

Quantum Circuits of AES with a Low-depth Linear Layer and a New Structure

Haotian Shi^{1,2} and Xiutao Feng(✉)¹

¹ Key Laboratory of Mathematics Mechanization, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China

² University of Chinese Academy of Sciences, Beijing, China
{shihaotian, fengxt}@amss.ac.cn

Abstract. In recent years quantum computing has developed rapidly. The security threat posed by quantum computing to cryptography makes it necessary to better evaluate the resource cost of attacking algorithms, some of which require quantum implementations of the attacked cryptographic building blocks. In this paper we manage to optimize quantum circuits of AES in several aspects. Firstly, based on de Brugière *et al.*'s greedy algorithm, we propose an improved depth-oriented algorithm for synthesizing low-depth CNOT circuits with no ancilla qubits. Our algorithm finds a CNOT circuit of AES MixColumns with depth 10, which breaks a recent record of depth 16. In addition, our algorithm gives low-depth CNOT circuits for many MDS matrices and matrices used in block ciphers studied in related work. Secondly, we present a new structure named compressed pipeline structure to synthesize quantum circuits of AES, which can be used for constructing quantum oracles employed in quantum attacks based on Grover's and Simon's algorithms. When the number of ancilla qubits required by the round function and its inverse is not very large, our structure will have a better trade-off of D - W cost. Moreover, our encryption oracle will have the lowest depth to date. We then give detailed encryption circuits of AES-128 under the guidance of our structure and make some comparisons with other circuits. Finally, the encryption part and the key schedule part have their own application scenarios. The Encryption oracle used in Simon's algorithm built with the former will have smaller round depth. For example, we can construct an AES-128 Encryption oracle with T -depth 33, while the previous best result is 60. A small variant of the latter, along with our method to make an Sbox input-invariant, can avoid the allocation of extra ancilla qubits for storing key words in the shallowed pipeline structure. Based on this, we achieve an encryption circuit of AES-128 with the lowest T of D - W cost 130720 to date.

Keywords: Quantum circuit · Depth · AES · Encryption oracle

1 Introduction

Quantum computers provide a great potential of solving certain important information processing tasks that are intractable for any classical computer. Shor's

algorithm [53] showed that a sufficiently large quantum computer allows to factor numbers and compute discrete logarithms in polynomial time, which represents an exponential speed-up compared to classical algorithms and can be devastating to many public-key encryption schemes in use today.

The possible emergence of large-scale quantum computing devices in the near future has brought new security threats and raised concerns about post-quantum security. Not only the public-key cryptosystem, the security of the symmetric-key cryptosystem is also under threat. A trivial application of Grover's algorithm [21] results in a quadratic speedup of the exhaustive search attack. Simon's algorithm [55] answers the question of how to find the period of a periodic function with n input bits in $O(n)$ quantum queries. As a result, many encryption structures and the most widely used modes of operation for authentication and authenticated encryption were attacked by using Simon's algorithm [29,31]. Both of these two algorithms require the quantum oracle of the symmetric building block to be attacked. Moreover, the National Institute of Standards and Technology (NIST) used the complexity of the quantum circuit for AES with a bound of depth called MAXDEPTH as a baseline to categorize the post-quantum public-key schemes into different security levels in the call for proposals to the standardization of post-quantum cryptography³. Both these reasons give rise to the growing appeals for studying the quantum implementations of symmetric-key building blocks as well as how to optimize them. This helps understand the quantum security of current encryption schemes and guides future post-quantum encryption designs.

The synthesis and optimization of quantum circuits have been studied for many years [50,44,27,5,4,57]. Given an n -qubit unitary operator and an available gate set \mathcal{G} , synthesis algorithms find one of its implementations described as a sequence of G quantum gates in \mathcal{G} with width (number of qubits) W , full depth FD and T -depth TD . The optimization of G and W is related to the saving of resources and qubits, while the optimization of FD, TD is also concerned due to the phenomenon of quantum decoherence. In addition, it is worth noting that there is a lot of work on optimizing quantum circuits on some noisy intermediate-scale quantum (NISQ) devices (see [42,58,60] for an incomplete list). In the process of quantum computation, since it has been difficult to isolate qubits for a long time, qubits would interact unintentionally with external elements, which would distort the results. Assuming that two non-overlapping gates can run in parallel, the running time of the circuit is related to its depth. Therefore, the proper execution of complex algorithms can be significantly facilitated by optimizing the depth of quantum circuits since the decoherence time is very limited. The reduction of T -depth is more important in fault-tolerant computations where the running time is dominated by T -depth [19].

Recent research on quantum implementation of symmetric ciphers mainly focuses on AES due to its popularity and importance. The main concerns are the structure, AES MixColumns and AES S-box. At the same time, there is also related work focusing on quantum implementations of other symmetric building

³ <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>

blocks [62]. One research line is to optimize the width. Grassl *et al.* proposed the first quantum circuit of AES under the so-called zig-zag structure with low width [20]. Zou *et al.* proposed the improved zig-zag structure [63], and then Huang *et al.* presented the OP-based round-in-place structure with a similar idea [22]. Jaques *et al.* first proposed the straight-line structure of key schedule process with no ancilla states and first adopted the pipeline structure [24]. Furthermore, Jang *et al.* proposed the shallowed pipeline structure to reduce the full depth [23]. Li *et al.* directly designed an in-place quantum circuit of AES S-box with low width and constructed a straight-line circuit of AES with the lowest width to date [35]. During the research on quantum circuits of AES, many researchers studied low-width quantum circuits of AES S-box.

The other research line is to reduce the circuit depth. In Clifford+ T circuits, T -depth is the main concern since Clifford gates are much cheaper than the T gate. The pipeline structure, first mentioned in reversible logic implementations of AES in [17], is straightforward to provide a low T -depth circuit of AES in many studies. To synthesize a low T -depth AES S-box, usually a low Toffoli-depth Sbox which produces some redundant states is designed. Jaques *et al.* constructed an Sbox with Toffoli-depth 6, and then gave a quantum circuit of AES-128 with T -depth 120 [24]. Li *et al.* [34] proposed an Sbox with Toffoli-depth 4. Huang *et al.* also gave an Sbox with Toffoli-depth 4, and further gave one with Toffoli-depth 3, which is the theoretical minimum. Therefore, the T -depth of quantum circuit of AES-128 is reduced to 60 [22]. The width of Sbox was further reduced by Jang *et al.*'s and Liu *et al.*'s techniques [23,38], while the saving of qubits made the optimized Sbox no longer input-invariant.

Full depth is a forward-looking time-cost measure for quantum circuits, so optimizing the depth of a CNOT circuit can reduce the full depth of the entire circuit. CNOT circuits which consist only of CNOT gates appear as subcircuits of larger circuits, such as quantum oracles of symmetric ciphers, stabilizer circuits [1], and CNOT+ T circuits. Patel *et al.* and Jiang *et al.* proposed methods to generate CNOT circuits with asymptotic optimal gate count and space-depth trade-off, respectively [41,27]. For specific matrices, Xiang *et al.*'s method [59] is based on some reduction rules for matrix decomposition, can effectively reduce the number of gates of given CNOT circuits and provides CNOT circuits with the smallest CNOT gates to date for many MDS matrices and matrices used in block ciphers. A lot of work on quantum circuits of AES (see [63,22,23,35] for an incomplete list) adopted the implementation of AES MixColumns with 92 CNOT gates provided by Xiang *et al.*'s method. Its depth estimation given by the $Q\#$ resource estimator is 30. However, their method does not take the circuit depth into account. Zhu *et al.* defined the exchange-equivalence of sequences, and proposed a framework of optimizing the depth of a given CNOT circuit [61] by exploring the possibility of exchanging CNOT gates. They started the optimization with the results of Xiang *et al.*'s method and gave a better estimation of depth 28 for AES MixColumns. Recently Liu *et al.* proposed a method for computing the depth of given quantum circuits and provided a circuit of AES MixColumns with depth 16 by computing the depth of many search

results of Xiang *et al.*'s method. Some CNOT circuits of AES MixColumns with ancilla qubits are synthesized on the basis of optimized low-depth classical circuits, and the state-of-art classical circuit of AES MixColumns with minimum depth 3 requires 99 XOR gates [51]. In addition, de Brugière *et al.* proposed a depth-oriented greedy method and a block algorithm for small and middle scale matrices, respectively [18]. However, their methods have not been tested on the linear layers of many cryptographic building blocks.

1.1 Our contributions

This paper mainly focuses on optimizing quantum circuits of AES and gives improvements in several aspects.

Improved greedy algorithm for finding low-depth CNOT circuits with no ancilla qubits. We first notice that related works of providing CNOT circuits of AES Mixcolumns either adopted non-depth-oriented search methods or determined the depth based on existing circuits. Instead, we use a depth-oriented search method. Since de Brugière *et al.* proposed a depth-oriented cost-minimization greedy algorithm that is suitable for random small scale matrices, we first apply their algorithm to AES Mixcolumns and find a circuit with depth 12, which is much better than a recent record of depth 16 in [38]. We then propose an improved greedy algorithm based on de Brugière *et al.*'s algorithm and successfully find a circuit with depth 10, which can be used to reduce the full depth of quantum circuits of AES. The improvement of our algorithm is reflected in three aspects. First, in addition to considering the logarithm of each row's Hamming weight, we also consider the *square* of each row's Hamming weight, which gives priority to rows or columns that are "far from being done" and is beneficial to reduce the circuit depth in many cases. Second, we treat two cases of row and column operations differently when evaluating the cost, that is, each column's Hamming weight is considered when column operations are performed. Finally, we give an equivalent condition of determining whether a matrix can be implemented with depth 1 to better handle sparse matrices. As applications, our improved greedy method provides low-depth CNOT circuits for many MDS matrices and matrices used in block ciphers (see Table 3, 4). Except for some matrices with depth 3, all the results are much better than those in [61]. De Brugière *et al.*'s algorithm is also applied to these matrices for comparison.

Compressed pipeline structure for quantum circuits of AES. We observe that the pipeline structure has a low depth but too many intermediate states, while the OP-based round-in-place structure has fewer intermediate states but a greater depth. To combine the advantages of the above two structures, we propose a new structure named compressed pipeline structure, which computes new states and eliminates intermediate states in parallel for qubit reuse. If the round function is taken as a unit, our structure will have lower $D-W$ cost than the above two structures when the number of ancilla qubits of a round function is small enough. To give detailed quantum circuits of AES-128, we propose iterative round functions for the encryption circuit under the guidance of our structure. Since two consecutive roundkeys are needed in the round functions, we also

present a new circuit for the key schedule of AES-128 which provides linear components of consecutive two roundkeys in one round. Both cases of NCT-based circuit and qAND-based circuit are considered. Our circuit only needs such quantum circuits of AES S-box where the output register can only be set to $|0\rangle$ initially and has lower $TD-W$ or $TofD-W$ cost when the number of ancilla qubits of AES S-box is small enough. The cost for the AES Grover oracle can be evaluated by referring to the cost of the encryption circuit.

The AES-128 Encryption oracle with lower T -depth. The encryption circuit in our structure can be used to construct the Encryption oracle employed in Simon’s algorithm with simplified cleaning of redundant states. If the round function is taken as a unit, our constructed Encryption oracle will have depth $r + 1$, which is almost half of the previous best result $2r$. When it comes to AES-128, the AES-128 Encryption oracle can be constructed with smaller T -depth. Since the redundant states of the encryption circuit can be cleaned by $|c\rangle$ with one layer of AES S-box, the AES-128 Encryption oracle can be constructed with T -depth 33, which breaks the previous record of T -depth 60 in [22].

Key schedule of the shallowed pipeline structure with input-invariant Sbox. In the key schedule of the shallowed pipeline structure, 10×32 qubits need to be allocated for storing the input register of low Toffoli-depth Sbox which cannot keep the input register unchanged. We find that adding some CNOT gates can make such Sbox input-invariant without increasing the Toffoli-depth and ancilla qubits, which ensures that the information of the input register is not lost. Based on this, we propose a new key schedule for the shallowed pipeline structure which is actually a small variant of the key schedule in the compressed pipeline structure. It can avoid the allocation of extra 10×32 qubits for storing key words and can be used to synthesize a quantum circuit of AES-128 with the lowest $TofD-W$ cost 130720 to date.

All the source codes and results of this paper are available at <https://gitee.com/Haotian-Shi/Quantum-circuits-of-aes-with-a-low-depth-linear-layer-and-a-new-structure>.

1.2 Organization

In Section 2 we introduce some background knowledge about quantum computation. In Section 3 we introduce some existing methods for optimizing the depth of CNOT circuits. Our new method and its application on some matrices are illustrated in Section 4. In Section 5 we propose our compressed pipeline structure for iterative building blocks and show its application in Encryption oracles and Grover oracles. Specific quantum implementations of AES in different scenarios and the resource costs are given in Section 6. We conclude our work in Section 7.

