A_{12} \lll 12) + E_{12}$ by using the rotating operation. Thirdly, we can compute the 32 work qubits to $(Con_{12} \lll 12)$ by modular subtraction of $(A_{12} \lll 12)$ and E_{12}. At last, we shall resume the 32 word qubits to $|0\rangle^{32}$ by xoring $(Con_{12} \lll 12)$. Since Con_{12} is constant, we can implement this operation with 15 NOT gates.

As shown in Fig. 5, we can implement the message expansion in the 12th round with 260 CNOT gates, while the 12th round function requires 10 poly-depth ADDER operations and 516 CNOT gates, 30 NOT gates. By using the ADDER operation [11], we only have an ancilla qubit in the ADDER operation. Since we have only a single ancilla qubit for CH and MAJ, we shall not only adopt the design of CH and MAJ proposed in [3], but also need to implement FF_i and GG_i in a serial mode. That is, the cost of 12th round can be computed with 33 ancilla qubits, 610 Toffoli depth, 610 Toffoli gates, 2346+1570=3916 CNOT gates and 610 NOT gates.

The cost of ith round can be calculated in a similar way as the 12th round (for $12 \leq i \geq 63$). However, there are some differences between different rounds. Firstly, we need to deal with different constant in the ith round (for $i \geq 16$). Secondly, the FF_i and GG_i (for $i \geq 16$) adopts CH and MAJ Boolean functions, while FF_i and GG_i (for $0 \leq i \leq 15$) adopt XOR operations. Since we shall clean up 33 ancilla qubits each round, we shall implement CH and MAJ twice in each round. In the following, we just omit the details and give the result due to the space limitation. That is, our memory-efficient quantum circuit implementation of SM3 requires $32+1 = 33$ ancilla qubits, $96 \times 48 + 96 \times 48 + 610 \times 64 = 48256$ Toffoli depth, 48256 Toffoli gates, $135504 + 157 \times 640 = 235,984$ CNOT gates, $2112 + 58 \times 640 = 39,232$ NOT gates. Since a Toffoli gate can be implemented with 9 T gates, 7 CNOT gates, and 3 T-depth, we can recalculate the above time and memory cost as 768 qubits, 33 ancilla qubits, 144768 T-depth, 434304 T gates, 573776 CNOT gates, 39,232 NOT gates. Compare with Song et al.'s quantum circuit of SM3 with 2721 qubits in [37], our new memory-efficient only require 768+33=801 qubits.

We call the above SM3's quantum circuit MQC_{SM3} in the following. After the above message expansion operation, we store $\{W_{52}, W_{53}, \cdots, W_{67}\}$ in the qubits. Similar to our memory-efficient quantum circuit of SM4, we can reverse the entire circuit to uncompute $\{W_{52}, W_{53}, \cdots, W_{67}\}$ to $\{W_0, W_1, \cdots, W_{15}\}$ in the circuit of the Grover algorithm. In other words, MQC_{SM3} can only be used as a subprogram in the Grover algorithm on SM3. In the following, we shall design a new circuit MRM_{SM3} to uncompute $\{W_{52}, W_{53}, \cdots, W_{67}\}$ to $\{W_0, W_1, \cdots, W_{15}\}$. Then, a stand-alone memory-efficient quantum circuit of SM3 can be constructed by combining MQC_{SM3} with MRM_{SM3}.

