Cryptography and Communications
https://doi.org/10.1007/s12095-023-00666-y

RESEARCH

Quantum Implementation and Analysis of Default

Kyungbae Jang[1] · Anubhab Baksi[2] · Jakub Breier[3] · Hwajeong Seo[1] · Anupam Chattopadhyay[2]

Received: 30 April 2022 / Accepted: 26 July 2023

Abstract
In this paper, we present the quantum implementation and analysis of the recently proposed block cipher, DEFAULT. This cipher consists of two components, namely DEFAULT-LAYER and DEFAULT-CORE. Two instances of DEFAULT-LAYER are used before and after DEFAULT-CORE (in the so-called 'sandwich construction'). We discuss the various choices made to keep the cost for the basic quantum circuit and that of the Grover's oracle search, and compare it with the levels of quantum security specified by the United States' National Institute of Standards and Technology (NIST). All in all, our work fits in the research trend of finding the possible quantum vulnerability of symmetric key ciphers.

Keywords Block cipher · Quantum circuit · Default · Grove's search

Mathematics Subject Classification (2010) 68Q12

1 Introduction

Recent trends in symmetric cryptography lead to designs that either allow efficient implementations of side-channel and fault attack countermeasures or offer a certain level of inherent protection against these physical attack vectors. This is especially important in the area of

The work was done while the author was with Silicon Austria Labs, Graz, Austria.

✉ Jakub Breier
jbreier@jbreier.com

Kyungbae Jang
starj1023@gmail.com

Anubhab Baksi
anubhab001@e.ntu.edu.sg

Hwajeong Seo
hwajeong84@gmail.com

Anupam Chattopadhyay
anupam@ntu.edu.sg

[1] Hansung University, Seoul, South Korea

[2] Nanyang Technological University, 50 Nanyang Avenue, Singapore

[3] TTControl GmbH, Vienna, Austria

Published online: 23 August 2023

lightweight cryptography, which is aimed to be deployed in embedded devices, and therefore, physical attacks are a real threat. DEFAULT is a lightweight symmetric cipher, which takes its basic structure from GIFT [10], proposed at Asiacrypt'21 [8] with the aim to offer protection against differential fault analysis (DFA) [12] (see also [7, Section 5.1]). The main design feature to provide this protection is an SBox with linear structures, to which we refer to as LS SBox. It was shown that no matter how many faults the attacker injects at the input of such an SBox, it is impossible to determine the input value exactly. The DFA security of DEFAULT is 2^{64}, and generally, by using the same construction, it is $2^{n/2}$ for an n-bit cipher.

The emergence of quantum computing constitutes a potent threat against cryptography. Public key algorithms are especially weakened by Shor's algorithm that reduces the key search space complexity to a polynomial time [32]. There have been a number of research works dedicated to explore the applicability of public key ciphers against a quantum adversary, such as [19]. In general, symmetric ciphers offer higher security when it comes to quantum attacks, with Grover's algorithm being able to perform a full key search with $2^{n/2}$ queries.

One may note that the quantum security of the symmetric key ciphers is not properly analyzed at the time of design (basically, the quantum security is taken for granted by the designers). Case in point, the lightweight ciphers not only consume fewer resources in classical circuits but also applies in the quantum circuit. Thus, it may just so happen that a lightweight cipher, despite having requisite classical security, may not have the requisite quantum security. It highlights the ever-growing need to implement and analyze the newly proposed symmetric key ciphers with respect to an adversary with quantum computing capability.

Quantum circuits for AES were first reported by Grassl et al. in [20].[1] Currently, the costs estimated by Grassl et al. are cited in the NIST's (National Institute of Standards and Technology) post-quantum security requirements for the complexity of quantum attacks. From potential quantum attacks, NIST evaluates the post-quantum security level according to the cost required for a quantum attack on the target cipher. Swimming with this tide, several studies have been conducted to efficiently implement AES quantum circuits to reduce the cost of quantum attacks. Recently, this research field has been extended to lightweight ciphers. This is an important research pursuit since, lightweight ciphers can be picked up as a choice of implementation due to the tight resource constraints of many platforms, which, on the other hand, can be susceptible to a quantum adversary. Therefore, an important open research question is whether lightweight cipher leads to a more efficient quantum-enabled attack.

1.1 Our contribution

DEFAULT has two variants, one was proposed in [8], and the other in [6] (the latter is adopted from [5, Chapter 8]). The contribution in the paper consists of both variants and can be summarized as follows:

1. We report the first quantum circuit implementation of the cipher DEFAULT [6, 8]. Based on the implemented quantum circuit, we estimate the Grover key search cost for DEFAULT in detail.
2. We use the optimal implementation considering the trade-offs in the number of qubits, quantum gates, and circuit depth. The quantum circuit for DEFAULT in [6] uses the minimum number of qubits and keeps the number of gates and depth low. The quantum

[1] They are probably the first to do so for a symmetric key cipher.

circuit for DEFAULT in [8] effectively lowers the number of gates and circuit depth by allocating additional qubits for the key schedule.
3. We compare the quantum circuits of DEFAULT and other recently implemented lightweight ciphers. With this, we analyze the quantum implementation characteristics of lightweight ciphers.
4. We evaluate the post-quantum security for DEFAULT based on NIST's post-quantum security requirements.

ProjectQ,[2] a Python-based quantum programming tool, is used in this work. The relevant source code is shared as an open-source project[3]

1.2 Related works

Quite a few implementation/analysis works on symmetric key ciphers with respect to a functional quantum computer have been reported so far. Examples of such works include [3, 4, 9, 15, 20, 21, 23–26, 29, 30, 35]. The range of these works cover stream and block ciphers (including the recent lightweight ciphers), as well as AEADs.

Additionally, the LIGHTER-R tool was proposed in [17] for finding the improved in-place implementation of SBoxes. Also, the work in [34], although not directly related to quantum, supports the quantum-friendly implementation of linear layers (with CNOT gates and with no anciall/garbage qubit).

2 Background

In simple words, a quantum algorithm (that has a theoretical advantage over the best-known classical algorithm) starts with a random set of inputs. Then, with the quantum circuit, it finds a solution that works with a high probability. Thus, all such algorithms are inherently probabilistic.