2 Preliminaries

2.1 Notations

We assume that the reader is familiar with AES [16]. Some notations used throughout the paper are listed as follows:

Table 1: Some notations used throughout the paper.

Notation	Definition
\mathbb{F}_2	The finite field with two elements 0 and 1
\oplus	The XOR computation
$\text{GL}(2, n)$	The set of all $n \times n$ invertible matrices over \mathbb{F}_2
$\text{GF}(n, \mathbb{F}_2)$	The finite field with 2^n elements
I_n	The n -by- n identity matrix over \mathbb{F}_2
$E(i + j)$	The resulting matrix by adding the j -th row to the i -th row of I_n (type-3 elementary matrix in $\text{GL}(2, n)$)
$E(i, j)$	The CNOT gate of adding the i -th qubit to the j -th qubit
$E(i \leftrightarrow j)$	The resulting matrix by exchanging the i -th and j -th row of I_n (type-1 elementary matrix in $\text{GL}(2, n)$), or the swapping of the i -th and j -th qubits
$ u\rangle$	A state vector u
$S(x)$	The function of AES S-box
$\mathcal{O}_{\mathcal{R}}$	The quantum oracle: $ x\rangle y\rangle \mapsto x\rangle y \oplus \mathcal{R}(x)\rangle$
$\mathcal{O}_{\mathcal{R}^{-1}}$	The quantum oracle: $ x\rangle y\rangle \mapsto x\rangle y \oplus \mathcal{R}^{-1}(x)\rangle$
k_j^l	The l -th (start from zero) 32-bit word of the j -th roundkey, or W_{4j+l} in the key schedule of AES

2.2 Quantum computation

The simplest quantum system is a single qubit state. It can be described as a unit vector $|u\rangle$ in a Hilbert Space $\mathcal{H} = \mathbb{C}^2$ and has two computational basis states $|0\rangle$ and $|1\rangle$. Then $|u\rangle = \alpha |0\rangle + \beta |1\rangle$, where $|\alpha|^2 + |\beta|^2 = 1$. An n -qubit state $|u\rangle$ can be described as a unit vector in $\mathcal{H}^{\otimes n}$, and a computational basis state can be described as a state of n -bit 0/1 string: $|x_1 x_2 \dots x_n\rangle$.

Several typical quantum gates include X, S, T , CNOT and Toffoli gates. In this paper we mainly focus on quantum circuits which compute classical vectorial boolean functions, so the input states we are concerned with are only computational basis states. The X gate, CNOT gate and Toffoli gate all act on computational basis states as shown in Figure 1:

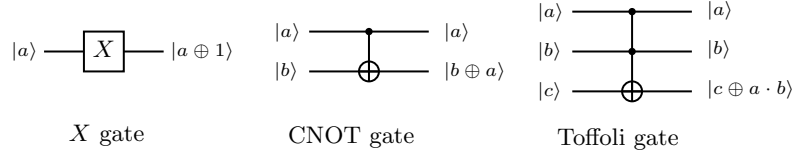


Fig. 1: Circuits of X gate, CNOT gate and Toffoli gate. The changed qubit is called the target qubit.

One can see that the roles they play in quantum computation are NOT gate, XOR gate and AND gate in classical computation, respectively.

There is another quantum gate called a quantum AND gate (qAND in short) which simulates the functionality of a classical AND gate. It differs from the Toffoli gate in that the target qubit must be $|0\rangle$. This gate together with its adjoint is illustrated in Figure 2.

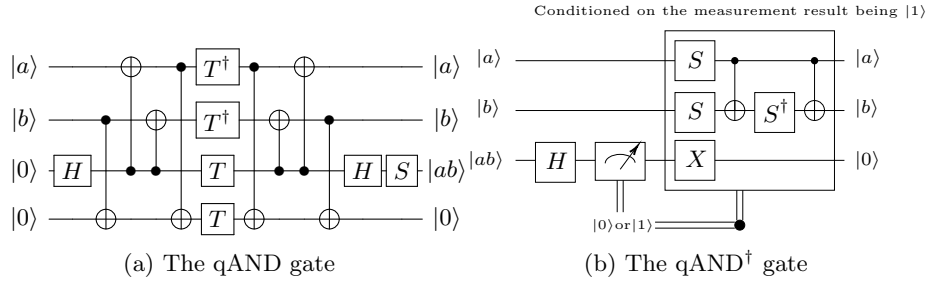


Fig. 2: The quantum AND gate together with its adjoint.

2.3 Optimization goals

Due to the limited decoherence time and qubit resources, it is crucial to reduce the time cost and storage cost in quantum circuits. In circuits containing X , CNOT and Toffoli gates, the metrics of width (W), Toffoli depth ($TofD$), and $TofD$ - W cost are crucial for evaluating the cost. In practice, the Toffoli gate can be decomposed with T -depth $3/4$ and full depth $9/8$, respectively, using 0 ancilla qubit [5], or decomposed with T -depth 1 using 4 ancilla qubits [49]. If the target qubit of a Toffoli gate is identically equal to $|0\rangle$, the Toffoli gate can be replaced by qAND with T -depth 1 using 1 ancilla qubit and its adjoint can be replaced by qAND † with T -depth 0 using 0 ancilla qubit. As is shown in [40], the Clifford gates are much cheaper than the T gate. Therefore, T -depth (TD) is a key parameter to measure the running time of a circuit. At the same time, a forward-looking perspective assumes that each gate has a unit depth and defines the full depth (FD) as a time-cost metric.

This paper mainly deals with quantum circuits of AES-128, including encryption circuits and Encryption oracles. Specifically, the encryption circuit is defined as:

$$|x\rangle |k\rangle |0\rangle \mapsto |x\rangle |k'\rangle |Enc_k(x)\rangle,$$

where $Enc_k(x)$ is the encryption of message x under the seed key k . In many cases the k' in $|k'\rangle$ is the roundkey of the last round. Note that $|x\rangle$, $|k\rangle$ can be in a superposition state. The encryption oracle differs from it in that the key register does not exist since the seed key is pre-fixed. It is defined as:

$$|x\rangle |0\rangle \mapsto |x\rangle |Enc(x)\rangle,$$

where $Enc(x)$ is the encryption of message x . In this paper, “quantum circuit” is seen as a general term for both two cases above.

3 State-of-art heuristics for optimizing the depth of CNOT circuits

In this section, we first introduce some background knowledge of CNOT circuits. We then introduce some existing methods for optimizing the depth of CNOT circuits, including methods for handling existing circuits [61,38] and the depth-oriented greedy algorithm [18].

3.1 CNOT circuits

A CNOT circuit is a quantum circuit that contains only CNOT gates. One can see from Figure 1 that a CNOT gate adds one boolean tuple to another, so for an n -qubit system, the CNOT gate $E(i, j)$ controlled by the i -th qubit and targeting on the j -th qubit can be seen as the type-3 elementary matrix $E(j+i)$. Actually the linear layer of a cipher is an n -bit reversible linear boolean function and can be interpreted as an invertible matrix A in $GL(2, n)$. Therefore, the CNOT circuit of a linear layer A can be synthesized by referring to a proper form of matrix decomposition of A . Recall that the general form of matrix decomposition is illustrated below as Theorem 1:

Theorem 1. *Any A in $GL(2, n)$ can be expressed as a product of type-1 and type-3 elementary matrices.*

Note that the only type-2 matrix in $GL(2, n)$ is the identity matrix and therefore does not appear in the matrix decomposition. It is easy to see that the type-3 elementary matrices can be adjacent by the following swapping Property 1, as shown in Theorem 2.

Property 1. $E(i+j)E(k \leftrightarrow l) = E(k \leftrightarrow l)E(f_{k,l}(i) + f_{k,l}(j))$, $E(k \leftrightarrow l)E(i+j) = E(f_{k,l}(i) + f_{k,l}(j))E(k \leftrightarrow l)$, where

$$f_{k,l}(x) = \begin{cases} k, & \text{if } x = l; \\ l, & \text{if } x = k; \\ x, & \text{else.} \end{cases} \quad (1)$$

Theorem 2. Any A in $GL(2, n)$ can be expressed as

$$A = PE(i_1 + j_1)E(i_2 + j_2) \dots E(i_L + j_L), \quad (2)$$

where P is a permutation matrix.

Since the swapping of two qubits can be realized by rewiring for free, it is easy to convert a decomposition of A in Theorem 2 to a CNOT circuit of A and vice versa. Based on this, Xiang *et al.* defined the sequential XOR (s-XOR) metric which describes the minimum gate cost of implementing A by updating input variables to output variables:

Definition 1 (s-XOR). [59] Let $M \in GL(n, \mathbb{F}_2)$ be an invertible matrix. Assume (x_1, x_2, \dots, x_n) are the n input bits of M . It is always possible to perform a sequence of XOR instructions $x_i = x_i \oplus x_j$ with $1 \leq i, j \leq n$, such that the n input bits are updated to the n output bits. The s-XOR count of M is defined as the minimum number of XOR instructions to update the inputs to the outputs.

Xiang *et al.* proposed some reduction rules to reduce the number of type-3 elementary matrices in a matrix decomposition and obtained in-place implementations of the many constructed MDS matrices and matrices used in block ciphers with minimum XOR gates up to date [59]. The specific results of their method can serve as a fairly good starting point for optimizing the quantum depth of the corresponding CNOT circuits.

3.2 Computing the depth of given circuits

Many efforts have been done on optimizing the depth of given CNOT circuits. Zhu *et al.* proposed an algorithm named One-way-opt, which involves iteratively extracts a layer of CNOT gates by exploring gates that can be exchanged forward and can run in parallel with the gates of the current layer [61]. One-way-opt is executed twice in both forward and backward directions of sequences. They compared their results with those given by the Q# resource estimator, which only explores the moving of parallelable gates. Recently, Jang *et al.* also adopted the idea of reordering gates to optimize the depth of linear layers [23]. Liu *et al.* proposed an algorithm FINDDEPTH to quickly determine the full depth of a given CNOT circuit [38]. It works by recording the updated qubits (target qubits) and used qubits (control qubits) of previous CNOT gates in each depth layer to determine the minimum depth of the current gate.

3.3 Greedy method

De Brugière *et al.* proposed a depth-oriented greedy method to find low-depth CNOT circuits of invertible linear layers [18]. It is a cost-minimization algorithm, that is, a cost function needs to be defined to evaluate the cost of reducing the matrix A to a permutation matrix, and then a strategy for exploring effective elementary transformations is designed according to the cost function.

The definition of the cost function is the only concern in the optimization of CNOT gate counts, while things are a little bit different when it comes to the optimization of the depth. Suppose a cost function to minimize has been defined to guide the search, the depth-oriented algorithm will find a series of elementary transformations of row layers and column layers to transform the target matrix A into a permutation matrix, as follows:

$$E_{i_d}^d E_{i_{d-1}}^d \cdots E_1^d \cdots E_{i_1}^1 E_{i_{1-1}}^1 \cdots E_1^1 A F_1^1 F_2^1 \cdots F_{j_1}^1 \cdots F_1^d F_2^d \cdots F_{j_d}^d = P,$$

where $E_1^t, E_2^t, \dots, E_{i_t}^t$ and $F_1^t, F_2^t, \dots, F_{j_t}^t$ for $t = 1, 2, \dots, d$ are layers of elementary row (column) operations that can act in parallel when converted to quantum CNOT gates, and P is a permutation matrix. Then one decomposition form in Theorem 2 of A is deduced as follows:

$$A = \tilde{P} \tilde{E}_1^1 \cdots \tilde{E}_{i_1-1}^1 \tilde{E}_{i_1}^1 \cdots \tilde{E}_1^d \cdots \tilde{E}_{i_d-1}^d \tilde{E}_{i_d}^d F_{j_d}^d F_{j_d-1}^d \cdots F_1^d \cdots F_{j_1}^1 F_{j_1-1}^1 \cdots F_1^1,$$

where \tilde{P}, \tilde{E}_q^p are obtained by Property 1. This decomposition corresponds to a CNOT circuit of A with depth $2d$ or $2d - 1^4$. One can see that unlike the case of the gate optimization, the depth-oriented algorithm searches row and column operations which reduce the cost function to generate a row layer and a column layer (sometimes only one row or column layer). Specifically, the authors defined two sets L_r and L_c , which record the recently applied row and column operations that can act in parallel. In each iteration of choosing an operation, only *available* row or column operations can be chosen, which are defined as follows:

Definition 2 (Available operation). *The available row (column, respectively) operations meet the following two conditions:*

- Reduce the cost function.
- Can act in parallel with recent operations in L_r (L_c , respectively).

If no available row or column operations exist, one resets L_r, L_c to empty. Each time a non-empty L_r or L_c is reset to empty, the depth count is increased by one. The algorithm ends when the cost function is equal to its minimum, that is, the current matrix is a permutation matrix, or when the depth counter exceeds a certain threshold, that is, the algorithm falls into a local minima.