Since the message expansion algorithm of SM3 is linear, we can design MRM_{SM3} to uncompute $\{W_{52}, W_{53}, \cdots, W_{67}\}$ to $\{W_0, W_1, \cdots, W_{15}\}$ as follows. According to $W_j = P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15))$ (for $16 \le j \le 67$), we can rewrite W_{j-16} as $W_{j-16} = P_1^{-1}(W_j \oplus W_{j-6} \oplus (W_{j-13} \lll 7)) \oplus W_{j-9} \oplus (W_{j-3} \lll 15)$. Then, given $\{W_{52}, W_{53}, \cdots, W_{67}\}$, we can compute W_{51} as follows. Firstly, we xor W_{61} and $(W_{54} \lll 7)$ to W_{67} to obtain $W_{67} \oplus W_{61} \oplus (W_{54} \lll 7)$, which requires 64 CNOT gates. Secondly, we apply P_1^{-1} to $W_{67} \oplus W_{61} \oplus (W_{54} \lll 7)$ to compute $P_1^{-1}(W_{67} \oplus W_{61} \oplus (W_{54} \lll 7))$. Since the cost of P_1^{-1} is the same as P_1, we can implement P_1^{-1} with 68 CNOT gates. Thirdly, we can obtain W_{51} by xoring W_{j-9} and $(W_{j-3} \lll 15)$ to $P_1^{-1}(W_{67} \oplus W_{61} \oplus (W_{54} \lll 7))$, which is 64 CNOT gates. That is, we require $68 + 64 \times 2 = 196$ CNOT gates to obtain W_{51}. After the above operation, W_{51} is stored in W_{67}. In general, we can generate W_j and store in W_{j+16} (for $0 \le j \le 51$) in a similar way, which also requires 196 CNOT gates. To sum up, we require $196 \times 52 = 10192$ CNOT gates to uncompute $\{W_{52}, W_{53}, \cdots, W_{67}\}$ to $\{W_0, W_1, \cdots, W_{15}\}$. Since a stand-alone memory-efficient quantum circuit of SM3 can be constructed by combining MQC_{SM3} with MRM_{SM3}, it requires 768 qubits, 33 ancilla qubits, 144768 T-depth, 434304 T gates, $573776 + 10192 = 583,968$ CNOT gates, and 39,232 $1qCliff$ gates.

7.2 Our depth-efficient quantum circuit implementations of SM3

In this subsection, we propose a new depth-efficient quantum circuit of SM3. That is, we need to construct a depth-efficient quantum circuit of the round function and the message expansion function of SM3. Our depth-efficient quantum circuit of the message expansion function of SM3 can be constructed as follows. Firstly, the message expansion function of SM3 is linear. Secondly, we can determine all the expanded message words uniquely by any consecutive 16 words. Since the nonlinear operations contain the T gate, we do not increase the T-depth even if we only store 16 words $\{W_i, W_{i+1}, \cdots W_{i+15}\}$ (for $0 \le i \le 52$). As a result, we just adopt the quantum circuit of the message expansion function of SM3 in Sect. 7.1 in our depth-efficient quantum circuit of SM3.

Our depth-efficient quantum circuit of the round function of SM3 can be constructed as follows. Firstly, we try to reduce the T-depth by replacing Toffoli gates with QAND/QAND† gates. Secondly, we show how to optimize the nonlinear operations, such as CH, MAJ and ADDER, because only the nonlinear operations contain the T gate. However, QAND gates require the output qubit to be zero, while Tof-