2.1 Key recovery using Grover's Algorithm

1. Hadamard gates are applied to all qubits of the n-qubit key to prepare the key in superposition state ($|\psi\rangle$). An n-qubit key has the same amplitude for 2^n states as:

$$|\psi\rangle = H^{\otimes n}|0\rangle^{\otimes n} = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{1}{2^{n/2}}\sum_{x=0}^{2^n-1}|x\rangle \tag{1}$$

2. In the oracle, the target cipher is implemented as a quantum circuit. The quantum circuit implemented in oracle generates ciphertexts for all key values by encrypting the known plaintext with the previously prepared key in the superposition state. The generated ciphertexts are compared with the known ciphertext and, if they match ($f(x) = 1$ in Eq. (2)), the sign of the key state to be recovered is changed to negative ($f(x) = 1$ in Eq. (3)). At the end of the oracle, the implemented quantum circuit works once more in

[2] https://projectq.ch/.

[3] https://github.com/starj1023/DEFAULT_QC.

reverse, returning the generated ciphertexts back to known plaintext.

$$f(x) = \begin{cases} 1 \text{ if } Enc_k(m) = c \\ 0 \text{ if } Enc_k(m) \neq c \end{cases} \tag{2}$$

$$U_f(|\psi|-\rangle) = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x|-\rangle \tag{3}$$

3. The diffusion operator amplifies the amplitude of the key state to be recovered, which is indicated by the oracle by changing the sign to negative. The quantum circuit for the diffusion operator is usually generic, so no special technique is required to implement it. Also, the diffusion operator is generally ignored when estimating the cost of Grover's algorithm because its overhead is negligible compared to the oracle. Finally, Grover's algorithm measures the solution key with high probability by repeating the oracle and diffusion sufficiently to increase the amplitude of the key to be recovered.

2.2 Quantum gate basics

Reversible computing that can return a given output to input is unitary of quantum computers. There are several representative quantum gates to implement reversible quantum circuits for ciphers, as shown in Fig. 1. The X gate can replace the classical NOT operation, inverting the state of the input qubit, i.e., $\mathrm{X}(x) = \sim x$. The CNOT gate can replace the classical XOR operation. This quantum gate inverts the state of the target qubit if the control qubit is 1, i.e., $\mathrm{CNOT}(x, y) = (x, x \oplus y)$. The Toffoli gate can replace the classical AND operation. This quantum gate inverts the state of the target qubit if both control qubits are 1, i.e., $\mathrm{Toffoli}(x, y, z) = (x, y, z \oplus (x \cdot y))$ Since the diagram of the Toffoli gate in Fig. 1 is simplified, it is actually decomposed into quantum gates such as X, CNOT, and H, T gates [1]; and the cost varies with the format of decomposition. The Swap gate swaps the state of the input qubits, i.e., $\mathrm{Swap}(x, y) = (y, x)$.

3 Description of DEFAULT

DEFAULT [8] is a 128-bit lightweight block cipher, aimed at providing inherent protection against DFA. It borrows its structure from GIFT- 128 [11], but changes the SBox to prevent the attacker from getting enough differential information to uniquely identify the key among

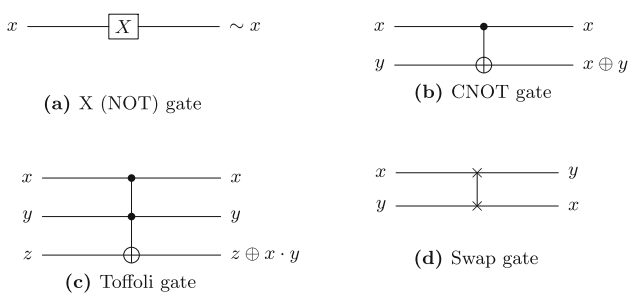


Fig. 1 Common quantum gates

the key candidates after injecting faults. The protection is achieved by using so-called linear structures (LS) in the DEFAULT SBox. By utilizing a 4×4 SBox with 4 LS, DEFAULT achieves a 2^{64} security for its 128-bit state size – this means that after injecting as many faults as the attacker wants, they still have to do an exhaustive search with a complexity of 2^{64}. Usage of such SBoxes, however, lowers the cipher's security against differential cryptanalysis, as it introduces linearity. Therefore, to provide protection against both, DEFAULT uses a sandwich construction – two outer blocks (called DEFAULTl) use some SBox with non-trivial LS (referred to as LS SBox for simplicity), while the inner block (called DEFAULTc) uses an SBox with no non-trivial (referred to as non-LS SBox). In the following, we provide a description of each of these.

The schematic diagrams of DEFAULT can be found in Fig. 2. In particular, the overall sandwich construction of DEFAULT (which is consisted by sandwiching an instance of DEFAULT-CORE between two instances of DEFAULT-LAYER)) is shown in Fig. 2(c).