Then for the definition of the cost function, the authors considered four choices to guide the optimization of the gate count in [13] and the depth in [18]:

- (1) $h_{sum}(A) = \sum_{i,j} a_{ij}$;
- (2) $H_{sum}(A) = h_{sum}(A) + h_{sum}(A^{-1})$;
- (3) $h_{prod}(A) = \sum_i \log_2(\sum_j a_{ij})$;
- (4) $H_{prod}(A) = h_{prod}(A) + h_{prod}(A^{-1})$.

⁴ i_d or j_d may equal to 0, or $\tilde{E}_1^d, \dots, \tilde{E}_{i_d-1}^d, \tilde{E}_{i_d}^d, F_{j_d}^d, F_{j_d-1}^d, \dots, F_1^d$ can act in parallel.

These four cost functions roughly estimate the cost of decomposition through the sparsity of a matrix A , and reach their minimum when A is a permutation matrix. Therefore, they can guide the search in the cost-minimization process. h_{sum} is a rough estimation since there can be too many operations which lead to the same cost, and the remaining cost functions have two major improvements over h_{sum} . On the one hand, the inverse of the current matrix is taken into account, which was first proposed in [47]. Since a row operation on A is equivalent to a corresponding column operation on A^{-1} , adding the cost of the inverse matrix can provide a more balanced estimation of the distance to a permutation matrix. On the other hand, the logarithm of every row's Hamming weight is taken into account, which was first presented in [13]. h_{prod} gives priority to "almost done" rows such that the overall efficiency of elimination is guaranteed. Note that in cost-minimization algorithms one may end up with a sparse matrix where the rows and columns have few nonzero entries with common indices. This type of matrix represents a local minima from which it might be difficult to escape. Both these two considerations can help to avoid getting stuck in a local minima and can lead to better results.

According to their experiments with random matrices, this depth-oriented greedy method behaves well for small n (roughly $n < 40$). When n is larger, it performs worse than their block algorithm and often falls into a local minima. Their block algorithm [18] and Jiang *et al.*'s algorithm [27] have asymptotic optimal bounds and can handle larger matrices better.

4 Our method and its applications

In this section, we first propose an improved greedy algorithm for finding low-depth CNOT circuits. Then we apply our algorithm to different linear building blocks with sizes of 16×16 and 32×32 that have been studied in [61]. Since the original greedy method in [18] is not applied to these matrices, we also apply their method to these matrices for comparison.

4.1 Our method

Our method is based on the framework of de Brugière *et al.*'s greedy method [18], and differs from it in three aspects. First, in addition to considering the logarithm of each row's Hamming weight, we also consider the *square* of each row's Hamming weight. Second, we treat two cases of row and column operations differently when evaluating the cost, that is, each column's Hamming weight is considered when column operations are performed. Finally, we add a judgement of whether the current matrix can be implemented with depth 1 to better handle sparse matrices.

We first make an intrinsic observation about the problem of synthesizing low-depth CNOT circuits. Though not strict, the larger Hamming weight of a row i , the more potential gates need to be done on the row i . Therefore, prioritizing the rows or columns with larger Hamming weights might be a preferable choice

to obtain lower circuit depth. Intuited by this, we propose a new cost function h_{sq} which is based on the square of every row's Hamming weight:

$$h_{sq}(A) = \sum_i \left(\sum_j a_{ij} \right)^2.$$

We also notice that focusing on the Hamming weight of the rows ignores the effect of the column operations. So we propose two cost functions H_{sqr}, H_{sqc} which are based on h_{sq} to evaluate row operations and column operations respectively:

$$\begin{aligned} H_{sqr}(A) &= h_{sq}(A) + h_{sq}((A^{-1})^T); \\ H_{sqc}(A) &= h_{sq}(A^T) + h_{sq}(A^{-1}). \end{aligned}$$

Note that the cost of the corresponding transformation of the inverse matrix is under consideration. In our method, the cost after row operations is evaluated by $H_{sqr}(A)$, and the cost after column operations is evaluated by $H_{sqc}(A)$. In addition, the cost of the current matrix is defined as the maximum evaluation of H_{sqr} and H_{sqc} to explore more possibilities.

According to our experiments, using row and column cost functions based on h_{prod} will sometimes yield better results than using row and column cost functions based on h_{sq} , which means that h_{sq} is not the best choice for all matrices. So in practice, we also use cost functions H_{prodr} and H_{prodc} defined as follows:

$$\begin{aligned} H_{prodr}(A) &= h_{prod}(A) + h_{prod}((A^{-1})^T); \\ H_{prodc}(A) &= h_{prod}(A^T) + h_{prod}(A^{-1}). \end{aligned}$$

In our algorithm, we adopt a hybrid strategy of randomly using H_{prodr}, H_{prodc} or H_{sqr}, H_{sqc} , since both cases are likely to give the best result.

In addition, we give an equivalent condition to test whether a matrix can have depth 1. This helps determine whether the last searched L_r and L_c can be implemented in parallel, since the original algorithm can only find this better case with probability $\frac{2}{2^t}$, where t is the total number of CNOT gates in the last searched L_r and L_c .

Theorem 3. *Suppose the implementation of a permutation matrix is free. Given an invertible matrix $A_{n \times n}$ on \mathbb{F}_2 , A can be implemented with depth 1 if and only if the following conditions hold:*

- (a) *The Hamming weights of all A 's rows are less than or equal to 2.*
- (b) *For any two rows i, j of A with Hamming weight 2, $a_{ik}a_{jk} = 0$ for all k .*

Proof. It is easy to see that a set of row operations on a permutation matrix can be interpreted as a set of column operations and vice versa. So we only need to consider the parallelism of row operations to reduce a matrix to a permutation matrix.

Necessity. It is easy to see that target rows of type-3 matrices have Hamming weight 2 and other rows have Hamming weight 1. (b) holds since the reduced matrix is a permutation matrix.

Sufficiency. Without loss of generality, assume that the i -th row has Hamming weight 2 for $0 \leq i < l$, and other rows have Hamming weight 1. For each i such that $a_{i,p_i} = 1, a_{i,q_i} = 1$, there exists only one corresponding row t_i with Hamming weight 1 such that either $a_{t_i,p_i} = 1$ or $a_{t_i,q_i} = 1$, since A is invertible. Applying $E(t_i, i)$ for $0 \leq i < l$ gives the implementation with depth 1. \square

Detailed steps of our improved greedy algorithm are illustrated as Algorithm 1. The running time of this algorithm is dominated by evaluating the cost of all possible row or column operations. Suppose the considered matrix A is n by n . Ignoring the limitation of parallelism, at most n^2 row operations and n^2 column operations need to be evaluated. Since A^{-1} can be computed first and updated according to the corresponding operations on A , the cost of the resulting matrix can be computed in $O(n)$ times based on the cost of A . Therefore, the complexity of determining of an operation to be done for a current matrix is upper bounded by $O(n^3)$. Similar to de Brugière *et al.*'s greedy method, our algorithm behaves well for small scale matrices (roughly $n < 40$), and our often falls into a local minima when n is larger. Our algorithm is repeated tens of thousands of time for a matrix and the best result is recorded.

4.2 Application to AES MixColumns

We first focus on CNOT circuits of AES MixColumns. Previous researchers synthesized quantum circuits of AES MixColumns with different methods. Grassal *et al.* found one circuit with depth 39 by the LUP decomposition method [20]. Zou *et al.* adopted the same circuit, while Jaque *et al.* obtained a circuit with depth 111⁵ [24]. Xiang *et al.*'s reduction framework produced a circuit with 92 CNOT gates and depth 41 [59], while Q#'s estimation of this circuit is 30, and Zhu *et al.* studied the exchange-equivalent sequence of this circuit [61] and obtained a new circuit with depth 28. Recently Liu *et al.* proposed a method for computing the depth of quantum circuits and then used it to evaluate many circuits generated by Xiang *et al.*'s method. They obtained a circuit with 98 CNOT gates and depth 16 [38]. If ancilla qubits are allowed, some CNOT circuits with low depth can be designed by converting optimized classical circuits (see [32,8,39] for an incomplete list) into quantum style. For example, Liu *et al.* converted the classical circuit in [32] with 105 XOR gates and classical depth 3 into quantum style with depth 11 and 105 ancilla qubits. Jang *et al.* optimized the circuit built upon the work in [32] and obtained an out-of-place circuit with 64 ancilla qubits and depth 8 [23].

⁵ In the Eurocrypt'20 paper [24], the authors remarked that they could not reproduce the result although they used the same technique. The authors of [23] analyzed that the reason may come from the encoding issue.

Algorithm 1 Improved greedy algorithm for synthesizing low-depth CNOT circuits.

Input: An invertible matrix $A_{n \times n}$
Output: A depth d and a vector of layers $Layers$ that implement A with length d .
 $Layers \leftarrow \emptyset, Layers_r \leftarrow \emptyset, Layers_c \leftarrow \emptyset;$
 $L_r \leftarrow \emptyset, L_c \leftarrow \emptyset;$
 $List \leftarrow \emptyset;$
 $B \leftarrow A;$
Randomly determine $H_r, H_c \leftarrow H_{sqr}, H_{sqc}$ or $H_{prodr}, H_{prod c}$.
 $cost \leftarrow \max\{H_r(B), H_c(B)\};$
 $can_one \leftarrow \mathbf{False};$ ▷ If the current matrix can be implemented with depth 1.
 $d \leftarrow 0;$
while True do
 $cost \leftarrow \max\{H_r(B), H_c(B)\};$
 $mincost \leftarrow$ the minimum resulting cost of all available row operations (can act in parallel with those in L_r) of adding the i -th row to the j -th row of B (denoted $\{i, j, 0\}$), and if not can_one , all available column operations (can act in parallel with those in L_c) adding the i -th column to the j -th column of B (denoted $\{i, j, 1\}$); ▷ If the current matrix can be implemented with depth 1, only row operations need to be considered.
 $List \leftarrow$ {All operations which can lead to the cost of the resulting matrix being $mincost$ };
if mincost == cost then
if not can_one and Can-depth-one(A) then
 $can_one \leftarrow \mathbf{True};$ ▷ If so, only row operations are considered in the next iteration.
end if
if $L_r.size()$ then
 $d \leftarrow d + 1, Layers_r.push_back(L_r), L_r.clear();$
end if
if $L_c.size()$ then
 $d \leftarrow d + 1, Layers_c.push_back(L_c), L_c.clear();$
end if
if $cost = 2n$ then
break;
end if
if $d \geq 100$ then
return $d, \emptyset;$ ▷ Too large d means the matrix may fall into a local minima.
end if
else
Randomly choose one operation $\{i, j, op\}$ that minimizes the cost function of the resulting matrix, add $\{i, j, op\}$ to L_r if $op == 0$, or L_c if $op == 1$;
 $B \leftarrow E(j+i)^{1-op} B E(i+j)^{op};$
end if
 $List.clear();$
end while
Record a permutation P , satisfying $P(i) == j$ if $B[i][j] == 1$;
for i from 0 to $(Layer_c.size() - 1)$ do
 $l \leftarrow \emptyset;$
for j from 0 to $(Layer_c[i].size() - 1)$ do
 $\{t, c, op\} \leftarrow Layer_c[i][j], l.push_back(\{c, t\});$
end for
 $Layers.push_back(l);$
end for
for i from $(Layer_r.size() - 1)$ down to 0 do
 $l \leftarrow \emptyset;$
for j from 0 to $(Layer_r[i].size() - 1)$ do
 $\{c, t, op\} \leftarrow Layer_r[i][j], l.push_back(\{P[c], P[t]\});$
end for
end for
return $d, Layers;$

We observe that related works of providing CNOT circuits of AES Mix-Columns either adopt non-depth-oriented search methods or determine the depth based on existing circuits. Instead, we use depth-oriented search algorithms to generate low-depth CNOT circuits with no ancilla qubits. We first apply the

greedy algorithm in [18] with cost function H_{prod} to AES MixColumns and obtain a circuit with depth 12 and 128 gates. Then our improved greedy algorithm finds a circuit with depth 10 and 131 gates. It can be used to reduce the full depth of quantum circuits of AES without increasing the circuit width. The comparison with previous results is shown in Table 2.

Table 2: Comparison of CNOT circuits of the AES MixColumns matrix.

Source	# CNOT	W	FD
[8,39]	206	135	13
[32]	210	137	11
[23]	169	96	8
[24]	277	32	111
[20,63]	277	32	39
[59]	92	32	30
[61]	92	32	28
[38]	98	32	16
[18]	128	32	12
This paper	131	32	10

4.3 Applications to many proposed matrices

Following the work of [61], we apply our method to various matrices in the literature including:

- some matrices used in block ciphers [16,10,52,28,6,48,9,25,7,15,2,12,11];
- some MDS matrices which are constructed in [54,45,46,26,37,33,14].

