Optimized quantum implementation of AES

Da Lin[1], Zejun Xiang[2,3*], Runqing Xu[3], Shasha Zhang[3] and Xiangyong Zeng[1]

[1]Faculty of Mathematics and Statistics, Hubei Key Laboratory of Applied Mathematics, Hubei University, Wuhan, 430062, China.
[2]State Key Laboratory of Cryptology, P. O. Box 5159, Beijing, 100878, China.
[3]School of Cyber Science and Technology, Hubei University, Wuhan, 430062, China.

*Corresponding author(s). E-mail(s): xiangzejun@hubu.edu.cn;
Contributing authors: linda@stu.hubu.edu.cn;
xurq5953@stu.hubu.edu.cn; amushasha@163.com;
xzeng@hubu.edu.cn;

Abstract

This work researches the implementation of the AES family with Pauli-X gates, CNOT gates and Toffoli gates as the underlying quantum logic gate set. First, the properties of quantum circuits are investigated, as well as the influence of Pauli-X gates, CNOT gates and Toffoli gates on the performance of the circuits constructed with those gates. Based on these properties and the observations on the hardware circuits built by Boyar *et al.* and Zou *et al.*, it is possible to construct quantum circuits for AES's Substitution-box (S-box) and its inverse (S-box^{-1}) by rearranging the classical implementation to three parts. Since the second part is treated as a 4-bit S-box in this paper and can be dealt with by existing tools, a heuristic is proposed to search optimized quantum circuits for the first and the third parts. In addition, considering the number of parallelly executed S-boxes, the trade-offs between the qubit consumption and $\boldsymbol{T} \cdot \boldsymbol{M}$ values for the round function and key schedule of AES are studied. As a result, quantum circuits of AES-128, AES-192 and AES-256 can be constructed with 269, 333 and 397 qubits, respectively. If more qubits are allowed, quantum circuits that outperform state-of-the-art schemes in the metric of $\boldsymbol{T} \cdot \boldsymbol{M}$ value for the AES family can be reported, and it needs only 474, 538 and 602 qubits for AES-128, AES-192 and AES-256, respectively.

Keywords: AES, quantum circuit, quantum gate, Toffoli depth

1 Introduction

The development of quantum technology challenges the security of modern cryptography, especially the overwhelming advantage of quantum computers in solving mathematical problems over the classical ones, which benefits from the quantum algorithms such as Grover's Algorithm [12], Simon's Algorithm [28] and Shor's algorithm [27]. In addition, the successful design of quantum processors such as *Sycamore* [3] further increases the need for modern cryptography to prepare in advance for the rapid development of the construction of quantum computers.

Developing ciphers that are secure in both classical and quantum environment is the main research goal of post-quantum cryptography (PQC). In 2016, NIST (National Institute of Standards and Technology) started a process to develop new cryptography standards, which was aimed at developing new standards that resist to quantum attacks. Based on the strength offered by the existing standards[1,2], NIST suggested classifying the security strength of the submissions into five categories in [24], where the categories 1, 3 and 5 are related to the quantum resource required to conduct an exhaustive key search on the AES family [8]. On the other hand, the Grover's algorithm can achieve a square root speed-up when searching for a certain element in an unordered set. Therefore, the research on designing quantum circuits for AES and evaluating the quantum resource of exhaustively searching for the key of the AES family combined with the Grover's algorithm have received wide attention.

The researches on the quantum implementation of the AES family mainly focus on building the circuits with the Pauli-X gate (or NOT gate), the controlled-NOT gate (also known as C-NOT gate or CNOT gate) and the Toffoli gate (see [23] for definitions) as the underlying quantum logic gate set (NCT gate set for short) [1, 11, 14, 15, 18–20, 30, 34]. In 2016, Grassl *et al.* [11] first systematically investigated the construction of quantum circuits for the three variants of AES. Afterwards, Almazrooie *et al.* [1] optimized the quantum circuit of the multiplicative inverse over finite fields with the help of Itoh-Tsujii algorithm [13] and designed a quantum circuit for AES-128 with fewer qubits. Based on the quantum circuits proposed in [11], the authors of [18] improved the cost of computing multiplicative inverse and researched the time-space complexity for searching the key of the AES family. In [19], the classical hardware implementation of AES S-box given in [5] was adopted to construct a quantum one, benefit from which, Langenberg *et al.* proposed optimized quantum implementations for the AES family with reduced consumption of qubits and quantum logic gates compared with the previous work.

[1] https://doi.org/10.6028/NIST.FIPS.197
[2] https://doi.org/10.6028/NIST.FIPS.202.

Along the research direction of designing quantum circuits for AES with the help of classical implementations, Zou *et al.* [34] presented optimized quantum circuits for the S-box and S-box^{-1} simultaneously at ASIACRYPT 2020, combined with their proposed methods to implement the key schedule and the round function, both the qubit cost and the $T \cdot M$ value (the product of the Toffoli depth and the number of qubits) of the quantum circuits built for the AES family were reduced. In [30], Wang *et al.* also reported a quantum circuit for the case that the output qubits of the S-box are not all 0s to optimize the implementation of the key schedule for AES-128, by which they saved quantum gates and qubits at the same time. Recently, new quantum circuits for AES S-box and its inverse were given to design quantum circuits for AES in [14]. Besides, the authors introduced a method to construct quantum circuit for the S-box^{-1} from the S-box circuit by adding some linear transformations. The circuits of AES S-box with low depth presented in [14] were also applied by Jang *et al.* [15], and the $T \cdot M$ value of the circuits constructed in [15] for the AES family decreased significantly. In addition, the arithmetic over Finite Fields has also been applied. In [7], Chung *et al.* focused on constructing quantum circuits for AES S-box with the tower-filed construction combined with their proposed strategies of the trade-off between depth and width. Similarly, by making use of the algebraic structure, Li *et al.* [20] designed various quantum circuits for AES S-box and its inverse to optimize the quantum implementation of AES.

As quantum computation technology develops, the number of qubits that can be handled by quantum simulators will gradually increase. However, the progress is very slow [3, 33, 35]. Some early researches investigated qubit reduction by proposing improved algorithms focus on saving input qubits for factoring an integer when Shor's algorithm is adopted, such as [10, 25], where the number of input qubits can be reduced from $2n$ to $(1 + o(1))n$ and $(1/2 + o(1))n$, respectively. Recently, the authors of [21] studied the problem of period finding with fewer output qubits based on Simon's algorithm and Shor's algorithm, where they can reduce the number of output qubits from n to 1. As the authors stated in [21], "although there is steady progress in constructing larger quantum computers, within the next years the number of qubits seems to be too limited for tackling problems of interesting size" and "quantum computers with a very limited number of qubits might still serve as a powerful oracle that assists us in speeding up classical computations". Note that the method of [21] assumed the oracle access of the quantum embedding of underlying functions, and reduced qubits from the structure of Simon's algorithm or Shor's algorithm. However, it is also of great significance to reduce the oracle qubit consumption of the underlying function itself. Only by combining these two efforts, a quantum circuit with a reduced qubit consumption can be achieved. It is widely believed that algorithms and circuits with better performance in qubit requirements may be physically implemented earlier in a real quantum computer [4, 33, 35]. Therefore, as the authors did in [1, 11, 18–20, 30, 34], this study focuses on constructing quantum circuits for AES with fewer qubits, as it is the core component to construct quantum embeddings of

oracles for quantum attacks. Note that the Clifford+T gate set is also adopted when designing quantum circuits of the AES family [14–16]. However, a Toffoli gate can be constructed by several Clifford gates and T gates. On the other hand, the classical AND gates can be simulated by Toffoli gates, which helps to make better use of classical circuits to construct quantum ones. Thus, this work adopts the Toffoli gate to investigate the AES quantum circuits. Since depth is also an important metric, as the authors did in [15, 34], $T \cdot M$ value (i.e., the product of the Toffoli depth and the number of qubits) is taken as a metric to evaluate the trade-off between depth and qubits.

1.1 Our Contributions

First, we outline the influence of Pauli-X gates, CNOT gates and Toffoli gates on the Toffoli depth of an NCT-based circuit, based on which we illustrate how the CNOT gate consumption is affected by the s-XOR operations. Meanwhile, the influence of the operation order on the Toffoli depth of NCT-based circuits and the conditions under which two consecutive operations are commutative are also discussed.

Then, we rearrange both the classical implementation of AES S-box and its inverse into three parts. Specifically, the tower fields architecture decomposes both the S-box of AES and its inverse into three functions, the top function, the middle function and the bottom function. The first step of rearranging the circuit is to derive the operations for calculating the multiplicative inverse over $\mathbb{F}_{2^4}$ from the circuit of the middle function and treat them as the second part, the first part of the rearranged circuit consists of the operations in the circuit of the middle function for generating the inputs of the second part, while the third part consists of the remaining operations in the circuit of the middle function and the bottom function. Both the first part and the third part of the rearranged circuit take the outputs of the top function as auxiliary variables.

Furthermore, the construction of optimized quantum circuits for AES S-box and its inverse are investigated based on the rearranged circuits with three parts. In this work, the second part that calculates the multiplicative inverse over $\mathbb{F}_{2^4}$ is treated as a 4-bit S-box for the first time, and the public tools LIGHTER [17] and LIGHTER-R [9] are taken into account to design its in-place implementation. Moreover, we detect a quantum style implementation of the third part by adding unit row vectors and making use of the heuristic in [32]. As far as we know, this is the first time that the heuristic proposed for searching optimized s-XOR implementation of linear layers has been applied to build quantum circuits for AES S-box and its inverse. In addition, an algorithm is proposed to search optimized NCT-based circuits for the remaining two parts based on our observations on quantum circuits. The heuristic is designed on the premise of optimizing the Toffoli depth. Moreover, the strategy of randomization is also used to save CNOT gates. Our researches on the construction of NCT-based circuits for S-box and S-box^{-1} enrich the method to build quantum implementations of AES S-box and its inverse based on the classical implementations produced by tower fields architecture.

We applied our methods to the hardware circuits of AES S-box and S-box^{-1} presented in [5] and [34], respectively. The results reveal that the circuits obtained in this paper consume fewer qubits, the CNOT gate consumption and the Toffoli depth are also optimized on the premise of saving qubits. The details of the quantum resource consumption of AES S-box and its inverse are listed in Table 7. In order to implement the key schedule without introducing extra storage qubits, we investigate the implementation of AES S-box with the initial values of outputs are not all 0s and report an optimized circuit that maps $|x\rangle|y\rangle|0\rangle^{\otimes 5}$ to $|x\rangle|y \oplus S(x)\rangle|0\rangle^{\otimes 5}$. Moreover, since removing the previous rounds when expanding the round function can save qubits, then, we investigate the implementation of AES S-box^{-1} with the initial values of outputs are not all 0s and report an optimized circuit that maps $|x\rangle|y\rangle|0\rangle^{\otimes 5}$ to $|x\rangle|y \oplus S^{-1}(x)\rangle|0\rangle^{\otimes 5}$. The comparison of the quantum resource consumption is shown in Table 8.

Finally, we investigate the implementation of AES with various number of S-boxes applied in parallel by the method we call partial zig-zag. Combined with our new technique, NCT-based circuits for all instances of the AES family can be constructed with 269, 333 and 397 qubits, respectively. Moreover, considering the metric of $T \cdot M$ value, our methods guarantee that the NCT-based circuits for the AES family outperform state-of-the-art schemes in the metric of $T \cdot M$ value. The corresponding schemes consume only 474, 538 and 602 qubits. The details are shown in Table 1, Table 2 and Table 3, where m is the number of S-boxes[3] applied in parallel.

1.2 Organization

Section 2 introduces the notations used throughout this paper and gives a brief introduction to AES. Then, some properties of quantum circuit are presented in Section 3. In Section 4, the heuristic for searching optimized quantum circuits for the first and the third parts of our rearranged circuits are reported, as well as the quantum circuits for AES S-box and its inverse. The method to implement the key schedule and the round function are introduced in Section 5, followed by the applications to the AES family in Section 6. Finally, the conclusion and the future work are discussed in Section 7.

[3]Applying m S-boxes in parallel when implementing the **SubBytes** of the current round also means that one can apply m S-box^{-1}es in parallel to remove the previous round, since the circuits designed in this work for AES S-box and its inverse can always be implemented with the same number of ancilla qubits.

Table 1 The quantum resource of different NCT-based circuits for AES-128.

Source		#Qubits	Toffoli Depth	#Toffoli	#CNOT	#Pauli-X	$T \cdot M$
[11]		984	12672	151552	166548	1456	12469248
[1]		976	NR	150528	192832	1370	NR
[19]		864	1880	16940	107960	1570	1624320
[34]		512	2016	19788	128517	4528	1032192
[30]		656	NR	18040	101174	1976	NR
		400	NR	19064	118980	4528	NR
[14]*		492	820	17888	126016	2528	403440
		374	1558				582692
[18]◇		984	11088	NR	NR	NR	10910592
		2208	1260	NR	NR	NR	2782080
[20]◇		270	11008	16508	81652	1072	2972160
		400	1108	15824	82928		443200
[15]◇		3936	76	12920	84120	800	299136
		6368	40	12240	81312		254720
This work	$m = 1$	269	7396	19608	77408	2224	1989524
	$m = 1^\dagger$	274	6480		78448		1775520
	$m = 2$	282	3720		77408		1049040
	$m = 2^\dagger$	287	3306		78416		948822
	$m = 3$	295	2622		77444		773490
	$m = 4$	308	1970		77408		606760
	$m = 4^\dagger$	313	1700		78272		532100
	$m = 5$	321	1736		77444		557256
	$m = 6$	334	1304		77552		435536
	$m = 7$	347	1304		77480		452488
	$m = 8$	360	1106		77408		398160
	$m = 8^\dagger$	365	908		77984		331420
	$m = 9$	373	872		77660		325256
	$m = 10$	386	872		77624		336592
	$m = 11$	399	872		77588		347928
	$m = 12$	412	872		77552		359264
	$m = 13$	425	872		77516		370600
	$m = 14$	438	872		77480		381936
	$m = 15$	451	872		77444		393272
	$m = 16$	464	674		77408		312736
	$m = 16^\dagger$	474	476		77984		225624

∗ The quantum resource consumption listed in the table is from Table 6 in [14].
◇ Only the circuit costs fewest qubits and the one with lowest $T \cdot M$ value in the reference are listed.
† The S-boxes for the key schedule that are applied in parallel with the S-boxes for the round function or the S-box^{-1}es for removing the previous round by adding 5 or 10 ancilla qubits.