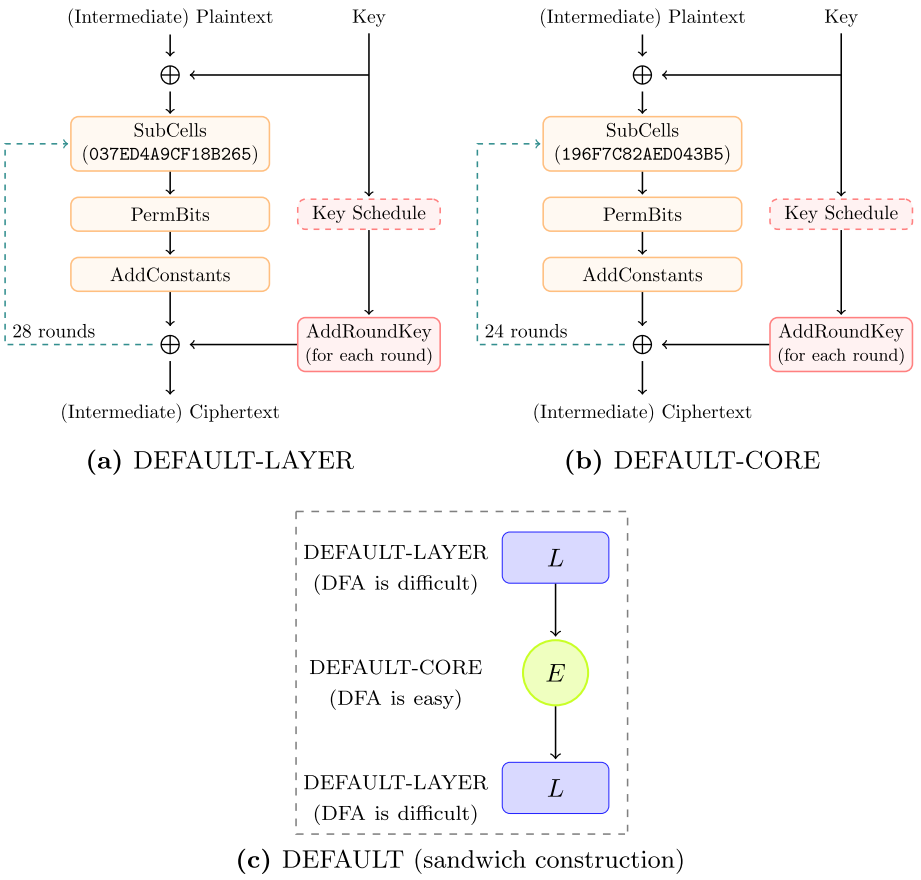


Fig. 2 Schematic structure of DEFAULT

3.1 DEFAULT-LAYER

DEFAULT-LAYER is a 28-round keyed permutation. Its round consists of 4 operations: SubCells applies a 4-bit SBox to the state; PermBits permutes the state bits the same way as GIFT- 128 [10]; The AddRoundConstants sub-routine XORs a 6-bit constant and another bit of the state is flipped at each round; AddRoundKey XORs the round key to the state. The schematic of DEFAULT-LAYER is depicted in Fig. 2(a). A detailed description of all the operations is provided in the following.

SubCells In this sub-routine, DEFAULT-LAYER uses a 4-bit LS SBox (S = 037ED4A9CF1 8B265), applied to every nibble of the state: $w_i \leftarrow S(w_i)$, $\forall i \in \{0, \ldots, 31\}$. The coordinate functions of this SBox are given respectively by:

$$\begin{aligned} y_0 &= x_0 \oplus x_1 \oplus x_2, \\ y_1 &= x_0 \oplus x_1 \oplus x_0x_1 \oplus x_0x_2 \oplus x_1x_3 \oplus x_2x_3, \\ y_2 &= x_1 \oplus x_2 \oplus x_3, \\ y_3 &= x_0x_1 \oplus x_2 \oplus x_0x_2 \oplus x_3 \oplus x_1x_3 \oplus x_2x_3. \end{aligned}$$

PermBits The bit-permutation is taken from that of the bit-permutation P_{128} in GIFT- 128, mapping bits from bit position i of the internal state to bit position $P_{128}(i)$: $b_{P_{128}(i)} \leftarrow b_i$, $\forall i \in \{0, ..., 127\}$. The specification of this permutation is as stated as follows: (0, 33, 66, 99, 96, 1, 34, 67, 64, 97, 2, 35, 32, 65, 98, 3, 4, 37, 70, 103, 100, 5, 38, 71, 68, 101, 6, 39, 36, 69, 102, 7, 8, 41, 74, 107, 104, 9, 42, 75, 72, 105, 10, 43, 40, 73, 106, 11, 12, 45, 78, 111, 108, 13, 46, 79, 76, 109, 14, 47, 44, 77, 110, 15, 16, 49, 82, 115, 112, 17, 50, 83, 80, 113, 18, 51, 48, 81, 114, 19, 20, 53, 86, 119, 116, 21, 54, 87, 84, 117, 22, 55, 52, 85, 118, 23, 24, 57, 90, 123, 120, 25, 58, 91, 88, 121, 26, 59, 56, 89, 122, 27, 28, 61, 94, 127, 124, 29, 62, 95, 92, 125, 30, 63, 60, 93, 126, 31).

AddRoundConstants This subroutine XORs a single bit "1" and a 6-bit round constant C = $c_5||c_4||c_3||c_2||c_1||c_0$ into the cipher state at bit positions 127, 23, 19, 15, 11, 7 and 3, respectively: $w_{127} = w_{127} \oplus 1$, $w_{23} = w_{23} \oplus c_5$, $w_{19} = w_{19} \oplus c_4$, $w_{15} = w_{15} \oplus c_3$. The round constants (6-bit) for DEFAULTc are given respectively by: (1, 3, 7, 15, 31, 62, 61, 59, 55, 47, 30, 60, 57, 51, 39, 14, 29, 58, 53, 43, 22, 44, 24, 48, 33, 2, 5, 11, 28); the first 24 of which are used by DEFAULTl. At every round, the corresponding constant is encoded to a 6-bit word and XORed to the cipher state, with c_0 being the least significant bit.

AddRoundKey This subroutine applies the bit-wise XOR the respective round key k to the state: $b_i \leftarrow b_i \oplus k_i^j$, $\forall i \in \{0, ..., 127\}$.

Key Schedule There are two variants of key schedule, depending on which version of DEFAULT is considered:

1. The key scheduling algorithm of DEFAULT defined in [8] operates as follows. It generates four 128-bit sub-keys K_0, K_1, K_2 and K_3 from the 128−bit master key K as follows: $K_0 = K$ and $K_{i+1} = \mathcal{R}'(\mathcal{R}'(\mathcal{R}'(\mathcal{R}'(K_i))))$ for $i \in [0, 1, 2]$, where $\mathcal{R}'$ denotes the $\mathcal{R}$ round function with no AddLayerKey layer and with the AddRoundConstants layer changed to only flipping the bit at position 127. Alternatively, $\mathcal{R}'$ can be seen as the $\mathcal{R}$ function with an all-zero round key and an all-zero round constant. Then, these four

sub-keys are used to generate the round keys as follows: for round i with $i \geq 0$, the sub-key $K_{i \bmod 4}$ is used as a round key input for AddLayerKey.

2. The key scheduling algorithm of DEFAULT defined in [6] uses only rotation operations. The 128-bit key (which is the same as the master key for the underlying cipher in the case of the ad-hoc layer) is split into 16-bit words: $k_7 \,||\, k_6 \,||\, \ldots \,||\, k_0$. Each round key is extracted first and then the next round key is generated by the update rule: $k_7 \,||\, k_6 \,||\, \ldots \,||\, k_0 \leftarrow (k_7 \,||\, k_6 \,||\, \ldots \,||\, k_0) \ggg 20$ followed by $k_7 \ggg 1$, where $\ggg i$ is an i-bit rotation to the right within the 16-bit word.