Based on the CNOT circuits of these matrices provided by Xiang *et al.*'s method, Zhu *et al.* evaluated their move-equivalent sequence depth by Q#[43], and investigated their exchange-equivalent sequences. Except for a few small-scale matrices, we can find much better results with lower depths for these matrices. For the matrices used in block ciphers, we have succeeded in reducing the circuit depth for all of them except for a few matrices that already have CNOT circuits with small depth (see Table 3). For the many constructed MDS matrices, we can optimize the depth of CNOT circuits for all of them (see Table 4). Overall, our improved greedy algorithm gives the best results with the lowest depth for all of the matrices⁶.

⁶ Note that for a few matrices, implementations with the same depth but fewer gates can be searched using de Brugière *et al.*'s algorithm. Therefore, their algorithm could be used in combination with our algorithm in order to search for better low depth CNOT circuits.

Table 3: Comparison of the depth/gate count of CNOT circuits for matrices used in block ciphers.

Cipher	Size	Q#	[61]	This paper	[18]
AES ^a [16]	32	30/92	28/92	10 /131	12/128
ANUBIS [10]	32	26/98	20/98	10 /119	14/136
CLEFIA M0 [52]	32	30/98	27/98	10 /110	13/126
CLEFIA M1 [52]	32	21/103	16/103	10 /128	13/127
FOX MU4 [28]	32	55/136	48/136	21 /265	21 /200
QARMA128 [6]	32	6/48	5/48	3 /48	3 /48
TWOFISH [48]	32	37/111	29/111	15 /175	18/187
WHIRLWIND M0 [9]	32	65/183	51/183	28 /331	28 /286
WHIRLWIND M1 [9]	32	69/190	54/190	22 /290	25/279
JOLTIK [25]	16	20/44	17/44	7 /52	9/48
MIDORI [7]	16	3 /24	3 /24	3 /24	3 /24
SmallScale AES [15]	16	20/43	19/43	10 /62	11/59
PRIDE L0 [2]	16	3 /24	3 /24	3 /24	3 /24
PRIDE L1 [2]	16	5/24	5/24	3 /24	3 /24
PRIDE L2 [2]	16	5/24	5/24	3 /24	3 /24
PRIDE L3 [2]	16	6/24	6/24	3 /24	3 /24
PRINCE M0 [12]	16	6/24	6/24	3 /24	3 /24
PRINCE M1 [12]	16	6/24	6/24	3 /24	3 /24
QARMA64 [6]	16	6/24	5/24	3 /24	3 /24
SKINNY [11]	16	3 /12	3 /12	3 /12	3 /12

^a A recent result of 16/98 is given in [38].

Table 4: Comparison of the depth/gate count of CNOT circuits for many constructed MDS matrices.

Matrices	Size	Move-eq	[61]	This paper	[18]
4 × 4 matrices in GF(4, F ₂)					
[14]	16	23/41	21/41	10 /59	12/57
[26]	16	24/41	18/41	9 /49	9 /48
[37]	16	27/41	26/41	11 /63	12/65
[54]	16	25/44	22/44	11 /59	11 /59
[33]	16	29/44	27/44	11 /62	12/65
[26](Involutory)	16	15/41	14/41	9 /54	13/54
[54](Involutory)	16	19/44	16/44	7 /52	9/48
[33](Involutory)	16	27/44	25/44	7 /52	9/48
[45](Involutory)	16	12/38	11/38	8 /46	8 /44
4 × 4 matrices in GF(8, F ₂)					
[14]	32	56/144	47/144	18 /208	20/188
[26]	32	26/82	22/82	9 /100	9 /96
[37]	32	67/121	54/121	21 /235	23/203
[33]	32	55/104	42/104	13 /164	16/167
[54]	32	23/90	20/90	10 /112	11/118
[45]	32	47/114	40/114	20 /218	20 /190
[26](Involutory)	32	18/83	14/83	9 /102	13/108
[54](Involutory)	32	18/91	16/91	8 /101	9/96
[33](Involutory)	32	19/87	19/87	8 /99	8 /98
[45](Involutory)	32	19/93	18/93	10 /121	12/119
8 × 8 matrices in GF(4, F ₂)					
[46]	32	54/183	44/183	29 /351	33/302
[54]	32	59/170	49/170	28 /349	29/286
[54](Involutory)	32	47/185	37/185	29 /337	30/300
8 × 8 matrices in GF(8, F ₂)					
[54](Involutory)	64	50/348	37/348	22 /484	25/412

5 The compressed pipeline structure for iterative primitives

In this section, we first introduce some structures used for quantum circuits of AES in previous work. Then we propose a new structure named compressed pipeline structure. Finally we make some comparisons in different levels and introduce its application in the Grover oracle and the Encryption oracle.

5.1 Existing structures

Many structures have been proposed to synthesize quantum circuits of AES. Some of them are based on out-of-place round functions, including the pipeline structure \mathcal{S}_p , the zig-zag structure \mathcal{S}_z and the out-of-place based (OP-based in short) round-in-place structure \mathcal{S}_i . These structures treat the round function and its inverse as a unit. Therefore, the metric "round depth" – the number of layers of the round function – is then used to describe the depth of a structure, and the width of a structure is closely related to the number of intermediate states and the number of parallel round functions.

The pipeline structure, which is first mentioned in reversible logic implementations of AES in [17], was proposed by Jaques *et al.* in [24]. It has low round depth and large width and is used to construct low T -depth quantum circuits of AES in [24,22,23,38]. The zig-zag structure was first put forward by Grassal *et al.* in [20] to reduce the number of intermediate states. It is used to construct low-width circuits of AES in [20,3,30]. To further reduce the number of intermediate states, Zou *et al.* presented the improved zig-zag structure [63], and Huang *et al.* proposed the OP-based round-in-place structure to construct in-place circuits on the basis of out-of-place circuits. Denote the j -th round function \mathcal{R}_j which satisfies $\mathcal{R}_j(c_{j-1}) = c_j$, then the out-of-place oracle $\mathcal{O}_{\mathcal{R}_j}$ takes $|x\rangle|y\rangle$ as input and outputs $|x\rangle|y \oplus \mathcal{R}_j(x)\rangle$. The input register and output register is distinguished by notation $\mathcal{R}_{j,i}$ and $\mathcal{R}_{j,o}$ respectively. For simplicity, the key schedule and the ancilla qubits are omitted. \mathcal{S}_p and \mathcal{S}_z compute the desired output $|c\rangle$ along with some redundant states, as illustrated in Figure 3, 4, respectively. The construction of the in-place function in \mathcal{S}_i is shown in Figure 5.

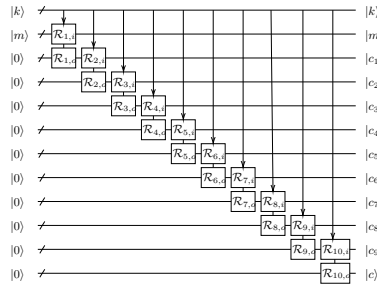


Fig. 3: The pipeline structure \mathcal{S}_p .

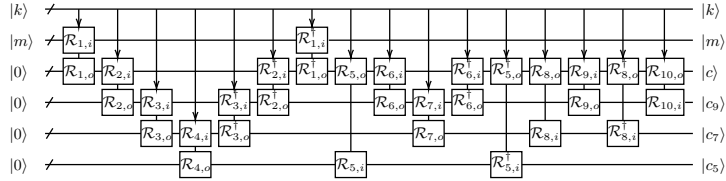


Fig. 4: The zig-zag structure \mathcal{S}_z .

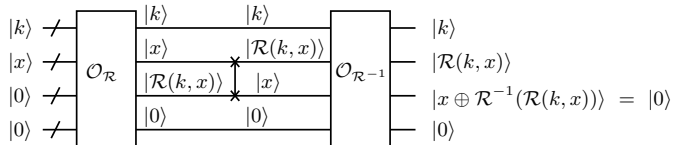


Fig. 5: OP-based round-in-place function in \mathcal{S}_i .

There are also some other structures which do not take the out-of-place round function as a unit. On the one hand, an in-place round function can be directly designed without out-of-place round functions. For example, Li *et al.* proposed an in-place quantum circuit of AES S-box with only 8 ancilla qubits, and then used it to synthesize a quantum circuit of AES under the straight-line structure, which has the lowest width to date [35]. On the other hand, some out-of-place round functions themselves can be decomposed into computation and uncomputation. The shallowed pipeline structure proposed by Jang *et al.* delays the uncomputation of one round function (if exists) to the next round in the pipeline structure to reduce the full depth [23]. Liu *et al.* improved this structure by sharing the ancilla qubits of the computation and uncomputation to save qubits [38].

5.2 Compressed pipeline structure

In this section we propose the compressed pipeline structure.

We first make some observations on existing structures. Though the pipeline structure has the lowest round depth, too many qubits store the intermediate states. At the same time, the zig-zag structure and its improvements clean some intermediate states to save qubits, but at a cost of almost twice the round depth of the pipeline structure. To combine the advantages of the above two structures, we propose a strategy of computing new states and eliminating intermediate states in parallel.

Specifically, when c_{j+1} ($j \geq 1$) is generated, the register storing c_{j-1} can be cleaned by $\mathcal{O}_{\mathcal{R}_{j-1}^{-1}}$ in parallel and then can be reallocated in further use. In fact, our structure can be thought of as adding the clean process to the

pipeline structure, hence the name compressed pipeline structure denoted by \mathcal{S}_{cp} . Since the input register of both $\mathcal{O}_{\mathcal{R}_{j-1}^{-1}}$ and $\mathcal{O}_{\mathcal{R}_j}$ is the same, $|c_j\rangle$ is copied by CNOT gates so that $\mathcal{O}_{\mathcal{R}_{j-1}^{-1}}$ and $\mathcal{O}_{\mathcal{R}_j}$ can act in parallel. Therefore, our structure requires four intermediate message registers, while a round function and its inverse function need to be executed in parallel. \mathcal{S}_{cp} is illustrated in Figure 6.

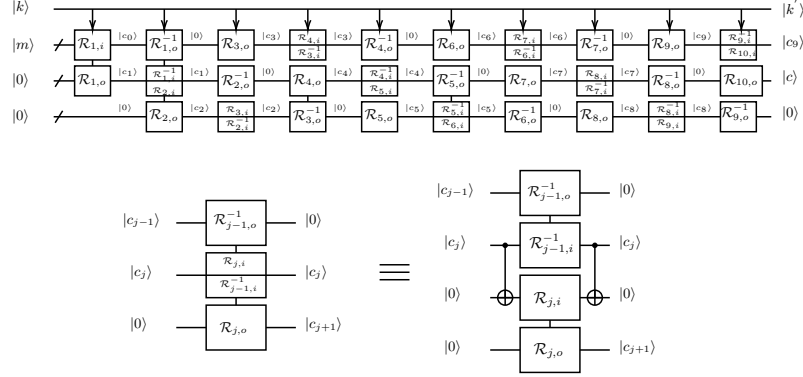


Fig. 6: The compressed pipeline structure \mathcal{S}_{cp} . For convenience, the copy of the $|c_j\rangle$ state is simplified as "split into two parts".

5.3 Comparison of different structures

In this subsection, we will compare the round depth and width of different structures that takes the round function as a unit and illustrate the use of our \mathcal{S}_{cp} in two scenarios: the Grover oracle and the Encryption oracle.

We first clarify some parameters for the different structures. Suppose $\mathcal{O}_{\mathcal{R}_j}$ has round depth 1 and needs α ancilla qubits. Since the components of $\mathcal{O}_{\mathcal{R}_j}$, $\mathcal{O}_{\mathcal{R}_j^{-1}}$ are almost the same, it is reasonable to regard them as having the same cost. Suppose the round function iterates for r rounds, and one message register needs n qubits. The width of the key schedule is set to k' to show the difference from the other structures, because in our structure, the parallel execution of a round function with its inverse means that two consecutive roundkeys are required.

The comparison with previous structures under the above parameters is outlined in Table 5. Since we treat the round function as a unit, only the comparison between \mathcal{S}_p , \mathcal{S}_z , \mathcal{S}_i with our \mathcal{S}_{cp} are considered. Our \mathcal{S}_{cp} has the same round depth r as \mathcal{S}_p , and at the same time needs 4 message registers instead of $r + 1$ message registers in \mathcal{S}_p . Therefore, our \mathcal{S}_{cp} will have lower width than \mathcal{S}_p and lower D - W cost than \mathcal{S}_z , \mathcal{S}_i if α and k' are small enough.

Table 5: The comparison of different structures, where t is the minimal number such that $\sum_{i=1}^t i > r$.

