Quantum Circuit Implementations of AES with Fewer Qubits

Jian Zou[1,2], Zihao Wei[3,4], Siwei Sun[3,4⋆], Ximeng Liu[1,2], Wenling Wu[5]

[1]Mathematics and Computer Science of Fuzhou University,
Fuzhou, Fujian Province, China
[2]Key Lab of Information Security of Network Systems (Fuzhou University),
Fujian Province, China `fzuzoujian15@163.com, snbnix@gmail.com`
[3]State Key Laboratory of Information Security, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, China
[4]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
`weizihao@iie.ac.cn, siweisun.isaac@gmail.com`
[5]Institute of Software, Chinese Academy of Sciences, Beijing, China
`wwl@tca.iscas.ac.cn`

Abstract. We propose some quantum circuit implementations of AES with the following improvements. Firstly, we propose some quantum circuits of the AES S-box and S-box^{-1}, which require fewer qubits than prior work. Secondly, we reduce the number of qubits in the zig-zag method by introducing the S-box^{-1} operation in our quantum circuits of AES. Thirdly, we present a method to reduce the number of qubits in the key schedule of AES. While the previous quantum circuits of AES-128, AES-192, and AES-256 need at least 864, 896, and 1232 qubits respectively, our quantum circuit implementations of AES-128, AES-192, and AES-256 only require 512, 640, and 768 qubits respectively, where the number of qubits is reduced by more than 30%.

Key words: AES, S-box, S-box^{-1}, quantum circuit, circuit complexity

1 Introduction

In the post-quantum era, we need to study the security of cryptographic systems against quantum attackers. In fact, many cryptographic schemes turn out to be less secure against attacks based on quantum computing. Some asymmetric cryptographic primitives face devastating attacks due to Shor's algorithm [24]. In contrast, the impact of quantum computing on secret-key cryptography seems to be less severe. Most of the existing works are based on Grover's algorithm [12] and Simon's algorithm [25]. Grover's algorithm can solve the search problem with quadratic speed-up, while Simon's algorithm can find the hidden period with polynomially many quantum queries. In such attacks, the corresponding quantum oracle of the target cipher has to be implemented. Due to the importance of AES, it is one of the most studied ciphers [11,17,3,18,15] in

⋆ Corresponding author.

the context of efficient synthesis of quantum circuits. These implementations can be potentially used in some quantum attacks against symmetric-key primitives involving AES [4,13,16,9]. In this paper, we construct some quantum circuits of AES with fewer qubits, and the techniques involved may provide more flexible qubit and circuit depth trade-offs for the quantum circuits of AES.

A quantum oracle for any classical vectorial Boolean function can be constructed with the Clifford + T gate set, which consists of the Hadamard gate (H), Phase gate (S), controlled-NOT gate (CNOT), and non-clifford T gate. There are some works on synthesizing optimal reversible circuits, such as reversible Boolean functions. Shende *et al.* [22] considered the synthesis of 3-bit reversible logic circuits using NOT gate, CNOT gate, and Toffoli gate. Golubitsky *et al.* [10] proposed an optimal 4-bit reversible circuits composed with NOT gate, CNOT gate, Toffoli gate, and the 4-bit Toffoli gate. The goal of synthesizing the optimal quantum circuit implementation is to reduce the circuit depth and number of qubits [11,17,3,18]. According to our current understanding of fault-tolerant quantum computing, the metric of T-depth is probably the most important. However, before practical quantum computers are built, the method for reducing the cost with respect to the number of qubits is also very meaningful, and it may provides more flexible qubit and depth trade-offs.

Recently, the construction of efficient quantum circuits of AES has attracted much attention. In [8], Datta *et al.* presented a reversible implementation of AES. In [15], Jaques *et al.* proposed a method to minimize the depth-times-width cost metric for quantum circuits of AES. In [11], Grassl et al. proposed a quantum circuit of AES aiming at the lowest possible number of qubits. In [17], Kim *et al.* showed some time-memory trade-offs for key search on AES. In [3], Almazrooie *et al.* presented a new quantum circuit of AES-128. By utilizing the classical algebraic structure of the S-box [5], Langenberg *et al.* in [18] showed a new way to construct the quantum circuit of AES's S-box, based on which Langenberg *et al.* proposed an efficient quantum circuit of AES-128. Compared to Almazrooie *et al.*'s and Grassl *et al.*'s estimates, the circuit proposed by Langenberg *et al.* could reduce the number of qubits and Toffoli gates simultaneously. Langenberg *et al.*'s work shows that we can construct an improved quantum circuit of AES by constructing a more efficient classical circuit of AES.

There are several works on how to reduce the gate number of AES in the classical setting [14,7,1,19,28]. In [14], Itoh and Tsujii proposed the tower field architecture for calculating multiplicative inverse in $\mathbb{F}_2$, which was a powerful technique for designing compact hardware implementation of S-box. By using the tower field technique, Canright in [7] showed an efficient method for computing the multiplicative inverse of the input. In [6], Boyar and Peralta proposed a depth 16 circuit for the S-box in AES by using the tower field implementation.

Contribution. Firstly, we propose an improved quantum circuit for the S-box^{-1} of AES based on the improved classical circuit of the inverse of the AES S-box [28,29]. Also, by exploiting some useful linear relationship, we propose some improved qubit-depth trade-offs for the quantum circuits of S-box/S-box^{-1} of

AES. The improvements of the S-box and its inverse lead to corresponding improvements of the quantum circuits of the round function and the key-schedule algorithm of AES. Taking AES-128 as an example, we can generate W_{4i} by XORing $SubWord(RotWord(W_{4i-1})), Rcon(i/s)$, W_{4i-1}, W_{4i-5}, $W4_{4i-9}$ to W_{4i-13} (for $4 \leq i \leq 10$). In other words, we can obtain W_{4i} without introducing new qubits or cleaning up W_{4i-13} (for $4 \leq i \leq 10$). That is, our quantum circuit for the key schedules of AES-128/-192/-256 need 128/192/256 qubit, and 6 ancillas qubits, which require fewer qubits than the previous works [11,3,18,15].

Secondly, we propose an improved zig-zag method with fewer qubits. To compute the output of the AES round function, we need 256 qubits to store the 128 qubits input and the 128 qubits output of the round function. In other words, we need at least 256 qubits in the zig-zag method. By using our quantum circuits of AES's S-box and S-box^{-1}, we propose an improved zigzag method for AES-128/-192/-256 with 256 qubits, which matches the minimum values. That is, our improved zig-zag method require 256/256/256 qubits for AES-128/-192/-256, while the prior work needed at least 528/656/656 qubits for AES-128/-192/-256, respectively.

We summarize the quantum resources to implement AES in Table 1. The # Toffoli/CNOT/NOT means the number of Toffoli gates, CNOT gates, and NOT gates, and # qubits means the number of qubits. We will adopt the same notations in the following tables. As shown in Table 1, our quantum circuit implementations of AES require fewer qubits than the prior works. Also, our quantum circuits of AES-128/-256 can obtain the best trade-off of $T \cdot M$, where T is the Toffoli depth and M is the number of qubits.

Table 1. Summary of the quantum resources to implement AES

Algorithm	# qubits	Toffoli Depth	# Toffoli	# CNOT	# NOT	$T \cdot M$	Source
AES-128	984	12672	151552	166548	1456	12469248	[11]
	976	not reported	150528	192832	1370	not reported	[3]
	864	1880	16940	107960	1570	1624320	[18]
	512	2016	19788	128517	4528	1032192	Sect. 6.1
AES-192	1112	11088	172032	189432	1608	12329856	[11]
	896	1640	19580	125580	1692	1469440	[18]
	640	2022	22380	152378	5128	1294080	Sect. 6.2
AES-256	1336	14976	215040	233836	1943	20007936	[11]
	1232	2160	23760	151011	1992	2661120	[18]
	768	2292	26774	177645	6103	1760256	Sect. 6.2

Remark. In this work, the Toffoli-count and Toffoli-depth are involved in our metric. A more fine-grained and accurate approach is to implement the entire circuit with the Clifford+T set, count the number of T gate, and measure the T-depth as was done in [15]. In [15], the quantum circuit was implemented with

Q# [26] and the cost of the quantum circuit was estimated by the resource estimator of Q#. However, it seems that there are some issues with the resource estimator (see https://github.com/microsoft/qsharp-runtime/issues/192). So we do not use it here.

Outline. In Section 2, we present the definitions of some quantum gates. Section 3 not only makes a brief introduction to AES, but also shows the algebraic structures of AES's S-box/S-box^{-1}. In Section 4, we propose our improved quantum circuits of AES's S-box and S-box^{-1}. Section 5 shows our improved ideas for the zig-zag method and the key schedule of AES. In Section 6, we show our improved quantum circuit implementations of AES. We conclude this paper in Section 7.

2 Notations

The classical circuits allow wires to be joined together, such as $a = a \oplus b$ and $a = a \wedge b$. Obviously these operations are not reversible and not unitary. Different from the classical circuits, quantum circuits shall be reversible and unitary, which can be constructed by replacing classical gates with quantum gates. For example, we shall simulate AND gates with the Toffoli gate, while a XOR gate can be simulated with the CNOT gate.

Some prior works [2] showed that the quantum circuit consisting only of Clifford gates were not advantageous over classical computing. In other words, we shall adopt some non-Clifford gates (i.e. Toffoli gate) to obtain the quantum benefit. Also, some works [23,27] showed the Toffoli gate and Clifford gates were universal. That is, we can implement any quantum computation by these gates. As shown in [20], the Clifford groups are much cheaper than the Toffoli gate (or T-gate). As a result, [11,17,18] defined the Toffoli depth as the time cost of the algorithm, while the memory cost is the total number of logical qubits required to perform the quantum algorithm. Similar to [11,17,18], we define the time and memory cost of our quantum circuit implementation of AES as follows.

Definition 1. *A unit of quantum computational time cost is defined as the time for running a nonparallelizable logical Toffoli gate.*

Definition 2. *The space cost of the quantum circuit is defined as the number of logical qubits for the entire quantum computational.*

Apart from the two definitions, we also clarify three kinds of qubits to avoid the confusions.

1. **Data qubits** are written as the input message, such as the round key or the input plaintext.
2. **Ancilla qubits** (or called garbage qubits) are initialized qubits those assist certain operation, which get written unwanted information after a certain operation. Note that we shall clean up the ancilla qubits at the end of the quantum circuit.

3. **Output qubits** contain the output information of a certain operation. Note that we do not need to clean up the output qubits.

Based on the definitions of three types of qubits, we adopt the following two strategies to reduce the number of qubits. First, we shall avoid applying the Toffoli gate to ancilla qubits, because these wires shall be cleaned up. However, we do not need to clean up the output qubits. As a result, we shall apply the Toffoli gates to output qubits to avoid involving them in the cleanup process. Second, some ancilla qubits remained idle until the end of the quantum circuit. By uncomputing these wires, we can reuse these ancilla qubits instead of introducing new ancilla qubits, which can reduce the number of qubits.

3 The AES Block Cipher

AES [21] is a family of iterative block ciphers based on the SPN structure. Its members with 128-bit, 192-bit, and 256-bit keys are denoted as AES-128 (10-round), AES-192 (12-round), and AES-256 (14-round), respectively. We will show the round function and key schedule of AES in the following. We refer the reader to [21] for the full description of AES.

3.1 Specification of AES

The AES round function consists the following four operations: **AddRoundKey**$\circ$ **MixColumns** $\circ$ **ShiftRows** $\circ$ **SubBytes**, where

- **AddRoundKey** exclusive-ors each round key to the state;
- **SubBytes** is the only non-linear transformation in AES, which applies an 8-bit S-box to the 16 bytes of the state in parallel. The algebraic structure of S-box is shown in Section 3.2.
- **ShiftRows** cyclically rotates the cells of the i-th row to the left by i-byte (for $0 \le i \le 3$).
- **MixColumns** does a linear transformation on each column of the state with the MDS matrix
$$M = \begin{bmatrix} \texttt{0x02} & \texttt{0x03} & \texttt{0x01} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x02} & \texttt{0x03} & \texttt{0x01} \\ \texttt{0x01} & \texttt{0x01} & \texttt{0x02} & \texttt{0x03} \\ \texttt{0x03} & \texttt{0x01} & \texttt{0x01} & \texttt{0x01} \end{bmatrix}.$$

Similar to the encryption procession of AES, the decryption process of AES also consists of four operations **AddRoundKey** $\circ$ **InvMC** $\circ$ **InvShiftRows** $\circ$ **InvSubBytes**, where

- **AddRoundKey** exclusive-ors the round key to the state;
- **InvSubBytes** is the inverse operation of SubBytes;
- **InvShiftRows** cyclically rotates the cells of the i-th row to the right by i-byte (for $0 \le i \le 3$).
- **InvMC** does a linear transformation on each column with the MDS matrix
$$M^{-1} = \begin{bmatrix} \texttt{0x0E} & \texttt{0x0B} & \texttt{0x0D} & \texttt{0x09} \\ \texttt{0x09} & \texttt{0x0E} & \texttt{0x0B} & \texttt{0x0D} \\ \texttt{0x0D} & \texttt{0x09} & \texttt{0x0E} & \texttt{0x0B} \\ \texttt{0x0B} & \texttt{0x0D} & \texttt{0x09} & \texttt{0x0E} \end{bmatrix}.$$

The key schedules of AES-128/-192/-256 are described in Algorithm 1 and Algorithm 2. The parameters s and t used in the key schedules of AES-128 are $s = 4$, $t = 43$, while AES-192 adopts $s = 6$, $t = 51$.

Algorithm 1 The key schedules of AES-128 and AES-192

```
For $i = s$ till $i = t$ do
    If $i \equiv 0 \mod s$, then
        $W_i = W_{i-s} \oplus \textbf{SubWord}(\textbf{RotWord}(W_{i-1})) \oplus \textbf{Rcon}(i/s)$;
    else $W_i = W_{i-s} \oplus W_{i-1}$.
```

Algorithm 2 The key schedules of AES-256

```
For $i = 8$ till $i = 59$ do
    If $i \equiv 0 \mod 8$, then
        $W_i = W_{i-8} \oplus \textbf{SubWord}(\textbf{RotWord}(W_{i-1})) \oplus \textbf{Rcon}(i/8)$;
    If $i \equiv 4 \mod 8$, then
        $W_i = W_{i-8} \oplus \textbf{SubWord}(W_{i-1})$;
    else $W_i = W_{i-8} \oplus W_{i-1}$.
```

The operations **RotWord**, **Rcon** and **SubWord** used in Algorithm 1 and Algorithm 2 are explained as follows.

- **RotWord** cyclically rotates the four bytes to the left by 1-byte;
- **Rcon** exclusive-ors the constant to each byte of the word;
- **SubWord** applies an S-box operation to each byte of the word.