3.2 DEFAULT-CORE

DEFAULT-CORE is a 24-round block cipher. It follows the same structure as DEFAULT-LAYER except for the following:

1. The SBox (196F7C82AED043B5) does not have any non-trivial linear structures. The coordinate functions of this SBox are given respectively by:

$$\begin{aligned} y_0 &= 1 \oplus x_1 \oplus x_0x_1 \oplus x_0x_2 \oplus x_3, \\ y_1 &= x_1 \oplus x_2 \oplus x_0x_2 \oplus x_3, \\ y_2 &= x_1 \oplus x_2 \oplus x_0x_3, \\ y_3 &= x_0 \oplus x_1x_2 \oplus x_3 \oplus x_0x_3 \oplus x_0x_1x_3 \oplus x_2x_3. \end{aligned}$$

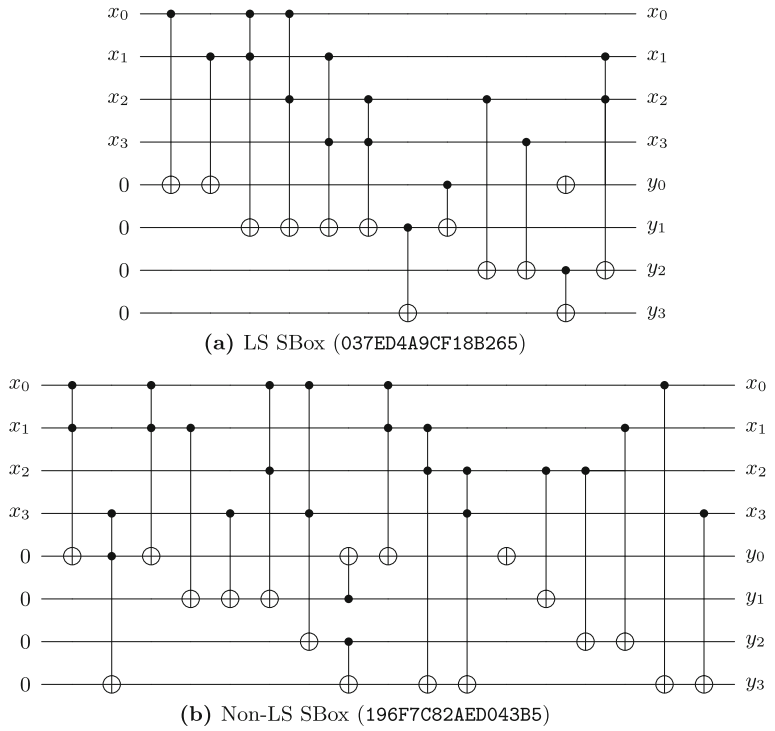
2. The key schedule is the same as DEFAULT-LAYER given in [6] (described in variant 2 earlier in Sect. 3.1).

The schematic of DEFAULT-CORE is depicted in Fig. 2(b). As most of the operations are identical to DEFAULT-CORE, we skip the detailed description to avoid repetition.

4 Quantum implementation

4.1 LS and Non-LS SBoxes

In this section, the implementation of the quantum circuit for DEFAULT is described in detail. In quantum circuits for block ciphers of SPN structure, the most resources are generally required for SBox implementation. To achieve an efficient quantum circuit for DEFAULT, in implementing SBox, we use the LIGHTER-R tool [17] instead of implementing the algebraic normal form defined in [8] as it is. Figure 3 shows the naïve quantum implementation of DEFAULT SBoxes using its coordinate functions (see Sect. 3). Notice that the input variables (x_0, x_1, x_2, x_3) are not used and the output variables (y_0, y_1, y_2, y_3) are initialized at 0. Figure 4 shows the quantum implementation of DEFAULT SBoxes using LIGHTER-R. Table 1 shows detailed resources for the quantum implementation of DEFAULT SBoxes.



(a) LS SBox (037ED4A9CF18B265)

(b) Non-LS SBox (196F7C82AED043B5)

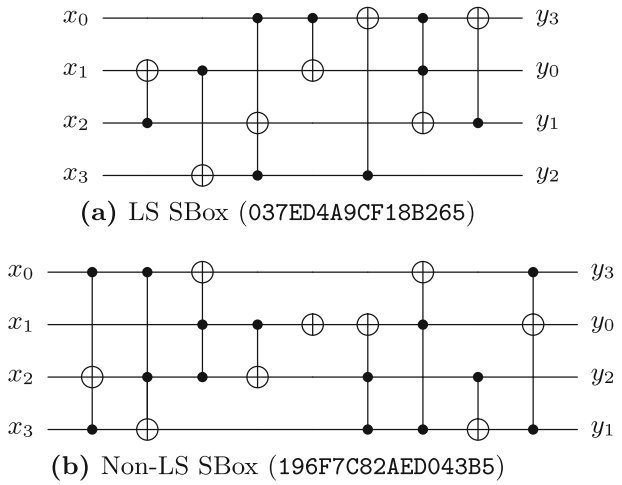
Fig. 3 Naïve quantum implementation of DEFAULT SBoxes

For detailed resource estimation, we decompose the Toffoli gate to the Clifford + T level. Following the method of [1], the Toffoli gate is decomposed into 7 T gates + 8 Clifford gates, with a T-depth of 4 and a total depth of 8. This decompose method applies equally to all Toffoli gates used in DEFAULT.

As shown in Table 1, the naïve implementation uses more resources in terms of gate count and circuit depth, not to mention it requires an additional 4 qubits for output. On the other hand, an implementation using LIGHTER-R [17] (the results shown here were found using NCT-gc option) can achieve in-place quantum circuits for the SBoxes with fewer resources. In-place LS and Non-LS SBoxes operate in parallel 32 times for 128-qubit in SubCell.

4.2 PermBits

Bit permutation can be implemented using quantum Swap gates, but since it can be replaced by changing the index of qubits, it is not counted as a resource in most cases [20, 26, 28, 31]. We do not use Swap gates by implementing logical swap that changes the index of qubits. In ProjectQ, the bit-permutation can be implemented as in Code 1.1.