Structure	Round depth	Width
\mathcal{S}_p	r	$k + (r + 1)n + \alpha$
\mathcal{S}_z	$\approx 2r$	$k + tn + \alpha \approx k + \sqrt{2rn} + \alpha$
\mathcal{S}_i	$2r$	$k + 2n + \alpha$
This paper	r	$k' + 4n + 2\alpha$

Circuits for Grover oracles. We first consider the Grover oracle: $|y\rangle |q\rangle \rightarrow |y\rangle |q \oplus f(y)\rangle$, where $f(y)$ is a boolean function which outputs one bit 1 or 0. An exhaustive key search Grover oracle has input state $|k\rangle$, and the correctness of the key is verified by some plaintext-ciphertext pairs. For simplicity we consider the case of one pair (m_0, c_0) . Denote an encryption circuit as C_* , the Grover oracle works as follows:

- C_* computes $|c\rangle$ with $|m_0\rangle$ and $|k\rangle$.
- A comparison process compares $|c\rangle$ with $|c_0\rangle$ to decide whether to flip $|q\rangle$.
- Do the uncomputation with C_*^\dagger .

It can be seen that the cost of the Grover oracle is almost twice that of the encryption circuit. Therefore, the cost of Grover oracles with different structures can be evaluated directly by referring to the cost of different encryption circuits in Table 5.

Since uncomputation of recovering m is necessary in the Grover oracle, the depth of the Grover oracle is dominated by that of C_* . Thus, \mathcal{S}_p is used to construct low-depth circuits of the Grover oracle in related research. Our circuit \mathcal{S}_{cp} greatly reduces the use of message registers to store intermediate states and will have lower width than \mathcal{S}_p if the number of ancilla qubits required in round functions is small enough.

Circuits for Encryption oracles. We then consider the Encryption oracle defined in [29]: $|m\rangle |0\rangle \rightarrow |m\rangle |Enc(m)\rangle$, where m is the plaintext and $Enc(m)$ is the encryption of m with a pre-fixed key. Encryption oracles allows the input register to be in a superposition $\sum_m |m\rangle$, and then the output will be a superposition $\sum_m |m\rangle |Enc(m)\rangle$. Note that the key register is not needed in the Encryption oracle since the roundkeys can be precomputed classically and the AddRoundKey can be realized by applying X gates on specified qubits. The construction of Encryption oracles using $\mathcal{S}_p, \mathcal{S}_z$ or \mathcal{S}_i is shown in Figure 7.

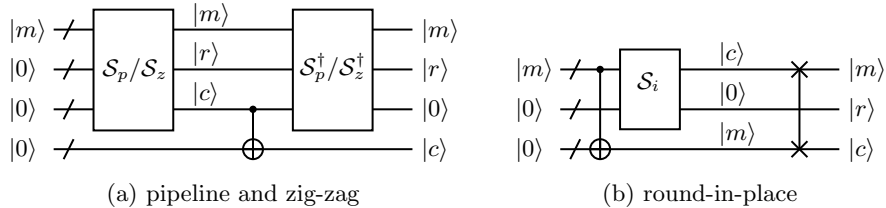


Fig. 7: The Encryption oracles based on different structures

We show that our structure \mathcal{S}_{cp} can be used to construct an Encryption oracle that has a smaller round depth. The round depth of an Encryption oracle using \mathcal{S}_i is $2r$ since there is no redundant intermediate states, and the round depth of an Encryption oracle using $\mathcal{S}_p/\mathcal{S}_z$ is twice the round depth of $\mathcal{S}_p/\mathcal{S}_z$ since uncomputation is needed to clean redundant states. However, the advantage of our structure is that \mathcal{S}_{cp} can greatly reduce the cost for cleaning redundant states. Since $|c_{j-1}\rangle$ is cleaned by $|c_j\rangle$, the remaining redundant states contains only $|c_{r-1}\rangle$, which can be cleaned by $|c\rangle$ using \mathcal{R}_r^{-1} . Thus, the round depth of an Encryption oracle using \mathcal{S}_{cp} and \mathcal{R}_r^{-1} is only $r + 1$, almost half the previous record. The comparison with previous results is outlined in Table 6.

Table 6: The depth and width of Encryption oracles with different structures

Encryption oracle	\mathcal{S}_p	\mathcal{S}_z	\mathcal{S}_i	This paper
round depth	$2r$	$\approx 4r$	$2r$	$r + 1$
width	$(r + 1)n + \alpha$	$\approx \sqrt{2}rn + \alpha$	$(1 + 2)n + \alpha$	$(1 + 4)n + 2\alpha$

6 Quantum circuits of AES

In this section we give several kinds of detailed quantum circuits of AES under the guidance of our structure in Section 5, including an Encryption oracle, encryption circuits under our structure and improved encryption circuits under the shallowed pipeline structure. The depth 10 circuit of AES MixColumns in Section 4 are used in all circuits to reduce the full depth, and the different circuits for AES S-box are introduced in Subsection 6.1. Encryption circuits for the encryption part and the key schedule of AES-128 are specified in Subsection 6.2, and the cost is compared with other circuits in different cases in Subsection 6.3⁷. In Subsection 6.4 we synthesize an Encryption oracle of AES-128 with the lowest T -depth to date, and in Subsection 6.5 we give an improved encryption

⁷ The main difference between the encryption circuit of AES-128, AES-192 and AES-256 lies in the key schedule. The key schedules and the cost analysis for encryption circuits of AES-192 and AES-256 are presented in Appendix B.

circuit under the shallowed pipeline structure with the lowest *TofD-W* cost to date.

6.1 Quantum circuits of AES S-box

We first introduce some knowledge on quantum circuits of AES S-box. The C_2 circuit of AES S-box defined by $C_2 : |x\rangle |y\rangle \mapsto |x\rangle |y \oplus S(x)\rangle$ is the main concern for out-of-place implementations, since C_1 circuits defined by $C_1 : |x\rangle |0\rangle \mapsto |x\rangle |S(x)\rangle$ are special cases of C_2 circuits and are easier to design. The C_3 circuit defined by $C_3 : |S(x)\rangle |x\rangle \mapsto |S(x)\rangle |0\rangle$ can be efficiently constructed on a C_1 circuit with a few more CNOT gates by the method in [22]. Moreover, some C_2 circuits of AES S-box can be decomposed into Sbox and SubS^\dagger . The Sbox circuit is defined by $\text{Sbox} : |x\rangle |0\rangle |y\rangle \mapsto |x'\rangle |r\rangle |y \oplus S(x)\rangle$, where $|r\rangle$ is the redundant state. Sbox can be decomposed into two parts denoted by SubS and SubC. SubS takes $|x\rangle |0\rangle$ as input and outputs $|x'\rangle |r\rangle$, where $|r\rangle$ contains linear components of $S(x)$. Then SubC adds them to $|y\rangle$ to get $|y \oplus S(x)\rangle$.

Many researchers have studied low Toffoli-depth Sboxes, which is illustrated in Table 7. In these related works, the target qubit of each Toffoli gate in SubS is always a new qubit $|0\rangle$, so the Toffoli gates can be replaced by qAND gates with T -depth 1, and SubS^\dagger can be realized with T -depth 0 using qAND † . Therefore, these related works can be used to synthesize low T -depth qAND-based C_2 circuits of AES S-box.

Note that the input register of some Sboxes can remain unchanged while others cannot. An Sbox is defined to be input-invariant if it can keep the input register unchanged, which means the information of the input register $|x\rangle$ is not lost before SubS^\dagger or Sbox^\dagger is done. We observe that, the reason why some Sboxes are not input-invariant is that the input register is updated by some ancilla qubits with CNOT gates to save qubits, and these ancilla qubits themselves are also updated. It is worthy to note that the updating of $|x\rangle$ can be uncomputed with only CNOT gates, as the target qubit of all Toffoli gates is always a new qubit $|0\rangle$ in related works of low Toffoli-depth Sboxes. Therefore, adding some CNOT gates can make this kind of Sbox input-invariant without increasing the number of ancilla qubits and the Toffoli-depth (see Table 7 for our results). The full depth is also not increased when the Toffoli gates are decomposed.

The process of finding a sequence of CNOT gates added to a given low Toffoli-depth Sbox to make it input-invariant can be integrated into an algorithm. It involves recording the qubits whose updatings should be memorized. All the input qubits are recorded at the beginning, and each qubit used to update a recorded qubit is recorded. Finally, a sequence of uncomputing the memorized updatings is returned. The detailed process is illustrated in Algorithm 2. Note that in order not to increase the Toffoli-depth, this algorithm is only suitable for such low Toffoli-depth Sboxes where the updating of input qubits is related with only CNOT gates.

Some C_1 circuits for low-width Toffoli-based circuits are also studied, see Table 8 for some recent works. If these C_1 circuits are used for AES S-box,

Table 7: Some low $TofD$ Sboxes

Source	#CNOT	#1qClifford	#Toffoli	$TofD$	Ancilla qubits	Input-invariant
[24]	186	4	34	6	120	✓
[22]	214	4	34	4	120	✓
[22]	356	4	78	3	182	✓
[38]	168	4	34	4	74	✗
This paper	179	4	34	4	74	✓
[38]	196	4	34	4	60	✗
This paper	207	4	34	4	60	✓
[23]	313	4	78	3	136	✗ ^b
[23]	162	4	34	4	68 ^a	✗ ^b

^a The full depth of this circuit is smaller when the Toffoli gates are decomposed.

^b Since the authors do not give specific implementations, we cannot give detailed costs for their input-invariant versions.

our encryption circuit of AES-128 in Subsection 6.3 will have the lowest $TofD$ - W/TD - W cost compared to other structures which take the round function as a unit.

Table 8: Some Toffoli-based C_1 circuits of AES S-box

Source	#CNOT	#1qClifford	#Toffoli	Toffoli-depth	Ancilla qubits
[36]	193	4	57	24	5
[36]	195	4	57	22	6
[35]	197	4	44	32	4

6.2 Round function and key schedule

In this subsection we give the detailed circuits of the iterative functions that we define for the encryption circuit and key schedule of AES-128. For AES-192 and AES-256, the main difference lies in the key schedule, which is illustrated in Appendix B.

For the encryption circuit, we define the beginning function B and the j -round function F_j which are shown in Figure 8. For simplicity, the result of applying multiple AES S-boxes on a qubit register $|x\rangle$ is denoted by $|S(x)\rangle$ throughout the rest of the paper. B and F_j acts as follows:

$$\begin{aligned}
 B &: |m\rangle |0\rangle |0\rangle |0\rangle \mapsto |c_0\rangle |S(c_0)\rangle |c_1\rangle |0\rangle, \\
 F_j &: |c_{j-1}\rangle |S(c_{j-1})\rangle |c_j\rangle |0\rangle \mapsto |c_j\rangle |S(c_j)\rangle |c_{j+1}\rangle |0\rangle.
 \end{aligned} \tag{3}$$

As a result, our encryption circuit is synthesized by connecting B, F_1, F_2, \dots, F_9 . Note that C_{cp} does not strictly adhere to the structure \mathcal{S}_{cp} in Figure 6. One can see that outputs of AES S-box are copied for cleaning the inputs in the next

Algorithm 2 Make an Sbox input-invariant.

Input: An NCT-based circuit $\mathcal{C} = \{g_0 g_1 \dots g_{t-1}\}$ of an Sbox with input qubits $|x_0 x_1 \dots x_7\rangle$, ancilla qubits $|r_0 r_1 \dots r_{m-1}\rangle$ and output qubits $|y_0 y_1 \dots y_7\rangle$. The updating of input qubits in the input Sbox should be related with only CNOT gates.

Output: A sequence of CNOT gates which is added to the Sbox to make it input-invariant.

```

seq ← [];
for i from 0 to m − 1 do
    updated[i] ← 0;
end for
for i from 0 to t − 1 do
    if  $g_i$  is not a CNOT gate then
        Continue;
    end if
    Let  $a$  be the control qubit and  $b$  be the target qubit of  $g_i$ .
    if  $b$  is some  $|x_i\rangle$  or some  $|r_j\rangle$  with  $updated[j] = 1$  then
        seq.append( $g_i$ );
        if  $a$  is some  $|r_j\rangle$  then
            updated[j] ← 1;
        end if
    end if
end for
seq.reverse();
return seq;

```

round, which saves the cost of the inverse of linear layers, and that our circuit C_{cp} is more compact, has clear linear and nonlinear layers, and has fewer linear layer components of AES than the circuit which adhere strictly to \mathcal{S}_{cp} . In a nonlinear layer, 16 C_1 circuits and 16 C_3 circuits are executed in parallel for the encryption circuit.