2 Preliminaries

2.1 Notations

$\mathbb{Z}_+$	the set of all positive integers
$\mathbb{F}_2$	the finite field containing elements 0 and 1
$\mathbb{F}_{2^k}$	the finite field containing 2^k elements
$a \oplus b$	the XOR of bits a and b over $\mathbb{F}_2$
$a \cdot b$	the AND of bits a and b over $\mathbb{F}_2$
$\overline{a}$	the inversion of bit a over $\mathbb{F}_2$

Definition 1 (NCT-based Circuit) *An NCT-based circuit is a quantum circuit constructed based on Pauli-X gates, CNOT gates and Toffoli gates.*

The circuit symbols and functions of the Pauli-X gate, CNOT gate and Toffoli gate are depicted in Figure 1, where $a, b, c \in \mathbb{F}_2$.

Table 2 The quantum resource of different NCT-based circuits for AES-192.

Source		#Qubits	Toffoli Depth	#Toffoli	#CNOT	#Pauli-X	$T \cdot M$
[11]		1112	11088	172032	189432	1608	12329856
[19]		896	1640	19580	125580	1692	1469440
[34]		640	2022	22380	152378	5128	1294080
[20]$^\diamond$		334	13144	19196	94180	1160	4390096
		464	1340	18400	95696		621760
[15]$^\diamond$		4256	92	14688	96112	896	391552
		6688	48	14008	92856		321024
This work	$m = 1$	333	8844	22800	90384	2568	2945052
	$m = 1^\dagger$	338	7904		91408		2671552
	$m = 2$	346	4444		90384		1537624
	$m = 2^\dagger$	351	4026		91360		1413126
	$m = 3$	359	3190		90428		1145210
	$m = 4$	372	2310		90384		859320
	$m = 4^\dagger$	377	2068		91184		779636
	$m = 5$	385	2112		90428		813120
	$m = 6$	398	1584		90560		630432
	$m = 7$	411	1584		90472		651024
	$m = 8$	424	1254		90384		531696
	$m = 8^\dagger$	429	1100		90832		471900
	$m = 9$	437	1056		90692		461472
	$m = 10$	450	1056		90648		475200
	$m = 11$	463	1056		90604		488928
	$m = 12$	476	1056		90560		502656
	$m = 13$	489	1056		90516		516384
	$m = 14$	502	1056		90472		530112
	$m = 15$	515	1056		90428		543840
	$m = 16$	528	726		90384		383328
	$m = 16^\dagger$	538	572		90832		307736

$\diamond$ Only the circuit costs fewest qubits and the one with lowest $T \cdot M$ value in the reference are listed.
† The S-boxes for the key schedule that are applied in parallel with the S-boxes for the round function or the S-box^{-1}es for removing the previous round by adding 5 or 10 ancilla qubits.

Table 3 The quantum resource of different NCT-based circuits for AES-256.

Source		#Qubits	Toffoli Depth	#Toffoli	#CNOT	#Pauli-X	$T \cdot M$
[11]		1336	14976	215040	233836	1943	20007936
[19]		1232	2160	23760	151011	1992	2661120
[34]		768	2292	26774	177645	6103	1760256
[20]$^\diamond$		398	15756	23228	114476	1367	6270888
		528	1540	22264	116288		813120
[15]$^\diamond$		4576	108	18088	117704	1103	494208
		6976	56	17408	113744		390656
This work	$m = 1$	397	10622	27816	109856	3069	4216934
	$m = 1^\dagger$	402	9322		111416		3747444
	$m = 2$	410	5324		109830		2182840
	$m = 2^\dagger$	415	4724		111312		1960460
	$m = 3$	423	3736		109908		1580328
	$m = 4$	436	2826		109856		1232136
	$m = 4^\dagger$	441	2436		111104		1074276
	$m = 5$	449	2488		109908		1117112
	$m = 6$	462	1864		110064		861168
	$m = 7$	475	1844		109920		875900
	$m = 8$	488	1556		109856		759328
	$m = 8^\dagger$	493	1270		110688		626110
	$m = 9$	501	1218		110220		610218
	$m = 10$	514	1218		110168		626052
	$m = 11$	527	1218		110116		641886
	$m = 12$	540	1218		110064		657720
	$m = 13$	553	1218		110012		673554
	$m = 14$	566	1218		109960		689388
	$m = 15$	579	1218		109908		705222
	$m = 16$	592	932		109856		551744
	$m = 16^\dagger$	602	646		110688		388892

$\diamond$ Only the circuit costs fewest qubits and the one with lowest $T \cdot M$ value in the reference are listed.
† The S-boxes for the key schedule that are applied in parallel with the S-boxes for the round function or the S-box^{-1}es for removing the previous round by adding 5 or 10 ancilla qubits.

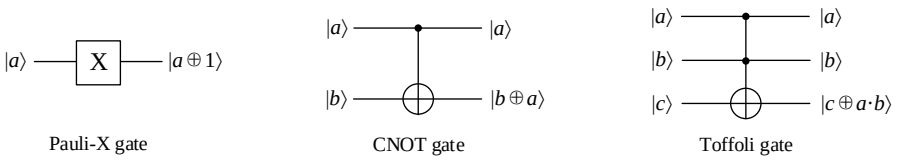

Fig. 1 The description of the underlying quantum gates.

Besides, a CNOT gate can be regarded as the transformation that maps $|a\rangle|b\rangle$ to $|a\rangle|b \oplus a\rangle$, the operand b is updated as $b = b \oplus a$. Consequently, the application of CNOT gates can be simulated by XOR operations under s-XOR metric, which is originally a concept for the implementation of matrices.

Definition 2 (s-XOR [17]) *Let M be an invertible matrix over $\mathbb{F}_2$ with size $n \times n$. Assuming that $x_0, x_1, ..., x_{n-1}$ are the n input bits of M. It is always possible to perform a sequence of* XOR *operations $x_i = x_i \oplus x_j$ with $0 \leq i, j \leq n-1$, such that the n input bits are updated to the n output bits. The s-*XOR *count of M is defined as the minimal number of such* XOR *operations to update the inputs to the outputs.*

2.2 Description of the AES Family

The AES family [8] contains three instances, denoted as AES-128, AES-192 and AES-256 respectively according to the length of the key.

Round Function The round function of the AES family consists of four transformations, i.e., **SubBytes**, **ShiftRows**, **MixColumns** and **AddRoundKey** as shown in Figure 2, where r is the round number and equals 10, 12 and 14 for AES-128, AES-192 and AES-256, respectively. The **SubBytes** replaces each byte in the state by another one according to the S-box. The **ShiftRows** changes the position of the bytes in the grid by cyclically rotating the bytes in the ith row to the left by i bytes, where $i = 0, 1, 2, 3$. The **MixColumns** is a linear transformation and it multiplies the right circulant matrix $(0x02, 0x03, 0x01, 0x01)$ over $\mathbb{F}_{2^8}$ with each column of the state. Note that the **MixColumns** is absent in the last round. The **AddRoundKey** adds the round key to the state by bitwise XOR.

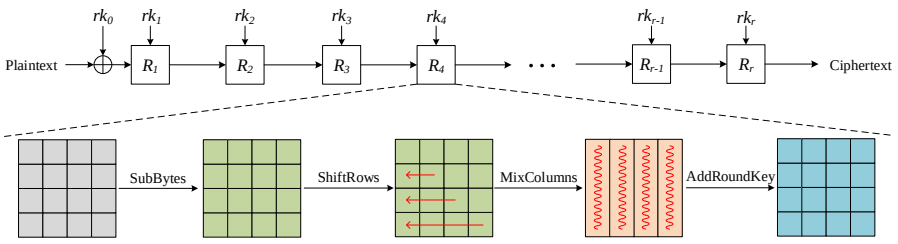

Fig. 2 The encryption of the AES family.

Key Schedule The key schedule of AES is based on 32-bit words. Denote the master key by $W_0, W_1, ..., W_{s-1}$, where $s = 4$ for AES-128, $= 6$ for AES-192, $= 8$ for AES-256. Except the given words (i.e., the words in the master key), 40, 46 and 52 words are required by AES-128, AES-192 and AES-256 respectively.

For AES-128, the word W_i can be calculated by

$$W_i = \begin{cases} W_{i-4} \oplus \mathbf{SubWord}(\mathbf{RotWord}(W_{i-1})) \oplus \mathbf{Rcon}(i/4), & \text{if } i \equiv 0 \bmod 4, \\ W_{i-4} \oplus W_{i-1}, & \text{otherwise,} \end{cases}$$

where $i = 4, 5, ..., 43$.

For AES-192, the word W_i can be calculated by

$$W_i = \begin{cases} W_{i-6} \oplus \mathbf{SubWord}(\mathbf{RotWord}(W_{i-1})) \oplus \mathbf{Rcon}(i/6), & \text{if } i \equiv 0 \bmod 6, \\ W_{i-6} \oplus W_{i-1}, & \text{otherwise,} \end{cases}$$

where $i = 6, 7, ..., 51$.

For AES-256, the word W_i can be calculated by

$$W_i = \begin{cases} W_{i-8} \oplus \mathbf{SubWord}(\mathbf{RotWord}(W_{i-1})) \oplus \mathbf{Rcon}(i/8), & \text{if } i \equiv 0 \bmod 8, \\ W_{i-8} \oplus \mathbf{SubWord}(W_{i-1}), & \text{if } i \equiv 4 \bmod 8, \\ W_{i-8} \oplus W_{i-1}, & \text{otherwise,} \end{cases}$$

where $i = 8, 9, ..., 59$.

The **SubWord** applies four S-boxes to the bytes in one word. The **RotWord** cyclically rotates the bytes in the word to the left by one byte. The **Rcon** adds the round constant to the word by bitwise XOR.

2.3 Classical Implementations of AES Building Blocks

2.3.1 Classical Implementations of MixColumn

The transformation of **MixColumn** can be represented as a 32×32 binary matrix over $\mathbb{F}_2$. Among the methods of matrix implementation, LUP-type decomposition [29] can be used to generate an implementation of **MixColumn** under s-XOR metric. In an s-XOR implementation, the outputs are stored in the input registers and no extra registers are needed. Meanwhile, one can easily simulate an XOR operation under s-XOR metric by a CNOT gate. This is an important reason why the LUP-type decomposition method is commonly used when constructing quantum circuits for **MixColumn** [1, 11, 16, 19, 31]. Also based on matrix decomposition theory, Xiang *et al.* [32] presented an implementation of **MixColumn** with 92 XOR operations. Considering the gate consumption and the convenience of being converted to a quantum implementation, the s-XOR implementation presented in [32] is used in this work to build the quantum circuit for the **MixColumns**.

2.3.2 Classical Implementations of AES S-box and S-box^{-1}

As the only nonlinear building block of AES, the implementation of S-box has a crucial impact on the overall implementation performance of the cipher. Due to the advantage in obtaining an efficient implementation of AES S-box with a lower gate count, tower fields architecture is widely used in the field of constructing circuits for AES in hardware application scenarios [5, 6, 22, 31]. Designing quantum circuits from these classical implementations seems to be a popular approach in recent years. This section investigates the construction of efficient quantum circuits for AES based on the circuit of the S-box reported in [5] and the circuit of the S-box^{-1} given in [34]. By exploiting the tower fields architecture, Boyar *et al.* [5] decomposed AES S-box into three transformations and represented the S-box as $S(x) = B \cdot F(U \cdot x)$, where x is the 8-bit input of the S-box. Similarly, Zou *et al.* [34] represented the S-box^{-1} of AES as $S^{-1}(x) = B^{'} \cdot F^{'}(U^{'} \cdot x)$, where x is the 8-bit input of S-box^{-1}. For simplicity, this section only lists the classical circuit reported in [5].

Top Function U Denote the input of S-box by $(x_0, x_1, ..., x_7)$, the function U takes $(x_0, x_1, ..., x_7)$ as its input and generates $(y_0, y_1, ..., y_{21})$, which can be calculated as

$$
\begin{array}{lllll}
y_0 = x_7, & y_{14} = x_3 \oplus x_5, & y_{13} = x_0 \oplus x_6, & y_9 = x_0 \oplus x_3, & y_8 = x_0 \oplus x_5, \\
t_0 = x_1 \oplus x_2, & y_1 = t_0 \oplus x_7, & y_4 = y_1 \oplus x_3, & y_{12} = y_{13} \oplus y_{14}, & y_2 = y_1 \oplus x_0, \\
y_5 = y_1 \oplus x_6, & y_3 = y_5 \oplus y_8, & t_1 = x_4 \oplus y_{12}, & y_{15} = t_1 \oplus x_5, & y_{20} = t_1 \oplus x_1, \\
y_6 = y_{15} \oplus x_7, & y_{10} = y_{15} \oplus t_0, & y_{11} = y_{20} \oplus y_9, & y_7 = x_7 \oplus y_{11}, & y_{17} = y_{10} \oplus y_{11}, \\
y_{19} = y_{10} \oplus y_8, & y_{16} = t_0 \oplus y_{11}, & y_{21} = y_{13} \oplus y_{16}, & y_{18} = x_0 \oplus y_{16}. &
\end{array}
$$

Middle Function F The function F takes $(y_0, y_1, ..., y_{21})$ as its inputs and generates $(z_0, z_1, ..., z_{17})$, which can be calculated as

$$
\begin{array}{lllll}
t_2 = y_{12} \cdot y_{15}, & t_3 = y_3 \cdot y_6, & t_4 = t_3 \oplus t_2, & t_5 = y_4 \cdot y_0, & t_6 = t_5 \oplus t_2, \\
t_7 = y_{13} \cdot y_{16}, & t_8 = y_5 \cdot y_1, & t_9 = t_8 \oplus t_7, & t_{10} = y_2 \cdot y_7, & t_{11} = t_{10} \oplus t_7, \\
t_{12} = y_9 \cdot y_{11}, & t_{13} = y_{14} \cdot y_{17}, & t_{14} = t_{13} \oplus t_{12}, & t_{15} = y_8 \cdot y_{10}, & t_{16} = t_{15} \oplus t_{12}, \\
t_{17} = t_4 \oplus y_{20}, & t_{18} = t_6 \oplus t_{16}, & t_{19} = t_9 \oplus t_{14}, & t_{20} = t_{11} \oplus t_{16}, & t_{21} = t_{17} \oplus t_{14}, \\
t_{22} = t_{18} \oplus y_{19}, & t_{23} = t_{19} \oplus y_{21}, & t_{24} = t_{20} \oplus y_{18}, & t_{25} = t_{21} \oplus t_{22}, & t_{26} = t_{21} \cdot t_{23}, \\
t_{27} = t_{24} \oplus t_{26}, & t_{28} = t_{25} \cdot t_{27}, & t_{29} = t_{28} \oplus t_{22}, & t_{30} = t_{23} \oplus t_{24}, & t_{31} = t_{22} \oplus t_{26}, \\
t_{32} = t_{31} \cdot t_{30}, & t_{33} = t_{32} \oplus t_{24}, & t_{34} = t_{23} \oplus t_{33}, & t_{35} = t_{27} \oplus t_{33}, & t_{36} = t_{24} \cdot t_{35}, \\
t_{37} = t_{36} \oplus t_{34}, & t_{38} = t_{27} \oplus t_{36}, & t_{39} = t_{29} \cdot t_{38}, & t_{40} = t_{25} \oplus t_{39}, & t_{41} = t_{40} \oplus t_{37}, \\
t_{42} = t_{29} \oplus t_{33}, & t_{43} = t_{29} \oplus t_{40}, & t_{44} = t_{33} \oplus t_{37}, & t_{45} = t_{42} \oplus t_{41}, & z_0 = t_{44} \cdot y_{15}, \\
z_1 = t_{37} \cdot y_6, & z_2 = t_{33} \cdot y_0, & z_3 = t_{43} \cdot y_{16}, & z_4 = t_{40} \cdot y_1, & z_5 = t_{29} \cdot y_7, \\
z_6 = t_{42} \cdot y_{11}, & z_7 = t_{45} \cdot y_{17}, & z_8 = t_{41} \cdot y_{10}, & z_9 = t_{44} \cdot y_{12}, & z_{10} = t_{37} \cdot y_3, \\
z_{11} = t_{33} \cdot y_4, & z_{12} = t_{43} \cdot y_{13}, & z_{13} = t_{40} \cdot y_5, & z_{14} = t_{29} \cdot y_2, & z_{15} = t_{42} \cdot y_9, \\
z_{16} = t_{45} \cdot y_{14}, & z_{17} = t_{41} \cdot y_8. &
\end{array}
$$