(a) LS SBox (`037ED4A9CF18B265`)

(b) Non-LS SBox (`196F7C82AED043B5`)

Fig. 4 Quantum implementation of DEFAULT SBoxes using LIGHTER-R

4.3 AddRoundConstants and AddRoundKey

AddRoundConstants is implemented using a few X gates and the depth is also low as one. Since round constants are defined in advance, it is implemented by performing X gates according to the position where the bit of the round constant is 1. The number of X gates used is determined by the round constant. (e.g., the number of X gates with a round constant of 7 is 3).

AddRoundKey, which XORs a 128-qubit round key, is implemented using 128 CNOT gates and has a depth of 1. Algorithms 1 and 2 describe the quantum circuits performing AddRoundKey and AddRoundConstants in our DEFAULT implementation.

Table 1 Quantum resources required for DEFAULT SBoxes implementations

SBox	Method	#CNOT	#1qCliff	#T	#qubits	Full depth
LS	Naïve	33	2	28	8	34
	LIGHTER-R⁎	17	2	14	4	19
Non-LS	Naïve	57	15	56	8	69
	LIGHTER-R⁎	38	11	42	4	50

⁎: Used in this work

Algorithm 1 AddRoundConstants

Input: 128-qubit x, Round constant RC
Output: x (after update)
if 1st bit of $RC = 1$ **then**
    $x[3] \leftarrow \mathrm{X}(x[3])$
if 2nd bit of $RC = 1$ **then**
    $x[7] \leftarrow \mathrm{X}(x[7])$
if 3rd bit of $RC = 1$ **then**
    $x[11] \leftarrow \mathrm{X}(x[11])$
if 4th bit of $RC = 1$ **then**
    $x[15] \leftarrow \mathrm{X}(x[15])$
if 5th bit of $RC = 1$ **then**
    $x[19] \leftarrow \mathrm{X}(x[19])$
if 6th bit of $RC = 1$ **then**
    $x[23] \leftarrow \mathrm{X}(x[23])$
$x[127] \leftarrow \mathrm{X}(x[127)$

Algorithm 2 AddRoundKey

Input: 128-qubit x, Round key RK
Output: x (after update)
for $i = 0$ to 127 **do**
    $x[i] \leftarrow \mathrm{CNOT}(RK[i], x[i])$

Code 1.1: Implementation of DEFAULT PermBits using logical swap

```
def Permutation (eng, x): # `x` is of 128-qubits

    index = [0, 5, 10, 15, 16, 21, 26, 31, 32, 37, 42, 47, 48, 53,
             58, 63, 64, 69, 74, 79, 80, 85, 90, 95, 96, 101, 106,
             111, 112, 117, 122, 127, 12, 1, 6, 11, 28, 17, 22, 27,
             44, 33, 38, 43, 60, 49, 54, 59, 76, 65, 70, 75, 92, 81,
             86, 91, 108, 97, 102, 107, 124, 113, 118, 123, 8, 13,
             2, 7, 24, 29, 18, 23, 40, 45, 34, 39, 56, 61, 50, 55,
             72, 77, 66, 71, 88, 93, 82, 87, 104, 109, 98, 103, 120,
             125, 114, 119, 4, 9, 14, 3, 20, 25, 30, 19, 36, 41, 46,
             35, 52, 57, 62, 51, 68, 73, 78, 67, 84, 89, 94, 83, 100,
             105, 110, 99, 116, 121, 126, 115]

    new_x = []
    for i in range (128):
        new_x.append (x[index[i]])

    return new_x
```

Table 2 Quantum resources required for DEFAULT-LAYER and DEFAULT-CORE implementations

Component		#CNOT	#1qCliff	#T	#qubits	Full depth
DEFAULT-LAYER	1 round	672	69	448	128	21
DEFAULT-CORE		1,344	356	1,344	128	52
DEFAULT-LAYER	Full	18,816	1,917	12,544	128	561
DEFAULT-CORE		32,256	8,561	32,256	128	1,225

4.4 DEFAULT-LAYER and DEFAULT-CORE

DEFAULT-LAYER is 28 rounds and DEFAULT-CORE is 24 rounds. A round consists of SubCell → PermBits → AddConstants → AddRoundKey and the quantum circuits described above are used. In DEFAULT-LAYER, LS SBox is used for SubCell and Non-LS SBox is used for SubCell in DEFAULT-CORE. Table 2 shows the quantum resources required to implement the quantum circuit for the DEFAULT-LAYER and DEFAULT-CORE. The reason why more quantum resources are used in DEFAULT-CORE despite fewer rounds is that the cost for a SubCell in which Non-LS SBox is high (Table 3).

4.5 Key Schedule

For key schedule, the on-the-fly approach is adopted in most quantum implementations [2, 20, 22, 26, 29, 30, 35]. In the quantum implementation, storing all round keys incurs significant overhead for qubits. For this reason, an on-the-fly approach that can save qubits by using a round key and replacing it with the next round key is frequently adopted. In addition, since the key schedule is performed in each round, in an ideal case, the key schedule can be operated in parallel with the round function. However, adopting the on-the-fly approach is inefficient for the key schedule in DEFAULT [8].

In DEFAULT, only 4 round keys including the master key are used in AddRoundKey. Of course, it is possible to adopt an on-the-fly approach to implementing a quantum circuit for the DEFAULT key schedule. However, the key schedule for generating the round key must be operated in every round, and the cost of the SubCells (LS SBox), which is performed 4 times for the key schedule, cannot be ignored. In implementing the DEFAULT key schedule, we take the overhead of qubits and initially generate 4 round keys and use them for each round. This is the best choice as we consider the qubit and (gate, depth) trade-off. For this, we allocate qubits for storing 3 round keys, excluding the master key (i.e., 3×128 qubits), but provide the optimal qubit and (gate, depth) trade-off. Figure 5 shows the quantum circuit for the DEFAULT key schedule in which 4 round keys are stored and used.

The two variants of DEFAULT differ only in the key schedule. The key schedule in [8] described above is heavier than that of [6]. Since the key schedule in [6] uses only rotation operations, quantum resources are not required. For this very reason, it is not described here.