We then present a new key schedule circuit which is suitable with our new encryption circuit, since in F_j , two consecutive roundkeys $|k_j\rangle, |k_{j+1}\rangle$ should be able to be computed simultaneously by CNOT gates. Instead of storing $|k_j\rangle, |k_{j+1}\rangle$ in eight 32-qubit registers, we store linear components of two consecutive roundkeys registers to save qubits. The linear components of two consecutive roundkeys $|k_j\rangle, |k_{j+1}\rangle$ include $|k_j^0\rangle |k_j^1\rangle |k_j^2\rangle |k_j^3\rangle |S(k_j^3)\rangle$, and the computation of $|k_j\rangle, |k_{j+1}\rangle$ with CNOT gates is based on the dependence of consecutive roundkeys illustrated below:

$$\begin{cases} k_{j+1}^0 = Const_{j+1} \oplus S(k_j^j) \oplus k_j^0 \\ k_{j+1}^1 = Const_{j+1} \oplus S(k_j^j) \oplus k_j^0 \oplus k_j^1 \\ k_{j+1}^2 = Const_{j+1} \oplus S(k_j^j) \oplus k_j^0 \oplus k_j^1 \oplus k_j^2 \\ k_{j+1}^3 = Const_{j+1} \oplus S(k_j^j) \oplus k_j^0 \oplus k_j^1 \oplus k_j^2 \oplus k_j^3 \end{cases}, \quad (4)$$

where $Const_{j+1}$ is the $(j + 1)$ -th round constant.

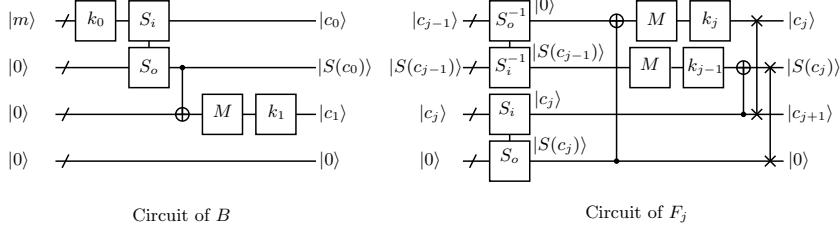


Fig. 8: Circuits of B and F_j . S_i, S_o and S_i^{-1}, S_o^{-1} stand for the input and output registers of C_1 circuits and C_3 circuits, respectively. MixColumns no longer acts on the first message register in F_9 . ShiftRows are omitted for simplicity throughout the rest of the paper.

By the dependence of consecutive roundkeys, we construct a circuit of key schedule which can compute linear components of two consecutive roundkey states with six 32-qubit registers. The beginning iteration K_0 and the j -th iteration K_j act as follows:

$$\begin{aligned}
 K_0 &: |k_0^0\rangle |k_0^1\rangle |k_0^2\rangle |k_0^3\rangle |0\rangle |0\rangle \mapsto |k_0^0\rangle |k_0^1\rangle |k_0^2\rangle |k_0^3\rangle |S(k_0^3)\rangle |0\rangle \\
 K_j &: |k_{j-1}^0\rangle |k_{j-1}^1\rangle |k_{j-1}^2\rangle |k_{j-1}^3\rangle |S(k_{j-1}^3)\rangle |0\rangle \mapsto |k_j^0\rangle |k_j^1\rangle |k_j^2\rangle |k_j^3\rangle |S(k_j^3)\rangle |0\rangle.
 \end{aligned} \tag{5}$$

The circuit of K_j is illustrated in Figure 9, and the circuit of K_0 is omitted due to its simplicity. In the nonlinear layer of K_j , 4 C_1 circuits and 4 reversed C_1 circuits are executed in parallel. Therefore, F_j and K_j can be synchronized. It is easy to see that $|k_j\rangle |k_{j+1}\rangle$ can be computed by the linear components with depth 5, thus a small increase in the depth of AddRoundKey trades for key register savings.

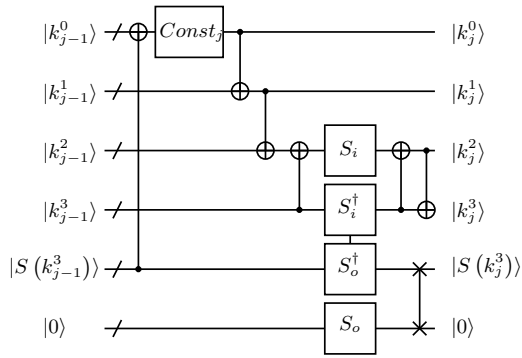


Fig. 9: The j -th iteration K_j of the key schedule.

Moreover, only five 32-qubit registers are enough if one uses the qAND-based Sbox, since Sbox^\dagger can clear the redundant states without increasing the T -depth. The beginning iteration K'_0 and the j -th iteration K'_j acts as follows:

$$\begin{aligned} K'_0 &: |k_0^0\rangle |k_0^1\rangle |k_0^2\rangle |k_0^3\rangle |0\rangle |0\rangle \mapsto |k_0^0\rangle |k_0^1\rangle |k_0^2\rangle |k_0^3\rangle |r_0\rangle |0\rangle \\ K_j &: |k_{j-1}^0\rangle |k_{j-1}^1\rangle |k_{j-1}^2\rangle |k_{j-1}^3\rangle |r_{j-1}\rangle |0\rangle \mapsto |k_j^0\rangle |k_j^1\rangle |k_j^2\rangle |k_j^3\rangle |r_j\rangle |0\rangle \end{aligned} \quad (6)$$

The corresponding circuit K'_j is shown in Figure 10. $|r_{j-1}\rangle$ and $|r_j\rangle$ are the redundant states within the computations of $|S(k_{j-1})\rangle$ and $|S(k_j)\rangle$, respectively. Similarly, K_0 's corresponding circuit K'_0 needs 4 Sboxes. It is worthy to note that the Sbox is actually input-invariant, which is easy to achieve based on our analysis in Subsection 6.1.

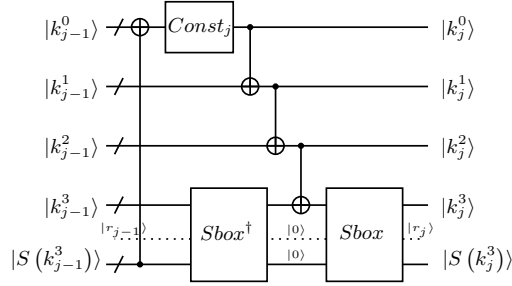


Fig. 10: K'_j with Sbox and Sbox^\dagger . The dashed line represents the ancilla qubits of qAND-based Sbox.

6.3 Encryption circuits of AES-128

In this subsection we give detailed encryption circuits of AES-128, which can be constructed with the iterative circuits F_j, B, K_j, K'_j defined by us. F_j runs in parallel with B_j , where K_j add specific 32-qubit registers to the message registers by CNOT gates. See Figure 11 for the hierarchy of our encryption circuit of AES-128.

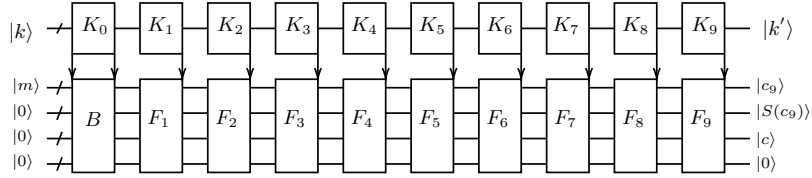


Fig. 11: Our encryption circuit of AES-128. The arrows indicate the AddRound-Key process at the beginning or end of K_j .

Denote our circuit under \mathcal{S}_{cp} as C_{cp} , and the circuit under \mathcal{S}_p and \mathcal{S}_i as C_p and C_i , respectively. We then compare the cost of our C_{cp} with C_p, C_i in both Toffoli-based and qAND-based AES S-box scenarios. For different circuits, the number of qubits required for the key registers and message registers, parallel C_1, C_2 circuits⁸ and layers of AES S-box are shown in Table 9. One can see that our circuit C_{cp} do not need C_2 circuits.

Table 9: Costs of encryption circuits of AES-128 for different structures

Circuits	C_p	C_i	C_{cp} with K_j	C_{cp} with K'_j
Qubits of key registers	128	128	192	160
Qubits of message registers	128×11	128×2	128×4	128×4
C_1 circuits in parallel	16	16	40	36
C_2 circuits in parallel	4	2	0	0
Layers of AES S-box	10	20	10	10

The cost of different structures can be computed when the number of ancilla qubits and the Toffoli/ T -depth of AES S-box are determined. For simplicity of comparison, assume that both C_1 and C_2 circuits require m ancilla qubits. In case of using Toffoli-based AES S-box, our circuit needs fewer ancilla qubits than C_p when $m < 42$, and has lower $TofD-W$ cost than C_i when $m < 16$. So our circuit will have lower $TofD-W$ -cost with state-of-art low-width AES S-box. In case of using qAND-based Sbox, our circuit needs fewer ancilla qubits than C_p when $m < 54$, and has lower $TD-W$ cost than C_i for all $m > 0$. In conclusion, among the structures which take the round function as a unit, our compressed pipeline structure can provide different encryption circuits with better $TofD/TD-W$ trade-offs when the width of AES S-box is small. An encryption circuit using [36]’s low-width S-Box with more balanced $TofD$ and W is given in Table 10⁹.

Since a Grover oracle is composed of an encryption circuit, its dagger and a small comparison process, the choice of parameters to make the $TD-W$ cost of Grover search lower is basically consistent with the above analysis.

6.4 AES Encryption oracle with lower T -depth

In this subsection we synthesize Encryption oracles of AES using the encryption part of our encryption circuit C_{cp} .

As introduced in Subsection 5.3, previous researchers synthesized AES-128 Encryption oracles which cannot break the limit of 2×10 layers of AES S-

⁸ Since a C_3 circuit can be constructed by a C_1 circuit with a few more CNOT gates using the method in [22], we regard them as the same type of C_1 circuits.

⁹ So far the circuit under the improved compressed pipeline structure with our improvement of the input-invariant Sbox has the lowest $TofD-W$ cost, since Sbox[†] in the round function oracle is delayed to the next round and the technique of combined Sbox and Sbox[†] can save many qubits.

box. Since the roundkeys can be precomputed in the Encryption oracle, the redundant states of our circuit C_{cp} only include $|c_9\rangle |S(c_9)\rangle$, which can be cleaned with only one layer of AES S-box. The clear function C shown in Figure 12 takes $|c_9\rangle |S(c_9)\rangle |c\rangle |0\rangle$ as input and outputs $|0\rangle |0\rangle |c\rangle |0\rangle$. Therefore, the AES-128 Encryption oracle can be constructed with $(10 + 1)$ layers of AES S-box.

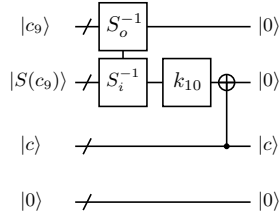


Fig. 12: The clear function C .

Using the qAND-based C_1 circuits and C_3 circuits with T -depth 3, we construct an AES-128 Encryption oracle with T -depth 33, which breaks the previous record of T -depth 60 in [22]. In the same way, an AES-192 (AES-256, respectively) Encryption oracle can be synthesized with T -depth 39 (45, respectively).

Since in AES-like Hashing the roundkeys are actually constants, our circuit can also be used to construct quantum oracles of AES-like Hashing with lower T -depth.

6.5 Key schedule of the shallowed pipeline structure with lower width

In this subsection we give an encryption circuit with the lowest $TofD-W$ cost to date under the shallowed pipeline structure. This encryption circuit uses an improved key schedule with our design of input-invariant Sbox in Subsection 6.1.

Jang *et al.* proposed the shallowed pipeline structure which delays the cleaning of redundant states by SubS^\dagger to the next round to reduce the full depth [23]. Then, Liu *et al.* improved the structure by sharing the ancilla qubits in Sbox and SubS^\dagger [38] to save qubits. They both adopted the straight line structure for the key schedule, where $|k_{j-1}\rangle$ will be updated by $|k_j\rangle$ in the j -th round. As introduced in Subsection 6.1, the Sbox given in [22] with unoptimized width is input-invariant, but the Sbox given in [38] with optimized width is no longer input-invariant, which leads to the lose of information in the input register. Since Sbox and SubS^\dagger need $|k_j^3\rangle$ and $|k_{j-1}^3\rangle$ for each j , respectively, Jang *et al.* allocated 32 additional qubits for storing $|k_{j-1}^3\rangle$ when using the input-invariant Sbox. Since the Sbox used by Liu *et al.* has smaller width but is no longer input-invariant, 10×32 qubits are allocated for storing all $|k_{j-1}^3\rangle$ with $1 \leq j \leq 10$. We show that the extra allocation of qubits for storing keywords is not necessary.