Bottom Function B Denote the output of the S-box by $(s_0, s_1, ..., s_7)$. The function B takes $(z_0, z_1, ..., z_{17})$ as inputs and generates $(s_0, s_1, ..., s_7)$, which can be calculated as

$$
\begin{array}{lllll}
t_{46} = z_{15} \oplus z_{16}, & t_{47} = z_{10} \oplus z_{11}, & t_{48} = z_5 \oplus z_{13}, & t_{49} = z_9 \oplus z_{10}, & t_{50} = z_2 \oplus z_{12}, \\
t_{51} = z_2 \oplus z_5, & t_{52} = z_7 \oplus z_8, & t_{53} = z_0 \oplus z_3, & t_{54} = z_6 \oplus z_7, & t_{55} = z_{16} \oplus z_{17}, \\
t_{56} = z_{12} \oplus t_{48}, & t_{57} = t_{50} \oplus t_{53}, & t_{58} = z_4 \oplus t_{46}, & t_{59} = z_3 \oplus t_{54}, & t_{60} = t_{46} \oplus t_{57}, \\
t_{61} = z_{14} \oplus t_{57}, & t_{62} = t_{52} \oplus t_{58}, & t_{63} = t_{49} \oplus t_{58}, & t_{64} = z_4 \oplus t_{59}, & t_{65} = t_{61} \oplus t_{62}, \\
t_{66} = z_1 \oplus t_{63}, & s_0 = t_{59} \oplus t_{63}, & s_6 = \overline{t_{56} \oplus t_{62}}, & s_7 = \overline{t_{48} \oplus t_{60}}, & t_{67} = t_{64} \oplus t_{65}, \\
s_3 = t_{53} \oplus t_{66}, & s_4 = t_{51} \oplus t_{66}, & s_5 = t_{47} \oplus t_{65}, & s_1 = \overline{t_{64} \oplus s_3}, & s_2 = \overline{t_{55} \oplus t_{67}}.
\end{array}
$$

3 Observations on NCT-based Circuits

Quantum Toffoli Depth Although linear operations themselves are considered not to increase the Toffoli depth, but the propagation of Toffoli depth caused by CNOT gates cannot be ignored. If the Toffoli depth of two variables are the same before they are taken as the inputs of a CNOT gate, the depth of these two variables remain unchanged after the CNOT gate, which is beyond doubt. But if the Toffoli depth of the operands of a CNOT gate are not the same, the Toffoli depth for one of the operands should be changed. We give the following properties to illustrate the update of Toffoli depth caused by logic gates in an NCT-based circuit.

Property 1 *For a Pauli-X gate that maps* $|a\rangle$ *to* $|a \oplus 1\rangle$*, the application of the Pauli-X gate will not change the Toffoli depth of* a*.*

Property 2 *For a CNOT gate that maps* $|a\rangle|b\rangle$ *to* $|a\rangle|b \oplus a\rangle$*, denote the input Toffoli depth of* a *and* b *by* d_a *and* d_b *respectively. After the application of the CNOT gate,* d_a *and* d_b *are updated as*

$$d_a = d_b = max\{d_a, d_b\}.$$

Property 3 *For a Toffoli gate that maps* $|a\rangle|b\rangle|c\rangle$ *to* $|a\rangle|b\rangle|c \oplus a \cdot b\rangle$*, denote the input Toffoli depth of* a*,* b *and* c *by* d_a*,* d_b *and* d_c *respectively. After the application of the Toffoli gate,* d_a*,* d_b *and* d_c *are updated as*

$$d_a = d_b = d_c = max\{d_a, d_b, d_c\} + 1.$$

Example 1 demonstrates the update of Toffoli depth caused by CNOT gates and Toffoli gates.

Example 1 Take *Circuit 1* and *Circuit 2* listed in Table 4 as an example. Suppose that the initial Toffoli depth of all variables is zero. Denote the Toffoli depth of $a, b, ..., g$ by $(d_a, d_b, ..., d_g)$, where d_i is the Toffoli depth of variable i and $i \in \{a, b, ..., g\}$. The evolution of the Toffoli depth vector at each step are listed in the 3rd and 6th columns in Table 4.

Table 4 The Toffoli depth of each operation.

No.	Circuit 1	Toffoli depth	No.	Circuit 2	Toffoli depth
1	$a = a \oplus b$	$(0,0,0,0,0,0,0)$	1	$b = b \oplus a$	$(0,0,0,0,0,0,0)$
2	$c = c \oplus a \cdot d$	$(1,0,1,1,0,0,0)$	2	$c = c \oplus b \cdot d$	$(0,1,1,1,0,0,0)$
3	$b = b \oplus e$	$(1,0,1,1,0,0,0)$	3	$b = b \oplus a \oplus e$	$(1,1,1,1,1,0,0)$
4	$f = f \oplus b \cdot g$	$(1,1,1,1,0,1,1)$	4	$f = f \oplus b \cdot g$	$(1,2,1,1,1,2,2)$

One can easily check that both circuits listed in Table 4 perform the same function. However, *Circuit 2* costs one more CNOT gate than *Circuit 1* (caused by the third operation in *Circuit 2*). Besides, the Toffoli depth of *Circuit 2* is two, while the Toffoli depth of *Circuit 1* is one. The only difference between *Circuit 1* and *Circuit 2* is the variable chosen to store the intermediate value $a \oplus b$ in the first operation, by which, the circuits in Table 4 show the effect of selecting a specific bit to store the result of an s-XOR operation on the overall Toffoli depth of a quantum circuit. This can be summarized with the following observation.

Observation 1 *Given a quantum circuit with Toffoli gates involved, the Toffoli depth and the CNOT gate consumption of the quantum circuit may be affected by the specific arrangement of CNOT gates.*

In addition, take the third operation of *Circuit 2* in Table 4 (i.e., $b = b \oplus a \oplus e$) as an example, among the three operands, the Toffoli depth of b is 1 while other operands are with Toffoli depth 0. The execution of the third operation causes the Toffoli depth of a and e to increase by 1 due to the influence of b, which has a higher Toffoli depth. But what if the value $b \oplus e$ (the target value of the third operation) can be obtained before the Toffoli depth of b is increased? This inspires us to investigate the effect of the order of operations on Toffoli depth and give rise to the following observation.

Observation 2 *Given a quantum circuit with Toffoli gates involved, the Toffoli depth of the circuit may be affected by the order of operations.*

Example 2 For a quantum circuit denoted by *Circuit 3* in Table 5, a is not the operand of the second operation, and d is not the operand of the first operation. Consequently, the first two operations in *Circuit 3* are commutative, as shown with *Circuit 4* in Table 5. Thus, the Toffoli depth can be reduced by 1 as listed in the sixth column of Table 5.

Table 5 The Toffolil depth of the operations.

No.	Circuit 3	Depth vector	No.	Circuit 4	Depth vector
1	$a = a \oplus b \cdot c$	$(1,1,1,0,0,0)$	1	$d = d \oplus b$	$(0,0,0,0,0,0)$
2	$d = d \oplus b$	$(1,1,1,1,0,0)$	2	$a = a \oplus b \cdot c$	$(1,1,1,0,0,0)$
3	$e = e \oplus d \cdot f$	$(1,1,1,2,2,2)$	3	$e = e \oplus d \cdot f$	$(1,1,1,1,1,1)$

Note that it is not always possible to exchange two consecutive operations. Denote qubit by variable t in the following facts, it follows that t_i and t_j are two different qubits if and only if $i \neq j$.

Fact 1 *Given a quantum circuit with m qubits $t_0, t_1, ..., t_{m-1}$, if two consecutive operations are in the form of $t_a = t_a \oplus t_b, t_c = t_c \oplus t_d$, where $a, b, c, d \in [0, m-1]$, $a \neq b$ and $c \neq d$, the order of these two operations can be exchanged when one of the following conditions holds: (i) $a = c$; (ii) $a \neq c, d$ and $b \neq c$.*

Fact 2 *Given a quantum implementation with the m involved qubits are denoted by $t_0, t_1, ..., t_{m-1}$, if two consecutive operations are in the form of $t_a = t_a \oplus t_b, t_c = t_c \oplus t_d \cdot t_e$ or vice versa, where $a, b, ..., e \in [0, m-1]$, $a \neq b$ and $c \neq d \neq e$, the order of these two operations can be exchanged when one of the following conditions holds: (i) $a = c$; (ii) $a \neq c, d, e$ and $b \neq c$.*

Fact 3 *Given a quantum circuit with m qubits $t_0, t_1, ..., t_{m-1}$, if two consecutive operations are in the form of $t_a = t_a \oplus t_b \cdot t_c, t_d = t_d \oplus t_e \cdot t_f$, where $a, b, ..., f \in [0, m-1]$, $a \neq b \neq c$, $d \neq e \neq f$, the order of these two operations can be exchanged when one of the following conditions holds: (i) $a = d$; (ii) $a \neq d$, e, f and $d \neq b$, c.*

The proof of Fact 1 is given in Appendix A, Fact 2 - 3 can be proved in the same way.

4 New NCT-based Circuits of AES S-box and S-box^{-1}

The quantum circuit of AES S-box^{-1} can be constructed from the classical one presented in [34], which was decomposed in the same way as the authors did in [5] to represent AES S-box, or from the quantum circuit designed for the S-box by adding some linear transformations [14], which dose not affect the structure of the classical circuit presented in [5]. Therefore, only the optimized quantum implementation of AES S-box is discussed in this section, the S-box^{-1} of AES can be implemented similarly.

4.1 Observations on the Adopted Classical Circuits of S-box

Middle Functions F For the implementation of F reported in [5] (as listed in Subsection 2.3), Zou *et al.* [34] pointed out that the outputs of F can be calculated with the knowledge of $t_{29}, t_{33}, t_{37}, t_{40}$ and the inputs of AES S-box. Furthermore, one can easily find that $t_{29}, t_{33}, t_{37}, t_{40}$ are the outputs of the multiplicative inverse in $\mathbb{F}_{2^4}$, and $t_{21}, t_{22}, t_{23}, t_{24}$ are the inputs. Essentially, the function that maps $(t_{21}, t_{22}, t_{23}, t_{24})$ to $(t_{29}, t_{33}, t_{37}, t_{40})$ is a permutation and thus can be regarded as a 4-bit S-box as shown in Table 6.

Table 6 The 4-bit S-box within F.

$(t_{21}, t_{22}, t_{23}, t_{24})$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$(t_{29}, t_{33}, t_{37}, t_{40})$	0	6	2	4	9	3	d	5	1	e	c	7	8	a	b	f

Compared with searching the s-XOR implementation for linear layers, the design of the quantum implementation of S-boxes is tricky, especially for large S-boxes. Nevertheless, for a 4-bit S-box, the public tools LIGHTER[4] and LIGHTER-R[5], which are proposed in [17] and [9] respectively, can be used to search an optimized reversible circuit with fewer logic gates. However, only LIGHTER is used in this paper for the 4-bit S-box shown in Table 6. The discussion on LIGHTER and LIGHTER-R is presented in Appendix B.

Bottom Functions B The function B generates the outputs of AES S-box, which are linear expressions of z_i, where $i = 0, 1, ..., 17$. As pointed in [34], B can be expressed as a matrix. Note that the matrix corresponding to B is of size 8×18 and rank 8. In order to obtain an optimized s-XOR implementation of B, one can extend its corresponding matrix to be invertible by adding unit row vectors. Then, the heuristic[6] proposed in [32] can be used.

4.2 Heuristic for Searching Optimized NCT-based Circuits for S-box

According to the analysis in Subsection 4.1, the middle functions F can be divided into three parts. The first part takes $(y_0, y_1, ..., y_{21})$ (i.e., the outputs of the top function U) as inputs and generates $(t_{21}, t_{22}, t_{23}, t_{24})$ as outputs. In this section, the first part of the middle function F and the top function U are combined and denoted by f_1, which takes $(x_0, x_1..., x_7)$ as inputs and generates $(t_{21}, t_{22}, t_{23}, t_{24})$ as outputs. The second part of the middle function F is a 4-bit S-box which is denoted by S_4 as shown in Table 6. S_4 takes $(t_{21}, t_{22}, t_{23}, t_{24})$ as inputs and generates $(t_{29}, t_{33}, t_{37}, t_{40})$ as outputs. Similarly, the third part of the middle function F, the top function U and the bottom function B are combined and denoted by f_2, which takes $(t_{29}, t_{33}, t_{37}, t_{40})$ (i.e., the outputs of the 4-bit S-box) and $(x_0, x_1..., x_7)$ as inputs and calculates $(s_0, s_1..., s_7)$ as outputs. The reversible circuit of S_4 with 2-input AND gates, 2-input XOR gates and 1-input NOT gates can be obtained with LIGHTER by introducing an additional variable. Consequently, this section focuses on constructing quantum circuits for f_1 and f_2 with a lower Toffoli depth as it is another important factor that affects the metric of $T \cdot M$ value. The main idea is to try to execute more nonlinear operations in parallel.