3.2 The algebraic structures of the S-box of AES

There are several ways to implement the S-box of AES. In [5], Boyar and Peralta showed an efficient way to compute AES's S-box by using the tower field architecture. Since we do not find a circuit with fewer AND gate than the classical circuit proposed by Boyar and Peralta [5], we adopt their classical circuit to construct our quantum circuit of AES's S-box in the Sect. 4. Their circuit represents AES's S-box as $S(x) = B_S \cdot F_S(U_S \cdot x)$, where the matrix U_S takes $x_0, x_1, \cdots, x_7$ as input and outputs $x_7, y_1, \cdots, y_{21}$.

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

The function $F_S : \mathbb{F}_2^{22} \to \mathbb{F}_2^{18}$ takes $x_7, y_1, \cdots, y_{21}$ as input and outputs $z_0, z_1, \cdots, z_{17}$.

$$\begin{array}{llll}
t_2 = y_{12} \cdot y_{15}, & t_3 = y_3 \cdot y_6, & t_4 = t_3 \oplus t_2, & t_5 = y_4 \cdot x_7, \\
t_6 = t_5 \oplus t_2, & t_7 = y_{13} \cdot y_{16}, & t_8 = y_5 \cdot y_1, & t_9 = t_8 \oplus t_7, \\
t_{10} = y_2 \cdot y_7, & t_{11} = t_{10} \oplus t_7, & t_{12} = y_9 \cdot y_{11}, & t_{13} = y_{14} \cdot y_{17}, \\
t_{14} = t_{13} \oplus t_{12}, & t_{15} = y_8 \cdot y_{10}, & t_{16} = t_{15} \oplus t_{12}, & t_{17} = t_4 \oplus y_{20}, \\
t_{18} = t_6 \oplus t_{16}, & t_{19} = t_9 \oplus t_{14}, & t_{20} = t_{11} \oplus t_{16}, & t_{21} = t_{17} \oplus t_{14}, \\
t_{22} = t_{18} \oplus y_{19}, & t_{23} = t_{19} \oplus y_{21}, & t_{24} = t_{20} \oplus y_{18}, & \\
t_{25} = t_{21} \oplus t_{22}, & t_{26} = t_{21} \cdot t_{23}, & t_{27} = t_{24} \oplus t_{26}, & t_{28} = t_{25} \cdot t_{27}, \\
t_{29} = t_{28} \oplus t_{22}, & t_{30} = t_{23} \oplus t_{24}, & t_{31} = t_{22} \oplus t_{26}, & t_{32} = t_{31} \cdot t_{30}, \\
t_{33} = t_{32} \oplus t_{24}, & t_{34} = t_{23} \oplus t_{33}, & t_{35} = t_{27} \oplus t_{33}, & t_{36} = t_{24} \cdot t_{35}, \\
t_{37} = t_{36} \oplus t_{34}, & t_{38} = t_{27} \oplus t_{36}, & t_{39} = t_{29} \cdot t_{38}, & t_{40} = t_{25} \oplus t_{39}, \\
t_{41} = t_{40} \oplus t_{37}, & t_{42} = t_{29} \oplus t_{33}, & t_{43} = t_{29} \oplus t_{40}, & t_{44} = t_{33} \oplus t_{37}, \\
t_{45} = t_{42} \oplus t_{41}, & z_0 = t_{44} \cdot y_{15}, & z_1 = t_{37} \cdot y_6, & z_2 = t_{33} \cdot x_7, \\
z_3 = t_{43} \cdot y_{16}, & z_4 = t_{40} \cdot y_1, & z_5 = t_{29} \cdot y_7, & z_6 = t_{42} \cdot y_{11}, \\
z_7 = t_{45} \cdot y_{17}, & z_8 = t_{41} \cdot y_{10}, & z_9 = t_{44} \cdot y_{12}, & z_{10} = t_{37} \cdot y_3, \\
z_{11} = t_{33} \cdot y_4, & z_{12} = t_{43} \cdot y_{13}, & z_{13} = t_{40} \cdot y_5, & z_{14} = t_{29} \cdot y_2, \\
z_{15} = t_{42} \cdot y_9, & z_{16} = t_{45} \cdot y_{14}, & z_{17} = t_{41} \cdot y_8. &
\end{array}$$

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

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

3.3 Our Improved Classical Circuit of the S-box^{-1} of AES

By using the tower technique, we propose an improved implementation of the S-box^{-1} (see in Table 2), which can be used to construct our quantum circuit of AES's S-box^{-1}. We can express AES's S-box^{-1} as $S^{-1}(x) = B' \cdot F'(U' \cdot x)$, where the matrix $U' \in F_2^{8 \times 22}$ takes $x_0, x_1, \cdots, x_7$ as input and outputs $y_0, y_1, \cdots, y_{21}$, where $U_i = x_i$ (for $0 \le i \le 7$).

$$\begin{array}{llll}
y_5 = \overline{U_1}, & y_4 = U_5 \oplus U_0, & y_{13} = U_2 \oplus y_5, & y_6 = y_4 \oplus y_{13}, \\
y_9 = y_5 \oplus y_4, & y_{20} = U_4 \oplus y_4, & y_{18} = \overline{U_6}, & y_2 = y_6 \oplus y_{18}, \\
t_0 = U_1 \oplus U_0, & y_7 = U_4 \oplus t_0, & y_{17} = y_6 \oplus y_7, & y_{16} = U_7 \oplus t_0, \\
y_3 = y_2 \oplus y_{16}, & y_{15} = y_5 \oplus y_7, & y_{11} = y_9 \oplus y_{17}, & y_{19} = y_{17} \oplus y_{16}, \\
t_1 = U_3 \oplus t_0, & y_1 = y_{20} \oplus t_1, & y_{14} = y_3 \oplus y_1, & y_{12} = U_2 \oplus t_1, \\
y_0 = y_2 \oplus y_{12}, & y_{10} = y_{14} \oplus y_{12}, & y_8 = y_1 \oplus y_0, & y_{21} = U_7 \oplus y_{12}.
\end{array}$$

The non-linear function $F' : \mathbb{F}_2^{22} \to \mathbb{F}_2^{18}$ takes $y_0, y_1, \cdots, y_{21}$ as input and outputs $z_0, z_1, \cdots, z_{17}$.

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

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

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

Table 2. Summary of the resources to implement AES's S-box^{-1}

Algorithm	# XOR/XNOR	XOR3	# NAND	# AND	# NOR	# NOT	Source
S-box	96	0	0	36	0	0	[19]
	80	0	34	0	6	0	[7]
	83	0	0	32	0	0	[5]
	81	0	0	32	0	0	[1]
	69	0	33	0	8	0	[28]
	51	9	33	0	8	0	[28]
S-box^{-1}	87	0	0	34	0	0	[1]
	81	0	34	0	0	6	[7]
	82	0	0	32	0	4	Sect. 3.3

4 The Quantum Circuits for the basic AES operations

4.1 Quantum Circuits for three linear transformations of AES

As pointed out in [11], the three linear transformations of AES can be implemented with the CNOT gates as follows. We just adopt their quantum circuit of three linear transformations in our quantum circuits of AES.

1. **AddRoundKey:** The AddRoundKey transformation xors 128-bit roundkey to the state, which can be executed with 128 CNOT gates in parallel.
2. **ShiftRows:** Since the ShiftRows transformation just permutes the order of the sixteen bytes of AES, we do not need any quantum gates to execute these operations.
3. **MixColumns:** The MixColumns transformation operates a column (32 bits) at a time, which can be specified with a 32×32 matrix. The resultant circuit of MixColumns has 277 CNOT gates with a total depth of 39, which can be estimated by an LUP-decomposition [11].

In the following, we present our improved quantum circuit implementations of AES's S-box and S-box^{-1}. The details of our implementation of AES S-box and S-box^{-1} are available at https://github.com/Asiacrypt2020submission370/aes/.

4.2 Improved Quantum Circuit Implementations of AES's S-box

In this subsection, we propose some improved quantum circuit implementations of AES's S-box. Our quantum circuit of AES's S-box considers the following two cases: $|x\rangle|0^a\rangle \longrightarrow |x\rangle|S(x)\rangle|0^{a-8}\rangle$ and $|x\rangle|b\rangle|0^{a-8}\rangle \longrightarrow |x\rangle|S(x) \oplus b\rangle|0^{a-8}\rangle$. Note that the prior works [11,18] only considered $|x\rangle|0^a\rangle \longrightarrow |x\rangle|S(x)\rangle|0^{a-8}\rangle$.

Firstly, we improve the quantum circuit sending $|x\rangle|0^8\rangle$ to $|x\rangle|S(x)\rangle$. In this part, we propose an improved quantum circuit of AES's S-box, which requires fewer qubits than the prior work. In detail, our quantum circuit of AES's S-box requires only 6 ancilla qubits, which maps $|x\rangle|0^{14}\rangle$ to $|x\rangle|S(x)\rangle|0^6\rangle$. The prior work needed at least 16 ancilla qubits to compute the Sbox, which maps $|x\rangle|0^{24}\rangle$ to $|x\rangle|S(x)\rangle|0^{16}\rangle$. Our improved quantum circuits of AES's S-box adopt the following two new observations, which are based on the algebraic structures of the S-box (see Section 3.2).

Observation 1. As shown in section 3.2, the 18 values of $z_0, \cdots, z_{17}$ can be obtained with the knowledge of t_{29}, t_{33}, t_{37}, t_{40}, t_{41}, t_{42}, t_{43}, t_{44}, t_{45} and $x_7, y_0, \cdots, y_{17}$, where $y_0, \cdots, y_{17}$ are the linear combination of $x_0, x_1, \cdots, x_7$. Besides, t_{41}, t_{42}, t_{43}, t_{44}, t_{45} can be obtained by the linear combination of $t_{29}, t_{33}, t_{37}, t_{40}$. In other words, we can obtain $z_0, \cdots, z_{17}$ only with the knowledge of $t_{29}, t_{33}, t_{37}, t_{40}$ and $x_0, x_1, \cdots, x_7$.

Observation 2. The $s_0, s_1, \cdots, s_7$ can be obtained by a linear combination of $z_0, \cdots, z_{17}$ as follows, where $\bar{s}$ applies the NOT operation on s.

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

The above two observations explore the linear relationship between different parameters in the algebraic structure of AES's S-box. According to Observation 1, we can obtain $z_0, \cdots, z_{17}$ with the knowledge of $t_{29}, t_{33}, t_{37}, t_{40}$ and $x_0, x_1, \cdots, x_7$. Obviously, we can obtain $t_{29}, t_{33}, t_{37}, t_{40}$ by storing all t_i (for $2 \leq i \leq 40$), which requires 39 ancilla qubits (see in Section 3.2). Algorithm 3 can output $t_{29}, t_{33}, t_{37}, t_{40}$ with 6 ancilla qubits by reusing some ancilla qubits.

As shown in our Algorithm 3 can be constructed with 6 ancilla qubit, 17 Tofoli gates, and 93 CNOT gates, while our previous Algorithm 3 required 6 ancilla qubits, 21 Toffoli gates, and 109 CNOT gates to calculate the same values. There are several t_i can be computed in parallel as follows. First, we can compute t_7 and t_9 in parallel. Second, we can compute t_2 and t_{18} in parallel. Third, t_{29} and t_{37} can also computed in parallel. To sum up, the Toffoli depth of Algorithm 3 is 14.

Since Algorithm 3 need to recompute t_{36} and t_2, we can obtain a new depth-qubit tradeoff of Algorithm 3 as follows. First, we observe that our new Algorithm 3 shall compute t_{36} three times. If we introduce a new ancilla qubit to store t_{36}, we do not need to recompute t_{36}. That is, we can save two Toffoli gates and two Toffoli depth by storing t_{36} in a new ancilla qubit. Second, our new Algorithm 3 need to compute t_2 twice. If we introduce a new ancilla qubit to store t_2, we can save one Toffoli gates and one Toffoli depth. That is, we can obtain a new depth-qubit tradeoff i of our new Algorithm 3 with $14 - i$ Toffoli depth, $6 + i$ ancilla qubits, $17 - (i + 1)$ Toffoli gates, and $93 + (i + 1)$ CNOT gates (for $1 \leq i \leq 2$).

Algorithm 3 Output t_{29}, t_{33}, t_{37}, t_{40} of S-box with 6 ancilla qubits

```
    Input, $U[i] = x[i]$; (for $0 \leq i \leq 7$);
    Input $T[j] = 0$; (for $0 \leq j \leq 5$);
 1: $U[0] = U[0] \oplus U[6]$;
 2: $U[6] = U[6] \oplus U[2] \oplus U[4] \oplus U[5]$;
 3: $T[0] = (U[0] \cdot U[6]) \oplus T[0]$;
 4: $T[1] = T[1] \oplus T[0]$;
 5: $U[1] = U[1] \oplus U[2] \oplus U[7]$;
 6: $U[2] = U[2] \oplus U[1] \oplus U[4] \oplus U[5] \oplus U[6]$;
 7: $T[1] = (U[1] \cdot U[2]) \oplus T[1]$;
 8: $U[0] = U[0] \oplus U[1] \oplus U[2] \oplus U[3]$;
 9: $U[6] = U[6] \oplus U[1] \oplus U[7]$;
10: $T[2] = (U[0] \cdot U[6]) \oplus T[2]$;
11: $T[3] = T[3] \oplus T[2]$;
12: $U[5] = U[5] \oplus U[3]$;
13: $U[0] = U[0] \oplus U[2] \oplus U[4] \oplus U[6] \oplus U[7]$;
14: $T[3] = (U[5] \cdot U[0]) \oplus T[3]$;
15: $T[1] = T[1] \oplus T[3]$;
16: $U[0] = U[0] \oplus U[3] \oplus U[4]$;
17: $T[1] = T[1] \oplus U[0]$;
18: $U[0] = U[0] \oplus U[1] \oplus U[2] \oplus U[6] \oplus U[7]$;
```

```
19: $U[6] = U[6] \oplus U[7]$;
20: $T[0] = (U[0] \cdot U[6]) \oplus T[0]$;
21: $U[0] = U[0] \oplus U[2] \oplus U[5]$;
22: $U[5] = U[5] \oplus U[0] \oplus U[3] \oplus U[4]$;
23: $T[4] = (U[0] \cdot U[5]) \oplus T[4]$;
24: $T[3] = T[3] \oplus T[4]$;
25: $U[0] = U[0] \oplus U[1] \oplus U[3]$;
26: $U[5] = U[5] \oplus U[7]$;
27: $T[3] = (U[0] \cdot U[5]) \oplus T[3]$;
28: $U[1] = U[1] \oplus U[2] \oplus U[4] \oplus U[5] \oplus U[6]$;
29: $T[3] = T[3] \oplus U[1]$;
30: $U[1] = U[1] \oplus U[2] \oplus U[4] \oplus U[6]$;
31: $U[0] = U[0] \oplus U[2]$;
32: $T[2] = (U[1] \cdot U[0]) \oplus T[2]$;
33: $T[0] = T[0] \oplus T[2]$;
34: $T[2] = T[2] \oplus T[4]$;
35: $U[0] = U[0] \oplus U[1] \oplus U[2] \oplus U[3] \oplus U[5]$;
36: $U[5] = U[5] \oplus U[7]$;
37: $T[4] = (U[0] \cdot U[5]) \oplus T[4]$;
38: $U[2] = U[2] \oplus U[3] \oplus U[4] \oplus U[5] \oplus U[6]$;
39: $T[0] = T[0] \oplus U[2]$;
40: $U[1] = U[1] \oplus U[3] \oplus U[5] \oplus U[7]$;
41: $T[2] = (U[1] \cdot U[7]) \oplus T[2]$;
42: $U[0] = U[0] \oplus U[2] \oplus U[4] \oplus U[6] \oplus U[7]$;
43: $T[2] = T[2] \oplus U[0]$;
44: $T[4] = (T[1] \cdot T[3]) \oplus T[4]$;
45: $T[1] = T[1] \oplus T[0]$;
46: $T[4] = T[4] \oplus T[2]$;
47: $T[5] = (T[1] \cdot T[4]) \oplus T[5]$;
48: $T[5] = T[5] \oplus T[0]$;
49: $T[1] = T[1] \oplus T[0] \oplus T[5]$;
50: $T[4] = T[4] \oplus T[2] \oplus T[0]$;
51: $T[5] = T[5] \oplus T[4]$;
52: $T[1] = (T[0] \cdot T[5]) \oplus T[1]$;
53: $T[3] = T[3] \oplus T[2]$;
54: $T[2] = (T[3] \cdot T[4]) \oplus T[2]$;
55: $T[4] = (T[0] \cdot T[5]) \oplus T[4]$;
56: $T[3] = (T[2] \cdot T[4]) \oplus T[3]$;
57: $T[4] = (T[0] \cdot T[5]) \oplus T[4]$;
58: $T[5] = T[5] \oplus T[4]$;
59: $U[3] = U[3] \oplus U[2] \oplus U[4] \oplus U[5] \oplus U[6]$;
60: $U[4] = U[4] \oplus U[0] \oplus U[3] \oplus U[7]$;
61: $U[2] = U[2] \oplus U[6]$;
62: $U[5] = U[5] \oplus U[7]$;
63: $U[6] = U[6] \oplus U[0] \oplus U[1] \oplus U[3] \oplus U[4] \oplus U[5]$;
64: Output $U[0] = y_19$, $U[1] = y_4$, $U[2] = y_2$, $U[3] = y_5$, $U[4] = y_14$, $U[5] = y_6$, $U[6] = y_21$, $U[7] = x_7$ and $T[0] = t_24$, $T[1] = t_37$, $T[2] = t_29$, $T[3] = t_40$, $T[4] = t_27$, $T[5] = t_33$.
```