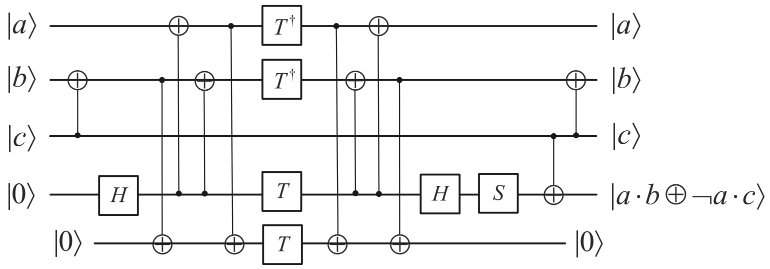


Fig. 6 New implementation of CH

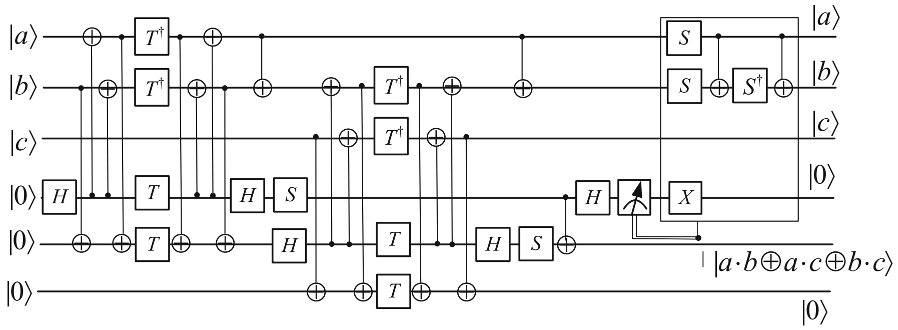


Fig. 7 New implementation of maj

foli gates in the previous CH and MAJ functions have no such restriction. In other words, we cannot replace Toffoli gates with QAND gates in the CH and MAJ functions directly. In order to solve this problem, we propose some new quantum circuits of the CH and MAJ functions to optimize the T-depth. As shown in Fig. 6, our new quantum circuit of the CH function can be constructed with the T-depth of 1, 1 QAND gate, and 3 CNOT gates. Similar to CH, our new quantum circuit of MAJ function (see in Fig. 7) can be constructed with the T-depth of 2, 2 QAND gate, 1 QAND† gate, and 3 CNOT gates. Note that our new depth-efficient quantum circuits implementations of the CH and MAJ functions can also be applied to SHA-1 and SHA-2.

We also try to reduce the T-depth by computing the nonlinear operations in a parallel mode. In other words, we shall introduce some new ancilla qubits for ADDER, CH and MAJ so as to implement these non-linear operations in a parallel mode. Take ADDER as an example, we can adopt a log-depth ADDER operation [13] with a Toffoli depth of 22 and 53 ancilla qubits to optimize the T-depth. In addition, we need to implement the MAJ in FF_i and the CH in GG_i in parallel (for $16 \leq i \leq 63$). That is, we need 32 ancilla qubits for FF_i, while GG_i requires 64 ancilla qubits. Apart from the nonlinear operations, we observe that the in-place implementations is an efficient way for the linear operations.

Similar to our memory-efficient quantum circuit of SM3, we need to compute the time and memory cost of the ith SM3's step function (see in Fig. 8). For simplicity, we just show how to compute the time and memory cost of the 16th step function as follows.