Table 3 Quantum resources required for DEFAULT key schedule implementation

Sub-routine	#CNOT	#1qCliff	#T	#qubits	Full depth
Key schedule [8]	6,912	780	5,376	512	232

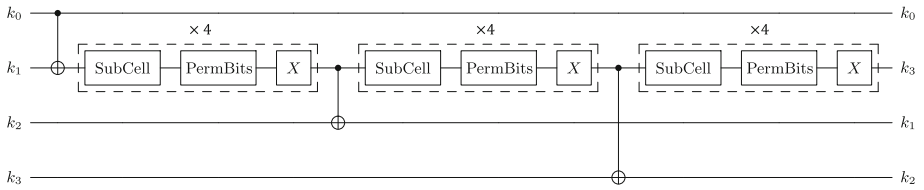


Fig. 5 DEFAULT key schedule in quantum

5 Performance and analysis

In this section, we present the performance of our DEFAULT quantum circuit implemented in this work. As noted earlier, the quantum programming tool, ProjectQ, is used to implement and simulate the quantum circuits. The ClassicalSimulator, a library of ProjectQ, is used to verify the implementation, and the ReousrceCounter is used to analyze the quantum resources used. Table 4 shows the resources required for our DEFAULT quantum circuit implementation and is compared with quantum circuits of other block ciphers. For consistency, Table 4 compares quantum resources at the NCT (NOT, CNOT, Toffoli) level in which the Toffoli gates are not decomposed.

For the quantum circuits of block ciphers, the required quantum resources are determined according to the encryption structure and implementation technique. As it can be seen from Table 4; SPECK, LEA, HIGHT and CHAM use more quantum resources than other block ciphers. SPECK, LEA, HIGHT, and CHAM are block ciphers of the ARX (Addition, Rotation, XOR) structure in which addition is used for encryption. Rotation can design a circuit without using quantum resources at all through a logical swap that changes the index between qubits. XOR is simply implemented as a CNOT gate. On the other hand, quantum addition is implemented with a combination of multiple Toffoli, CNOT, and X gates, and the circuit depth is high. Unlike classical computers, addition is a complex operation in quantum computers, and since there are various design methods, many studies have been proposed to effectively implement it [16, 18, 33].

Conversely, in the quantum circuit implementation for block ciphers of the SPN (Substitution-Permutation Network) structure, relatively few quantum resources are used if the SBox can be implemented efficiently. This is convincing as fewer quantum resources

Table 4 Summary of quantum resources required for DEFAULT and other block ciphers

Cipher	#CNOT	#X	#Toffoli	Toffoli depth	#qubits	Depth
DEFAULT [6]	13,824	1,131	8,192	256	256	644
DEFAULT [8]	23,040	1,143	8,960	280	640	754
GIFT-128/128 [26]	6,144	10,853	6,144	N/A	256	528
PRESENT-64/128 [26]	4,838	1,164	2,232	N/A	192	311
PIPO-64/128 [28]	2,248	1,477	1,248	N/A	192	248
SPECK-128/128 [2]	25,862	75	7,938	N/A	256	10,144
LEA-128/128 [27]	32,616	11,152	10,248	N/A	388	6,505
HIGHT-64/128 [27]	22,614	4,496	5,824	N/A	228	2,479
CHAM-128/128 [27]	28,760	4,880	4,880	N/A	292	5,307

Table 5 Details of quantum resources required for DEFAULT and other block ciphers

Cipher	#CNOT	#1qCliff	#T	T depth	#qubits	Full depth
DEFAULT [6]	62,976	12,395	57,344	1,024	256	2,291
DEFAULT [8]	76,800	13,175	62,720	1,120	640	2,497
GIFT-128/128 [26]	35,840	19,377	35,840	N/A	256	1,520
PRESENT-64/128 [26]	18,230	5,628	15,624	N/A	128	1,179
PIPO-64/128 [28]	9,928	3,973	8,736	N/A	192	1,041
SPECK-128/128 [2]$^{\diamond}$	73,490	15,951	55,566	N/A	256	36,358
LEA-128/128 [27]	94,104	31,588	71,736	N/A	388	47,401
HIGHT-64/128 [27]	57,558	16,144	40,540	N/A	228	14,058
CHAM-128/128 [27]	58,040	14,640	34,160	N/A	292	37,766

$\diamond$: Extrapolated result

are used for GIFT, PRESENT, PIPO and DEFAULT in this work than SPECK, LEA, HIGHT, and CHAM.

This becomes more evident when estimating full gates and full depth by decomposing the Toffoli gates. Table 5 shows a detailed analysis at the Clifford + T level for the quantum resources required for our DEFAULT quantum circuit and quantum circuits of other block ciphers. As described in Sect. 4.1, following the method of [1], the Toffoli gate is decomposed into 7 T gates + 8 Clifford gates, with a T-depth of 4 and a total depth of 8.

6 Grover's key search

In this part, the cost of Grover's key search for DEFAULT is estimated in detail. Grover's key search quantum circuit for block cipher consists of iterations of oracle and diffusion operators.Cost estimation for Grover's key search generally ignores diffusion operator and counts only quantum resources for oracle [20, 27, 29, 30, 35]. This is because the overhead of the diffusion operator is negligible, so the total cost is determined by the oracle.

Figure 6 shows the quantum circuit for the oracle of Grover's key search for DEFAULT. In oracle, plaintext 128-qubit $|m\rangle$ is encrypted with 128-qubit key $|k\rangle$ in the superposition state to generate ciphertext in the superposition state. 3 clean qubits (i.e., $|0\rangle$'s) are used to store round keys. In the middle of the oracle, it checks whether the generated ciphertext matches the known ciphertext. Since the overhead for this task is negligible, in our estimation, it is

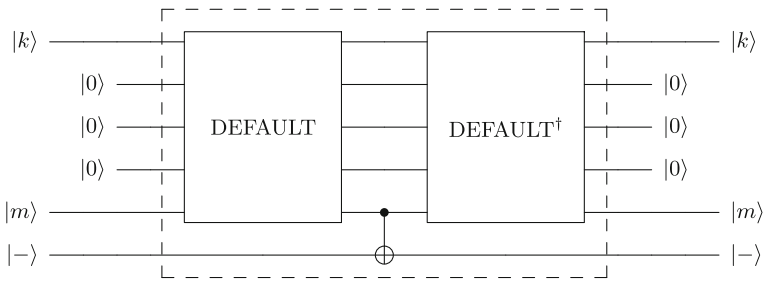


Fig. 6 Quantum circuit to run Grover's oracle on DEFAULT

Table 6 Quantum gates required for Grover's oracle

Source	#CNOT	#1qCliff	#T	T depth	#qubits	Full depth
DEFAULT [6]	125,952	24,790	114,688	2,048	257	4,582
DEFAULT [8]	153,600	26,350	125,440	2,240	641	4,994

excluded from the cost for simplicity. At the end of the oracle, in order to prepare for the next iteration, the reverse operation of DEFAULT is performed (i.e., DEFAULT†) to return to the initial state.