Note that Langenberg *et al.* in [18] also utilized the linear relationship between z_i and s_j (for $0 \le i \le 17$ and $0 \le j \le 7$) to reduce the number of Toffoli gates. However, they did not explore the whole linear relationship like Observation 2. As a result, they needed to introduce a new ancilla qubit Z in their work. According to Observation 2, we can construct Algorithm 4 for AES's S-box with the output of Algorithm 3.

Algorithm 4 Compute AES's S-box, when the output qubits are zero.

```
    Input $T[0] = t_{29}$, $T[1] = t_{37}$, $T[2] = t_{40}$, $T[3] = t_{33}$, $T[4] = t_{24}$, $T[5] = t_{27}$;
    Input $U[0] = y_5$, $U[1] = y_{19}$, $U[2] = y_{14}$, $U[3] = y_2$, $U[4] = y_6$, $U[5] = y_{21}$, $U[6] = y_4$, $U[7] = x_7$;
1: $U[1] = U[1] \oplus U[0] \oplus U[4] \oplus U[2]$;
2: $T[3] = T[3] \oplus T[1]$;
3: $S[5] = (T[3] \cdot U[1]) \oplus S[5]$;
4: $S[6] = S[6] \oplus S[5]$;
5: $U[1] = U[1] \oplus U[0] \oplus U[4] \oplus U[2]$;
6: $T[3] = T[3] \oplus T[1]$;
7: $S[2] = (T[2] \cdot U[2]) \oplus S[2]$;
8: $S[5] = S[2] \oplus S[5]$;
9: $S[2] = S[2] \oplus S[6]$;
10: $S[4] = (T[1] \cdot U[5]) \oplus S[4]$;
11: $S[1] = S[1] \oplus S[4]$;
12: $S[3] = S[3] \oplus S[4]$;
13: $U[5] = U[5] \oplus U[7]$;
14: $T[1] = T[1] \oplus T[5]$;
15: $S[7] = (T[1] \cdot U[5]) \oplus S[7]$;
16: $S[1] = S[1] \oplus S[7]$;
17: $S[3] = S[3] \oplus S[7]$;
18: $S[4] = S[4] \oplus S[7]$;
19: $U[5] = U[5] \oplus U[7]$;
20: $T[1] = T[1] \oplus T[5]$;
21: $S[7] = (T[5] \cdot U[7]) \oplus S[7]$;
22: $S[2] = S[2] \oplus S[7]$;
23: $S[5] = S[5] \oplus S[7]$;
24: $S[6] = S[6] \oplus S[7]$;
25: $U[1] = U[1] \oplus U[3] \oplus U[0] \oplus U[4] \oplus U[5] \oplus U[6]$;

26: $S[7] = (T[2] \cdot U[1]) \oplus S[7]$;
27: $S[2] = S[2] \oplus S[7]$;
28: $S[4] = S[4] \oplus S[7]$;
29: $S[5] = S[5] \oplus S[7]$;
30: $U[1] = U[1] \oplus U[3] \oplus U[0] \oplus U[4] \oplus U[5] \oplus U[6]$;

31: $U[2] = U[2] \oplus U[3]$;
32: $T[3] = T[3] \oplus T[2]$;
33: $S[7] = (T[3] \cdot U[2]) \oplus S[7]$;
34: $S[2] = S[2] \oplus S[7]$;
35: $S[5] = S[5] \oplus S[7]$;
36: $U[2] = U[2] \oplus U[3]$;
37: $T[3] = T[3] \oplus T[2]$;
38: $S[7] = (T[3] \cdot U[3]) \oplus S[7]$;
39: $S[6] = S[6] \oplus S[7]$;
40: $U[6] = U[6] \oplus U[3] \oplus U[2]$;
41: $T[2] = T[3] \oplus T[2]$;
42: $S[0] = (T[2] \cdot U[6]) \oplus S[0]$;
43: $S[4] = S[4] \oplus S[0]$;
44: $S[6] = S[6] \oplus S[0]$;
45: $S[7] = S[7] \oplus S[0]$;
```

```
46: $U[6] = U[6] \oplus U[3] \oplus U[2]$;
47: $T[2] = T[3] \oplus T[2]$;
48: $U[1] = U[1] \oplus U[0] \oplus U[4] \oplus U[2] \oplus U[5]$;
49: $S[0] = (T[3] \cdot U[1]) \oplus S[0]$;
50: $S[1] = S[1] \oplus S[0]$;
51: $S[2] = S[2] \oplus S[0]$;
52: $S[3] = S[3] \oplus S[0]$;
53: $S[4] = S[4] \oplus S[0]$;
54: $S[5] = S[5] \oplus S[0]$;
55: $S[6] = S[6] \oplus S[0]$;
56: $U[1] = U[1] \oplus U[0] \oplus U[4] \oplus U[2] \oplus U[5]$;
57: $U[3] = U[7] \oplus U[3] \oplus U[0] \oplus U[4] \oplus U[5] \oplus U[6] \oplus U[1]$;
58: $T[5] = T[5] \oplus T[2]$;
59: $S[0] = (T[5] \cdot U[3]) \oplus S[0]$;
60: $S[2] = S[2] \oplus S[0]$;
61: $S[5] = S[5] \oplus S[0]$;
62: $S[6] = S[6] \oplus S[0]$;
63: $U[3] = U[7] \oplus U[3] \oplus U[0] \oplus U[4] \oplus U[5] \oplus U[6] \oplus U[1]$;
64: $T[5] = T[5] \oplus T[2]$;
65: $U[3] = U[7] \oplus U[3] \oplus U[2] \oplus U[5] \oplus U[6]$;
66: $T[5] = T[5] \oplus T[2] \oplus T[1] \oplus T[3]$;
67: $S[0] = (T[5] \cdot U[3]) \oplus S[0]$;
68: $S[3] = S[3] \oplus S[0]$;
69: $S[4] = S[4] \oplus S[0]$;
70: $S[5] = S[5] \oplus S[0]$;
71: $S[6] = S[6] \oplus S[0]$;
72: $U[3] = U[7] \oplus U[3] \oplus U[2] \oplus U[5] \oplus U[6]$;
73: $T[5] = T[5] \oplus T[2] \oplus T[1] \oplus T[3]$;
74: $U[2] = U[2] \oplus U[3] \oplus U[4]$;
75: $T[5] = T[5] \oplus T[1]$;
76: $S[0] = (T[5] \cdot U[2]) \oplus S[0]$;
77: $S[5] = S[5] \oplus S[0]$;
78: $U[2] = U[2] \oplus U[3] \oplus U[4]$;
79: $T[5] = T[5] \oplus T[1]$;
80: $U[1] = U[1] \oplus U[3] \oplus U[4] \oplus U[2]$;
81: $S[0] = (T[1] \cdot U[1]) \oplus S[0]$;
82: $S[2] = S[2] \oplus S[0]$;
83: $S[6] = S[6] \oplus S[0]$;
84: $S[7] = S[7] \oplus S[0]$;
85: $U[1] = U[1] \oplus U[3] \oplus U[4] \oplus U[2]$;
86: $U[1] = U[1] \oplus U[2]$;
87: $T[5] = T[5] \oplus T[2]$;
88: $S[0] = (T[5] \cdot U[1]) \oplus S[0]$;
89: $S[2] = S[2] \oplus S[0]$;
90: $U[1] = U[1] \oplus U[2]$;
91: $T[5] = T[5] \oplus T[2]$;
92: $T[5] = T[5] \oplus T[2] \oplus T[1] \oplus T[3]$;
93: $S[0] = (T[5] \cdot U[4]) \oplus S[0]$;
94: $S[1] = S[1] \oplus S[0]$;
95: $S[3] = S[3] \oplus S[0]$;
96: $S[4] = S[4] \oplus S[0]$;
97: $S[5] = S[5] \oplus S[0]$;
98: $S[6] = S[6] \oplus S[0]$;
99: $S[7] = S[7] \oplus S[0]$;
100: $T[5] = T[5] \oplus T[2] \oplus T[1] \oplus T[3]$;
101: $U[1] = U[1] \oplus U[4] \oplus U[2]$;
102: $T[3] = T[3] \oplus T[1]$;
103: $S[2] = (T[3] \cdot U[1]) \oplus S[2]$;
104: $U[1] = U[1] \oplus U[4] \oplus U[2]$;
105: $T[3] = T[3] \oplus T[1]$;
106: $S[5] = (T[5] \cdot U[1]) \oplus S[5]$;
107: Compute $\overline{S[1]}$; $\overline{S[2]}$; $\overline{S[6]}$; $\overline{S[7]}$;
108: Adopt Algorithm 3 to set $T[i] = 0$ (for $0 \le i \le 5$) and $U[j] = x_j$ (for $0 \le j \le 7$);
109: Output $S[0]$, $S[1]$, $S[2]$, $S[3]$, $S[4]$, $S[5]$, $S[6]$, $S[7]$.
```

We can obtain the time and memory cost of Algorithm 4 as follows.

1. It needs 18 Toffoli gates and 140 CNOT gates to obtain z_i for $1 \le i \le 17$.
2. Since Algorithm 4 adopt Algorithm 3 twice to clean up the ancilla qubits, we can obtain a new depth-qubit trade-off i of Algorithm 4 as follows.
   a. When $i = 0$, Algorithm 4 can compute the output of S-box with 6 ancilla qubits, 52 Toffoli gates, 326 CNOT gates, and 4 NOT gates. The Toffoli depth of Algorithm 4 in this case is $2 \times 14 + 13 = 41$.
   b. When $1 \le i \le 2$, Algorithm 4 can compute the output of S-box with $6+i$ ancilla qubits, $52 - 2(i+1)$ Toffoli gates, $326 + 2(i+1)$ CNOT gates, 4 NOT gates. The Toffoli depth of Algorithm 4 in this case is $41 - 2i$.

Next, we improve the quantum circuit sending $|x\rangle|b\rangle$ to $|x\rangle|S(x) \oplus b\rangle$. In this part, we propose a new quantum circuit of AES's S-box, which maps $|x\rangle|b\rangle|0^7\rangle$ to $|x\rangle|S(x) \oplus b\rangle|0^7\rangle$ with the output of Algorithm 3. Since the qubits encoding b are not necessarily zero, we cannot adopt Algorithm 4 directly. According to Observation 2, this problem can be solved by introducing a new ancilla qubit Z, which can be used to store each z_i. After filling Z with z_i, we just XOR Z to s_j according to linear relationship in Observation 2. Note that we shall clean up Z each time so as to store new z_i.

Since this Algorithm 5 is similar to Algorithm 4, we just give a brief description of Algorithm 5 in the following pseudo code.