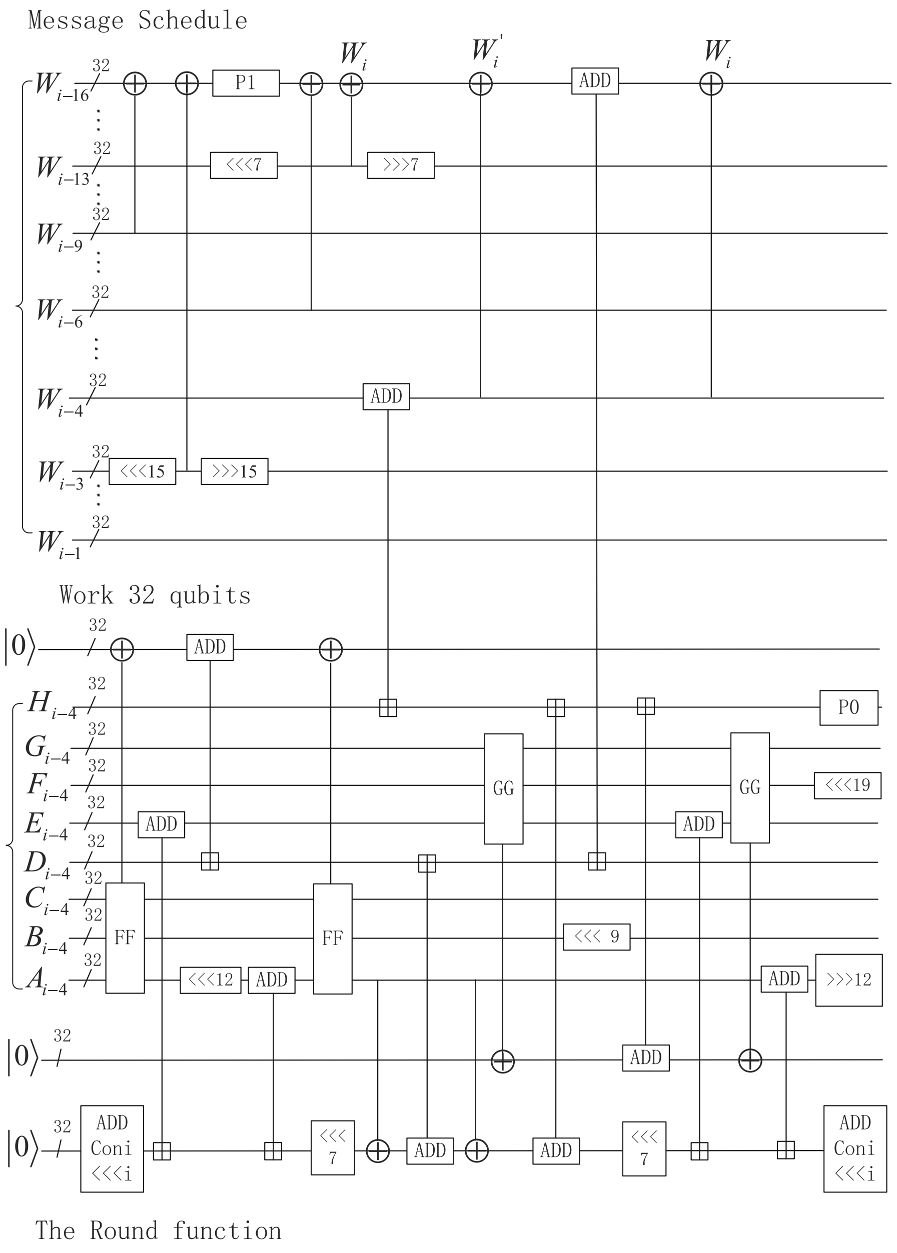


Fig. 8 Parallel implementation of SM3 step function

1. According to the message expansion algorithm of SM3, we need to compute $W_{20} = P_1(W_4 \oplus W_{11} \oplus (W_{17} \lll 15)) \oplus (W_7 \lll 7) \oplus W_{14}$, which requires 196 CNOT gates. Since we do not need W_4 after the 16th round, we can store W_{20} in W_4. In other words, we do not need to introduce new qubits for storing W_{20}.
2. After computing W_{20}, we shall copy $Con_{16} \lll 16$ to the work qubits $|0\rangle^{32}$. Since $Con_{16} \lll 16$ is constant, we can implement the above operation with 17 NOT gates. After the above assignment, we can compute $(Con_{16} \lll 16) + E_{16}$ with a log-depth ADDER operation. In addition, we need to compute FF_{16} in parallel by storing in the new work qubits $|0\rangle^{32}$. According to our new implementation of MAJ, we need to 64 QAND gate, 32 QAND† gate, and 96 CNOT gates to compute FF_{16} in parallel. Note that we still need 32 CNOT gates to xor FF_{16} to the new 32 ancilla qubits. In order to reduce the T-depth, we need to implement FF_{16} and $(Con_{16} \lll 16) + E_{16}$ in parallel by introducing more ancilla qubits. To sum up, this step require 32+64+53=149 ancilla qubits.
3. Given FF_{16}, we can compute $D_{16} = D_{16} + FF_{16}$ and $((Con_{16} \lll 16) + E_{16} + (A_{16} \lll 12)) \lll 7$ in parallel with two log-depth ADDER operations. This step require 32+53+53=138 ancilla qubits.
4. After computing $D_{16} = D_{16} + FF_{16}$, we do not need FF_{16} any more. Then, we need to uncompute the 32 ancilla qubits by xoring FF_{16} again, which requires 64 QAND gate, 32 QAND† gate, and 96 CNOT gates. In addition, we need to compute $(((Con_{16} \lll 16) + E_{16} + (A_{16} \lll 12)) \lll 7) \oplus (A_{16} \lll 12)$. Furthermore, we can compute $D_{16} = D_{16} + ((((Con_{16} \lll 12) + (A_{16} \lll 12) + F_{16}) \lll 7) \oplus (A_{16} \lll 12))$. At last, we can compute $H_{16} = H_{16} + W_{16}$. Note that this three operations can be implemented in parallel with 32+64+53+53=202 ancilla qubits.
5. Similar to FF_{16}, we need 32 QAND gate, 128 CNOT gates and 32 ancilla qubits to compute and xor GG_{16} to the work qubits $|0\rangle^{32}$. In addition, we can compute $W'_{16} = W_{16} \oplus W_{20}$ with 32 CNOT gates, where W'_{16} is stored in W_{20}. After obtaining W'_{16}, we can compute $D_{16} = D_{16} + W'_{16}$ with a log-depth ADDER operation. Since we do not need W'_{16} after $D_{16} = D_{16} \oplus W'_{16}$, we shall recompute $W_{20} = W'_{16} \oplus W_{16}$ with 32 CNOT gates. Furthermore, we can compute $H_{16} = H_{16} + (((Con_{16} \lll 16) + (A_{16} \lll 12) + E_{16}) \lll 7)$ with a poly-depth ADDER operation. The above three operations can be implemented in parallel with 32+32+53+53=170 ancilla qubits.
6. Given GG_{16}, we can compute $H_{16} = H_{16} + GG_{16}$ with a log-depth ADDER. In addition, we need to recompute the ancilla qubits to $((Con_{16} \lll 16) + (A_{16} \lll 12) + E_{16})$ by the rotation operation. Furthermore, we need to remove E_{16} with a modular subtraction. Note that this two operations can be implemented in parallel with 32+53+53=138 ancilla qubits.
7. After computing new value of H_{16}, we need to recompute the 32 work qubits with GG_{16} to $|0\rangle^{32}$ again, which requires 32 QAND gate, and 128 CNOT gates. Besides, we need to remove $(A_{16} \lll 12)$ with a modular subtraction. Furthermore, we still need to implement some linear operations, such as P_0 and the XOR operation. The two operations can be implemented in parallel with 32+53+53=138 ancilla qubits.