In the following, f_1 is taken as an example to illustrate how to get an optimized quantum circuit. Denote X and S the input set and the output set of f_1, i.e., $X = \{x_0, x_1, ..., x_7\}$ and $S = \{t_{21}, t_{22}, t_{23}, t_{24}\}$. According to the classical implementation of the S-box, the implementation of f_1 is listed as

[4] http://jeremy.jean.free.fr/pub/fse2018_layer_implementations.tar.gz
[5] https://github.com/vdasu/lighter-r
[6] https://github.com/xiangzejun/Optimizing_Implementations_of_Linear_Layers

follows.

$$
\begin{array}{llll}
t_{21} = t_{21} \oplus y_{12} \cdot y_{15}, & t_{22} = t_{22} \oplus t_{21}, & t_{21} = t_{21} \oplus y_3 \cdot y_6, & t_{22} = t_{22} \oplus y_4 \cdot y_0, \\
t_{22} = t_{22} \oplus y_8 \cdot y_{10}, & t_{23} = t_{23} \oplus y_{14} \cdot y_{17}, & t_{21} = t_{21} \oplus t_{23}, & t_{23} = t_{23} \oplus y_5 \cdot y_1, \\
t_{23} = t_{23} \oplus y_{13} \cdot y_{16}, & t_{24} = t_{24} \oplus y_2 \cdot y_7, & t_{24} = t_{24} \oplus y_{13} \cdot y_{16}, & t_{24} = t_{24} \oplus y_8 \cdot y_{10}, \\
a = a \oplus y_9 \cdot y_{11}, & t_{21} = t_{21} \oplus a, & t_{22} = t_{22} \oplus a, & t_{23} = t_{23} \oplus a, \\
t_{24} = t_{24} \oplus a, & a = a \oplus y_9 \cdot y_{11}, & t_{21} = t_{21} \oplus y_{20}, & t_{22} = t_{22} \oplus y_{19}, \\
t_{23} = t_{23} \oplus y_{21}, & t_{24} = t_{24} \oplus y_{18}, & &
\end{array}
$$

where a is an ancilla qubit, and y_i $(i = 0, 1, ..., 21)$ is the output of the top function U and linear related to $x_0, x_1, ..., x_7$.

The circuit shown above is obtained by simply eliminating redundant temporary variables in the classical implementation and rewriting it in a quantum style. Note that one ancilla qubit is allocated for f_1, this is due to the fact that the 4-bit S-box S_4 is an odd permutation, and at least one ancilla qubit is needed to construct its in-place implementation [26]. Thus, this ancilla qubit can be used in f_1 before the implementation of S_4, however, it should be reset to 0 and be reused to construct the quantum circuit for S_4.

Denote by Y the set of auxiliary variables, it follows that $Y = \{y_0, y_1, ..., y_{21}\}$ for f_1. For the sake of saving qubits, there is no need to precompute all the values of y_i when implementing f_1, as this needs at least $22 - 8 = 14$ extra qubits. Specifically, the values of y_i are computed on the fly. Taking $t_{21} = t_{21} \oplus y_{12} \cdot y_{15}$ as an example, the values of y_{12} and y_{15} are computed in an in-place manner when needed, that is the s-XOR metric is adopted to update the value of two qubits of $(x_0, x_1, ..., x_7)$ to be equal to y_{12} and y_{15}. After the computation of t_{21} is completed, $(x_0, x_1, ..., x_7)$ can be updated for the following operations in a similar way. Moreover, executing nonlinear operations in parallel as much as possible is the main idea of this work to reduce the depth. Assume that the operations $t_{21} = t_{21} \oplus y_{12} \cdot y_{15}$ and $t_{22} = t_{22} \oplus y_4 \cdot y_0$ are executed parallelly, it requires that $(x_0, x_1, ..., x_7)$ should be updated such that four of which equal to the value of y_{12}, y_{15}, y_4 and y_0. However, this is not always possible.

Property 4 *Let $y_i, i \in [0, m-1]$ be m linear combinations of $x_0, x_1, ..., x_{n-1}$, with $m \leq n$. $x_0, x_1, ..., x_{n-1}$ can be updated under s-XOR metric such that m of which are equal to $y_0, y_1, ..., y_{m-1}$ if and only if $y_0, y_1, ..., y_{m-1}$ are linear independent. In this case, the s-XOR implementations of $y_0, y_1, ..., y_{m-1}$ can be stored in m qubits of $x_0, x_1, ..., x_{n-1}$.*

Algorithm 1 presents a procedure to classify the nonlinear operations of f_1 and f_2 that can be performed concurrently, the usage of which is illustrated by taking f_1 as an example.

Algorithm 1 Classification of the Nonlinear Operations

Input: The implementation (denoted by Imp) for f_i (i = 1, 2) with input set X and output set S, the expressions of the auxiliary variables in Y;

Output: The classification of the nonlinear operations **Classify**(Imp, Y) of Imp;

$E \leftarrow \varnothing$; ▷ The set of classified nonlinear operations;
$l \leftarrow |Imp|$; ▷ The number of operations in Imp;
$N \leftarrow 0$; ▷ The number of elements in E;
$C_0 \leftarrow \varnothing$; ▷ The first set of nonlinear operations to be applied in parallel;
for $i = 0, l-1$ **do**
  $flag \leftarrow false$;
  if the ith operation o_i is nonlinear, i.e., formed as $t_{i_0} = t_{i_0} \oplus y_{j_0} \cdot y_{j_1}$ **then**
    if $C_0 = \varnothing$ **then**
      $C_0 = C_0 \cup \{o_i\}$;
    else
      for $j = 0, N$ **do**
        if o_i can be moved to be adjacent to the last operation in C_j **then**
          if o_i shares no operand with any operation in C_j **then**
            if all y's in $o_i \cup C_j$ are linear independent **then**
              $C_j = C_j \cup \{o_i\}$;
              $flag \leftarrow true$;
              break;
            end if
          end if
        end if
      end for
      if $flag = false$ **then**
        $N = N + 1$;
        $C_N \leftarrow \varnothing$;
        $C_N = C_N \cup \{o_i\}$;
      end if
    end if
  end if
end for
for $i = 0, N$ **do**
  $E = E \cup \{C_i\}$;
end for
return $E = \{C_0, C_1, ..., C_N\}$;

Example 3 First, the set E used to store the classification of the nonlinear operations should be initialized to be empty. Update C_0 as $C_0 = \{t_{21} = t_{21} \oplus y_{12} \cdot y_{15}\}$ since the first operation is nonlinear and the set E is empty. The next nonlinear operation $t_{21} = t_{21} \oplus y_3 \cdot y_6$ can not be moved to be adjacent with the operation in C_0 due to the second operation in the implementation. Thus, it should be added to C_1. The third nonlinear operation $t_{22} = t_{22} \oplus y_4 \cdot y_0$ can be moved to be adjacent with the operation in C_0, and y_{12}, y_{15}, y_4, y_0 are linear independent. According to Property 4, the third nonlinear operation can be executed in parallel with the operation in C_0. Hence, $t_{22} = t_{22} \oplus y_4 \cdot y_0$ can be added to C_0. The fourth nonlinear operation $t_{22} = t_{22} \oplus y_8 \cdot y_{10}$ shares the operand t_{22} with the second operation in C_0 and can be added to C_1. The remaining nonlinear operations can be analyzed similarly and the process ends

Algorithm 2 Search Optimized NCT-based Circuits

Input: The implementation (denoted by Imp) for f_i $(i = 1, 2)$ with input set X and output set S, the expressions of the auxiliary variables in Y;
Output: Optimized NCT-based circuit of f_i;

$E \leftarrow \varnothing$; ▷ The set to be expanded;
Rearrange Imp randomly according to Fact 1 - 3;
$E \leftarrow$ **Classify**(Imp, Y); ▷ Algorithm 1
$N \leftarrow |E|$; ▷ The number of elements in E;
for $i = 0, N-1$ **do**
    Move the operations in C_i to be adjacent;
    $Index \leftarrow \varnothing$;
    $l \leftarrow |C_i|$; ▷ The number of elements in C_i;
    for $j = 0, l-1$ **do**
        $t \leftarrow$ the number of auxiliary variables in the jth element of C_i;
        for $k = 0, t-1$ **do**
            if y_k is linear related to δ elements of X, denoted by $x_{i_0}, ..., x_{i_{\delta-1}}$ **then**
                $x_{i'} \leftarrow rand(x_{i_0}, x_{i_1}, ..., x_{i_{\delta-1}})$; ▷ to store the value of y_k;
                while $x_{i'} \in Index$ **do**
                    $x_{i'} \leftarrow rand(x_{i_0}, x_{i_1}, ..., x_{i_{\delta-1}})$;
                end while
                $Index = Index \cup \{x_{i'}\}$;
                add the s-XOR implementation of y_k to Imp before operations in C_i;
                update X and replace y_k by $x_{i'}$ in the operation of C_i;
            end if
        end for
    end for
end for
return Imp;

by returning $E = \{\{t_{21} = t_{21} \oplus y_{12} \cdot y_{15}, t_{22} = t_{22} \oplus y_4 \cdot y_0, t_{23} = t_{23} \oplus y_{14} \cdot y_{17}, t_{24} = t_{24} \oplus y_2 \cdot y_7\}, \{t_{21} = t_{21} \oplus y_3 \cdot y_6, t_{22} = t_{22} \oplus y_8 \cdot y_{10}, t_{24} = t_{24} \oplus y_{13} \cdot y_{16}\}, \{t_{23} = t_{23} \oplus y_5 \cdot y_1, t_{24} = t_{24} \oplus y_8 \cdot y_{10}, a = a \oplus y_9 \cdot y_{11}\}, \{t_{23} = t_{23} \oplus y_{13} \cdot y_{16}, a = a \oplus y_9 \cdot y_{11}\}\}$.

Based on the classification of nonlinear operations and the observations introduced in Section 3, Algorithm 2 presents a procedure to search optimized NCT-based circuits for f_1 and f_2.

Due to space limitations, the set E returned in Example 3 is taken as an example to illustrate the procedure of implementing the operations in C_0.

Example 4 According to Example 3, $C_0 = \{t_{21} = t_{21} \oplus y_{12} \cdot y_{15}, t_{22} = t_{22} \oplus y_4 \cdot y_0, t_{23} = t_{23} \oplus y_{14} \cdot y_{17}, t_{24} = t_{24} \oplus y_2 \cdot y_7\}$. First, initialize $Index$ to be empty and move the operations in C_0 to be adjacent. According to the classical implementation of auxiliary variables, y_{12} can be calculated as $y_{12} = x_0 \oplus x_3 \oplus x_5 \oplus x_6$. Suppose that x_0 is chosen randomly from $\{x_0, x_3, x_5, x_6\}$ to store the value of y_{12} under s-XOR metric. Thus, x_0 can not be used to store the value of any other auxiliary variables in C_0. Then $Index$ is updated as $Index = \{x_0\}$ and $x_0 = x_0 \oplus x_3 \oplus x_5 \oplus x_6$ is added to Imp

before $t_{21} = t_{21} \oplus y_{12} \cdot y_{15}$. Next, consider y_{15} which can be recomputed as $x_0 \oplus x_4 \oplus x_5$, where x_0 has been updated as $x_0 = x_0 \oplus x_3 \oplus x_5 \oplus x_6$. Since x_0 has been used to store the value of y_{12}, only x_4 or x_5 can be chosen to store the value of y_{15} under s-XOR metric. Assuming that x_4 is chosen, then add x_4 to $Index$ and insert $x_4 = x_4 \oplus x_5 \oplus x_0$ before the operation $t_{21} = t_{21} \oplus y_{12} \cdot y_{15}$. The remaining elements of C_0 can be updated in the same way. Replace the y's in C_0 by the corresponding elements in $Index$ and Algorithm 2 returns $\{x_0 = x_0 \oplus x_3 \oplus x_5 \oplus x_6, x_4 = x_4 \oplus x_5 \oplus x_0, x_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_7, x_3 = x_3 \oplus x_5, x_2 = x_2 \oplus x_0 \oplus x_6, x_6 = x_6 \oplus x_5 \oplus x_1 \oplus x_0, x_5 = x_5 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4, t_{21} = t_{21} \oplus x_0 \cdot x_4, t_{22} = t_{22} \oplus x_1 \cdot x_7, t_{23} = t_{23} \oplus x_3 \cdot x_2, t_{24} = t_{24} \oplus x_6 \cdot x_5\}$ as one of the in-place implementation of the elements of C_0.

The strategy of randomization is adopted in Algorithm 2. The step of randomly rearranging Imp by using Fact 1 - 3 (i.e., line 2) is aimed at providing different input for Algorithm 1, which is related to the Toffoli depth. According to Observation 1, each time a variable is randomly selected from the input set for calculating an auxiliary variable from Y (i.e., line 13 to line 16), a different implementation of the auxiliary variable will be obtained. Therefore, for each call to Algorithm 2, a different NCT-based circuit may be returned. Thus, one can run Algorithm 2 several times and keep the best one with the Toffoli depth as the primary consideration.

4.3 Reversible Circuits of AES S-box

4.3.1 Circuits for $|x\rangle|0\rangle^{\otimes n} \xrightarrow{\text{S-box}} |x\rangle|S(x)\rangle|0\rangle^{\otimes(n-8)}$

In this section, five qubits are allocated to build the quantum circuit for f_1, four of which are used to store the values of $t_{21}, t_{22}, t_{23}, t_{24}$, and the rest one is an ancilla qubit. Applying Algorithm 2 to f_1, an NCT-based circuit of f_1 can be obtained, which costs 5 ancilla qubits, 12 Toffoli gates, and 45 CNOT gates. The Toffoli depth of the circuit is 3. The implementation is listed in Appendix D.1.

The quantum circuit of S_4 with Toffoli depth 6 is listed in Appendix D.2, which only costs one ancilla qubit (denoted by a). The ancilla qubit allocated for this part can reuse the one from f_1. Since f_2 requires no ancilla qubits, it is not necessary to reset the ancilla qubit in the quantum circuits of S_4, and this also helps to save Toffoli gates and reduce Toffoli depth. The circuit for S_4 consumes 6 Toffoli gates and 4 CNOT gates. If 2 ancilla qubits are allocated for S_4, the Toffoli depth of the circuit listed in Appendix D.2 can be reduced from 6 to 5. As listed in Appendix D.3, in which a and b represent ancilla qubits, the circuit consumes 6 Toffoli gates and 5 CNOT gates.

Different from f_1, when devising a quantum style implementation of f_2, the first step generates an implementation of the bottom function B based on the observation presented in Subsection 4.1. The bottom function B takes $(z_0, z_1, ..., z_{17})$ as inputs and generates the outputs of AES S-box. Among the 18 inputs of B, 8 of them store the outputs of AES S-box under s-XOR metric. Using the implementation of the bottom function, a quantum style implementation of f_2 can be derived, which is listed in Appendix C.

It is worth noting that the auxiliary variable set Y for f_2 consists of $t_{29}, t_{33}, t_{37}, t_{40}, t_{41}, t_{42}, t_{43}, t_{44}, t_{45}, y_0, y_1, ..., y_{21}$ where t_i $(i = 41, 42, ..., 45)$ are linear combinations of the outputs of the 4-bit S-box, i.e., $t_{29}, t_{33}, t_{37}, t_{40}$, and y_j $(j = 0, 1, ..., 21)$ are linear expressions of the inputs of AES S-box. Thus, the input set for f_2 is $X = \{t_{29}, t_{33}, t_{37}, t_{40}, x_0, x_1, ..., x_7\}$. Then, an optimized NCT-based circuit can be obtained by applying Algorithm 2, which costs 21 Toffoli gates, 55 CNOT gates and 4 Pauli-X gates. The Toffoli depth of the circuit is 6. The implementation is listed in Appendix D.4.