The quantum resources required for the sequential operation of quantum circuits DEFAULT and DEFAULT† are shown in Table 6, which is the oracle cost.

The cost of Grover's key search is calculated from the estimated oracle cost. The number of iterations required for Grover's key search for an n-bit search space is about $\sqrt{2^n}$. In [14], the authors suggested that the optimal number of iterations for the n-bit search space is $\lfloor \frac{\pi}{4}\sqrt{2^n} \rfloor$ (reduced) through tight analysis of the Grover's algorithm. The total cost of Grover's key search is estimated as Table 6 with multiplying by $\lfloor \frac{\pi}{4}\sqrt{2^{128}} \rfloor$ (the cost of the diffusion operator is not counted).

It should be pointed out that r plaintext-ciphertext pairs are needed to find the unique key (which is not a spurious solution for the key). The cost estimation of Grover's key search for block ciphers was first presented for AES-128, -192, and -256 in [20], and the authors set $r = 3, 4$, and 5. After that, [29, 30] suggest that $r = \lceil$key size/block size$\rceil$ is sufficient to find a unique key. We also estimate the cost of Grover's key search by setting $r = \lceil$key size/block size$\rceil$ for DEFAULT using 128-bit block and key (i.e., $r = 1$). Finally, the quantum resources required for Grover's key search for DEFAULT are shown in Table 7.

NIST has utilized the cost of Grover's key search to estimate post-quantum security strength for symmetric key cryptography. NIST has defined the post-quantum security level as follows according to the relative cost of quantum attacks that violate the security of AES-128, -192, -256:

- ◇ Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128-bit key (e.g., AES-128).
- ◇ Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 192-bit key (e.g., AES-192).
- ◇ Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 256-bit key (e.g., AES-256).

NIST focuses on the overhead of quantum gates and circuit depth due to numerous iterations of Grover's algorithm and considers the number of qubits less as they are not affected

Table 7 Quantum resources required for Grover's key search

Source	r	Total gates	Total depth	Cost	NIST security
DEFAULT [6]	1	$1.59 \cdot 2^{81}$	$1.757 \cdot 2^{75}$	$1.397 \cdot 2^{157}$	2^{170}
DEFAULT [8]	1	$1.83 \cdot 2^{81}$	$1.915 \cdot 2^{75}$	$1.752 \cdot 2^{157}$	

by iterations. In a nutshell, during the operation of Grover's algorithm, gates and depth are continuously increased, but the number of qubits is fixed. One thing that makes this clear comes from the observation that NIST estimates the cost for Grover's key search as the product of the total gates and total depth of the quantum circuit, excluding the number of qubits. By following Grover's key search by Grassl et al. work for AES [20]; the costs for Levels 1, 3, and 5 are estimated respectively as 2^{170}, 2^{233}, and 2^{298} search complexities. NIST recommends meeting Level 1 and/or Level 3 as it is likely to provide sufficient security for the foreseeable future.[4]

If we compare the cost of Grover's key search of DEFAULT (2^{157}) with the post-quantum security level of NIST, Level 1 (2^{170}) cannot be achieved. However, it should be pointed out that the estimated costs (from the work of Grassl et al. [20]) for the levels (Level 1: 2^{170}, Level 3: 2^{233}, and Level 5: 2^{298}) are considerably high, and the level is defined according to the relative attack cost for AES. This is evident from the significantly reduced cost of most recent quantum implementations of ciphers [2, 9, 13, 15, 21, 22, 24, 26, 27, 29, 30, 35].

NIST has noted that these preliminary classifications should be evaluated conservatively if the cost of best-known attacks is significantly reduced. Understandably, the quantum cost for AES family has been updated with the passage of time. Notably, the cost for AES-128 has been reduced in [30], followed by [29],[5] then [35], and finally in [24]. Based on [24], the current estimate for quantum security of AES-128 is of the order of 2^{157}. The estimated cost of DEFAULT being about the same, we conclude that it attains the post-quantum security at Level 1.

7 Conclusion

In this paper, we present a detailed implementation and analysis of the newly proposed block cipher, DEFAULT [6, 8]. Along with optimizations, we explore the possible vulnerability of DEFAULT against a quantum adversary. We show DEFAULT can be expected to meet the NIST-specified quantum security of Level 1. As one can expect this line of research only to grow with the passage of time, we are optimistic that our work would become useful for upcoming researchers.

Acknowledgements This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2022R1A6A3A13062701) for Kyungbae Jang. This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (<Q|Crypton>, No.2019-0-00033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity) for Hwajeong Seo.

Author Contributions All of the authors contributed to this paper in various ways, ultimately leading to the current version. Most of the contributions (implementation in quantum circuits, benchmark etc.) were done by the first author. Rest of the authors contributed in planning/preparation and writing/proofreading the manuscript.

Funding No funding information to declare.

Availability of supporting data Not applicable.

[4] See https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf for more details.

[5] However, it has been reported in [21, 24, 35] that their implementation contains Q# related bugs.

Declarations

Conflict of interest Not applicable.

Consent for publication We, the authors of the paper, consent this paper to be published.

Ethical Approval and Consent to participate We, the authors of the paper, hereby approve this paper as ready to submit/go through the review process.