On the one hand, we have succeeded to make a low Toffoli-depth Sbox input-invariant with a few more CNOT gates in Subsection 6.1. On the other hand, by the dependence of consecutive roundkeys in Equation 4 used in K_j 's, we have $k_{j-1}^3 = k_j^2 \oplus k_j^3$. Therefore, unchanged $|k_j^3\rangle$ and the feasibility of computing $|k_{j-1}^3\rangle$ with $|k_j^2\rangle, |k_j^3\rangle$ means that the allocation of extra qubits for the key schedule in [23,38] is unnecessary. Our K_j'' 's for the shallowed pipeline structure which are similar to K_j 's only need 128 qubits for the key registers and work as follows:

$$\begin{aligned} K_0'' : |k_0^0\rangle |k_0^1\rangle |k_0^2\rangle |k_0^3\rangle |0\rangle &\mapsto |k_1^0\rangle |k_1^1\rangle |k_1^2\rangle |k_1^3\rangle |r_0, 0\rangle. \\ K_j'' : |k_j^0\rangle |k_j^1\rangle |k_j^2\rangle |k_j^3\rangle |r_{j-1}, 0\rangle &\mapsto |k_{j+1}^0\rangle |k_{j+1}^1\rangle |k_{j+1}^2\rangle |k_{j+1}^3\rangle |r_j, 0\rangle. \end{aligned} \quad (7)$$

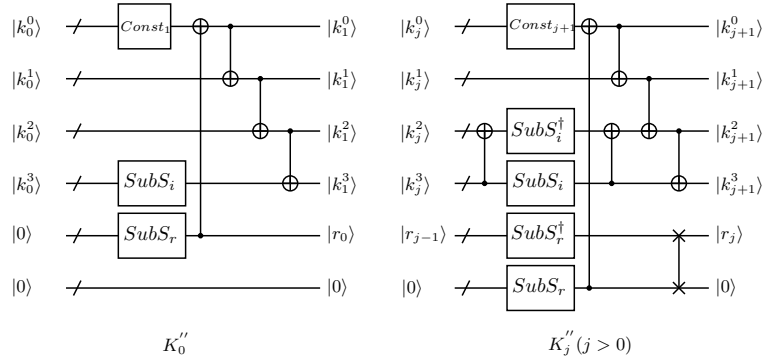


Fig. 13: K_j'' for the shallowed pipeline structure. $SubS_i$ represents the input register of $SubS$, and $SubS_r$ represents the register that will store the redundant state.

Using our K_j'' and our input-invariant Sbox, we achieve an encryption circuit of AES-128 under the shallowed pipeline structure with the lowest $ToFD-W$ cost 130720 to date¹⁰. A comparison with previous results is shown in Table 10.

7 Conclusion

In this work, quantum circuits of AES are studied and optimized. We first propose an improved greedy algorithm based on de Brugière *et al.*'s greedy algorithm. When applied to many MDS matrices and matrices used in block ciphers, our improved greedy algorithm gives the best results with the lowest depth for

¹⁰ We contacted the author of [23] and learned that their not-yet-public circuit of Sbox with 68 ancilla qubits makes the combined Sbox and $SubS^\dagger$ require 93 ancilla qubits. Our circuit uses the input-invariant version of Sbox and has a maximum width of 3268 which is internally optimized via ProjectQ [56].

Table 10: Comparison of encryption circuit metrics from various sources

Source	#CNOT	#X	#Toffoli	$TofD$	W	$TofD-W$ cost
[20]	166,548	1,456	151,552	12,672	984	12,469,248
[3]	192,832	1,370	150,528	-	976	-
[35]	53,360	1,072	16,688	12,168	264	3,212,352
[30]	107,960	1,570	16,940	1,880	864	1,624,320
[63]	128,517	4,528	19,788	2,016	512	1,032,192
[22]($p = 9$)	126,016	2,528	17,888	1,558	374	582,692
[35]	53,496	1,072	16,664	1,472	328	482,816
[22]($p = 18$)	126,016	2,528	17,888	820	492	403,440
[23]	81,312	800	12,240	40	6,368	254,720
[36]($m = 16$)	77,984	2,224	19,608	476	474	225,624
This paper^c	96,364	2,172	21,660	220	944	207,680
[38] (out-of-place)	75,024	800	12,920	40	4,823	192,920
[38] (in-place)	65,736	800	12,920	40	3,667	146,680
[23]	63,868	816	12,380	40	3,428	137,120
This paper^a	67,150	800	12,920	40	3,368	134,720
This paper^b	64,750	800	12,920	40	3,268	130,720

^a Using our improved shallowed pipeline structure and the input-invariant version of combined Sbox and Sbox[†] in [38].

^b Using our improved shallowed pipeline structure and the input-invariant version of combined Sbox and Sbox[†] with fewer qubits in [23].

^c Using our compressed pipeline structure and the C_1 circuit in [36] with $TofD$ 22 and 6 ancilla qubits.

all of them. For example, our improved method finds an in-place CNOT circuit of AES MixColumns with depth 10, which breaks the recent record of depth 16 and helps to reduce the full depth of AES. To further optimize quantum circuits of AES, we propose a new compressed pipeline structure for iterative building blocks whose round function can be taken as a unit. The Encryption oracle under the compressed pipeline structure will have the lowest round depth to date, and the encryption circuit under our structure will have better depth-width trade-off when the number of ancilla qubits of a round function is small enough. Detailed encryption circuits of AES under the guidance of our structure are given and compared with other circuits. Then an AES-128 Encryption oracle with T -depth 33 is synthesized. Finally, the shallowed pipeline structure is improved in two aspects of the key schedule and the Sbox, which leads to an encryption circuit with the lowest $TofD-W$ cost. Further optimization of the Sbox is left as a future work. Our methods in this paper can be used to optimize quantum circuits of other iterative building blocks.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This research

was supported by National Key Research and Development Project under Grant No. 2018YFA0704705 and CAS Project for Young Scientists in Basic Research (Grant No. YSBR-035).

References

1. Aaronson, S., Gottesman, D.: Improved simulation of stabilizer circuits. *Physical Review A* **70**(5), 052328 (2004)
2. Albrecht, M.R., Driessen, B., Kavun, E.B., Leander, G., Paar, C., Yalçın, T.: Block ciphers—focus on the linear layer (feat. pride). In: *Advances in Cryptology—CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2014, Proceedings, Part I* 34. pp. 57–76. Springer (2014)
3. Almazrooie, M., Samsudin, A., Abdullah, R., Mutter, K.N.: Quantum reversible circuit of AES-128. *Quantum information processing* **17**, 1–30 (2018)
4. Amy, M., Maslov, D., Mosca, M.: Polynomial-time T-depth optimization of clifford+ T circuits via matroid partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **33**(10), 1476–1489 (2014)
5. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **32**(6), 818–830 (2013)
6. Avanzi, R.: The QARMA block cipher family. Almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. *IACR Transactions on Symmetric Cryptology* pp. 4–44 (2017)
7. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: *Advances in Cryptology—ASIACRYPT 2015: 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29–December 3, 2015, Proceedings, Part II* 21. pp. 411–436. Springer (2015)
8. Banik, S., Funabiki, Y., Isobe, T.: Further results on efficient implementations of block cipher linear layers. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **104**(1), 213–225 (2021)
9. Barreto, P., Nikov, V., Nikova, S., Rijmen, V., Tischhauser, E.: Whirlwind: a new cryptographic hash function. *Designs, codes and cryptography* **56**, 141–162 (2010)
10. Barreto, P.S.: The Anubis block cipher. *NESSIE* (2000)
11. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: *Advances in Cryptology—CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14–18, 2016, Proceedings, Part II* 36. pp. 123–153. Springer (2016)
12. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., et al.: Prince—a low-latency block cipher for pervasive computing applications. In: *Advances in Cryptology—ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2–6, 2012, Proceedings* 18. pp. 208–225. Springer (2012)
13. de Brugière, T.G., Baboulin, M., Valiron, B., Martiel, S., Allouche, C.: Gaussian elimination versus greedy methods for the synthesis of linear reversible circuits. *ACM Transactions on Quantum Computing* **2**(3), 1–26 (2021)

14. Christof, B., Thorsten, K., Gregor, L.: Lightweight multiplication in $gf(2^n)$ with applications to mds matrices; *crypto 2016*. Incs 9814 (2016)
15. Cid, C., Murphy, S., Robshaw, M.J.: Small scale variants of the AES. In: *FSE*. vol. 3557, pp. 145–162. Springer (2005)
16. Daemen, J., Rijmen, V.: *The design of Rijndael*, vol. 2. Springer (2002)
17. Datta, K., Shrivastav, V., Sengupta, I., Rahaman, H.: Reversible logic implementation of AES algorithm. In: 2013 8th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS). pp. 140–144. IEEE (2013)
18. De Brugiére, T.G., Baboulin, M., Valiron, B., Martiel, S., Allouche, C.: Reducing the depth of linear reversible quantum circuits. *IEEE Transactions on Quantum Engineering* **2**, 1–22 (2021)
19. Fowler, A.G.: Time-optimal quantum computation. *arXiv preprint arXiv:1210.4626* (2012)
20. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying grover’s algorithm to AES: quantum resource estimates. In: *International Workshop on Post-Quantum Cryptography*. pp. 29–43. Springer (2016)
21. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. pp. 212–219 (1996)
22. Huang, Z., Sun, S.: Synthesizing quantum circuits of AES with lower T-depth and less qubits. In: *International Conference on the Theory and Application of Cryptology and Information Security*. pp. 614–644. Springer (2022)
23. Jang, K., Baksi, A., Kim, H., Song, G., Seo, H., Chattopadhyay, A.: Quantum analysis of AES. *Cryptology ePrint Archive* (2022)
24. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and LowMC. 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* 30. pp. 280–310. Springer (2020)
25. Jean, J., Nikolić, I., Peyrin, T.: Joltik v1. 3. *CAESAR Round 2* (2015)
26. Jean, J., Peyrin, T., Sim, S.M., Tourteaux, J.: Optimizing implementations of lightweight building blocks. *IACR Transactions on Symmetric Cryptology* **2017**(4), 130–168 (2017)
27. Jiang, J., Sun, X., Teng, S.H., Wu, B., Wu, K., Zhang, J.: Optimal space-depth trade-off of CNOT circuits in quantum logic synthesis. In: *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. pp. 213–229. SIAM (2020)
28. Junod, P., Vaudenay, S.: FOX: a new family of block ciphers. In: *Selected Areas in Cryptography: 11th International Workshop, SAC 2004, Waterloo, Canada, August 9–10, 2004, Revised Selected Papers* 11. pp. 114–129. Springer (2005)
29. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14–18, 2016, Proceedings, Part II* 36. pp. 207–237. Springer (2016)
30. Langenberg, B., Pham, H., Steinwandt, R.: Reducing the cost of implementing the advanced encryption standard as a quantum circuit. *IEEE Transactions on Quantum Engineering* **1**, 1–12 (2020)
31. Leander, G., May, A.: Grover meets simon–quantumly attacking the FX-construction. In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International*

- Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23. pp. 161–178. Springer (2017)
32. Li, S., Sun, S., Li, C., Wei, Z., Hu, L.: Constructing low-latency involutory MDS matrices with lightweight circuits. *IACR Transactions on Symmetric Cryptology* pp. 84–117 (2019)
 33. Li, Y., Wang, M.: On the construction of lightweight circulant involutory MDS matrices. In: *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*. pp. 121–139. Springer (2016)
 34. Li, Z., Cai, B., Sun, H., Liu, H., Wan, L., Qin, S., Wen, Q., Gao, F.: Novel quantum circuit implementation of advanced encryption standard with low costs. *Science China Physics, Mechanics & Astronomy* **65**(9), 290311 (2022)
 35. Li, Z., Gao, F., Qin, S., Wen, Q.: New record in the number of qubits for a quantum implementation of AES. *Frontiers in Physics* **11**, 1171753 (2023)
 36. Lin, D., Xiang, Z., Xu, R., Zhang, S., Zeng, X.: Optimized quantum implementation of aes. *Quantum Information Processing* **22**(9), 352 (2023)
 37. Liu, M., Sim, S.M.: Lightweight MDS generalized circulant matrices. In: *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*. pp. 101–120. Springer (2016)
 38. Liu, Q., Preneel, B., Zhao, Z., Wang, M.: Improved quantum circuits for AES: Reducing the depth and the number of qubits. In: *International Conference on the Theory and Application of Cryptology and Information Security*. pp. 67–98. Springer (2023)
 39. Liu, Q., Wang, W., Fan, Y., Wu, L., Sun, L., Wang, M.: Towards low-latency implementation of linear layers. *IACR Transactions on Symmetric Cryptology* pp. 158–182 (2022)
 40. Nielsen, M.A., Chuang, I.L.: *Quantum Computation and Quantum Information*. Cambridge University Press (2000)
 41. Patel, K.N., Markov, I.L., Hayes, J.P.: Optimal synthesis of linear reversible circuits. *Quantum Inf. Comput.* **8**(3), 282–294 (2008)
 42. Preskill, J.: Quantum computing in the NISQ era and beyond. *Quantum* **2**, 79 (2018)
 43. Q#, M.: Quantum development <https://devblogs.microsoft.com/qsharp/>
 44. Saeedi, M., Markov, I.L.: Synthesis and optimization of reversible circuits—a survey. *ACM Computing Surveys (CSUR)* **45**(2), 1–34 (2013)
 45. Sarkar, S., Syed, H.: Lightweight diffusion layer: Importance of Toeplitz Matrices. *IACR Transactions on Symmetric Cryptology* **2016**(1), 95–113 (2016)
 46. Sarkar, S., Syed, H.: Analysis of Toeplitz MDS Matrices. In: *Australasian Conference on Information Security and Privacy*. pp. 3–18. Springer (2017)
 47. Schaeffer, B., Perkowski, M.: A cost minimization approach to synthesis of linear reversible circuits. *arXiv preprint arXiv:1407.0070* (2014)
 48. Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: A 128-bit block cipher. *NIST AES Proposal* **15**(1), 23–91 (1998)
 49. Selinger, P.: Quantum circuits of T-depth one. *Physical Review A* **87**(4), 042302 (2013)
 50. Shende, V.V., Bullock, S.S., Markov, I.L.: Synthesis of quantum logic circuits. In: *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*. pp. 272–275 (2005)