When devising a complete NCT-based circuit for AES S-box, the circuits of f_1, S_4 and f_2 should be applied sequentially at the very beginning to get the outputs of AES S-box. The above three circuits overwrite the values stored in the ancilla qubits allocated for the S-box, which means those ancilla qubits cannot be reused. To reset ancilla qubits, several operations in the circuits of f_1, S_4 and f_2 should then be applied again in reverse order. Notably, after being updated by the f_1 circuit in an in-place manner, the inputs of AES S-box (i.e., $x_0, x_1, ..., x_7$) are then updated by the f_2 circuit with s-XOR operations. Besides, the outputs of S_4 are also updated by the f_2 circuit similarly. Consequently, the linear operations applied to $t_{29}, t_{33}, t_{37}, t_{40}$ and $x_0, x_1, ..., x_7$ in the f_2 circuit should be applied one more time in reverse order to reset their values to be equal to the outputs of S_4 and f_1 respectively. Finally, the ancilla qubits can be reset by applying the inverse circuits of S_4 and f_1.

Circuits for the S-box^{-1} When designing quantum circuit for the S-box^{-1} with the classical one proposed in [34], Algorithm 2 returns a circuit with Toffoli depth 26. If the method proposed in [14] is adopted, the 4 Pauli-X gates and linear transformation L^{-1} (given in [14]) applied to the inputs of the S-box can be combined with the top function U of the classical circuit given in [5], without changing the middle function F and the bottom function B. Thus, the quantum circuit constructed for the S_4 of AES S-box can also be used for designing the circuit of the S-box^{-1}. By applying Algorithm 2, a circuit with Toffoli depth 24 can be acquired. The circuit is listed in Appendix E and will be used to construct the NCT-based circuits for AES in this paper.

The quantum resource consumption of different NCT-based circuits are summarized in Table 7.

4.3.2 Circuits for $|x\rangle|y\rangle|0\rangle^{\otimes(n-8)} \xrightarrow{\text{S-box}} |x\rangle|y \oplus S(x)\rangle|0\rangle^{\otimes(n-8)}$

As shown in Subsection 2.3, B generates the outputs of the S-box with the outputs of F. Therefore, the only difference between the circuits for $|x\rangle|0\rangle^{\otimes n} \xrightarrow{\text{S-box}} |x\rangle|S(x)\rangle|0\rangle^{\otimes(n-8)}$ and $|x\rangle|y\rangle|0\rangle^{\otimes(n-8)} \xrightarrow{\text{S-box}} |x\rangle|y \oplus S(x)\rangle|0\rangle^{\otimes(n-8)}$ is the implementation of F and B.

The construction of the NCT-based circuit proposed in this work for function B is based on the heuristic given in [32], and the output qubits $s_0, s_1, ..., s_7$ have never been involved in any nonlinear operation. That is, the influence of y can be removed by applying a sequence of CNOT gates for the circuit shown in Appendix D.

Table 7 The comparison of different NCT-based circuits for outputs are $|0\rangle^{\otimes 8}$.

Operation	Source	#Qubits	#Toffoli	#CNOT	#Pauli-X	Toffoli Depth
S-box	[19]	16	55	314	4	40
	[30]	16	55	322	4	40
	[16]	120	34	186	4	6
	[34]	6	52	326	4	41
		7	48	330	4	39
		8	46	332	4	37
	[20]	6	48	236	4	36
		7	48	238	4	34
		8	46	240	4	32
	[14]	120	34	212	4	4
		202	78	355	4	3
	This work	5	57	193	4	24
		6	57	195	4	22
S-box^{-1}	[14]	6	52	368	8	41
	This work	5	58	187	10	26*
		5	57	205	8	24†
		6	57	207	8	22†

∗ Constructed based on the classical circuit given in [34].
† Constructed based on the classical circuit given in [5].

Take the output bit s_0 in the circuit shown in Appendix D.4 as an example. The bit s_0 is only used to update the values of s_1, s_2 and s_6 by applying CNOT gates. As a result, the influence of the initial value in s_0 can be removed by XORing s_0 to s_1, s_2 and s_6 before s_0 is updated. In short, before applying the circuit shown in Appendix D.4, adding the operations formed as $s_i = s_i \oplus s_j$ in the circuit listed in Appendix D.4 in an inverse order can remove the propagation of initial values, where $i, j \in [0, 7]$ and $i \neq j$. Finally, the circuit built for the S-box when outputs are all 0s can be transformed to the one that maps $|x\rangle|y\rangle|0\rangle^{\otimes 5}$ to $|x\rangle|y \oplus S(x)\rangle|0\rangle^{\otimes 5}$. The operations added before the circuit shown in Appendix D.4 are listed as follows.

$$
\begin{aligned}
&s_1 = s_1 \oplus s_0,\ s_4 = s_4 \oplus s_3,\ s_6 = s_6 \oplus s_7,\ s_7 = s_7 \oplus s_4,\ s_3 = s_3 \oplus s_1,\\
&s_0 = s_0 \oplus s_3,\ s_2 = s_2 \oplus s_0,\ s_5 = s_5 \oplus s_2,\ s_2 = s_2 \oplus s_6,\ s_4 = s_4 \oplus s_6,\\
&s_3 = s_3 \oplus s_5,\ s_1 = s_1 \oplus s_6,\ s_7 = s_7 \oplus s_2,\ s_6 = s_6 \oplus s_0,\ s_0 = s_0 \oplus s_4.
\end{aligned}
$$

Compared with the circuit that maps $|x\rangle|0\rangle^{\otimes 13}$ to $|x\rangle|S(x)\rangle|0\rangle^{\otimes 5}$, the circuit for the S-box with nonzero output values costs 15 CNOT gates more than the one shown in Appendix D.

Similarly, the circuit for the transformation that maps $|x\rangle|y\rangle|0\rangle^{\otimes 5}$ to $|x\rangle|y \oplus S^{-1}(x)\rangle|0\rangle^{\otimes 5}$ can be deduced from the one shown in Appendix E by adding the operations listed in Appendix F. The cost of different NCT-based circuits built for the S-box and the S-box^{-1} with outputs are not all 0s are listed in Table 8.

Table 8 The comparison of different NCT-based circuits for outputs are not $|0\rangle^{\otimes 8}$.

Operation	Source	#Qubits	#Toffoli	#CNOT	#Pauli-X	Toffoli Depth
S-box	[34]	7	68	352	4	60
		8	64	356	4	58
		9	62	358	4	56
	[30]	16	55	322	4	40
	[20]	6	48	272	4	36
		7	48	274	4	34
		8	46	276	4	32
	[14]	6	52	336	4	41
	This work	5	57	208	4	24
		6	57	210	4	22
S-box^{-1}	[34]	7	69	335	24	62
		8	67	337	24	60
		9	65	339	24	60
		10	63	341	24	60
	[20]	6	48	272	8	36
		7	48	274	8	34
		8	46	276	8	32
	This work	5	58	200	10	26*
		5	57	226	8	24†
		6	57	228	8	22†

∗ Constructed based on the classical circuit given in [34].
† Constructed based on the classical circuit given in [5].

5 Schemes for the Round Function and the Key Schedule

5.1 The Partial Zig-zag Method for Round Function

The pipeline, zig-zag and improved zig-zag methods are often used to design the overall structure for AES with a complete round function and its inverse for reducing depth. However, those methods require much qubits. In order to save qubits, constructing a partial round function and its inverse is adopted. The mechanism was adopted in [2] to design quantum circuits for SHA-2/SHA-3, and also be discussed in [14] to construct quantum circuits for AES based on *double-depth S-box circuits*, by which, two sequential S-boxes will be applied. In this paper, the strategy of implementing a partial round function and its inverse will be called partial zig-zag method and be discussed more extensively.

Assuming $a_0, a_1, ..., a_{15}$ are the 16 8-qubit inputs, and a_{16} is an 8-qubit tuple. The partial zig-zag method works as follows. First, the circuit $|x\rangle|0\rangle \rightarrow |x\rangle|S(x)\rangle$ to $|a_0\rangle|a_{16}\rangle$ is applied to get the output of the first S-box. Then, the circuit $|x\rangle|y\rangle \rightarrow |x\rangle|y \oplus S^{-1}(x)\rangle$ is applied to $|a_{16}\rangle|a_0\rangle$. This means once the S-box circuit has been applied to update a certain byte, the qubits of the corresponding input byte can be reset to zero by using the quantum circuit of S-box^{-1}, in this case, the S-box output of the first byte is stored in a_{16} and the input byte a_0 is reset to zero. Thus, a_0 can be reused to store the S-box output of the second byte in a similar way. Therefore, the partial zig-zag method can execute the S-box layer of AES in sequential, and one S-box is performed each time. Moreover, one can parallelly execute more S-boxes if more ancilla qubits are available. In the following, denote by m the number

of S-boxes that are parallelly executed. Clearly, $m = 1$ is the case described as above, $m = 16$ is equivalent to the improved zig-zag method. Generally, more S-boxes are applied in parallel means more qubits are needed to store the outputs of S-boxes. At the same time, more ancilla qubits are needed for these parallelly executed S-boxes. In the case that m S-boxes are applied in parallel, the number of allocated storage qubits for the next round is $8m$. In other words, only $128 + 8m$ qubits are required using the partial zig-zag method.

Denote the state of the ith round by R_i, the partial zig-zag method for AES-128 when $m = 4$ is shown in Figure 3, where $R_i^{\frac{1}{4}}$ represents the application of S-boxes to four bytes for the ith round, and $R_i^{-\frac{1}{4}}$ means resetting four bytes of the ith round.

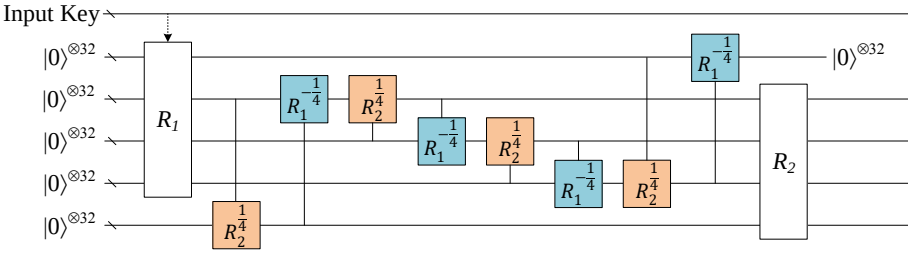

Fig. 3 The procedure for the **SubBytes** when $m = 4$.

5.2 Scheme for the Key Schedule

The research in [16] reveals that a quantum circuit that maps $|x\rangle|y\rangle|0\rangle^{\otimes(n-8)}$ to $|x\rangle|y \oplus S(x)\rangle|0\rangle^{\otimes(n-8)}$ can be used to reduce the qubit consumption of the key schedule. Based on such circuit, the authors of [15, 16, 20, 30] implemented the key schedule without introducing storage qubits. In this paper, the framework presented in [16] is adopted to implement the key schedules for all instances of AES. The scheme for AES-128 is shown in Figure 4 as an example to illustrate the procedure.

6 NCT-based Circuits of AES

6.1 The Scheme for the AES Family

This section estimates the circuit cost with m parallel S-boxes. For a given m, the allocated qubits for AES are also determined, i.e., k qubits for the master key ($k = 128, 192$ and 256 for the three instances of AES, respectively), 128 qubits for the first round, $(8 + 5)m$ qubits[7] for the m parallel S-boxes, where

[7]Note that an NCT-based circuit that costs 6 ancilla qubits is also designed for AES S-box, however, in order to save qubits, only 5 ancilla qubits are allocated for each S-box in the very beginning.

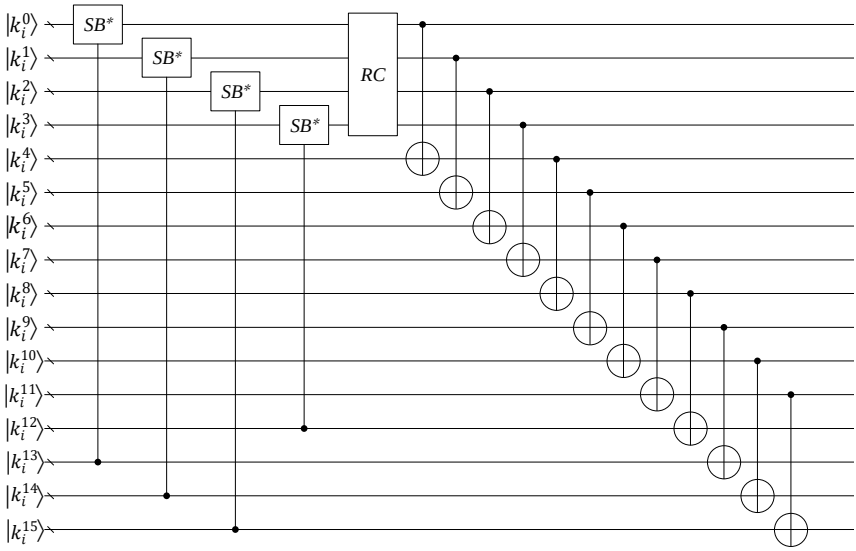

Fig. 4 The scheme for the key schedule of AES-128, where k_i^j represents the jth byte in the ith round key, SB* is the modified **SubBytes**, RC is the XOR of the round constant.

$m \in [1, 16]$. The case of $m = 4$ for AES-128 is taken as an example to illustrate the encryption of the AES family.

The First Round In the process of the key whitening, the plaintext is XORed to the master key for saving qubits. For a given plaintext, the key whitening can be implemented by inverting the qubits in the master key corresponding to the bits in the plaintext with a value of 1. Therefore, at most 128 Pauli-X gates are required to implement the key whitening. For $m = 4$, there are $128 + 4 \times 13 = 180$ qubits with zero value for the first round. The first round requires 20 S-boxes, 4 for the key schedule and 16 for the round function. Due to the qubit consumption of the quantum circuits constructed for the S-box in Section 4, 180 qubits with zero value is enough for us to implement the first round within an S-box depth of 2. The implementation of the first round is depicted in Figure 5, where X represents the Pauli-X gate, Anc^n represents the usage of n ancilla qubits, S_{in}^j and S_{out}^j are the inputs and the outputs of j S-boxes. Specifically, the first round starts with applying 12 S-boxes to the bytes in the state, after which 84 qubits with value zero are left. Inverting the bits in the state according to the plaintext again can recover 64 bits of the master key, by which, partial words of the round key can be generated. Note that the first word of the round key, i.e.,W_4, should be calculated with the knowledge of W_0 and W_3. Hence, among the 12 S-boxes applied in step (b), 8 should be applied to the first and the fourth words in the state as shown in Figure 5 with step (b).