References

1. Amy, M., Maslov, D., Mosca, M., Roetteler, 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)
2. Anand, R., Maitra, A., Mukhopadhyay, S.: Evaluation of quantum cryptanalysis on SPECK. In: Bhargavan, K., Oswald, E., Prabhakaran, M. (eds.) Progress in Cryptology - INDOCRYPT 2020, pp. 395–413. Springer International Publishing, Cham (2020)
3. Anand, R., Maitra, A., Mukhopadhyay, S.: Grover on SIMON. Quantum Information Processing. **19**(9) (2020). https://doi.org/10.1007/s11128-020-02844-w
4. Anand, R., Maitra, S., Maitra, A., Mukherjee, C.S., Mukhopadhyay, S.: Resource estimation of grovers-kind quantum cryptanalysis against fsr based symmetric ciphers. Cryptology ePrint Archive, Report 2020/1438 (2020). https://eprint.iacr.org/2020/1438
5. Baksi, A.: Classical and Physical Security of Symmetric Key Cryptographic Algorithms. PhD thesis, School of Computer Science & Engineering, Nanyang Technological University, Singapore (2021). https://dr.ntu.edu.sg/handle/10356/152003
6. Baksi, A.: Classical and Physical Security of Symmetric Key Cryptographic Algorithms. (2022). https://link.springer.com/book/10.1007/978-981-16-6522-6
7. Baksi, A., Bhasin, S., Breier, J., Jap, D., Saha, D.: Fault attacks in symmetric key cryptosystems. Cryptology ePrint Archive, Report 2020/1267 (2020)
8. Baksi, A., Bhasin, S., Breier, J., Khairallah, M., Peyrin, T., Sarkar, S., Sim, S.M.: Default: Cipher level resistance against differential fault attack. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021, pp. 124–156. Springer International Publishing, Cham (2021)
9. Baksi, A., Jang, K., Song, G., Seo, H., Xiang, Z.: Quantum implementation and resource estimates for rectangle and knot. Quantum Inf. Process. **20**(12), 395 (2021)
10. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: Gift: A small present. Cryptology ePrint Archive, Report 2017/622 (2017). https://eprint.iacr.org/2017/622
11. Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, Proceedings. 32–345 (2017) 25-28 Sept 2017
12. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In Kaliski, BurtonS., J., ed.: Advances in Cryptology - CRYPTO '97. Volume 1294 of Lecture Notes in Computer Science. Springer Berlin Heidelberg 513–525 (1997)
13. Bijwe, S., Chauhan, A.K., Sanadhya, S.K.: Quantum search for lightweight block ciphers: Gift, skinny, saturnin. Cryptology ePrint Archive (2020)
14. Boyer, M., Brassard, G., Hoyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik **46**(4–5), 493–505 (1998)
15. Chauhan, A.K., Sanadhya, S.K.: Quantum resource estimates of grover's key search on aria. In: International Conference on Security, Privacy, and Applied Cryptography Engineering, Springer 238–258 (2020)
16. Cuccaro, S.A., Draper, T.G., Kutin, S.A., Moulton, D.P.: A new quantum ripplecarry addition circuit. (2004) arXiv preprint quant-ph/0410184
17. Dasu, V.A., Baksi, A., Sarkar, S., Chattopadhyay, A.: LIGHTER-R: optimized reversible circuit implementation for sboxes. In: 32nd IEEE International Systemon-Chip Conference, SOCC 2019, Singapore, 260–265 (2019) Accessed 3-6 Sept 2019
18. Draper, T.G., Kutin, S.A., Rains, E.M., Svore, K.M.: A logarithmic-depth quantum carry-lookahead adder. arXiv preprint quant-ph/0406142 (2004)

19. Gidney, C., Ekerå, M.: How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. Quantum **5**, 433 (2021)
20. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: Quantum resource estimates. In Takagi, T.: (ed.): Post-Quantum Cryptography, Cham, Springer International Publishing 29–43 (2016)
21. Huang, Z., Sun, S.: Synthesizing quantum circuits of aes with lower t-depth and less qubits. Cryptology ePrint Archive, Report 2022/620 (2022). https://eprint.iacr.org/2022/620
22. Jang, K., Choi, S., Kwon, H., Kim, H., Park, J., Seo, H.: Grover on korean block ciphers. Applied Sciences **10**(18), (2020)
23. Jang, K., Kim, H., Eum, S., Seo, H.: Grover on GIFT. Cryptology ePrint Archive, Report 2020/1405 (2020). https://eprint.iacr.org/2020/1405
24. Jang, K., Baksi, A., Song, G., Kim, H., Seo, H., Chattopadhyay, A.: Quantum analysis of aes. Cryptology ePrint Archive, Paper 2022/683 (2022). https://eprint.iacr.org/2022/683
25. Jang, K., Choi, S., Kwon, H., Seo, H.: Grover on SPECK: Quantum resource estimates. Cryptology ePrint Archive, Report 2020/640 (2020). https://eprint.iacr.org/2020/640
26. Jang, K., Song, G., Kim, H., Kwon, H., Kim, H., Seo, H.: Efficient implementation of PRESENT and GIFT on quantum computers. Applied Sciences **11**(11) (2021)
27. Jang, K., Song, G., Kim, H., Kwon, H., Kim, H., Seo, H.: Parallel quantum addition for Korean block cipher. IACR Cryptol. ePrint Arch. 1507 (2021)
28. Jang, K., Song, G., Kwon, H., Uhm, S., Kim, H., Lee, W.K., Seo, H.: Grover on pipo. Electronics **10**(10), 1194 (2021)
29. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and lowmc. In Canteaut, A., Ishai, Y.: (eds.): Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, Proceedings, Part II. Volume 12106 of Lecture Notes in Computer Science., Springer 280–310 (2020). Accessed 10-14 May 2020
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 (01 2020)
31. Rahman, M., Paul, G.: Grover on katan: Quantum resource estimation. IEEE Transactions on Quantum Engineering **3**, 1–9 (2022)
32. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. Proceedings of the 35th Annual Symposium on the Foundations of Computer Science (1994)
33. Takahashi, Y., Tani, S., Kunihiro, N.: Quantum addition circuits and unbounded fan-out. (2009). arXiv:0910.2530
34. Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. Cryptology ePrint Archive, Report 2020/903 (2020). https://eprint.iacr.org/2020/903
35. Zou, J., Wei, Z., Sun, S., Liu, X., Wu, W.: Quantum circuit implementations of AES with fewer qubits. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2020, pp. 697–726. Springer International Publishing, Cham (2020)