Algorithm 5 Compute AES's S-box, when output qubits are not zero.

```
    Input: the output of Algorithm 3;
1:  $Z = Toffoli(t_{41}, y_{10}, Z)$;
2:  $S[2] = CNOT(S[2], Z)$;
3:  $S[5] = CNOT(S[5], Z)$;
4:  $S[6] = CNOT(S[6], Z)$;
5:  $Z = Toffoli(t_{41}, y_{10}, Z)$;
6:  $Z = Toffoli(t_{29}, y_2, Z)$;
7:  $S[2] = CNOT(S[2], Z)$;
8:  $S[5] = CNOT(S[5], Z)$;
9:  $Z = Toffoli(t_{29}, y_2, Z)$;
10: $Z = Toffoli(t_{37}, y_6, Z)$;
11: $S[1] = CNOT(S[1], Z)$;
12: $S[3] = CNOT(S[3], Z)$;
13: $S[4] = CNOT(S[4], Z)$;
14: $Z = Toffoli(t_{37}, y_6, Z)$;
15: $Z = Toffoli(t_{44}, y_{15}, Z)$;
16: $S[1] = CNOT(S[1], Z)$;
17: $S[2] = CNOT(S[2], Z)$;
18: $S[3] = CNOT(S[3], Z)$;
19: $S[5] = CNOT(S[5], Z)$;
20: $S[7] = CNOT(S[7], Z)$;
21: $Z = Toffoli(t_{44}, y_{15}, Z)$;
22: $Z = Toffoli(t_{33}, x_7, Z)$;
23: $S[2] = CNOT(S[2], Z)$;
24: $S[4] = CNOT(S[4], Z)$;
25: $S[5] = CNOT(S[5], Z)$;
26: $S[7] = CNOT(S[7], Z)$;
27: $Z = Toffoli(t_{33}, x_7, Z)$;
28: $Z = Toffoli(t_{29}, y_7, Z)$;
29: $S[4] = CNOT(S[4], Z)$;
30: $S[6] = CNOT(S[6], Z)$;
31: $S[7] = CNOT(S[7], Z)$;
32: $Z = Toffoli(t_{29}, y_7, Z)$;
33: $Z = Toffoli(t_{43}, y_{13}, Z)$;
34: $S[2] = CNOT(S[2], Z)$;
35: $S[5] = CNOT(S[5], Z)$;
36: $S[6] = CNOT(S[6], Z)$;
37: $S[7] = CNOT(S[7], Z)$;
38: $Z = Toffoli(t_{43}, y_{13}, Z)$;
39: $Z = Toffoli(t_{40}, y_5, Z)$;
40: $S[6] = CNOT(S[6], Z)$;
41: $S[7] = CNOT(S[7], Z)$;
42: $Z = Toffoli(t_{40}, y_5, Z)$;
43: $Z = Toffoli(t_{43}, y_{16}, Z)$;
44: $S[0] = CNOT(S[0], Z)$;
45: $S[3] = CNOT(S[3], Z)$;
46: $S[5] = CNOT(S[5], Z)$;
47: $S[7] = CNOT(S[7], Z)$;
48: $Z = Toffoli(t_{43}, y_{16}, Z)$;
49: $Z = Toffoli(t_{40}, y_1, Z)$;
50: $S[0] = CNOT(S[0], Z)$;
51: $S[3] = CNOT(S[3], Z)$;
52: $S[4] = CNOT(S[4], Z)$;
53: $S[5] = CNOT(S[5], Z)$;
54: $S[6] = CNOT(S[6], Z)$;
55: $Z = Toffoli(t_{40}, y_1, Z)$;
56: $Z = Toffoli(t_{42}, y_{11}, Z)$;
57: $S[0] = CNOT(S[0], Z)$;
58: $S[1] = CNOT(S[1], Z)$;
59: $S[2] = CNOT(S[2], Z)$;
60: $Z = Toffoli(t_{42}, y_{11}, Z)$;
61: $Z = Toffoli(t_{45}, y_{17}, Z)$;
62: $S[0] = CNOT(S[0], Z)$;
63: $S[1] = CNOT(S[1], Z)$;
64: $S[5] = CNOT(S[5], Z)$;
65: $S[6] = CNOT(S[6], Z)$;
66: $Z = Toffoli(t_{45}, y_{17}, Z)$;
67: $Z = Toffoli(t_{44}, y_{12}, Z)$;
68: $S[0] = CNOT(S[0], Z)$;
69: $S[1] = CNOT(S[1], Z)$;
70: $S[3] = CNOT(S[3], Z)$;
71: $S[4] = CNOT(S[4], Z)$;
72: $Z = Toffoli(t_{44}, y_{12}, Z)$;
73: $Z = Toffoli(t_{37}, y_3, Z)$;
74: $S[0] = CNOT(S[0], Z)$;
75: $S[1] = CNOT(S[1], Z)$;
76: $S[3] = CNOT(S[3], Z)$;
77: $S[4] = CNOT(S[4], Z)$;
78: $S[5] = CNOT(S[5], Z)$;
79: $Z = Toffoli(t_{37}, y_3, Z)$;
80: $Z = Toffoli(t_{42}, y_9, Z)$;
81: $S[0] = CNOT(S[0], Z)$;
82: $S[1] = CNOT(S[1], Z)$;
83: $S[2] = CNOT(S[2], Z)$;
84: $S[3] = CNOT(S[3], Z)$;
85: $S[4] = CNOT(S[4], Z)$;
86: $S[5] = CNOT(S[5], Z)$;
87: $S[6] = CNOT(S[6], Z)$;
88: $S[7] = CNOT(S[7], Z)$;
89: $Z = Toffoli(t_{42}, y_9, Z)$;
90: $Z = Toffoli(t_{45}, y_{14}, Z)$;
91: $S[0] = CNOT(S[0], Z)$;
92: $S[1] = CNOT(S[1], Z)$;
93: $S[3] = CNOT(S[3], Z)$;
94: $S[4] = CNOT(S[4], Z)$;
95: $S[5] = CNOT(S[5], Z)$;
96: $S[6] = CNOT(S[6], Z)$;
97: $S[7] = CNOT(S[7], Z)$;
98: $Z = Toffoli(t_{45}, y_{14}, Z)$;
99: $S[5] = Toffoli(t_{33}, y_4, S[5])$;
100: $\overline{S[2] = Toffoli(t_{41}, y_8, S[2])}$;
101: Compute $\overline{S[1]}$; $\overline{S[2]}$; $\overline{S[6]}$; $\overline{S[7]}$;
102: Adopt Algorithm 3 to set $T[i] = 0$ (for $0 \le i \le 5$) and $U[j] = x_j$ (for $0 \le j \le 7$);
103: Output $S[0]$, $S[1]$, $S[2]$, $S[3]$, $S[4]$, $S[5]$, $S[6]$, $S[7]$.
```

Similar to Algorithm 4, we can obtain the time and memory cost of Algorithm 5 as follows

1. Algorithm 5 calculates each z_i (for $0 \le i \le 17$) in the same order as Algorithm 4. That is, Algorithm 5 needs the same cost to compute each t_i and y_j as Algorithm 4.

2. Since z_{11} (or z_{17}) only appears in S_5 (or S_2) (see in Observation 2), we can store z_{11} (or z_{17}) in S_5 (or S_2) without affecting the other output qubits. In other words, we can compute z_{11} and z_{17} in parallel with other z_i. Because we do not need to store z_{11} and z_{17} in Z, we just need to clean up Z sixteen times so as to store new z_i. That is, Algorithm 5 needs 34 Toffoli gates to calculate each z_i (for $0 \leq i \leq 17$).
3. Algorithm 5 shall adopt Algorithm 3 twice to compute S-box and clean up these ancilla qubits.

Similar to Algorithm 4, We can obtain a new depth-qubit trade-off i of Algorithm 5 as follows.

1. When $i = 0$, Algorithm 5 can compute the output of S-box with 7 ancilla qubits, 68 Toffoli gates, 352 CNOT gates, 4 NOT gates, and 60 Toffoli depth.
2. When $1 \leq i \leq 2$, we can compute S-box with $7+i$ ancilla qubits, $68-2(i+1)$ Toffoli gates, $352 + 2(i + 1)$ CNOT gates, 4 NOT gates, and $60 - 2i$ Toffoli depth.

4.3 Improved Quantum Circuit Implementation of the S-box^{-1}

Here we propose an new quantum circuit of AES's S-box$^-1$ with 7 ancilla qubits, which maps $|x\rangle|S(x)\rangle|0^7\rangle$ to $|x \oplus S^{-1}(S(x))\rangle|S(x)\rangle|0^7\rangle = |0^8\rangle|S(x)\rangle|0^7\rangle$. We can adopt our quantum circuit of S-box^{-1} to remove some state values. We will use this property to improve the zig-zag method. Our quantum circuit of AES's S-box^{-1} benefits from the following observations, which are based on our improved classical circuit of AES's S-box^{-1}.

Observation 3. The 18-bit $z_0, \cdots, z_{17}$ for computing S-box^{-1} can be obtained with the knowledge of t_{29}, t_{33}, t_{37}, t_{40}, t_{41}, t_{42}, t_{43}, t_{44}, t_{45} and $y_0, \cdots, y_{21}$. Note that $y_0, \cdots, y_{21}$ are the linear combination of $x_0, \cdots, x_7$. Besides, t_{41}, t_{42}, t_{43}, t_{44}, t_{45} can be obtained by the linear combination of t_{29}, t_{33}, t_{37}, t_{40}. That is, we can obtain $z_0, \cdots, z_{17}$ with the knowledge of $t_{29}, t_{33}, t_{37}, t_{40}$ and $x_0, \cdots, x_7$.

Observation 4. The 8-bit output of S-box^{-1} $s_0, \cdots, s_7$ can be seen as a linear combination of the 18-bit $z_0, \cdots, z_{17}$ as follows.

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

According to Observation 3, we can obtain $z_0, \cdots, z_{17}$ by $t_{29}, t_{33}, t_{37}, t_{40}$ and $x_0, \cdots, x_7$. We propose Algorithm 6 to compute the $t_{29}, t_{33}, t_{37}, t_{40}$. As shown in Algorithm 6, we can compute the $z_0, \cdots, z_{17}$ of the S-box^{-1} with 6 ancilla qubits, 17 Toffoli gates, 110 CNOT gates and 12 NOT gates.

Algorithm 6 Compute t_{29}, t_{33}, t_{37}, t_{40} of S-box^{-1} with 6 ancilla qubits

```
    Input $U[0] = x_7$, $U[1] = x_6$, $U[2] = x_5$, $U[3] = x_4$, $U[4] = x_3$, $U[5] = x_2$, $U[6] = x_1$ $U[7] = x_0$;
    Input $T[j] = 0$; (for $0 \le j \le 5$);
 1: $U[7] = \overline{U[7] \oplus U[3]}$;
 2: $U[6] = U[6] \oplus U[5] \oplus U[4] \oplus U[3] \oplus U[1] \oplus U[0]$;
 3: $T[0] = (U[6] \cdot U[7]) \oplus T[0]$;
 4: $T[2] = T[2] \oplus T[0]$;
 5: $U[5] = \overline{U[5] \oplus U[3] \oplus U[2]}$;
 6: $U[7] = U[7] \oplus U[6] \oplus U[5] \oplus U[4] \oplus U[3] \oplus U[2] \oplus U[1]$;
 7: $T[0] = (U[5] \cdot U[7]) \oplus T[0]$;
 8: $T[1] = T[1] \oplus T[0]$;
 9: $U[7] = \overline{U[7] \oplus U[2] \oplus U[0]}$;
10: $U[6] = U[6] \oplus U[5] \oplus U[4] \oplus U[3] \oplus U[2] \oplus U[0]$;
11: $U[6] = \overline{U[6]}$;
12: $T[2] = (U[6] \cdot U[7]) \oplus T[2]$;
13: $T[3] = T[3] \oplus T[2]$;
14: $U[5] = U[5] \oplus U[6] \oplus U[2] \oplus U[1]$;
15: $U[7] = U[7] \oplus U[6] \oplus U[5] \oplus U[4] \oplus U[3] \oplus U[2] \oplus U[1]$;
16: $T[0] = (U[5] \cdot U[7]) \oplus T[0]$;
17: $T[2] = (U[5] \cdot U[7]) \oplus T[2]$;
18: $U[7] = U[7] \oplus U[3] \oplus U[4]$;
19: $U[5] = \overline{U[5] \oplus U[7] \oplus U[1] \oplus U[0]}$;
20: $T[1] = (U[5] \cdot U[7]) \oplus T[1]$;
21: $T[3] = (U[5] \cdot U[7]) \oplus T[3]$;
22: $U[5] = U[5] \oplus U[6] \oplus U[3] \oplus U[0]$;
23: $U[7] = U[7] \oplus U[5] \oplus U[2] \oplus U[1] \oplus U[0]$;
24: $T[0] = (U[5] \cdot U[7]) \oplus T[0]$;
25: $U[7] = \overline{U[7] \oplus U[5] \oplus U[3] \oplus U[1] \oplus U[0]}$;
26: $U[1] = \overline{U[1] \oplus U[7]}$;
27: $T[1] = (U[7] \cdot U[1]) \oplus T[1]$;
28: $U[1] = U[1] \oplus U[7] \oplus U[6] \oplus U[3]$;
29: $U[4] = \overline{U[4] \oplus U[3] \oplus U[2] \oplus U[1]}$;
30: $T[2] = (U[1] \cdot U[4]) \oplus T[2]$;
31: $U[5] = \overline{U[5] \oplus U[3] \oplus U[2] \oplus U[1]}$;
32: $U[6] = U[6] \oplus U[4]$;
33: $T[3] = (U[6] \cdot U[5]) \oplus T[3]$;
34: $U[7] = U[7] \oplus U[3] \oplus U[4] \oplus U[1] \oplus U[0]$;
35: $T[0] = T[0] \oplus U[7]$;
36: $U[5] = U[5] \oplus U[3]$;
37: $T[1] = T[1] \oplus U[5]$;
38: $U[7] = U[7] \oplus U[4] \oplus U[1]$;
39: $T[2] = T[2] \oplus U[7]$;
40: $U[6] = U[6] \oplus U[3] \oplus U[4] \oplus U[1]$;
41: $T[3] = T[3] \oplus U[6]$;
42: $T[4] = (T[1] \cdot T[3]) \oplus T[4]$;
43: $T[3] = T[3] \oplus T[2]$;
44: $T[4] = T[4] \oplus T[0]$;
45: $T[5] = T[5] \oplus T[2]$;
46: $T[5] = (T[3] \cdot T[4]) \oplus T[5]$;
47: $T[4] = T[4] \oplus T[0]$;
48: $T[4] = T[4] \oplus T[2]$;
49: $T[3] = T[3] \oplus T[2]$;
50: $T[5] = T[5] \oplus T[4]$;
51: $T[4] = (T[5] \cdot T[2]) \oplus T[4]$;
52: $T[2] = (T[1] \cdot T[3]) \oplus T[2]$;
53: $T[1] = T[0] \oplus T[1]$;
54: $T[0] = (T[1] \cdot T[2]) \oplus T[0]$;
55: $T[1] = (T[0] \cdot T[4]) \oplus T[1]$;
56: $T[4] = T[4] \oplus T[2]$;
57: $T[2] = T[2] \oplus T[5]$;
58: $T[3] = T[3] \oplus T[2]$;
59: $T[4] = T[4] \oplus T[3]$;
60: $U[7] = \overline{U[7] \oplus U[6] \oplus U[5] \oplus U[2] \oplus U[1]}$;
61: $U[5] = U[5] \oplus U[7] \oplus U[3] \oplus U[2] \oplus U[1] \oplus U[0]$;
62: $U[5] = \overline{U[5]}$;
63: $U[6] = U[6] \oplus U[3] \oplus U[4] \oplus U[1]$;
64: $U[0] = U[0] \oplus U[5] \oplus U[3] \oplus U[7]$;
65: $U[3] = U[6] \oplus U[5] \oplus U[3] \oplus U[4] \oplus U[1]$;
66: $U[2] = \overline{U[6] \oplus U[5] \oplus U[3] \oplus U[4] \oplus U[2] \oplus U[0]}$;
67: Output $U[0] = y_7$, $U[1] = y_5$, $U[2] = y_4$, $U[3] = y_6$, $U[4] = y_1$, $U[5] = y_2$, $U[6] = y_0$, $U[7] = y_3$; and $T[0] = t_{29}$, $T[1] = t_{40}$, $T[2] = t_{33}$, $T[3] = t_{34}$, $T[4] = t_{37}$, $T[5] = t_{35}$.
```

As shown in the above, we can obtain the 14 outputs of Algorithm 6 with 7 ancilla qubits, 17 Toffoli gates, 110 CNOT gates and 12 NOT gates. The Toffoli depth of Algorithm 6 is 14, because we can compute some t_i in parallel as follows. First, we can compute the two t_6 in parallel. Second, we can compute the two t_7 in parallel. Third, we can compute the t_8 and t_{10} in parallel.