Besides, the round keys are generated in an in-place way, and no additional storage qubits are required by the key schedule. It means that the 4 S-boxes for computing W_4 and the remaining 4 S-boxes applies to the bytes in the

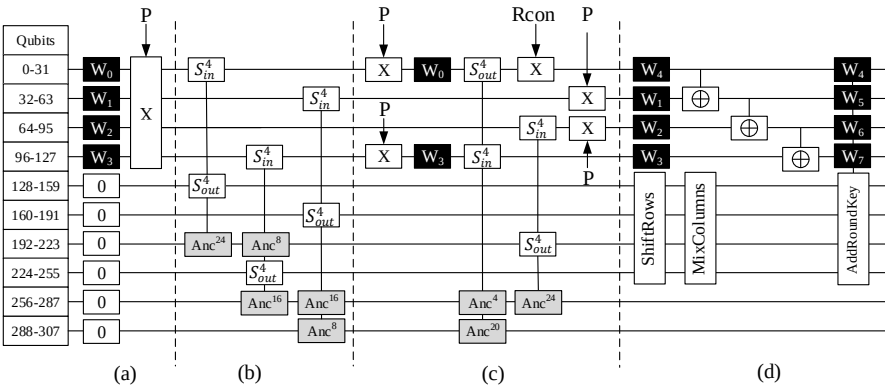

Fig. 5 The quantum circuit for the first round of AES-128.

state can be implemented in parallel. The procedure is shown in Figure 5 with step (c), after which 52 qubits with zero value are left. The first round is completed with step (d), which contains the implementation of **ShiftRows**, **MixColumns** and **AddRoundKey**.

The Rest Rounds The implementation of the second round is shown in Figure 6, where S_{in}^{-j} and S_{out}^{-j} are the inputs and the outputs of j S-box^{-1}es.

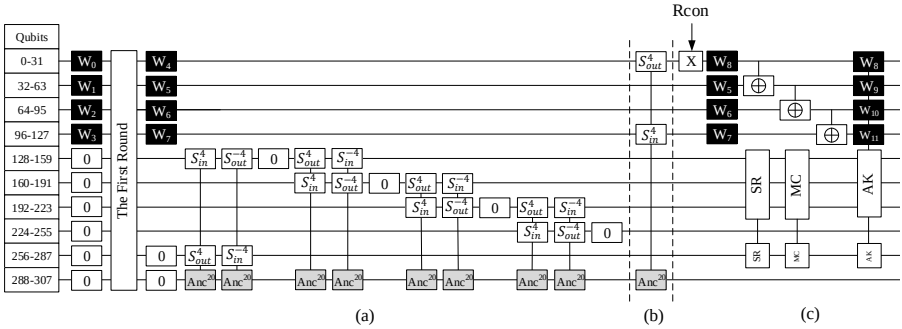

Fig. 6 The quantum circuit for the second round of AES-128.

After the first round, there are 52 qubits with zero value. Each application of 4 S-boxes for the round function will increase both the S-box depth and the S-box^{-1} depth by 4 (as shown in Figure 6 with step (a)), while the key schedule only increases the S-box depth by 1 (as shown in Figure 6 with step (b)). The remaining operations of the second round are shown in Figure 6 with step (c). The rest rounds can be implemented in the same way as the second round.

6.2 The Quantum Resource Estimate

The circuits constructed for AES S-box and it inverse are the only two nonlinear components used for designing NCT-based circuits of AES. However, due to the number of ancilla qubits allocated for each S-box or S-box^{-1}, different

quantity of various circuits constructed for S-box and S-box^{-1} will be applied. The S-boxes in the first round can be implemented with different circuits that consume 5 or 6 ancilla qubits, which will be discussed later. For the rest rounds, it can be easily verified that the last (16 mod m) S-boxes in the round function, the last (16 mod m) S-box^{-1}es for removing the previous round and the 4 S-boxes for the key schedule can alway be implemented by the quantum circuits that consume 6 ancilla qubits if $\frac{16}{m} \notin \mathbb{Z}_+$. For the case that $\frac{16}{m} \in \mathbb{Z}_+$, the 4 S-boxes for the key schedule can also be implemented by the quantum circuit that consumes 6 ancilla qubits. Denote the circuits constructed for $|x\rangle|0\rangle^{\otimes(n+8)} \xrightarrow{\text{S-box}} |x\rangle|S(x)\rangle|0\rangle^{\otimes n}$ and $|x\rangle|S(x)\rangle|0\rangle^{\otimes n} \xrightarrow{\text{S-box}^{-1}} |x\rangle|0\rangle^{\otimes(n+8)}$ by S_n and S_n^{-1*} respectively, where $n \in \{5, 6\}$ is the number of allocated ancilla qubits. Similarily, the circuit for $|x\rangle|y\rangle|0\rangle^{\otimes 14} \xrightarrow{\text{S-box}} |x\rangle|y \oplus S(x)\rangle|0\rangle^{\otimes 6}$ is denoted by S_6^*. Denote by $Cnot_{S_5}$ the CNOT gate consumption of the circuit constructed for S_5, the cost of other gates are denoted in the same way. The total number of applied **SubWord** operations and the number of applied **SubWord** except the first round are denoted by w and $w^{'}$, where $w = 10, 8, 13, w^{'} = 9, 7, 13$ for the three instances of AES respectively. Denote by r the round number and $r = 10, 12, 14$ for AES-128, AES-192 and AES-256, respectively. For simplicity, (16 mod m) is denoted by z and $(16 - (16 \bmod m))$ is denoted by $z^{'}$ in the following equations.

The number of CNOT gates consumed by an NCT-based circuit of AES except the nonlinear component in the first round can be calculated by

$$\begin{cases} 128r + 4 \cdot Cnot_{S_6^*} \cdot w^{'} + (4 \cdot 92 + 16 \cdot Cnot_{S_5} + \\ 16 \cdot Cnot_{S_5^{-1*}})(r-1) + t, & \text{if } \dfrac{16}{m} \in \mathbb{Z}_+, \\ 128r + 4 \cdot Cnot_{S_6^*} \cdot w^{'} + (4 \cdot 92 + z^{'} \cdot Cnot_{S_5} + \\ z \cdot Cnot_{S_6} + z^{'} \cdot Cnot_{S_5^{-1*}} + z \cdot Cnot_{S_6^{-1*}})(r-1) + t, & \text{otherwise,} \end{cases}$$

where $t = 3 \cdot 32w$ for AES-128 and AES-256, $= 3 \cdot 32w - 2 \cdot 32 + 4 \cdot 32(r - w)$ for AES-192.

The number of Pauli-X gates consumed by an NCT-based circuit of AES except the nonlinear component in the first round can be calculated by

$$\begin{cases} 128 \cdot 2 + HW(Rcon) + 4 \cdot X_{S_6^*} \cdot w^{'} + 16(X_{S_5} + X_{S_5^{-1*}})(r-1), & \text{if } \dfrac{16}{m} \in \mathbb{Z}_+, \\ 128 \cdot 2 + HW(Rcon) + 4 \cdot X_{S_6^*} \cdot w^{'} + (z^{'} \cdot X_{S_5} + z \cdot X_{S_6} + \\ z^{'} \cdot X_{S_5^{-1*}} + z \cdot X_{S_6^{-1*}})(r-1), & \text{otherwise,} \end{cases}$$

where $HW(Rcon)$ is the Hamming weight of all the round constants.

The number of Toffoli gates consumed by an NCT-based circuit of AES except the nonlinear component in the first round can be calculated by

$$\begin{cases} 4 \cdot Toffoli_{S_6^*} \cdot w^{'} + 16(Toffoli_{S_5} + Toffoli_{S_5^{-1*}})(r-1), & \text{if } \frac{16}{m} \in \mathbb{Z}_+, \\ 4 \cdot Toffoli_{S_6^*} \cdot w^{'} + (z^{'} \cdot Toffoli_{S_5} + z \cdot Toffoli_{S_6} + \\ z^{'} \cdot Toffoli_{S_5^{-1*}} + z \cdot Toffoli_{S_6^{-1*}})(r-1), & \text{otherwise.} \end{cases}$$

For better understanding of the gate cost, the number of CNOT gates consumed by the AES-128 circuit for the case that $\frac{16}{m} \in \mathbb{Z}_+$ is taken as an example and be presented in Appendix G.

Assuming that the partial zig-zag method executes m S-boxes in parallel, and l extra ancilla qubits are allocated for the key schedule (which will be explained later). The number of consumed qubits is

$$128 + k + 13m + l,$$

where k is the key length.

Denote by d_{S_5} the Toffoli depth of the circuit constructed for S_5, the Toffoli depth of other circuits designed for the S-box and the S-box^{-1} are denoted in the same way.

Case for $l = 0$ Assuming that m S-boxes are applied each time. If $\frac{16}{m} \in \mathbb{Z}_+$, the 16 S-boxes in the round function and the 16 S-box^{-1}es for removing the previous round will be implemented with the circuits that consume 5 ancilla qubits. The **SubWord** of the key schedule can be implemented by using the circuit that costs 6 ancilla qubits within $\lceil \frac{24}{13m} \rceil$ S-box depth (case 1). Otherwise, if $\frac{16}{m} \notin \mathbb{Z}_+$, 2 of the S-boxes for the key schedule can be implemented in parallel with last $(16 \bmod m)$ S-boxes for the **SubBytes**, and the remaining 2 S-boxes can be implemented in parallel with last $(16 \bmod m)$ S-box^{-1}es for removing the previous round (case 2). In this case, only the circuits that consume 6 ancilla qubits will be used, since $(16 \bmod m) \cdot 14 + 2 \cdot 6 \geq 13m$ and $((16 \bmod m) \cdot 6 + 2 \cdot 6) \geq (13m - (16 \bmod m) \cdot 8)$ always hold. The Toffoli depth of the circuit except the first round can be calculated by

Case 1:

$$(\frac{16}{m} \cdot d_{S_5} + \frac{16}{m} \cdot d_{S_5^{-1*}})(r-1) + \lceil \frac{24}{13m} \rceil \cdot d_{S_6^*} \cdot w^{'}.$$

Case 2:

$$\lfloor \frac{16}{m} \rfloor (d_{S_5} + d_{S_5^{-1*}})(r-1) + (d_{S_6} + d_{S_6^{-1*}})(r - w^{'} - 1) + (max\{d_{S_6}, d_{S_6^*}\} + max\{d_{S_6^{-1*}}, d_{S_6^*}\})w^{'}.$$

Case for $l > 0$ According to the analysis for $l = 0$, the S-boxes in the key schedule do not increase the S-box depth if $\frac{16}{m} \notin \mathbb{Z}_+$. Therefore, only the cases

that $\frac{16}{m} \in \mathbb{Z}_+$ are discussed for $l > 0$. In this case, the increased S-box depth caused by updating the key schedule can be reduced by adding some ancilla qubits. For the cases that $m = 1, 2, 4$ or 8, one S-box for the key schedule can be executed in parallel with m S-boxes for the round function or with m S-box^{-1}es for removing the previous round. Only 5 ancilla qubits are required. For the case that $m = 16$, the encryption of one round can be completed with an S-box depth and S-box^{-1} depth of 1. The Toffoli depth can be reduced by applying 2 S-boxes for the key schedule with 16 S-boxes for the round function and another 2 S-boxes with 16 S-box^{-1}es. This requires 10 ancilla qubits. Note that for $l > 0$, once the 4 S-boxes for the key schedule have been applied, the ancilla qubits for the key schedule can be used by the round function if the 16 S-boxes or 16 S-box^{-1}es have not been fully applied. In this case, the circuits of S-box and S-box^{-1} that cost 6 ancilla qubits can be applied to reduce the Toffoli depth if $l \geq m$, since all the m S-boxes or S-box^{-1}es after this can be applied in parallel by using the circuits with Toffoli depth 22. The Toffoli depth of the circuit except the first round is

$$\begin{cases} (2(max\{d_{S_5}, d_{S_5^*}\} + max\{d_{S_5^{-1*}}, d_{S_5^*}\}) + (\frac{16}{m} - 2)(d_{S_6} + & \\ d_{S_6^{-1*}}))(r-1), & \text{if } m = 1, 2, 4, 8, \\ (max\{d_{S_5}, d_{S_5^*}\} + max\{d_{S_5^{-1*}}, d_{S_5^*}\})(r-1), & \text{if } m = 16. \end{cases}$$

Depth of the First Round The first round of AES dose not need to apply S-box^{-1}, and only AES-128 and AES-192 apply **SubWord** in the first round. Assuming that l ($l = 0, 5, 10$) ancilla qubits are allocated for the S-boxes in the key schedule of AES, there are $128 + 13m + l$ zero qubits available for the first round. The S-box depth for the first round of AES with various m are presented in Table 9. Each S-box and S-box^{-1} are allocated 6 ancilla qubits unless otherwise specified.

Table 9 The S-box depth of the first round of AES.

	AES-128/AES-192				AES-256					
S-box depth \ m / l	1	2	3	≥ 4	1	2	3-5	6	7	≥ 8
0	4	3	3	2	3	2†	2	2	1⋆	1
5	3⋆	3	2※	2	3	2	2	1⋆	1	1
10	3†	3	2	2	3	2	2	1⋆	1	1

⋆ All the S-boxes and S-box^{-1}es are allocated 5 ancilla qubits.
※ Only the 13 S-boxes in the first S-box depth are allocated 5 ancilla qubits.
† Only the 11 S-boxes in the first S-box depth are allocated 5 ancilla qubits.

The resource estimate of different NCT-based circuits constructed for three instances of the AES family are listed in Table 1, Table 2 and Table 3.

7 Conclusion

This work researched the construction of optimized NCT-based circuits for the AES family. First of all, quantum circuits for AES S-box and its inverse were investigated based on classical ones. To this end, the properties as well as the factors that affect the Toffoli depth and CNOT gate consumption of the quantum circuit were investigated. Moreover, both the classical implementation of AES S-box and its inverse were divided into three parts, and the application of the existing tools or heuristic on those parts were investigated. In addition, a heuristic was proposed to search optimized NCT-based circuits for the first part and the third part of the rearranged S-box and S-box^{-1} circuits. The experimental results reveal that the quantum circuits designed in this work for AES S-box and S-box^{-1} with optimized CNOT gate consumption and Toffoli depth have advantages in qubit consumption. Then, this work researched the implementation of the key schedule and the round function of AES. By applying the framework based on partial round functions which is called partial zig-zag method in this paper, different NCT-based circuits were constructed for the AES family. The results show that it requires only 269, 333 and 397 qubits to implement the three instances of AES with NCT gate set. Besides, taking the trade-off of Toffoli depth and qubits into consideration, NCT-based circuits for AES-128, AES-192 and AES-256 that outperform state-of-the-art schemes in the metric of $T \cdot M$ value can be constructed with only 474, 538 and 602 qubits.