As shown in the above, we need to implement at least six ADDER operations in a serial mode (also see in Fig. 8). To sum up, we require $254 \times 10 = 2540$ Toffoli gates, 192 QAND gates, 64 QAND† gates, 396 T-depth, 202 ancilla qubits, $836+123\times 10 = 2066$ CNOT gates, $34+62\times 10 = 654$ NOT gates to implement the 16th round function of SM3.

We can implement the other round in a similar way. Due to the space limitation, we just omit the details and give the result directly. To sum up, our depth-efficient quantum circuit of SM3 require $254 \times 640 = 162560$ Toffoli gates, 12288 QAND gates, 4096 QAND† gates, 25344 T-depth, 202 ancilla qubits, $47296 + 123 \times 640 = 126016$ CNOT gates, and $2112 + 62 \times 640 = 41792$ NOT gates. Note that a QAND gate consists of 4 T gates, 8 CNOT gates and 3 $1qCliff$ gates, while a QAND† needs 0 T gates, 2 CNOT gates and 4 $1qCliff$ gates. Besides, a Toffoli gate consists of 7 T gates, 7 CNOT gates and 2 $1qCliff$ gates. As a result, we can recalculate the above time and memory cost as 202 ancilla qubits, $162560 \times 7 + 12288 \times 4 = 1187072$ T gates, 25344 T-depth, $126016 + 162560 \times 7 + 12288 \times 8 + 4096 \times 2 = 1370432$ CNOT gates, and $41792 + 162560 \times 2 + 12288 \times 3 + 4096 \times 4 = 420160$ $1qCliff$ gates.

Similar to our memory-efficient quantum circuit of SM3, after the above message expansion operation, the values in the qubits for message are $\{W_{52}, W_{53}, \cdots, W_{67}\}$. Obviously, we can reverse the entire circuit to uncompute $\{W_{52}, W_{53}, \cdots, W_{67}\}$ to $\{W_0, W_1, \cdots, W_{15}\}$ in the circuit of the Grover algorithm. In other words, the above depth-efficient quantum circuit of SM3 can only be used as a subprogram in the Grover algorithm on SM3.