Similar to Algorithm 3, we can obtain a new depth-qubit trade-off of Algorithm 6 by introducing more ancilla qubits. Note that Algorithm 6 need to compute t_6, t_7, t_{26} twice. If we introduce 3 more ancilla qubits to store these values, we do not need to recompute t_6, t_7, t_{26} again. That is, we can obtain a new depth-qubit trade-off of Algorithm 6, which needs $7 + i$ ancilla qubits, $17 - i$

Toffoli gates, $110+i$ CNOT gates and 12 NOT gates (for $0 \leq i \leq 3$). The Toffoli depth of this new trade-off Algorithm 6 is 13 (for $1 \leq i \leq 3$).

After obtaining the 14-bit output of Algorithm 6, we can construct Algorithm 7 by using Observation 4. Since our algorithm for S-box^{-1} can not make sure the output bits are zero, we shall introduce a new ancilla qubit Z to store each z_i in this algorithm.

Algorithm 7 Compute the 8-bit output of the S-box^{-1} of AES

```
    Input $T[0] = t_{29}$, $T[1] = t_{40}$, $T[2] = t_{33}$, $T[3] = t_{34}$, $T[4] = t_{37}$, $T[5] = t_{35}$;
    Input $U[0] = y_7$, $U[1] = y_5$, $U[2] = y_4$, $U[3] = y_6$, $U[4] = y_1$, $U[5] = y_2$, $U[6] = y_0$, $U[7] = y_3$.
1: $Z = (T[1] \cdot U[2]) \oplus Z$;
2: $S[0] = S[0] \oplus Z$;
3: $S[6] = S[6] \oplus Z$;
4: $Z = (T[1] \cdot U[2]) \oplus Z$;
5: $T[0] = T[0] \oplus T[1]$;
6: $U[2] = U[2] \oplus U[1]$;
7: $Z = (T[0] \cdot U[2]) \oplus Z$;
8: $S[1] = S[1] \oplus Z$;
9: $S[3] = S[3] \oplus Z$;
10: $S[6] = S[6] \oplus Z$;
11: $Z = (T[0] \cdot U[2]) \oplus Z$;
12: $T[0] = T[0] \oplus T[1]$;
13: $U[2] = U[2] \oplus U[1]$;
14: $Z = (T[0] \cdot U[1]) \oplus Z$;
15: $S[0] = S[0] \oplus Z$;
16: $S[1] = S[1] \oplus Z$;
17: $S[3] = S[3] \oplus Z$;
18: $Z = (T[0] \cdot U[1]) \oplus Z$;
19: $T[4] = T[4] \oplus T[1] \oplus T[0] \oplus T[2]$;
20: $U[2] = U[2] \oplus U[3] \oplus U[1] \oplus U[0]$;
21: $Z = (T[4] \cdot U[2]) \oplus Z$;
22: $S[0] = S[0] \oplus Z$;
23: $S[2] = S[2] \oplus Z$;
24: $S[3] = S[3] \oplus Z$;
25: $S[5] = S[5] \oplus Z$;
26: $S[6] = S[6] \oplus Z$;
27: $S[7] = S[7] \oplus Z$;
28: $Z = (T[4] \cdot U[2]) \oplus Z$;
29: $T[4] = T[4] \oplus T[1] \oplus T[0] \oplus T[2]$;
30: $U[2] = U[2] \oplus U[3] \oplus U[1] \oplus U[0]$;
31: $T[0] = T[0] \oplus T[2]$;
32: $U[1] = U[1] \oplus U[0]$;
33: $Z = (T[0] \cdot U[1]) \oplus Z$;
34: $S[1] = S[1] \oplus Z$;
35: $S[6] = S[6] \oplus Z$;
36: $Z = (T[0] \cdot U[1]) \oplus Z$;
37: $T[0] = T[0] \oplus T[2]$;
38: $U[1] = U[1] \oplus U[0]$;
39: $T[4] = T[4] \oplus T[1]$;
40: $U[3] = U[3] \oplus U[2]$;
41: $Z = (T[4] \cdot U[3]) \oplus Z$;
42: $S[0] = S[0] \oplus Z$;
43: $S[1] = S[1] \oplus Z$;
44: $S[2] = S[2] \oplus Z$;
45: $S[3] = S[3] \oplus Z$;
46: $S[5] = S[5] \oplus Z$;
47: $S[7] = S[7] \oplus Z$;
48: $Z = (T[4] \cdot U[3]) \oplus Z$;
49: $T[4] = T[4] \oplus T[1]$;
50: $U[3] = U[3] \oplus U[2]$;
51: $Z = (T[4] \cdot U[3]) \oplus Z$;
52: $S[2] = S[2] \oplus Z$;
53: $S[3] = S[3] \oplus Z$;
54: $S[5] = S[5] \oplus Z$;
55: $S[7] = S[7] \oplus Z$;
56: $Z = (T[4] \cdot U[3]) \oplus Z$;
57: $Z = (T[2] \cdot U[0]) \oplus Z$;
58: $S[2] = S[2] \oplus Z$;
59: $S[5] = S[5] \oplus Z$;
60: $S[7] = S[7] \oplus Z$;
61: $Z = (T[2] \cdot U[0]) \oplus Z$;
62: $Z = (T[1] \cdot U[6]) \oplus Z$;
63: $S[2] = S[2] \oplus Z$;
64: $S[3] = S[3] \oplus Z$;
65: $S[6] = S[6] \oplus Z$;
66: $Z = (T[1] \cdot U[6]) \oplus Z$;
67: $T[1] = T[1] \oplus T[0]$;
68: $U[6] = U[6] \oplus U[4]$;
69: $Z = (T[1] \cdot U[6]) \oplus Z$;
70: $S[4] = S[4] \oplus Z$;
71: $S[5] = S[5] \oplus Z$;
72: $S[6] = S[6] \oplus Z$;
73: $Z = (T[1] \cdot U[6]) \oplus Z$;
74: $T[1] = T[1] \oplus T[0]$;
75: $U[6] = U[6] \oplus U[4]$;
76: $Z = (T[0] \cdot U[4]) \oplus Z$;
77: $S[2] = S[2] \oplus Z$;
78: $S[3] = S[3] \oplus Z$;
79: $S[4] = S[4] \oplus Z$;
80: $S[5] = S[5] \oplus Z$;
81: $Z = (T[0] \cdot U[4]) \oplus Z$;
82: $T[4] = T[4] \oplus T[1] \oplus T[0] \oplus T[2]$;
83: $U[7] = U[7] \oplus U[6] \oplus U[5] \oplus U[4]$;
84: $Z = (T[4] \cdot U[7]) \oplus Z$;
85: $S[4] = S[4] \oplus Z$;
86: $S[6] = S[6] \oplus Z$;
87: $S[7] = S[7] \oplus Z$;
88: $Z = (T[4] \cdot U[7]) \oplus Z$;
89: $T[4] = T[4] \oplus T[1] \oplus T[0] \oplus T[2]$;
90: $U[7] = U[7] \oplus U[6] \oplus U[5] \oplus U[4]$;
91: $T[0] = T[0] \oplus T[2]$;
92: $U[7] = U[7] \oplus U[4]$;
93: $Z = (T[0] \cdot U[7]) \oplus Z$;
94: $S[1] = S[1] \oplus Z$;
95: $S[6] = S[6] \oplus Z$;
96: $S[7] = S[7] \oplus Z$;
97: $Z = (T[0] \cdot U[7]) \oplus Z$;
98: $T[0] = T[0] \oplus T[2]$;
99: $U[7] = U[7] \oplus U[4]$;
100: $T[1] = T[1] \oplus T[4]$;
101: $U[6] = U[6] \oplus U[5]$;
102: $Z = (T[1] \cdot U[6]) \oplus Z$;
103: $S[1] = S[1] \oplus Z$;
```

```
104: $S[4] = S[4] \oplus Z$;
105: $Z = (T[1] \cdot U[6]) \oplus Z$;
106: $T[1] = T[1] \oplus T[4]$;
107: $U[6] = U[6] \oplus U[5]$;
108: $Z = (T[4] \cdot U[5]) \oplus Z$;
109: $S[2] = S[2] \oplus Z$;
110: $S[3] = S[3] \oplus Z$;
111: $S[4] = S[4] \oplus Z$;
112: $S[7] = S[7] \oplus Z$;
113: $Z = (T[4] \cdot U[5]) \oplus Z$;
114: $T[4] = T[4] \oplus T[2]$;
115: $U[7] = U[7] \oplus U[5]$;
116: $Z = (T[4] \cdot U[7]) \oplus Z$;
117: $S[1] = S[1] \oplus Z$;
118: $S[4] = S[4] \oplus Z$;
119: $S[5] = S[5] \oplus Z$;
120: $S[7] = S[7] \oplus Z$;
121: $Z = (T[4] \cdot U[7]) \oplus Z$;
122: $T[4] = T[4] \oplus T[2]$;
123: $U[7] = U[7] \oplus U[5]$;
124: $Z = (T[2] \cdot U[7]) \oplus Z$;
125: $S[1] = S[1] \oplus Z$;
126: $S[2] = S[2] \oplus Z$;
127: $S[3] = S[3] \oplus Z$;
128: $S[5] = S[5] \oplus Z$;
129: $Z = (T[2] \cdot U[7]) \oplus Z$;
130: $T[4] = T[4] \oplus T[2]$;
131: $U[0] = U[0] \oplus U[3]$;
132: $S[3] = (T[4] \cdot U[0]) \oplus S[3]$;
133: $T[4] = T[4] \oplus T[2]$;
134: $U[0] = U[0] \oplus U[3]$;
135: Adopt the Algorithm 6 to set $T[i] = 0$ (for $0 \le i \le 5$) and $U[j] = x_j$ (for $0 \le j \le 7$);
136: Output $S[0]$, $S[1]$, $S[2]$, $S[3]$, $S[4]$, $S[5]$, $S[6]$, $S[7]$.
```

The time and space cost of Algorithm 7 can be computed as follows. First, Algorithm 7 needs 35 Toffoli gates and 115 CNOT gates to compute each z_i for $0 \le i \le 17$. Second, Algorithm 7 needs to adopt Algorithm 6 twice to compute S-box^{-1} and clean up the ancilla qubits, which set $T[i] = 0$ and $U[j] = x_j$ for $0 \le i \le 5$ and $0 \le j \le 7$. To sum up, Algorithm 7 can output S-box^{-1} with 7 ancilla qubits, 69 Toffoli gates, 335 CNOT and 24 NOT gates. The depth of Algorithm 7 is 62. Given more ancilla qubits, we can also propose a new depth-qubit trade-off of Algorithm 7, which needs $7 + i$ ancilla qubits, $69 - 2i$ Toffoli gates, $335 + 2i$ CNOT, and 24 NOT gates (for $0 \le i \le 3$). The Toffoli depth of the above algorithm is 60 (for $1 \le i \le 3$).

5 Our Strategies for the Zig-zag method and the Key Schedule of AES

5.1 Zig-zag method with Improved Depth-Qubit Trade-Offs

The prior quantum circuit of AES [11,3,18] adopted the zig-zag method to reduce the number of qubits. As shown in Fig. 1, the prior zig-zag method needed 512 qubits by reusing some qubits. However, they could not remove the Round 4, Round 7 and Round 9, unless the entire process was reversed. The reason for this drawback is that the prior work only considered the encryption algorithm in their zig-zag method. That is, they should know Round $i-1$ so as to remove Round i. In this subsection, we propose an improved zig-zag method (see in Fig. 2), which just needs 256 qubits. We can achieve this goal by applying our quantum circuit of S-box^{-1} in our zig-zag method.

Denote the j-th output of the 16 S-box in Round i as s_j^i (for $0 \le j \le 15$), while the j-th byte of Round $i-1$ is denoted as r_j^{i-1} (for $0 \le j \le 15$). Given $|r^{i-1}\rangle|0^{128}\rangle$, we can explain how to obtain Round i and remove Round $i-1$ within these 256 qubits.

1. Given $|r^{i-1}\rangle$, we can compute the first r bytes of $s_0^i, s_1^i, \cdots, s_{r-1}^i$ with our Algorithm 4. We can store $s_0^i, s_1^i, \cdots, s_{r-1}^i$ in the first $8 \cdot r$ qubits of $|0^{128}\rangle$,

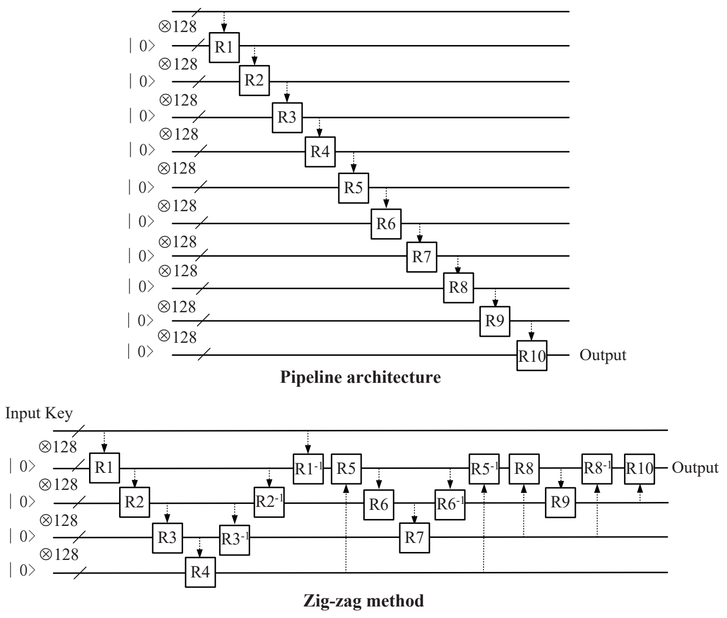

Fig. 1. Comparison between the pipeline architecture and the zig-zag method. The round i is indicated by R_i, while R_i^{-1} means to remove the round i.

while the left $|0^{128-8\cdot r}\rangle$ qubits can be used for ancilla qubits. We can choose r to obtain a improved depth-qubits trade-off for our quantum circuit.