When evaluating the depth of the quantum circuit, this work focuses on the Toffoli depth. Since a Toffoli gate can be decomposed into several Clifford gates and T gates, one can also research the construction of quantum circuits for AES with Clifford+T gates and the T-depth should be considered in this case. On the other hand, construction of the NCT-based circuits for odd permutations can also be a direction for future research.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Natural Science Foundation of China (Grant No.61802119) and the Wuhan Science and Technology Bureau (Grant No.2022010801020328).

References

[1] Almazrooie, M., Samsudin, A., Abdullah, R., Mutter, K.N.: Quantum reversible circuit of AES-128. Quantum Inf. Process. **17**(5), 112 (2018)

[2] Amy, M., Matteo, O Di., 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., Howard M. Heys, (eds.), Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's,

NL, Canada, August 10-12, 2016, Revised Selected Papers, vol. 10532 of Lecture Notes in Computer Science, pp. 317–337. Springer, (2016)

[3] Arute, F., Arya, K., Babbush, R. et al.: Quantum supremacy using a programmable superconducting processor. Nature. **574**(7779), 505–510 (2019)

[4] Bernstein, D.J., Biasse, JF., Mosca, M.: A low-resource quantum factoring algorithm. In: Lange, T., Takagi, T. (eds.), Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings, vol 10346 of Lecture Notes in Computer Science, pp. 330–346. Springer, (2017)

[5] Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: Festa, Paola., (eds.), Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings, vol. 6049 of Lecture Notes in Computer Science, pages 178–189. Springer, (2010)

[6] Canright, David.: A very compact s-box for AES. In Rao, Josyula R., Sunar, Berk., editors, Cryptographic Hardware and Embedded Systems - CHES 2005, 7th InternationalWorkshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings, vol. 3659 of Lecture Notes in Computer Science, pp. 441—455. Springer, (2005)

[7] Chung, D., Lee, S., Choi, D., Lee, J.: Alternative Tower Field Construction for Quantum Implementation of the AES S-box. IEEE Transactions on Computers. **71**(10), 2553–2564 (2021)

[8] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography. Springer. (2002)

[9] Dasu, V.A., Baksi, A., Sarkar, S., Chattopadhyay, A.: LIGHTER-R: optimized reversible circuit implementation for sboxes. In: Zhao, D. (eds.), SOCC 2019 - 32nd IEEE International System-on-Chip Conference, Singapore, September 3-6, pp. 260-265. IEEE, (2019)

[10] Ekerå ,M., Håstad, J.: Quantum algorithms for computing short discrete logarithms and factoring RSA integers. In: Lange, T., Takagi, T. (eds.), Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings, vol. 10346 of Lecture Notes in Computer Science, pp. 347–363. Springer, (2017)

[11] Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover's algorithm to AES: quantum resource estimates. In: Takagi,T., (eds.), Post-Quantum Cryptography - 7th International Workshop,

PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings, vol. 9606 of Lecture Notes in Computer Science, pp. 29–43. Springer, (2016)

[12] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Gary L. Miller, (ed), In: Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996, pp. 212–219. ACM, (1996)

[13] Guajardo, J., Paar, C.: Itoh-tsujii inversion in standard basis and its application in cryptography and codes. Des. Codes Cryptogr. **25**(2), 207–216 (2002)

[14] Huang, Z., Sun, S.: Synthesizing quantum circuits of aes with lower t-depth and less qubits. IACR Cryptol. ePrint Arch. **2022**, 620 (2022)

[15] Jang, K., Baksi, A., Song, G., Kim, H., Seo, H., Chattopadhyay, A.: Quantum analysis of aes. IACR Cryptol. ePrint Arch. **2022**, 683 (2022)

[16] Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and lowmc. In: Canteaut, A., Ishai, Y., (eds.), Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II, volume 12106 of Lecture Notes in Computer Science, pp. 280–310. Springer, (2020)

[17] Jean, J., Peyrin, T., Sim, S.M., Tourteaux, J.: Optimizing implementations of lightweight building blocks. IACR Trans. Symmetric Cryptol. **2017**(4), 130–168 (2017)

[18] 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), 1–39 (2018)

[19] Langenberg, B., Pham, H., Steinwandt, R.: Reducing the cost of implementing the advanced encryption standard as a quantum circuit. IEEE Transactions on Quantum Engineering. **2020**(1), 1–12 (2020)

[20] Li, Z., Cai, B., Sun, H., Liu, H., Wan, L., Qin, S., Wen, Q., Gao, F.: Novel quantum circuit implementation of Advanced Encryption Standard with low costs. Science China Physics, Mechanics & Astronomy. **65**(9), 290311 (2022)

[21] May, A., Schlieper, L.: Quantum period finding is compression robust. IACR Trans. Symmetric Cryptol. **2022**(1), 183–211 (2022)

[22] Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: A Systematic Evaluation of Compact Hardware Implementations for the Rijndael S-Box. In: Menezes, A. (eds.), Topics in Cryptology - CT-RSA 2005 - The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings, volume 3376 of Lecture Notes in Computer Science, pp. 323–333. Springer, (2005)

[23] Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information (10th Anniversary edition). Cambridge University Press. Cambridge (2016)

[24] NIST: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. (2016)

[25] Seifert J.: Using fewer qubits in shor's factorization algorithm via simultaneous diophantine approximation. In: Naccache, D. (eds.), Topics in Cryptology - CT-RSA 2001 - The Cryptographers' Track at the RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings, volume 2020 of Lecture Notes in Computer Science, pp. 319–327. Springer,(2001)

[26] Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **22**(6), 710–722 (2003)

[27] Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)

[28] Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(5), 1474–1483 (1997)

[29] Trefethen, L.N., Bau, D.: Numerical linear algebra. SIAM. (1997)

[30] Wang, Z., Wei, S., Long, G.: A quantum circuit design of aes requiring fewer quantum qubits and gate operations. Frontiers of Physics. **17**(4), 1–7 (2022)

[31] Wei, Z., Sun, S., Hu, L., Wei, M., Boyar, J., Peralta, R.: Scrutinizing the tower field implementation of the $\mathbb{F}_2^8$ inverter - with applications to AES, Camellia, and SM4. IACR Cryptol. ePrint Arch. (2019)

[32] Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. IACR Trans. Symmetric Cryptol. **2020**(2), 120–145 (2020)

[33] Zou, J., Li, L., Wei, Z., Luo, Y., Liu, Q., Wu, W.: New quantum circuit implementations of SM4 and SM3. Quantum Inf. Process. **21**(5), 1–38 (2022)

[34] 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, pp. 697–726. Springer, (2020)

[35] Zou, J., Wei, Z., Sun, S., Luo, Y., Liu, Q., Wu, W.: Some efficient quantum circuit implementations of camellia. Quantum Inf. Process. **21**(4), 1–27 (2022)

A The Proof of Fact 1

Proof Based on the values of a and c, the proof proceeds in two cases:

Case 1: if $a = c$, the two operations can be rewritten as $t_a = t_a \oplus t_b$, $t_a = t_a \oplus t_c$, after which the value of qubit t_a is $t_a \oplus t_b \oplus t_c$. Assume that the operations are changed to $t_a = t_a \oplus t_c$, $t_a = t_a \oplus t_b$, the final value of t_a is not changed. Thus, the order of these two operations can be exchanged.

Case 2: if $a \neq c, d$ and $b \neq c$, it follows that a, c, d are pairwise distinct since $a \neq b$ and $c \neq d$. In addition, the operations have no influence on the values of t_b and t_d. Therefore, exchanging the order of these two operations does not result in any change of the values stored in t_a and t_c. □

B Discussion on LIGHTER and LIGHTER-R

Before illustrating the method of using LIGHTER, the following definition is introduced.

Definition 3 (odd permutation [26]) *A permutation is called odd if it can be written as the product of an odd number of transpositions.*

The even permutation can be defined in the similar way.

It is obviously that the 4-bit S-box shown in Table 6 is odd (as well as the one derived from the inverse of AES S-box). The researches of [26] reveal that the NCT-based circuit for an even permutation can be constructed without temporary storage, but for an odd permutation, one wire of temporary storage is required. It means that one can not construct a quantum circuit for an odd permutation by using the tool LIGHTER-R only based on NCT gate set. To this end, the following strategies are investigated to construct an NCT-based circuit for an odd permutation.

Strategy 1 First, expand a 4-bit odd permutation to be a 5-bit one by adding one bit in the most significant bit of the inputs, whose corresponding output bit is identical to the input. There is no doubt that the resulting 5-bit permutation is even. Then, modify the code to make the tool LIGHTER-R compatible with 5-bits permutation as its input and search the NCT-based circuit for the resulting 5-bit even permutation. Unfortunately, due to the large search space, none implementation for the S-box shown in Table 6 returned.

Strategy 2 The underlying logic gate set of the tool LIGHTER can be customized as needed. Considering the relation between the NCT gate set and the classical And gate, Xor gate and Not gate, one can specify that the tool LIGHTER only uses And gates, Xor gates and Not gates to search an optimized in-place implementation for a 4-bit odd permutation. Certainly, this comes at the cost of an auxiliary variable, which means an ancilla qubit will be consumed by LIGHTER in this case.

C The Quantum Style Circuit of f_2 of AES S-box

$s_6 = s_6 \oplus t_{44} \cdot y_{15}$, $s_1 = s_1 \oplus t_{37} \cdot y_6$, $s_0 = s_0 \oplus t_{43} \cdot y_{16}$, $s_4 = s_4 \oplus t_{40} \cdot y_1$,
$s_3 = s_3 \oplus t_{44} \cdot y_{12}$, $s_5 = s_5 \oplus t_{37} \cdot y_3$, $s_2 = s_2 \oplus t_{43} \cdot y_{13}$, $s_7 = s_7 \oplus t_{40} \cdot y_5$,
$s_0 = s_0 \oplus s_4$, $s_6 = s_6 \oplus s_0$, $s_2 = s_2 \oplus t_{42} \cdot y_9$, $s_0 = s_0 \oplus t_{42} \cdot y_{11}$,
$s_5 = s_5 \oplus t_{45} \cdot y_{14}$, $s_0 = s_0 \oplus t_{45} \cdot y_{17}$, $s_7 = s_7 \oplus s_2$, $s_1 = s_1 \oplus s_6$,
$s_2 = s_2 \oplus t_{29} \cdot y_2$, $s_3 = s_3 \oplus s_5$, $s_6 = s_6 \oplus t_{33} \cdot y_0$, $s_4 = s_4 \oplus s_6$,
$s_4 = s_4 \oplus t_{29} \cdot y_7$, $s_5 = s_5 \oplus t_{33} \cdot y_4$, $s_3 = s_3 \oplus t_{42} \cdot y_9$, $s_6 = s_6 \oplus t_{45} \cdot y_{17}$,
$s_6 = s_6 \oplus t_{41} \cdot y_{10}$, $s_7 = s_7 \oplus t_{45} \cdot y_{14}$, $s_2 = s_2 \oplus s_6$, $s_5 = s_5 \oplus s_2$,
$s_2 = s_2 \oplus s_0$, $s_0 = s_0 \oplus s_3$, $s_3 = s_3 \oplus s_1$, $s_7 = s_7 \oplus s_4$,
$s_2 = s_2 \oplus t_{41} \cdot y_8$, $s_6 = s_6 \oplus s_7$, $s_4 = s_4 \oplus s_3$, $s_1 = s_1 \oplus s_0$.

D The Reversible Circuit of AES S-box

D.1 The Reversible Circuit for Generating $t_{21}, t_{22}, t_{23}, t_{24}$.

$x_6 = x_6 \oplus x_5 \oplus x_3 \oplus x_0$, $x_4 = x_6 \oplus x_5 \oplus x_4$, $x_1 = x_7 \oplus x_3 \oplus x_2 \oplus x_1$,
$x_5 = x_5 \oplus x_3$, $x_2 = x_5 \oplus x_2 \oplus x_0$, $x_3 = x_3 \oplus x_1 \oplus x_0$,
$x_0 = x_4 \oplus x_3 \oplus x_2 \oplus x_0$, $t_{21} = t_{21} \oplus x_6 \cdot x_4$, $t_{22} = t_{22} \oplus x_1 \cdot x_7$,
$t_{23} = t_{23} \oplus x_5 \cdot x_2$, $t_{24} = t_{24} \oplus x_3 \cdot x_0$, $t_{22} = t_{22} \oplus t_{21}$,
$t_{21} = t_{21} \oplus t_{23}$, $x_5 = x_6 \oplus x_5$, $x_4 = x_4 \oplus x_2$,
$x_1 = x_6 \oplus x_1$, $x_7 = x_7 \oplus x_4 \oplus x_2$, $x_3 = x_5 \oplus x_3 \oplus x_1$,
$x_0 = x_7 \oplus x_4 \oplus x_0$, $x_6 = x_6 \oplus x_5 \oplus x_3$, $x_2 = x_2 \oplus x_0$,
$t_{23} = t_{23} \oplus x_5 \cdot x_4$, $t_{21} = t_{21} \oplus x_1 \cdot x_7$, $t_{24} = t_{24} \oplus x_3 \cdot x_0$,
$a = a \oplus x_6 \cdot x_2$, $t_{21} = t_{21} \oplus a$, $t_{22} = t_{22} \oplus a$,
$t_{23} = t_{23} \oplus a$, $t_{24} = t_{24} \oplus a$, $x_1 = x_3 \oplus x_1$,
$x_7 = x_7 \oplus x_0$, $t_{22} = t_{22} \oplus x_3 \cdot x_0$, $t_{23} = t_{23} \oplus x_1 \cdot x_7$,
$t_{24} = t_{24} \oplus x_5 \cdot x_4$, $a = a \oplus x_6 \cdot x_2$, $x_6 = x_6 \oplus x_2$,
$t_{21} = t_{21} \oplus x_6$, $x_3 = x_3 \oplus x_0$, $t_{22} = t_{22} \oplus x_3$,
$x_5 = x_5 \oplus x_4$, $t_{23} = t_{23} \oplus x_5$, $x_5 = x_7 \oplus x_5 \oplus x_1$,
$t_{24} = t_{24} \oplus x_5$.

D.2 The Reversible Circuit for S_4 with Toffoli Depth 6.

$t_{23} = t_{23} \oplus t_{22} \cdot t_{24}$, $t_{24} = t_{24} \oplus t_{23}$, $t_{22} = t_{22} \oplus t_{21} \cdot t_{24}$,
$t_{24} = t_{24} \oplus t_{22} \cdot t_{23}$, $t_{23} = t_{23} \oplus t_{24}$ (t_{33}), $t_{22} = t_{22} \oplus t_{21}$ (t_{40}),
$t_{21} = t_{21} \oplus t_{22} \cdot t_{24}$, $a = a \oplus t_{23} \cdot t_{22}$, $t_{24} = t_{24} \oplus a \cdot t_{21}$ (t_{37}),
$t_{21} = t_{21} \oplus t_{22}$ (t_{29}).

D.3 The Reversible Circuit for S_4 with Toffoli Depth 5.