Given $\{W_{52}, W_{53}, \cdots, W_{67}\}$, we can still use MRM_{SM3} to uncompute $\{W_{52}, W_{53}, \cdots, W_{67}\}$ to $\{W_0, W_1, \cdots, W_{15}\}$ in this part. In other words, we can design a stand-alone depth-efficient quantum circuit of SM3 by combining MRM_{SM3} with the above depth-efficient quantum circuit of SM3. Note that MRM_{SM3} require $196\times 52 = 10192$ CNOT gates to uncompute $\{W_{52}, W_{53}, \cdots, W_{67}\}$ to $\{W_0, W_1, \cdots, W_{15}\}$. To sum up, our stand-alone depth-efficient quantum circuit of SM3 needs 768 qubits, 202 ancilla qubits, 1187072 T gates, 25344 T-depth, $1370432 + 10192 = 1380624$ CNOT gates, and 420160 $1qCliff$ gates.

8 Conclusion

In this paper, we propose some new efficient quantum circuit implementations of SM4 block cipher. Firstly, we not only propose an improved classical circuit of SM4's S-box, but also construct some new quantum circuit implementations of SM4's S-box. Secondly, we propose some new in-place implementations of the linear permutations L and L' of SM4, which can reduce the number of qubits in our quantum circuit of SM4.

Similar to SM4, we also propose some new efficient quantum circuit implementations of SM3 hash function. Firstly, we propose some new in-place implementations of the linear permutations P_0 and P_1 of SM3. Secondly, we reduce the number of qubits in our quantum circuit of SM3 by exploring the linear message expansion of

SM3. Thirdly, we propose some new depth-efficient quantum circuit implementations of the CH and MAJ Boolean functions, which can be used to SHA-1 and SHA-2.

Acknowledgements We would like to thank anonymous referees for their helpful comments and suggestions. Jian Zou is supported by the National Natural Science Foundation of China (No.61902073), Natural Science Foundation of Fujian Province (2021J01623). Yiyuan Luo is supported by the National Natural Science Foundation of China (No.62072207). Wenling Wu is supported by the National Natural Science Foundation of China (No.62072445).

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

1. Abbasi, I., Afzal, M.: A compact s-box design for SMS4 block cipher. IACR Cryptol. ePrint Arch. **2011**, 522 (2011)
2. Aaronson, S., Gottesman, D.: Improved simulation of stabilizer circuits. CoRR, arXiv:quant-ph/0406196 (2004)
3. Amy, M., Di Matteo, O., Gheorghiu, V., Mosca, M., Parent, A., Schanck, J.M.: Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In: Avanzi, R., Heys, H.M. (Eds.) Selected Areas in Cryptography—SAC 2016—23rd International Conference, St. John's, NL, Canada, August 10–12, 2016, Revised Selected Papers, volume 10532 of Lecture Notes in Computer Science, pp. 317–337. Springer (2016)
4. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(6), 818–830 (2013)
5. Almazrooie, M., Samsudin, A., Abdullah, R., Mutter, K.N.: Quantum reversible circuit of AES-128. Quantum Inf. Process. **17**(5), 112 (2018)
6. Banegas, G., Bernstein, D.J., van Hoof, I., Lange, T.: Concrete quantum cryptanalysis of binary elliptic curves. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2021**(1), 451–472 (2021)
7. Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: Festa, P. (Ed.) Proceedings of the Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20–22, 2010, volume 6049 of Lecture Notes in Computer Science, pp. 178–189. Springer (2010)
8. Boyar, J., Peralta, R.: A small depth-16 circuit for the AES s-box. In: Gritzalis, D., Furnell, S., Theoharidou, M. (Ed.) Proceedings of the Information Security and Privacy Research—27th IFIP TC 11 Information Security and Privacy Conference (SEC 2012), Heraklion, Crete, Greece, June 4–6, 2012, volume 376 of IFIP Advances in Information and Communication Technology, pp. 287–298. Springer (2012)
9. Bai, X., Xu, Y., Li, G.: Securing sms4 cipher against differential power analysis and its vlsi implementation. In: IEEE Singapore International Conference on Communication Systems (2009)
10. Canright, D.: A very compact s-box for AES. In: Rao, J.R., Sunar, B. (Ed.) , Proceedings of the Cryptographic Hardware and Embedded Systems—(CHES 2005), 7th International Workshop, Edinburgh, UK, August 29–September 1, 2005, volume 3659 of Lecture Notes in Computer Science, pp. 441–455. Springer (2005)
11. Cuccaro, S.A., Draper, T.G., Kutin, S.A., Moulton, D.P.: A new quantum Ripple–Carry addition circuit. (2004). arXiv:quant-ph/0410184
12. Cao, X.-Y., Jie, G., Yu-Shuo, L., Yin, H.-L., Chen, Z.-B.: Coherent one-way quantum conference key agreement based on twin field. New J. Phys. **23**(4), 043002 (2021)
13. Draper, T.G., Kutin, S.A., Rains, E.M., Svore, K.M.: A logarithmic-depth quantum carry-lookahead adder. Quantum Inf. Comput. **6**(4), 351–369 (2006)
14. Fu, Y., Yin, H.-L., Chen, T.-Y., Chen, Z.-B.: Long-distance measurement-device-independent multiparty quantum communication. Phys. Rev. Lett. **114**(9), 090501 (2015)
15. Gu, J., Cao, X.-Y., Yin, H.-L., Chen, Z.-B.: Differential phase shift quantum secret sharing using a twin field. Opt. Express **29**(6), 9165–9173 (2021)

16. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R: Applying grover's algorithm to AES: quantum resource estimates. In: Takagi, T. (Ed.) Proceedings of the post-Quantum Cryptography—7th International Workshop (PQCrypto 2016), Fukuoka, Japan, February 24–26, 2016, volume 9606 of Lecture Notes in Computer Science, pp. 29–43. Springer (2016)
17. Grice, W.P., Qi, B.: Quantum secret sharing using weak coherent states. Phys. Rev. A **100**(2), 022339 (2019)
18. Gu, J., Xie, Y.-M., Liu, W.-B., Fu, Y., Yin, H.-L., Chen, Z.-B.: Secure quantum secret sharing without signal disturbance monitoring. Opt. Express **29**(20), 32244–32255 (2021)
19. Google AI Quantum and collaborators: Quantum supremacy using a programmable superconducting processor. Nature **574**, 505–510 (2019)
20. Hosoyamada, A., Sasaki, Y.: Cryptanalysis against symmetric-key schemes with online classical queries and offline quantum computations. In: Smart, N.P. (Ed.) Topics in Cryptology - CT-RSA 2018—The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16–20, 2018, Proceedings, volume 10808 of Lecture Notes in Computer Science, pp. 198–218. Springer (2018)
21. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and lowmc. In: Canteaut, A., Ishai, Y. (Ed.) Proceedings of the Advances in Cryptology—EUROCRYPT 2020—39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Part II, volume 12106 of Lecture Notes in Computer Science, pp. 280–310. Springer (2020)
22. Kim, P., Han, D., Jeong, K.C.: Time-space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2. Quantum Inf. Process. **17**(12), 339 (2018)
23. Li, Z., Cao, X.-Y., Li, C.-L., Weng, C.-X., Jie, G., Yin, H.-L., Chen, Z.-B.: Finite-key analysis for quantum conference key agreement with asymmetric channels. Quantum Scie. Technol. **6**(4), 045019 (2021)
24. Lu, Y.-S., Cao, X.-Y., Weng, C.-X., Gu, J., Xie, Y.-M., Zhou, M.-G., Yin, H.-L., Chen, Z.-B.: Efficient quantum digital signatures without symmetrization step. Opt. Express **29**(7), 10162–10171 (2021)
25. Langenberg, B., Pham, H., Steinwandt, R.: Reducing the cost of implementing AES as a quantum circuit. IACR Cryptol. ePrint Arch. **2019**, 854 (2019)
26. Lucamarini, M., Yuan, Z.L., Dynes, J.F., Shields, A.J.: Overcoming the rate-distance limit of quantum key distribution without quantum repeaters. Nature **557**(7705), 400–403 (2018)
27. Martínez-Herrera, A.F., Mex-Perera, J.C., Nolazco-Flores, J.A.: Some representations of the s-box of camellia in $GF(((2^2)^2)^2)$. In: Pieprzyk, J., Sadeghi, A-R., Manulis, M. (Eds.) Proceedings of the Cryptology and Network Security, 11th International Conference, CANS 2012, Darmstadt, Germany, December 12–14, 2012, volume 7712, pp. 296–309. Springer (2012)
28. Martínez-Herrera, A.F., Mex-Perera, J.C., Nolazco-Flores, J.A.: Merging the camellia, SMS4 and AES s-boxes in a single s-box with composite bases. In: Desmedt, Y. (Ed.) Proceedings of the Information Security, 16th International Conference, ISC 2013, Dallas, Texas, USA, November 13–15, 2013, volume 7807 of Lecture Notes in Computer Science, pp. 209–217. Springer (2013)
29. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information (10th Anniversary Edition). Cambridge University Press (2016)
30. NIST: Advanced Encryption Standard (AES), FIPS PUB 197 (2001)
31. NIST: Secure Hash Standard (SHS), FIPS PUB 180-4 (2015)
32. Office of state commercial cryptography administration: Announcement of 6 cryptographic standards **(in Chinese)**. http://www.oscca.gov.cn/News/201204/News1228.htm
33. Peng, Q., Guo, Y., Liao, Q., Ruan, X.: Satellite-to-submarine quantum communication based on measurement-device-independent continuous-variable quantum key distribution. Quantum Inf. Process. **21**(2), 1–19 (2022)
34. Proietti, M., Ho, J., Grasselli, F., Barrow, P., Malik, M., Fedrizzi, A: Experimental quantum conference key agreement. Sci. Adv. **7**(23):eabe0395 (2021)
35. Roberts, G.L., Lucamarini, M., Yuan, Z.L., Dynes, J.F., Comandar, L.C., Sharpe, A.W., Shields, A.J., Curty, M., Puthoor, I.V., Andersson, E.: Experimental measurement-device-independent quantum digital signatures. Nat. Commun. **8**(1), 1–7 (2017)
36. Roetteler, M., Naehrig, M., Svore, K.M., Lauter, K.E.: Quantum resource estimates for computing elliptic curve discrete logarithms. In: Takagi, T., Peyrin, T. (Ed.), Proceedings of the Advances in Cryptology—ASIACRYPT 2017—23rd International Conference on the Theory and Applications of