2. After computing $s_0^i, \cdots, s_{r-1}^i$, we can remove the first r bytes in Round $i-1$ by using our Algorithm 7. That is, we can compute $|s_j^i\rangle|r_j^{i-1}\rangle|0^7\rangle \xrightarrow{Algorithm\ 7} |s_j^i\rangle|r_j^{i-1} \oplus Sbox^{-1}(s_j^i)\rangle|0^7\rangle$ for $0 \le j \le r-1$). Note that we can still use the left $|0^{128-8\cdot r}\rangle$ qubits as ancilla qubits. Since $r_j^{i-1} = Sbox^{-1}(s_j^i)$, we have $|s_j^i\rangle|r_j^{i-1} \oplus Sbox^{-1}(s_j^i)\rangle|0^7\rangle = |s_j^i\rangle|0^8\rangle|0^7\rangle$
3. These re-zero $|0^{8\cdot r}\rangle$ in Round $i-1$ can be used as ancilla qubits for obtaining $s_r^i, s_{r+1}^i, \cdots, s_{15}^i$ and removing the left $16-r$ bytes in Round $i-1$.
4. After computing the 16 bytes of $s_0^i, s_1^i, \cdots, s_{15}^i$, we can compute the 16 bytes of Round i by computing $AK \circ MC \circ SR(S_i)$, where S_i is the 16 bytes output of the S-box in Round i, and AK, MC and SR are the abbreviations for AddRoundKey, MixColumns and ShiftRows.

After generating Round i, we can compute Round $i+1$ and remove Round i in a similar way. We can assign the newly calculated Round $i+1$ to these 128 re-initialized zero qubits of Round $i-1$. We can compute the ciphertext of AES-128 by repeating the above operation 10 times. Obviously, we can construct the zig-zag method for AES-192/-256 with 256 qubits in a similar way, where the prior zig-zag method needs 656 qubits for AES-192/-256 both.

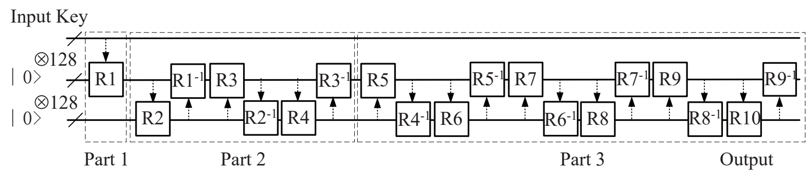

Fig. 2. Our method for improving the zig-zag method. The round i is indicated by R_i, while R_i^{-1} means to remove the round i.

5.2 Improved Quantum Circuits for the Key Schedule of AES

In this subsection, we propose some improved quantum circuit implementations for the key schedule of AES-128/-192/-256.

Our Strategy for The Key Schedule of AES-128. Our quantum circuit for the key-schedule of AES-128 only requires 128 qubits, while the prior works needed at least 224 qubits. We can achieve this improvement by combining our quantum circuit of S-box (Algorithm 5) with the property proposed by Langenberg *et al.* [18] (see Table 3).

We take W_{16} as an example to explain Table 3, where $W_{16} : W_{15}, W_{11}, W_7, W_3$. It means W_{16} can be computed with the knowledge of W_{15}, W_{11}, W_7, and W_3. According to Algorithm 1, we can rewrite W_{16} as $W_{16} = W_{15} \oplus W_{11} W_7 \oplus W_3 \oplus SubWord(RotWord(W_{15})) \oplus Rcon(4)$. We can obtain the other W_i in Table 3 similarly.

According to Table 3, we can compute all W_j (for $4 \leq i \leq 43$) with these ten 32-bit W_{4i+3} (for $1 \leq i \leq 10$). In [11], Grassl *et al.* just stored these ten 32-bit W_{4i+3} (for $1 \leq i \leq 10$) with $32 \times 10 = 320$ qubits to generate each roundkey of AES-128. In [18], Langenberg *et al.* showed that they could generate all round keys of AES-128 with 224 qubits by reusing some qubits as follows. After computing seven 32-qubit words W_7, W_{11}, W_{15}, W_{19}, W_{23}, W_{27}, W_{31}, they just cleaned up W_7 so as to assign W_{32} to these 32 re-zero qubits. Then they could compute W_{33}, W_{34} and W_{35} one by one with the knowledge of W_{19}, W_{23}, W_{27}, W_{31}. Obviously, they could compute the left round-keys similarly.

When the output qubits were not zero, Langenberg *et al.* could not apply their quantum circuit of S-box to compute AES's S-box. As a result, they should remove W_7 to generate W_{32}. Based on Algorithm 5, our improved quantum circuit for the key schedule of AES-128 can be explained as follows.

1. As shown in Section 6, we can generate four 32-qubit words W_7, W_{11}, W_{15}, W_{19} in the 128 zero qubits.
2. Since we have no zero qubits left, we shall remove W_7, W_{11}, W_{15}, W_{19} to generate new W_{4i+3} (for $5 \leq i \leq 10$). In detail, we can compute W_{20} by XORing $SubWord(RotWord(W_{19}))$, $Rcon(5)$, W_{19}, W_{11}, W_{15} to W_7. We shall adopt Algorithm 5 to compute SubWord, because the output qubits are not zero.

As a result, we can assign the newly calculated W_{20} to W_7 without introducing new qubits.
3. After generating W_{20}, we can compute W_{21}, W_{22} and W_{23} one by one with W_{11}, W_{15}, W_{19} (see Table 3). Since we only store W_{23} in the memory, we can assign the newly calculated W_{20+j} to W_{20+j-1} (for $1 \leq j \leq 3$).
4. The left round keys W_i (for $24 \leq i \leq 43$) can be generated in a similar way. After generating W_{4i-1}, W_{4i-5}, W_{4i-9}, and W_{4i-13}, we can assign the newly calculated W_{4i} to W_{4i-13} (for $5 \leq i \leq 10$) without introducing new qubits. After computing W_{4i}, we can generate W_{4i+1}, W_{4i+2}, W_{4i+3} as follows: $W_{4i+1} = W_{4i} \oplus W_{4i-1} \oplus W_{4i-9}$, $W_{4i+2} = W_{4i+1} \oplus W_{4i-1} \oplus W_{4i-5}$, $W_{4i+3} = W_{4i+2} \oplus W_{4i-1}$. We can assign the newly calculated W_{4i+j} to W_{4i+j-1} (for $1 \leq j \leq 3$).

Our Strategy for The Key Schedule of AES-192 and AES-256. Similar to AES-128, we can obtain a property for AES-192 (or AES-256) in Table 4 (or Table 5).

The quantum circuit for the key schedule of AES-192 is similar to AES-128. After generating W_{11}, W_{17}, W_{23}, W_{29}, W_{35} and W_{41} in the 192 qubits, we can compute W_{42} by xoring $SubWord(RotWord(W_{41}))$, $Rcon(7)$, W_{35}, W_{17} to W_{11}. Then we can compute the round-key W_{42+j} (for $1 \leq j \leq 5$) one by one with the knowledge of W_{42+j-1}, W_{17}, W_{23}, W_{29}, W_{35} and W_{41}. Obviously, we can compute left round keys for AES-192 in a similar way. To sum up, we can compute the 12 round-key of AES-192 with 192 qubits.

The quantum circuit for the key schedule of AES-256 can be constructed as follows. After generating the eight round-keys W_{11}, W_{15}, W_{19}, W_{23}, W_{27}, W_{31}, W_{35} and W_{39} in the quantum memory, we can compute W_{40} for AES-256 by XORing $SubWord(RotWord(W_{39}))$, $Rcon(5)$, W_{35}, W_{27}, W_{19} to W_{11}. Then we can compute the round-key W_{40+j} ($1 \leq j \leq 3$) for Round 10 one by one with the knowledge of W_{39}, W_{35}, W_{27}, W_{19}. Similar to W_{40}, we can obtain the left round key W_{44}, W_{48}, W_{52}, and W_{56} without introducing new qubits. To sum up, we can compute the 14 round-key of AES-256 with 256 qubits.

Table 3. The keys required to construct each round-key of AES-128.

$W_4 : W_3, W_0$	$W_5 : W_4, W_1$	$W_6 : W_5, W_2$	$W_7 : W_6, W_3$
$W_8 : W_7, W_3, W_2, W_1$	$W_9 : W_8, W_7, W_3, W_2$	$W_{10} : W_7, W_3$	$W_{11} : W_{10}, W_7$
$W_{12} : W_{11}, W_7, W_2$	$W_{13} : W_{12}, W_{11}, W_3$	$W_{14} : W_{13}, W_{11}, W_7$	$W_{15} : W_{14}, W_{11}$
$W_{16} : W_{15}, W_{11}, W_7, W_3$	$W_{17} : W_{16}, W_{15}, W_7$	$W_{18} : W_{17}, W_{15}, W_{11}$	$W_{19} : W_{18}, W_{15}$
$W_{20} : W_{19}, W_{15}, W_{11}, W_7$	$W_{21} : W_{20}, W_{19}, W_{11}$	$W_{22} : W_{21}, W_{19}, W_{15}$	$W_{23} : W_{22}, W_{19}$
$W_{24} : W_{23}, W_{19}, W_{15}, W_{11}$	$W_{25} : W_{24}, W_{23}, W_{15}$	$W_{26} : W_{25}, W_{23}, W_{19}$	$W_{27} : W_{26}, W_{23}$
$W_{28} : W_{27}, W_{23}, W_{19}, W_{15}$	$W_{29} : W_{28}, W_{27}, W_{19}$	$W_{30} : W_{29}, W_{27}, W_{23}$	$W_{31} : W_{30}, W_{27}$
$W_{32} : W_{31}, W_{27}, W_{23}, W_{19}$	$W_{33} : W_{32}, W_{31}, W_{23}$	$W_{34} : W_{33}, W_{31}, W_{27}$	$W_{35} : W_{34}, W_{31}$
$W_{36} : W_{35}, W_{31}, W_{27}, W_{23}$	$W_{37} : W_{36}, W_{35}, W_{27}$	$W_{38} : W_{37}, W_{35}, W_{31}$	$W_{39} : W_{38}, W_{35}$
$W_{40} : W_{39}, W_{35}, W_{31}, W_{27}$	$W_{41} : W_{40}, W_{39}, W_{31}$	$W_{42} : W_{41}, W_{39}, W_{35}$	$W_{43} : W_{42}, W_{39}$

Table 4. The keys required to construct round-key of AES-192.

$W_6 : W_5, W_0$	$W_7 : W_6, W_1$	$W_8 : W_7, W_2$
$W_9 : W_8, W_3$	$W_{10} : W_9, W_4$	$W_{11} : W_{10}, W_5$
$W_{12} : W_{11}, W_1, W_2, W_3, W_4, W_5$	$W_{13} : W_{12}, W_{11}, W_2, W_3, W_4, W_5$	$W_{14} : W_{13}, W_{11}, W_3, W_4, W_5$
$W_{15} : W_{14}, W_{11}, W_4, W_5$	$W_{16} : W_{15}, W_{11}, W_5$	$W_{17} : W_{16}, W_{11}$
$W_{18} : W_{17}, W_{11}, W_2, W_4$	$W_{19} : W_{18}, W_{17}, W_3, W_5$	$W_{20} : W_{19}, W_{17}, W_{11}, W_4$
$W_{21} : W_{20}, W_{17}, W_5$	$W_{22} : W_{21}, W_{17}, W_{11}$	$W_{23} : W_{22}, W_{17}$
$W_{24} : W_{23}, W_{17}, W_3, W_4$	$W_{25} : W_{24}, W_{23}, W_4, W_5$	$W_{26} : W_{25}, W_{23}, W_{17}, W_{11}, W_5$
$W_{27} : W_{26}, W_{23}, W_{11}$	$W_{28} : W_{27}, W_{23}, W_{17}$	$W_{29} : W_{28}, W_{23}$
$W_{30} : W_{29}, W_{23}, W_4$	$W_{31} : W_{30}, W_{29}, W_5$	$W_{32} : W_{31}, W_{29}, W_{23}, W_{17}, W_{11}$
$W_{33} : W_{32}, W_{29}, W_{17}$	$W_{34} : W_{33}, W_{29}, W_{23}$	$W_{35} : W_{34}, W_{29}$
$W_{36} : W_{35}, W_{29}, W_{11}, W_5$	$W_{37} : W_{36}, W_{35}, W_{11}$	$W_{38} : W_{37}, W_{35}, W_{29}, W_{23}, W_{17}$
$W_{39} : W_{38}, W_{35}, W_{29}$	$W_{40} : W_{39}, W_{35}, W_{29}$	$W_{41} : W_{40}, W_{35}$
$W_{42} : W_{41}, W_{35}, W_{17}, W_{11}$	$W_{43} : W_{42}, W_{41}, W_{17}$	$W_{44} : W_{43}, W_{41}, W_{35}, W_{29}, W_{23}$
$W_{45} : W_{44}, W_{41}, W_{35}$	$W_{46} : W_{45}, W_{41}, W_{35}$	$W_{47} : W_{46}, W_{41}$
$W_{48} : W_{47}, W_{41}, W_{23}, W_{17}$	$W_{49} : W_{48}, W_{47}, W_{23}$	$W_{50} : W_{49}, W_{47}, W_{41}, W_{35}, W_{29}$
$W_{51} : W_{50}, W_{47}, W_{41}$		

Table 5. The keys required to construct round-key of AES-256.

$W_8 : W_7, W_0$	$W_9 : W_8, W_1$	$W_{10} : W_9, W_2$	$W_{11} : W_{10}, W_3$
$W_{12} : W_{11}, W_4$	$W_{13} : W_{12}, W_5$	$W_{14} : W_{13}, W_6$	$W_{15} : W_{14}, W_7$
$W_{16} : W_{15}, W_{11}, W_3, W_2, W_1$	$W_{17} : W_{16}, W_{11}, W_3, W_2$	$W_{18} : W_{17}, W_{11}, W_3$	$W_{19} : W_{18}, W_{11}$
$W_{20} : W_{19}, W_{15}, W_7, W_6, W_5$	$W_{21} : W_{20}, W_{15}, W_7, W_6$	$W_{22} : W_{21}, W_{15}, W_7$	$W_{23} : W_{22}, W_{15}$
$W_{24} : W_{23}, W_{19}, W_{11}, W_2$	$W_{25} : W_{24}, W_{19}, W_3$	$W_{26} : W_{25}, W_{19}, W_{11}$	$W_{27} : W_{26}, W_{19}$
$W_{28} : W_{27}, W_{23}, W_{15}, W_6$	$W_{29} : W_{28}, W_{23}, W_7$	$W_{30} : W_{29}, W_{23}, W_{15}$	$W_{31} : W_{30}, W_{23}$
$W_{32} : W_{31}, W_{27}, W_{19}, W_{11}, W_3$	$W_{33} : W_{32}, W_{31}, W_{11}$	$W_{34} : W_{33}, W_{27}, W_{19}$	$W_{35} : W_{34}, W_{27}$
$W_{36} : W_{35}, W_{31}, W_{23}, W_{15}, W_7$	$W_{37} : W_{36}, W_{35}, W_{15}$	$W_{38} : W_{37}, W_{31}, W_{23}$	$W_{39} : W_{38}, W_{31}$
$W_{40} : W_{39}, W_{35}, W_{27}, W_{19}, W_{11}$	$W_{41} : W_{40}, W_{39}, W_{19}$	$W_{42} : W_{41}, W_{35}, W_{27}$	$W_{43} : W_{42}, W_{35}$
$W_{44} : W_{43}, W_{39}, W_{31}, W_{23}, W_{15}$	$W_{45} : W_{44}, W_{43}, W_{23}$	$W_{46} : W_{45}, W_{39}, W_{31}$	$W_{47} : W_{46}, W_{39}$
$W_{48} : W_{47}, W_{43}, W_{35}, W_{27}, W_{19}$	$W_{49} : W_{48}, W_{47}, W_{27}$	$W_{50} : W_{49}, W_{43}, W_{35}$	$W_{51} : W_{50}, W_{43}$
$W_{52} : W_{51}, W_{47}, W_{39}, W_{31}, W_{23}$	$W_{53} : W_{52}, W_{51}, W_{31}$	$W_{54} : W_{53}, W_{47}, W_{39}$	$W_{55} : W_{54}, W_{47}$
$W_{56} : W_{55}, W_{51}, W_{43}, W_{35}, W_{27}$	$W_{57} : W_{56}, W_{55}, W_{35}$	$W_{58} : W_{57}, W_{51}, W_{43}$	$W_{59} : W_{58}, W_{51}$

6 Improved Quantum Circuit Implementations of AES

6.1 Our Improved Quantum Circuit of AES-128.

As shown in Fig. 2, we can divide our quantum circuit of AES-128 into three parts. Part 1 only contains Round 1, which does not need the S-box^{-1} operation. Part 2 contains Round 2, Round 3 and Round 4. Part 3 contains the left 6 rounds, which shall use Algorithm 5 to compute the round-keys.