$t_{23} = t_{23} \oplus t_{22} \cdot t_{24}$, $t_{24} = t_{24} \oplus t_{23}$, $t_{22} = t_{22} \oplus t_{21} \cdot t_{24}$,
$t_{24} = t_{24} \oplus t_{22} \cdot t_{23}$, $t_{23} = t_{23} \oplus t_{24}$ (t_{33}), $t_{22} = t_{22} \oplus t_{21}$ (t_{40}),
$b = b \oplus t_{22}$, $a = a \oplus t_{23} \cdot t_{22}$, $t_{21} = t_{21} \oplus b \cdot t_{24}$,
$t_{24} = t_{24} \oplus a \cdot t_{21}$ (t_{37}), $t_{21} = t_{21} \oplus t_{22}$ (t_{29}).

D.4 The Reversible Circuit for the Outputs of AES S-box.

$x_3 = x_3 \oplus x_1 \oplus x_0$, $x_0 = x_4 \oplus x_2 \oplus x_0$, $x_6 = x_6 \oplus x_2$,
$t_{22} = t_{22} \oplus t_{21}$, $t_{23} = t_{24} \oplus t_{23}$, $t_{21} = t_{24} \oplus t_{23} \oplus t_{21}$,
$s_0 = s_0 \oplus t_{22} \cdot x_4$, $s_5 = s_5 \oplus t_{24} \cdot x_3$, $s_6 = s_6 \oplus t_{23} \cdot x_0$,
$s_2 = s_2 \oplus t_{21} \cdot x_6$, $x_5 = x_7 \oplus x_5 \oplus x_4 \oplus x_1$, $x_3 = x_6 \oplus x_3 \oplus x_1$,
$t_{23} = t_{23} \oplus t_{22}$, $t_{24} = t_{24} \oplus t_{23} \oplus t_{21}$, $s_2 = s_2 \oplus t_{22} \cdot x_5$,
$s_5 = s_5 \oplus t_{23} \cdot x_3$, $s_4 = s_4 \oplus t_{24} \cdot x_7$, $s_0 = s_0 \oplus s_4$,
$s_6 = s_6 \oplus s_0$, $s_7 = s_7 \oplus s_2$, $s_1 = s_1 \oplus s_6$,
$s_3 = s_3 \oplus s_5$, $x_7 = x_7 \oplus x_4 \oplus x_2$, $x_4 = x_4 \oplus x_0$,
$t_{22} = t_{24} \oplus t_{22} \oplus t_{21}$, $s_6 = s_6 \oplus t_{22} \cdot x_7$, $s_3 = s_3 \oplus t_{21} \cdot x_6$,
$s_0 = s_0 \oplus t_{23} \cdot x_4$, $s_4 = s_4 \oplus s_6$, $x_5 = x_5 \oplus x_1$,
$t_{22} = t_{22} \oplus t_{21}$, $s_6 = s_6 \oplus t_{23} \cdot x_4$, $s_0 = s_0 \oplus t_{21} \cdot x_2$,
$s_7 = s_7 \oplus t_{24} \cdot x_1$, $s_2 = s_2 \oplus t_{22} \cdot x_5$, $x_4 = x_4 \oplus x_2$,
$x_0 = x_7 \oplus x_0$, $x_7 = x_7 \oplus x_2$, $x_1 = x_5 \oplus x_3 \oplus x_1$,
$t_{23} = t_{23} \oplus t_{21}$, $t_{24} = t_{24} \oplus t_{23}$, $t_{21} = t_{24} \oplus t_{22} \oplus t_{21}$,
$s_6 = s_6 \oplus t_{23} \cdot x_4$, $s_1 = s_1 \oplus t_{24} \cdot x_0$, $s_4 = s_4 \oplus t_{22} \cdot x_7$,
$s_3 = s_3 \oplus t_{21} \cdot x_1$, $s_2 = s_2 \oplus s_6$, $s_5 = s_5 \oplus s_2$,
$s_2 = s_2 \oplus s_0$, $s_0 = s_0 \oplus s_3$, $s_3 = s_3 \oplus s_1$,
$s_7 = s_7 \oplus s_4$, $x_6 = x_6 \oplus x_3$, $x_5 = x_6 \oplus x_5 \oplus x_3$,
$t_{22} = t_{24} \oplus t_{23} \oplus t_{22} \oplus t_{21}$, $t_{21} = t_{24} \oplus t_{21}$, $s_2 = s_2 \oplus t_{23} \cdot x_6$,
$s_7 = s_7 \oplus t_{22} \cdot x_3$, $s_5 = s_5 \oplus t_{21} \cdot x_5$, $s_6 = s_6 \oplus s_7$,
$s_4 = s_4 \oplus s_3$, $s_1 = s_1 \oplus s_0$, $s_6 = s_6 \oplus 1$,
$s_7 = s_7 \oplus 1$, $s_1 = s_1 \oplus 1$, $s_2 = s_2 \oplus 1$.

E The Reversible Circuit of AES S-box^{-1}

E.1 The Reversible Circuit for Generating $t_{21}, t_{22}, t_{23}, t_{24}$.

$x_6 = x_7 \oplus x_6 \oplus x_1 \oplus x_0 \oplus 1$, $x_1 = x_5 \oplus x_3 \oplus x_2 \oplus x_1$, $x_3 = x_6 \oplus x_3 \oplus x_0$,
$x_0 = x_5 \oplus x_2 \oplus x_0 \oplus 1$, $x_4 = x_4 \oplus x_1 \oplus x_0$, $x_5 = x_7 \oplus x_6 \oplus x_5 \oplus x_4 \oplus 1$,
$x_7 = x_7 \oplus x_5 \oplus x_2 \oplus x_1$, $x_2 = x_3 \oplus x_2 \oplus 1$, $t_{21} = t_{21} \oplus x_6 \cdot x_1$,
$t_{22} = t_{22} \oplus x_3 \cdot x_0$, $t_{23} = t_{23} \oplus x_4 \cdot x_5$, $t_{24} = t_{24} \oplus x_7 \cdot x_2$,
$x_6 = x_6 \oplus x_4$, $x_5 = x_5 \oplus x_1$, $x_3 = x_6 \oplus x_4 \oplus x_3$,
$x_0 = x_1 \oplus x_0$, $x_7 = x_7 \oplus x_6 \oplus x_3$, $x_2 = x_5 \oplus x_2 \oplus x_0$,
$x_4 = x_7 \oplus x_4$, $x_1 = x_5 \oplus x_2 \oplus x_1$, $t_{22} = t_{22} \oplus t_{21}$,
$t_{21} = t_{21} \oplus t_{23}$, $t_{23} = t_{23} \oplus x_6 \cdot x_5$, $t_{21} = t_{21} \oplus x_3 \cdot x_0$,
$t_{24} = t_{24} \oplus x_7 \cdot x_2$, $a = a \oplus x_4 \cdot x_1$, $t_{21} = t_{21} \oplus a$,
$t_{22} = t_{22} \oplus a$, $t_{23} = t_{23} \oplus a$, $t_{24} = t_{24} \oplus a$,
$x_3 = x_7 \oplus x_3$, $x_0 = x_2 \oplus x_0$, $a = a \oplus x_4 \cdot x_1$,
$t_{22} = t_{22} \oplus x_7 \cdot x_2$, $t_{23} = t_{23} \oplus x_3 \cdot x_0$, $t_{24} = t_{24} \oplus x_6 \cdot x_5$,
$x_4 = x_4 \oplus x_1$, $t_{21} = t_{21} \oplus x_4$, $x_2 = x_7 \oplus x_2$,
$t_{22} = t_{22} \oplus x_2$, $x_5 = x_6 \oplus x_5$, $t_{23} = t_{23} \oplus x_5$,
$x_5 = x_5 \oplus x_3 \oplus x_0$, $t_{24} = t_{24} \oplus x_5$.

E.2 The Reversible Circuit for the Outputs of AES S-box^{-1}.

$t_{22} = t_{22} \oplus t_{21}$, $t_{21} = t_{23} \oplus t_{21}$, $t_{23} = t_{24} \oplus t_{23}$,
$x_5 = x_6 \oplus x_5 \oplus x_3 \oplus x_0$, $x_4 = x_4 \oplus x_1$, $x_2 = x_7 \oplus x_5 \oplus x_2 \oplus x_1$,
$x_7 = x_7 \oplus x_3$, $s_2 = s_2 \oplus t_{22} \cdot x_5$, $s_4 = s_4 \oplus t_{21} \cdot x_4$,
$s_3 = s_3 \oplus t_{23} \cdot x_2$, $s_7 = s_7 \oplus t_{24} \cdot x_7$, $t_{24} = t_{24} \oplus t_{23} \oplus t_{22} \oplus t_{21}$,
$t_{23} = t_{23} \oplus t_{22}$, $x_7 = x_7 \oplus x_4 \oplus x_3$, $s_4 = s_4 \oplus t_{22} \cdot x_6$,
$s_1 = s_1 \oplus t_{21} \cdot x_4$, $s_0 = s_0 \oplus t_{24} \cdot x_0$, $s_7 = s_7 \oplus t_{23} \cdot x_7$,
$s_2 = s_2 \oplus s_0$, $s_3 = s_3 \oplus s_2$, $s_6 = s_6 \oplus s_4$,
$s_5 = s_5 \oplus s_3$, $s_1 = s_1 \oplus s_7$, $t_{24} = t_{24} \oplus t_{22} \oplus t_{21}$,
$t_{22} = t_{23} \oplus t_{22}$, $t_{21} = t_{24} \oplus t_{21}$, $x_0 = x_5 \oplus x_1 \oplus x_0$,
$x_7 = x_7 \oplus x_6$, $x_5 = x_5 \oplus x_2$, $x_6 = x_6 \oplus x_3$,
$s_3 = s_3 \oplus t_{24} \cdot x_0$, $s_1 = s_1 \oplus t_{22} \cdot x_7$, $s_2 = s_2 \oplus t_{23} \cdot x_5$,
$s_4 = s_4 \oplus t_{21} \cdot x_6$, $s_0 = s_0 \oplus s_3$, $t_{22} = t_{24} \oplus t_{22}$,
$t_{24} = t_{24} \oplus t_{21}$, $x_0 = x_1 \oplus x_0$, $x_2 = x_2 \oplus x_1 \oplus x_0$,
$s_3 = s_3 \oplus t_{23} \cdot x_5$, $s_0 = s_0 \oplus t_{21} \cdot x_0$, $s_5 = s_5 \oplus t_{22} \cdot x_2$,
$s_2 = s_2 \oplus t_{24} \cdot x_1$, $t_{24} = t_{24} \oplus t_{23}$, $x_1 = x_5 \oplus x_1$,
$x_7 = x_7 \oplus x_6 \oplus x_3$, $s_3 = s_3 \oplus t_{24} \cdot x_1$, $s_6 = s_6 \oplus t_{23} \cdot x_7$,
$s_4 = s_4 \oplus s_3$, $s_7 = s_7 \oplus s_4$, $t_{22} = t_{24} \oplus t_{22}$,
$t_{21} = t_{24} \oplus t_{23} \oplus t_{21}$, $x_7 = x_7 \oplus x_4$, $x_6 = x_6 \oplus x_4$,
$s_4 = s_4 \oplus t_{24} \cdot x_7$, $s_6 = s_6 \oplus t_{22} \cdot x_3$, $s_7 = s_7 \oplus t_{21} \cdot x_6$,
$s_6 = s_6 \oplus s_2$, $s_0 = s_0 \oplus s_6$, $s_1 = s_1 \oplus s_4$,
$s_4 = s_4 \oplus s_0$, $s_2 = s_2 \oplus s_5$, $s_0 = s_0 \oplus s_3$,
$s_4 = s_4 \oplus s_7$, $s_2 = s_2 \oplus s_7$, $s_7 = s_7 \oplus s_1$,
$s_1 = s_1 \oplus s_6$, $s_1 = s_1 \oplus s_5$, $s_3 = s_3 \oplus s_6$,
$s_5 = s_5 \oplus s_0$.

F The Reversible Circuit Added If Not All Output Qubits Are 0s.

$s_5 = s_5 \oplus s_0$, $s_3 = s_3 \oplus s_6$, $s_1 = s_1 \oplus s_5$, $s_1 = s_1 \oplus s_6$, $s_7 = s_7 \oplus s_1$,
$s_2 = s_2 \oplus s_7$, $s_4 = s_4 \oplus s_7$, $s_0 = s_0 \oplus s_3$, $s_2 = s_2 \oplus s_5$, $s_4 = s_4 \oplus s_0$,
$s_1 = s_1 \oplus s_4$, $s_0 = s_0 \oplus s_6$, $s_6 = s_6 \oplus s_2$, $s_7 = s_7 \oplus s_4$, $s_4 = s_4 \oplus s_3$,
$s_0 = s_0 \oplus s_3$, $s_1 = s_1 \oplus s_7$, $s_5 = s_5 \oplus s_3$, $s_6 = s_6 \oplus s_4$, $s_3 = s_3 \oplus s_2$,
$s_2 = s_2 \oplus s_0$.

G An Example of Calculating the CNOT Gate Consumption of AES-128.

For the NCT-based circuit of AES-128, the number of CNOT gates consumed by the **AddRoundKey** and the **MixColumns** are $128 \times r$ and $4 \times 92 \times (r-1)$, respectively, where r is the round number. Suppose that 5 ancilla qubits are allocated for each of the m parallel S-boxes in the round function, one can

also use the S-box^{-1} circuit that consumes 5 ancilla qubits to remove the previous round, after which, there are $8 \times m + 5 \times m$ qubits with value zero available for the S-boxes in the key schedule. For the case that $\frac{16}{m} \in \mathbb{Z}_+$, $8 \times m + 5 \times m > 6$ always hold. It follows that the S-boxes in the key schedule can be implemented with the circuit that consumes 6 ancilla qubits. In each round of AES-128, it requires 16 S-boxes to implement the **SubBytes** in the round function, 16 S-box^{-1}es to remove the previous round, and 4 S-boxes for the key schedule. Denote by $Cnot_{S_5^*}$, $Cnot_{S_5^{-1*}}$, $Cnot_{S_6^*}$ the CNOT gate count of the S-box circuit that consumes 5 ancilla qubits, the CNOT gate count of the S-box^{-1} circuit that consumes 5 ancilla qubits, and the CNOT gate count of the S-box circuit that consumes 6 ancilla qubits, respectively. The CNOT consumption of the nonlinear components except the first round can be calculated as $(16 \times Cnot_{S_5{}^*} + 16 \times Cnot_{S_5^{-1*}}) \times (r-1) + 4 \times Cnot_{S_6^*} \times w'$, where $w^{'}$ is the number of **SubWord** operations used in the key schedule except the first round and equals 9 for AES-128. In addition, word-wise XOR is applied in the key schedule to implement $W_i = W_{i-4} \oplus W_{i-1}$, which means $3 \times 32 \times w$ CNOT gates are required, where w is the number of **SubWord** operations used in the key schedule and equals 10 for AES-128.