Cryptology and Information Security, Hong Kong, China, December 3–7, 2017, Part II, volume 10625 of Lecture Notes in Computer Science, pp. 241–270. Springer (2017)

37. Song, G., Jang, K., Kim, H., Lee, W.-K., Zhi, H., Seo, H.: Grover on SM3. IACR Cryptol. ePrint Arch. **2021**, 668 (2021)
38. Specification of sm3 cryptographic hash function **(in Chinese)**. http://www.oscca.gov.cn/UpFile/20101222141857786.pdf/
39. Toffoli, T: Reversible computing. In: de Bakker, J.W., van Leeuwen, J. (Ed.), Proceedings of the Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherlands, July 14–18, 1980, volume 85 of Lecture Notes in Computer Science, pp. 632–644. Springer (1980)
40. Wei, Z., Sun, S., Lei, H., Wei, M., Boyar, J., Peralta, R.: Scrutinizing the tower field implementation of the f_{2^8} inverter—with applications to aes, camellia, and SM4. IACR Cryptol. ePrint Arch. **2019**, 738 (2019)
41. Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. IACR Trans. Symm. Cryptol. **2020**(2), 120–145 (2020)
42. Yin, H.-L., Yao, F., Chen, Z.-B.: Practical quantum digital signature. Phys. Rev. A **93**, 032316 (2016)
43. Zou, J., Dong, L., Wenling, W.: New algorithms for the unbalanced generalised birthday problem. IET Inf. Secur. **12**(6), 527–533 (2018)
44. Zou, J., Liu, Y., Dong, L.: An efficient quantum multi-collision search algorithm. IEEE Access **8**, 181619–181628 (2020)
45. Zou, J., Wei, Z., Sun, S., Liu, X., Wu, W.: Quantum circuit implementations of aes with fewer qubits. In: Advances in Cryptology—ASIACRYPT 2020—the 26th Annual International Conference on the Theory and Application of Cryptology and Information Security, Lecture Notes in Computer Science. Springer (2020)