51. Shi, H., Feng, X., Xu, S.: A framework with improved heuristics to optimize low-latency implementations of linear layers. *IACR Transactions on Symmetric Cryptology* **2023**(4), 489–510 (2023)
52. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit block cipher CLEFIA. In: *Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26–28, 2007, Revised Selected Papers 14*. pp. 181–195. Springer (2007)
53. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: *Proceedings 35th annual symposium on foundations of computer science*. pp. 124–134. IEEE (1994)
54. Sim, S.M., Khoo, K., Oggier, F., Peyrin, T.: Lightweight MDS involution matrices. In: *Fast Software Encryption: 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8–11, 2015, Revised Selected Papers 22*. pp. 471–493. Springer (2015)
55. Simon, D.R.: On the power of quantum computation. *SIAM journal on computing* **26**(5), 1474–1483 (1997)
56. Steiger, D.S., Häner, T., Troyer, M.: Projectq: an open source software framework for quantum computing. *Quantum* **2**, 49 (2018)
57. Sun, X., Tian, G., Yang, S., Yuan, P., Zhang, S.: Asymptotically optimal circuit depth for quantum state preparation and general unitary synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023)
58. Wu, B., He, X., Yang, S., Shou, L., Tian, G., Zhang, J., Sun, X.: Optimization of CNOT circuits under topological constraints. *arXiv preprint arXiv:1910.14478* (2019)
59. Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. *IACR Transactions on Symmetric Cryptology* pp. 120–145 (2020)
60. Zhang, A., Feng, X., Xu, S.: Size optimization of CNOT circuits on NISQ. *arXiv preprint arXiv:2210.05184* (2022)
61. Zhu, C., Huang, Z.: Optimizing the depth of quantum implementations of linear layers. In: *International Conference on Information Security and Cryptology*. pp. 129–147. Springer (2022)
62. Zou, J., Li, L., Wei, Z., Luo, Y., Liu, Q., Wu, W.: New quantum circuit implementations of SM4 and SM3. *Quantum Information Processing* **21**(5), 181 (2022)
63. Zou, J., Wei, Z., Sun, S., Liu, X., Wu, W.: Quantum circuit implementations of AES with fewer qubits. In: *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26*. pp. 697–726. Springer (2020)

A The depth 10 implementation of AES MixColumns.

Operation	Operation	Operation	Operation	Operation	Operation
Depth 1	(16, 0)	(22, 23)	(3, 23)	(1, 18)	(13, 21)
(12, 28)	(24, 8)	Depth 4	(8, 24)	(9, 26)	(12, 20)
(20, 4)	(26, 18)	(20, 27)	(17, 1)	(14, 23)	(24, 9)
(19, 3)	(15, 23)	(22, 31)	(31, 19)	(31, 8)	(19, 27)
(27, 11)	(2, 10)	(16, 17)	(22, 30)	(29, 13)	(8, 17)
(21, 5)	(29, 5)	(10, 18)	(7, 4)	(28, 12)	(26, 18)
(13, 29)	(3, 11)	(4, 21)	(2, 12)	(11, 4)	Depth 10
(6, 22)	(25, 17)	(25, 9)	(25, 18)	(6, 15)	(21, 5)
(30, 14)	(1, 9)	(7, 0)	(5, 21)	(20, 27)	(13, 29)
(23, 31)	(22, 14)	(29, 6)	Depth 7	(16, 25)	(30, 14)
(15, 7)	Depth 3	(2, 26)	(23, 24)	(0, 24)	(6, 22)
(18, 2)	(17, 26)	(11, 19)	(7, 18)	(10, 3)	(15, 7)
(26, 10)	(18, 19)	(3, 23)	(1, 2)	Depth 9	(23, 31)
(24, 1)	(20, 13)	(8, 24)	(16, 9)	(31, 7)	(27, 3)
(0, 8)	(31, 12)	Depth 5	(22, 19)	(29, 5)	(20, 4)
(9, 25)	(11, 3)	(18, 26)	(31, 20)	(2, 10)	(19, 11)
(16, 17)	(8, 9)	(27, 12)	(10, 27)	(22, 14)	(12, 28)
Depth 2	(21, 30)	(17, 9)	(5, 13)	(15, 23)	(9, 25)
(31, 7)	(28, 4)	(7, 3)	(28, 29)	(4, 28)	(26, 2)
(19, 27)	(2, 27)	(31, 15)	(25, 26)	(3, 11)	(18, 10)
(12, 20)	(7, 25)	(22, 8)	Depth 8	(25, 1)	(24, 16)
(13, 21)	(14, 6)	(23, 20)	(5, 22)	(30, 6)	(8, 0)
(4, 28)	(29, 15)	(24, 16)	(7, 17)	(0, 16)	(17, 1)
(30, 6)	(24, 16)	Depth 6			

Table 11: The implementation of AES MixColumns with quantum depth 10. (i, j) stands for the CNOT gate adding the i -th qubit to the j -th qubit. The outputs $|y_0\rangle, |y_1\rangle, \dots, |y_{31}\rangle$ are represented by 0, 17, 18, 27, 28, 21, 22, 15, 16, 25, 26, 3, 20, 29, 30, 7, 24, 1, 10, 19, 12, 5, 6, 31, 8, 9, 2, 11, 4, 13, 14, 23, respectively.

B The key schedules and cost analysis of our compressed pipeline structure for AES-192 and AES-256.

The key schedule of AES is based on 32-bit words. Denote the master key by W_0, W_1, \dots, W_{s-1} , where $s = 4, 6, 8$ for AES-128, AES-192 and AES-256, respectively. Except the given words (i.e., the words in the master key), 44, 52, 60 words are required by AES-128, AES-192 and AES-256, respectively. For all AES-128, AES-192 and AES-256, k_j^i equals W_{4j+i} and is the i -th 32-bit word of the j -th roundkey.

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

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

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

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

$$W_i = \begin{cases} W_{i-6} \oplus \text{SubWord}(\text{RotWord}(W_{i-1})) \oplus \text{Const}(i/4), & \text{if } i \equiv 0 \pmod{6}, \\ W_{i-6} \oplus W_{i-1}, & \text{otherwise,} \end{cases}$$

where $i = 4, 5, \dots, 51$.

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

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

where $i = 4, 5, \dots, 59$.

Fewer additional key registers are needed for AES-192's and AES-256's key schedules compared to AES-128's. For AES-192's key schedule, only 1 additional 32-bit key register is needed. The circuits of the first 5 rounds are shown below in Figure 14, and the following rounds are designed similarly. In each round, there is one layer of 4 parallel S-boxes. Note that the 8 key registers needed after K_i are $W_{4i}, W_{4i+1}, W_{4i+2}, \dots, W_{4i+7}$. $W_{4i+2}, \dots, W_{4i+7}$ are computed in the circuit, and for W_{4i}, W_{4i+1} we have:

- $W_{4i+1} = W_{4i+6} \oplus W_{4i+7}$;
- $W_{4i} = W_{4i+5} \oplus W_{4i+6}$, if $4i \pmod{6} \neq 0$;
- W_{4i} is stored in the first key register and cleaned in the next round, if $4i \pmod{6} = 0$.

One can see that the additional key register is necessary due to the case of $4i \pmod{6} = 0$, if only one layer of S-boxes is allowed in each round.

For AES-256's key schedule, no additional 32-bit key registers are needed. The circuits of K_1, K_2 are shown below in Figure 15, and the following rounds are designed similarly. Note that the circuit of K_0 is an identity. In each round, there is one layer of 4 parallel S-boxes. After K_i the 8 key registers store two consecutive roundkeys exactly.

The resource costs of different structures for encryption circuits of AES-192 and AES-256 are given in Table 12 and Table 13, respectively. The bound m for better trade-offs of our AES-192 and AES-256 are more relaxed than AES-128 since the extra costs of key schedules are smaller. For AES-192, our circuit needs fewer ancilla qubits than C_p when $m < 70$, and has lower *TofD-W* cost than C_i for all m ; For AES-256, our circuit needs fewer ancilla qubits than C_p when $m < 88$, and has lower *TofD-W* cost than C_i for all m .

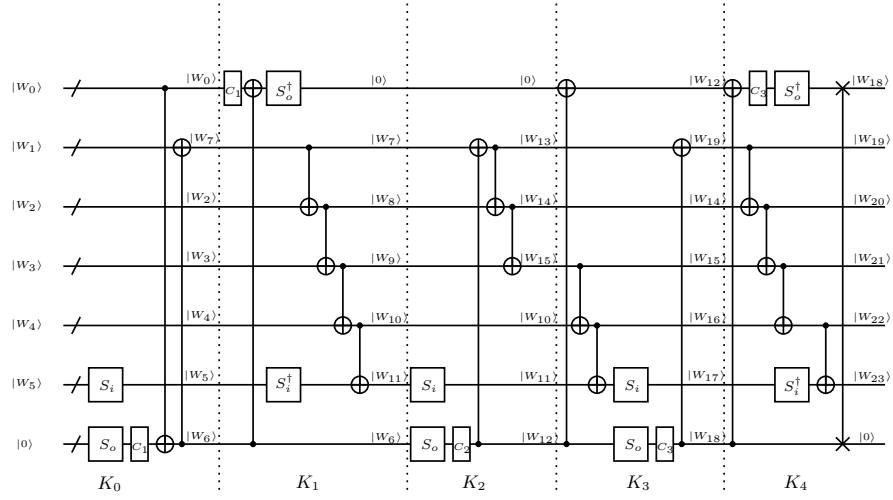


Fig. 14: The key schedule of our compressed pipeline structure for AES-192. S_i and S_o stands for the input and output registers of 4 S-boxes, and C_j stands for $Const_j$.

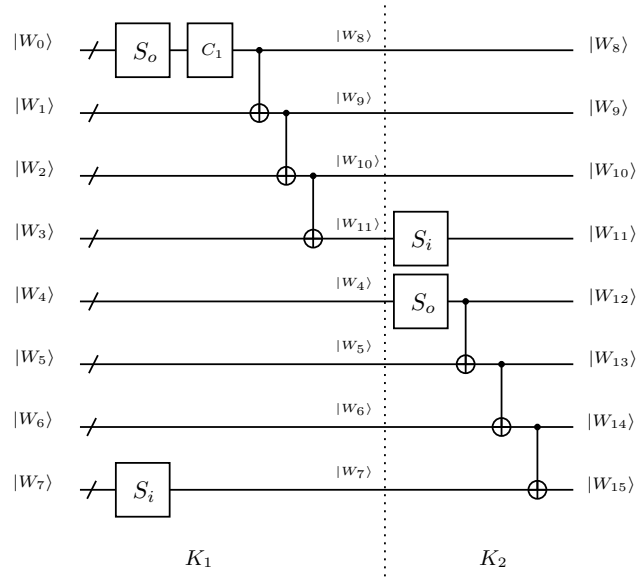


Fig. 15: The key schedule of our compressed pipeline structure for AES-256. S_i and S_o stands for the input and output registers of 4 S-boxes, and C_j stands for $Const_j$.

Table 12: Costs of encryption circuits of AES-192 for different structures

Circuits	C_p	C_i	our C_{cp}
Qubits of key registers	192	192	224
Qubits of message registers	128×13	128×2	128×4
\mathcal{C}_1 circuits in parallel	16	16	32
\mathcal{C}_2 circuits in parallel	4	2	4
Layers of AES S-box	12	24	12

Table 13: Costs of encryption circuits of AES-256 for different structures

Circuits	C_p	C_i	our C_{cp}
Qubits of key registers	256	256	256
Qubits of message registers	128×15	128×2	128×4
\mathcal{C}_1 circuits in parallel	16	16	32
\mathcal{C}_2 circuits in parallel	4	2	4
Layers of AES S-box	14	28	14