After denoting r_i^j and s_i^{j+1} as the i-th byte of Round j and the S-box operations in Round $j+1$ (for $0 \le j \le 9$ and $0 \le i \le 15$), the time and memory cost of each parts can be computed as follows.

The time and space cost of Part 1. We just compute Round 1 and remove Round 0 in Part 1 (see in Fig. 3).

1. We can obtain Round 0 by implementing at most 128 Pauli-X gates (or called NOT gate) on the input keys W_0, W_1, W_2, W_3.

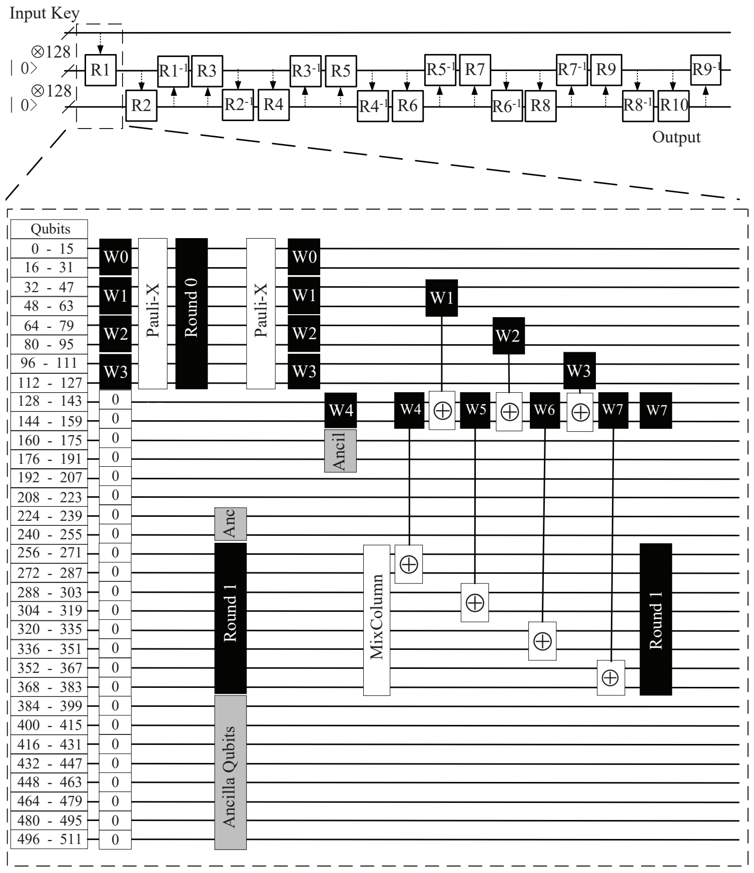

Fig. 3. Our method for computing Round 1.

2. We can adopt Algorithm 4 in parallel to compute s_i^1 (for $0 \leq i \leq 15$), because we have 384 zero qubits (from the 128 to 511 qubits in initial state in Fig. 3). Since we need 128 qubits to store these 16 bytes s_i^1 (for $0 \leq i \leq 15$), we have $384 - 128 = 256$ qubits left for ancilla qubits. In other words, we can obtain a depth-qubit trade-off $i = 2$ for these 16 S-box operations. That is, we can implement these 16 S-box operations with 128 ancilla qubits, 736 Toffoli gates and 5,312 CNOT gates. The Toffoli depth of these 16 S-box operations is $41 - 4 = 37$, because we can implement the 16 S-box in parallel.
3. After obtaining s_i^1 (for $0 \leq i \leq 15$), we can apply at most 128 NOT gates to Round 0 so as to obtain W_0, W_1, W_2, W_3 again. Then we can compute the round-key W_4, W_5, W_6, W_7 for Round 1 with the knowledge of W_0, W_1, W_2, W_3. Similar to step 2, we can obtain a depth-qubit trade-off $i = 2$ for these 4 S-box operations for W_4, because we have 224 ancilla qubits left. That is, we need 184 Toffoli gates and 1328 CNOT gates to implement these 4 S-box operations. The Toffoli depth of this operation is 37.

4. We not only require $3 \times 32 = 96$ CNOT gates and 1 NOT gate to produce W_4, W_5, W_6, W_7, but also need 128 CNOT gates to implement the AddRoundKey operation. In addition, we still need $277 \times 4 = 1108$ CNOT gates to implement 4 times MixColumns operations.

To sum up, we can implement Part 1 with 920 Toffoli gates, 7,972 CNOT gates, and 337 NOT gates. Since the 16 S-box in Round 1 and W_4 cannot be implemented in parallel, the Toffoli depth of the above operation is 74.

The time and space cost of Part 2 Part 2 contains three similar rounds from Round 2 to Round 4.

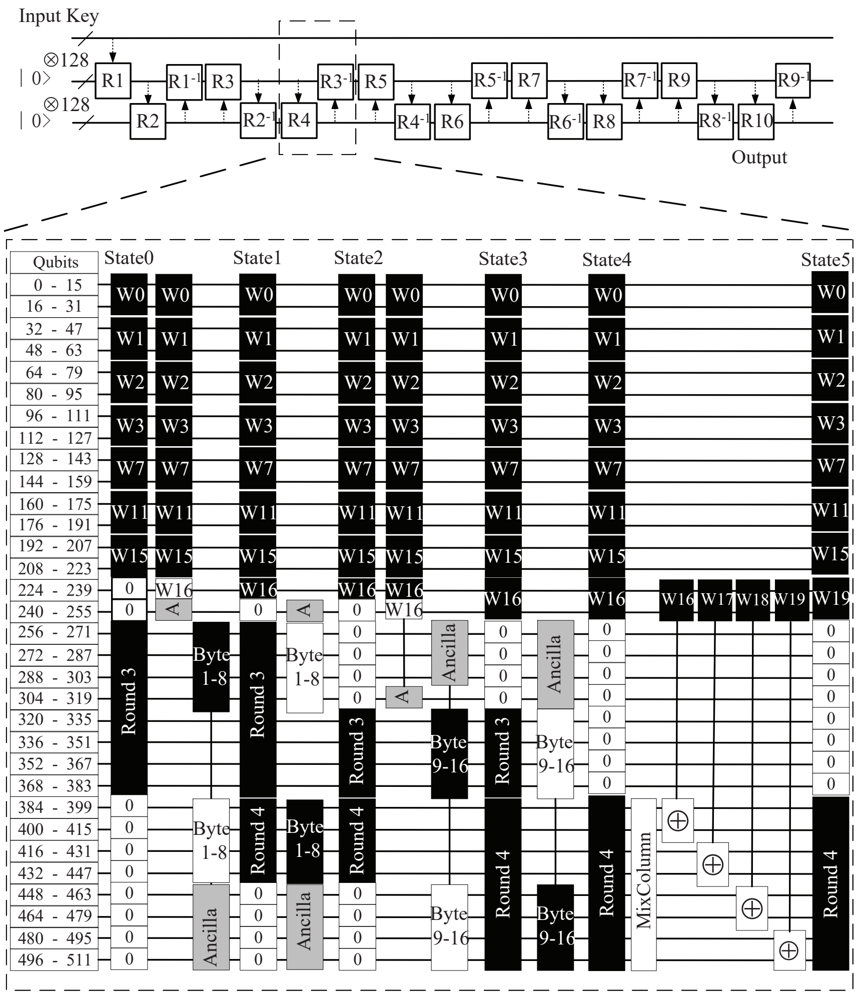

Fig. 4. Our method for computing Round 4 and removing Round 3 of AES-128.

In the following, we show the time and memory cost of computing Round 4 and removing Round 3, which can be divided into 5 phases (see in Fig. 4).

1. We can compute $s_0^4, \cdots, s_7^4$ in Round 4 and the first two bytes S-box operations of W_{16}, which requires 80 qubits to store these 10 bytes output of S-box. Since we have 160 zero qubits (the 224-255 and 384-511 qubits in state0 in Fig. 4), we have $160 - 80 = 80$ qubits left for ancilla qubits. As a result, we can obtain a depth-qubit trade-off $i = 2$ for these 10 S-box operations. That is, we can implement these 10 S-box operations with 80 ancilla qubits, 460 Toffoli gates, 3320 CNOT gates and 40 NOT gates. The Toffoli depth of these 10 S-box operations is 37.
2. We can remove $r_0^3, \cdots, r_7^3$ in Round 3 by adopting Algorithm 7. Since we have 80 zero qubits (the 240-255 and 448-511 qubits in state1 in Fig. 4), we can obtain a depth-qubit trade-off $i = 3$ for these 8 S-box^{-1} operations. That is, we can implement these 8 S-box^{-1} operations with 80 ancilla qubits, 504 Toffoli gates, 2728 CNOT gates and 192 NOT gates. The Toffoli depth of the 8 S-box^{-1} operations is 60.
3. We can compute $s_8^4 \cdots, s_{15}^4$ in Round 4 and the last two bytes of W_{16}, which requires 80 qubits to store these 10 bytes output of S-box. Since we have 144 zero qubits (the 240-319 and 448-511 qubits in state2 in Fig. 4), we have $144 - 80 = 64$ qubits left for ancilla qubits. In other words, we can obtain the depth-qubit trade-off $i = 1$ (and $i = 0$) for the first 4 S-box (the left 6 S-box) operations. That is, we can implement the first 4 S-box operations with $4 * 7$ =28 ancilla qubits, 192 Toffoli gates, 1320 CNOT gates and 16 NOT gates, while the left 6 S-box operations can be implemented with 36 ancilla qubits, 312 Toffoli gates, 1956 CNOT gates and 24 NOT gates. To sum up, we can implement these 10 S-box operations with 64 ancilla qubits, 504 Toffoli gates, 3276 CNOT gates and 40 NOT gates. The Toffoli depth of these 10 S-box operations is 41.
4. We can remove the $r_8^3, \cdots, r_{15}^3$ in Round 3 by adopting Algorithm 7. Since we have 64 zero qubits here (the 256-319 qubits in state3 in Fig. 4), we can obtain a depth-qubit trade-off $i = 1$ for these 8 S-box^{-1} operations. That is, we can implement these 8 S-box^{-1} operations with 64 ancilla qubits, 544 Toffoli gates, 2688 CNOT gates and 192 NOT gates. The Toffoli depth of the 8 S-box^{-1} operations is 61.
5. We shall implement the MixColumns and AddRoundKey operations so as to obtain Round 4. The MixColumns operation for 128-bit state requires $277 \times 4 = 1108$ CNOT operations. According to the round-key algorithm of AES-128, after the SubWord operation, we still need $32 \times 8 = 256$ CNOT gates and 1 NOT gate to compute W_{16}, W_{17}, W_{18}, W_{19}. As a result, we can implement the AddRoundKey operation with 256+128=384 CNOT gates and 1 NOT gate.

To sum up, we need 2012 Toffoli gates, 13504 CNOT gates and 465 NOT gates to obtain Round 4 and remove Round 3. The Toffoli depth of the above five steps is 199. Since the time and memory cost of the left two rounds in Part 2 is similar to the above operation, we just provide some results and ignore the details. First, we require 1928 Toffoli gates, 13556 CNOT gates and 465 NOT gates to obtain Round 3 and remove Round 2. The Toffoli depth of this transformation is 194. Second, we require 1968 Toffoli gates, 13548 CNOT gates and 465 NOT gates to obtain Round 2 and remove Round 1, while the Toffoli depth is 157.

The time and space cost of Part 3. Part 3 contains 6 similar rounds operations. In the following, we will show the time and memory cost of obtaining Round 5 and removing Round 4.

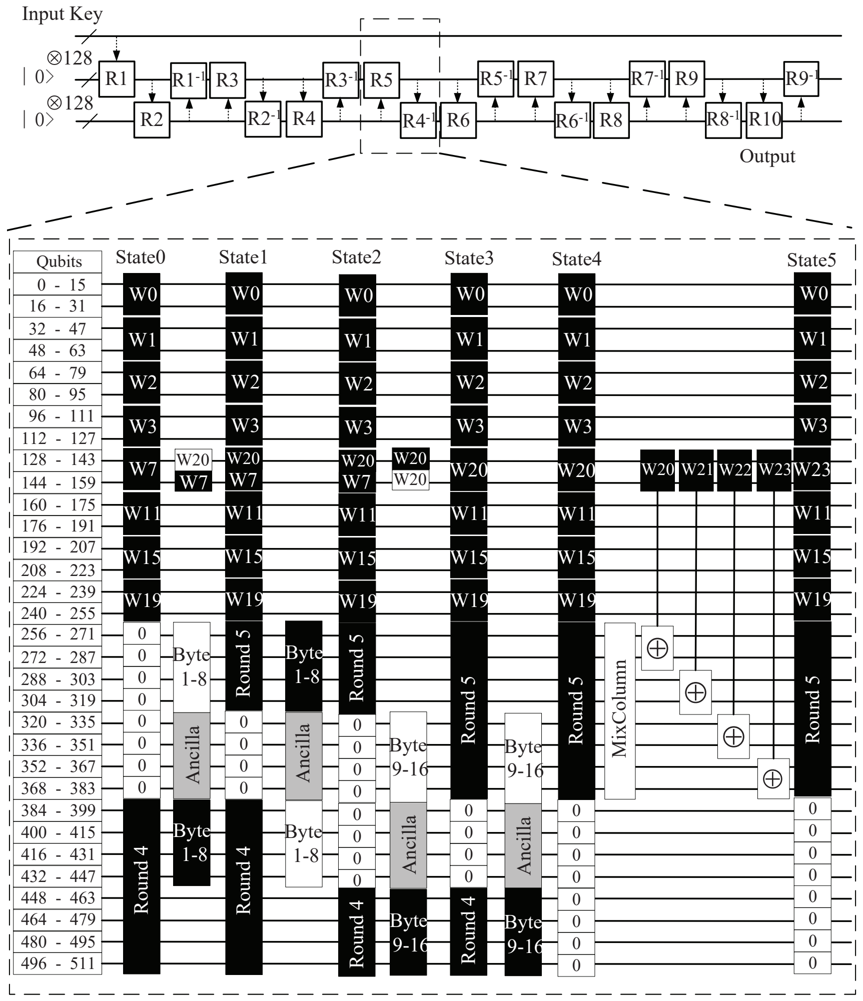

Fig. 5. Our method for computing Round 5 and removing Round 4 of AES-128.

Then we can compute the time and memory cost of the other rounds in Part 3 in a similar way. As shown in Fig. 5, we can divide the above transformation into 5 phases.

1. We can compute the $s_0^5, \cdots, s_7^5$ in Round 5 and the first two S-box operations of W_{20}. Since we have 128 zero bits (from the 256 to 383 qubits in state0 in Fig. 5), we have 128-64= 64 qubits left for ancilla qubits, because we need $|0\rangle^{\otimes 64}$ qubits to store $s_0^5, \cdots, s_7^5$. Since Algorithm 4 and Algorithm 5 require 6 and 7 ancilla qubits respectively, we need $6 \times 8 + 2 \times 7 = 62$ qubits to run Algorithm 4 eight times and Algorithm 5 twice in parallel. Then we have $64 - 48 - 14 = 2$ ancilla qubits left, which can introduce one more ancilla qubit for the first 2 S-box of W_{20}. That is, we can implement the first 2 S-box of W_{20} with 16 ancilla qubits, 128 Toffoli gates, 706 CNOT gates and 8 NOT gates, while the 8 S-box of Round 5 can be implemented with 48 ancilla qubits, 416 Toffoli gates, 2608 CNOT gates and 32 NOT gates. To sum up, we can implement these 10 S-box operations with 64 ancilla qubits, 544 Toffoli gates, 3314 CNOT gates and 40 NOT gates. The Toffoli depth of these 10 S-box operations is 56, which is determined by Algorithm 5.
2. We can remove the $r_0^4, \cdots, r_7^4$ in Round 4 by computing eight times S-box^{-1} operations with Algorithm 7. Since we have 64 qubits left for ancilla qubits (see in state1 in Fig. 5), we can obtain a depth-qubit trade-off $i = 1$ for these 8 S-box^{-1} operations. That is, we can implement these 8 S-box^{-1} operations with 64 ancilla qubits, 536 Toffoli gates, 2696 CNOT gates and 192 NOT gates. The Toffoli depth of these 8 S-box^{-1} operations is 60, because we can implement these 8 S-box^{-1} in parallel.
3. We can compute the $s_8^5, \cdots, s_{15}^5$ in Round 5 and the last two bytes of W_{20}. Similar to Step 1, we also have 2 ancilla qubits left, which can obtain a depth-qubit trade-off $i = 1$ for the last 2 S-box operations in W_{20}. Similar to step 1, we can implement these 10 S-box operations with 64 ancilla qubits, 544 Toffoli gates, 3264 CNOT gates and 40 NOT gates. The Toffoli depth of these 10 S-box operations is 56.
4. We shall remove the $r_8^4, \cdots, r_{15}^4$ of Round 4 in state3 by implementing eight times S-box^{-1} operations with Algorithm 7. Since we have 64 ancilla qubits here, we can implement these 8 S-box^{-1} operations with 64 ancilla qubits, 536 Toffoli gates, 2696 CNOT gates and 192 NOT gates. The Toffoli depth of the 8 S-box^{-1} operation is 60.
5. We shall implement the MixColumns and AddRoundKey operations so as to obtain Round 5. The 4 times MixColumns operation requires $277 \times 4 = 1108$ CNOT operations. According to the key algorithm of AES-128, after the $SubWord$ operation, we still need $32 \times 8 = 256$ CNOT gates and 1 NOT gate to compute W_{20}, W_{21}, W_{22}, W_{23}. As a result, we can implement the AddRoundKey operation with 256+128=384 CNOT gates and 1 NOT gate.

That is, we need 2160 Toffoli gates, 13512 CNOT gates, 465 NOT gates to obtain Round 5 and remove Round 4, while the Toffoli depth is 232. We can compute the time and space cost of the left 5 rounds in Part 3 in a similar way. However, different rounds of AES-128 require different cost in the AddRoundKey operation. According to the key schedule of AES-128, we need $256 \times 3 = 768$ CNOT gates and $1 \times 3 = 3$ NOT gate to generate the 3 round-keys of Round 6, Round 7 and Round 8, while the round-key of Round 9 and Round 10 require $256 \times 2 = 512$ CNOT gates and $4 \times 2 = 8$ NOT gates.

The time and memory cost of our quantum circuit of AES-128 can be obtained by summing Part 1, Part 2 and Part 3. All in all, our quantum circuit of AES-128 needs 512 qubits, 19788 Toffoli gates, 128517 CNOT gates and 4528 NOT gates. The Toffoli depth of our quantum circuit of AES-128 is 2016 (see in Table 6).

6.2 Quantum Circuit Implementations of AES-192 and AES-256

Since our quantum circuit implementation of AES-192 and AES-256 are similar to AES-128, we just show the conclusions and omit the details (see in Table 6). Our quantum circuit of AES-192 requires 640 qubits, 22380 Toffoli gates, 152378 CNOT gates and 5128 NOT gates. The Toffoli depth of our quantum circuit implementation of AES-192 is 2022. Our quantum circuit of AES-256 requires 768 qubits, 26774 Toffoli gates, 177645 CNOT gates and 6103 NOT gates. The Toffoli depth of our quantum circuit implementation of AES-256 is 2292.

7 Conclusion

In this paper, we propose some improved quantum circuit implementations of AES. In the future, there are still several research directions. First, we can explore some possible time-space trade-offs for our quantum circuit of AES by using Kim *et al.*'s work. Second, we can explore some improved quantum circuits for the other construction, such as the Feistel-SPN. Third, we can explore some improved quantum circuits of the S-box of the other block cipher, such as SM4 and Camellia.

Acknowledgments. We would like to thank anonymous referees for their helpful comments and suggestions. Jian Zou is supported by the National Natural Science Foundation of China (No.61902073). Zihao Wei and Siwei Sun are supported by the National Key Research and Development Program of China (Grant No. 2018YFA0704704), the Chinese Major Program of National Cryptography Development Foundation (Grant No. MMJJ20180102), the National Natural Science Foundation of China (61772519, 61802400), and the Youth Innovation Promotion Association of Chinese Academy of Sciences. Wenling Wu is supported by the National Natural Science Foundation of China (No.61672509).

References

1. Circuit minimization team (cmt) http://www.cs.yale.edu/homes/peralta/CircuitStuff/CMT.html.

Table 6. The quantum resource for AES-128 AES-192 and AES-256.

Algorithm	Operation	Toffoli Depth	# Toffoli	# CNOT	# NOT
AES-128	Obtain Round 1 and Remove Round 0	74	920	7972	337
	Obtain Round 2 and Remove Round 1	157	1968	13548	465
	Obtain Round 3 and Remove Round 2	194	1928	13529	465
	Obtain Round 4 and Remove Round 3	199	2012	13504	465
	Obtain Round 5 and Remove Round 4	232	2160	13512	465
	Obtain Round 6 and Remove Round 5	232	2160	13512	465
	Obtain Round 7 and Remove Round 6	232	2160	13512	465
	Obtain Round 8 and Remove Round 7	232	2160	13512	465
	Obtain Round 9 and Remove Round 8	232	2160	13512	468
	Obtain Round 10 and Remove Round 9	232	2160	12404	468
	Sum of 10 rounds	2016	19788	128517	4528
AES-192	Obtain Round 1 and Remove Round 0	74	920	7940	81
	Obtain Round 2 and Remove Round 1	97	1744	12132	448
	Obtain Round 3 and Remove Round 2	97	2080	13908	465
	Obtain Round 4 and Remove Round 3	157	1928	13620	465
	Obtain Round 5 and Remove Round 4	157	1744	12260	448
	Obtain Round 6 and Remove Round 5	157	1968	13676	465
	Obtain Round 7 and Remove Round 6	194	1928	13529	465
	Obtain Round 8 and Remove Round 7	194	1928	13573	448
	Obtain Round 9 and Remove Round 8	199	2012	13472	465
	Obtain Round 10 and Remove Round 9	232	2160	13284	465
	Obtain Round 11 and Remove Round 10	232	1808	12228	448
	Obtain Round 12 and Remove Round 11	232	2160	12756	465
	Sum of 12 rounds	2022	22380	152378	5128
AES-256	Obtain Round 1 and Remove Round 0	37	736	6568	64
	Obtain Round 2 and Remove Round 1	97	1774	12152	465
	Obtain Round 3 and Remove Round 2	97	1774	12152	464
	Obtain Round 4 and Remove Round 3	97	1774	12344	465
	Obtain Round 5 and Remove Round 4	97	2080	13684	464
	Obtain Round 6 and Remove Round 5	157	1928	13588	465
	Obtain Round 7 and Remove Round 6	157	1968	13548	464
	Obtain Round 8 and Remove Round 7	194	1928	13461	465
	Obtain Round 9 and Remove Round 8	199	2012	13536	464
	Obtain Round 10 and Remove Round 9	232	2160	13544	465
	Obtain Round 11 and Remove Round 10	232	2160	13544	464
	Obtain Round 12 and Remove Round 11	232	2160	13544	465
	Obtain Round 13 and Remove Round 12	232	2160	13544	464
	Obtain Round 14 and Remove Round 13	232	2160	12436	465
	Sum of 14 rounds	2292	26774	177645	6103

2. Aaronson, S., Gottesman, D.: Improved simulation of stabilizer circuits. CoRR **quant-ph/0406196** (2004)
3. Almazrooie, M., Samsudin, A., Abdullah, R., Mutter, K.N.: Quantum reversible circuit of AES-128. Quantum Inf. Process. **17**(5), 112 (2018)
4. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of AES. IACR Trans. Symmetric Cryptol. **2019**(2), 55–93 (2019)
5. Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In: Festa, P. (ed.) Experimental Algorithms, 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6049, pp. 178–189. Springer (2010)
6. Boyar, J., Peralta, R.: A small depth-16 circuit for the AES s-box. In: Gritzalis, D., Furnell, S., Theoharidou, M. (eds.) Information Security and Privacy Research - 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. Proceedings. IFIP Advances in Information and Communication Technology, vol. 376, pp. 287–298. Springer (2012)
7. Canright, D.: A very compact s-box for AES. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3659, pp. 441–455. Springer (2005)
8. Datta, K., Shrivastav, V., Sengupta, I., Rahaman, H.: Reversible logic implementation of AES algorithm. In: Proceedings of the 8th International Conference on Design & Technology of Integrated Systems in Nanoscale Era, DTIS 2013, 26-28 March, 2013, Abu Dhabi, UAE. pp. 140–144. IEEE (2013)
9. Dong, X., Sun, S., Shi, D., Gao, F., Wang, X., Hu, L.: Quantum collision attacks on AES-like hashing with low quantum random access memories. In: Advances in Cryptology - ASIACRYPT 2020 - the 26th Annual International Conference on the Theory and Application of Cryptology and Information Security (2020)
10. Golubitsky, O., Maslov, D.: A study of optimal 4-bit reversible toffoli circuits and their synthesis. IEEE Trans. Computers **61**(9), 1341–1353 (2012)
11. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover's algorithm to AES: quantum resource estimates. In: Takagi, T. (ed.) Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9606, pp. 29–43. Springer (2016)
12. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Miller, G.L. (ed.) 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. Hosoyamada, A., Sasaki, Y.: Finding hash collisions with quantum computers by using differential trails with smaller probability than birthday bound. In: 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. pp. 249–279 (2020)
14. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in gf($2\hat{m}$) using normal bases. Inf. Comput. **78**(3), 171–177 (1988)
15. 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. Lecture Notes in Computer Science, vol. 12106, pp. 280–310. Springer (2020)

16. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Quantum differential and linear cryptanalysis. IACR Trans. Symmetric Cryptol. **2016**(1), 71–94 (2016)
17. Kim, P., Han, D., Jeong, K.C.: Time-space complexity of quantum search algorithms in symmetric cryptanalysis: applying to AES and SHA-2. Quantum Inf. Process. **17**(12), 339 (2018)
18. Langenberg, B., Pham, H., Steinwandt, R.: Reducing the cost of implementing AES as a quantum circuit. IACR Cryptol. ePrint Arch. **2019**, 854 (2019)
19. Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: A systematic evaluation of compact hardware implementations for the rijndael s-box. In: Menezes, A. (ed.) Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3376, pp. 323–333. Springer (2005)
20. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information (10th Anniversary edition). Cambridge University Press (2016)
21. NIST: Specification for the advanced encryption standard (aes), federal information processing standards publication 197 (2001)
22. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Synthesis of reversible logic circuits. IEEE Trans. on CAD of Integrated Circuits and Systems **22**(6), 710–722 (2003)
23. Shi, Y.: Both toffoli and controlled-not need little help to do universal quantum computing. Quantum Inf. Comput. **3**(1), 84–92 (2003)
24. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997)
25. Simon, D.R.: On the power of quantum computation. SIAM J. Comput. **26**(5), 1474–1483 (1997)
26. Svore, K.M., Geller, A., Troyer, M., Azariah, J., Granade, C.E., Heim, B., Kliuchnikov, V., Mykhailova, M., Paz, A., Roetteler, M.: Q#: Enabling scalable quantum computing and development with a high-level DSL. In: Proceedings of the Real World Domain Specific Languages Workshop, RWDSL@CGO 2018, Vienna, Austria, February 24-24, 2018. pp. 7:1–7:10 (2018)
27. Toffoli, T.: Reversible computing. In: de Bakker, J.W., van Leeuwen, J. (eds.) Automata, Languages and Programming, 7th Colloquium, Noordweijkerhout, The Netherlands, July 14-18, 1980, Proceedings. Lecture Notes in Computer Science, vol. 85, pp. 632–644. Springer (1980)
28. 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**, 738 (2019)
29. Wei, Z., Sun, S., Hu, L., Wei, M., Peralta, R.: Searching the space of tower field implementations of the $\mathbb{F}_{2^8}$ inverter with applications to AES, Camellia, and SM4. International Journal of Information and Computer Security (IJICS) (2020)