Quantum Analysis of AES

Lowering Limit of Quantum Attack Complexity

Kyungbae Jang[1], Anubhab Baksi[2], Hyunji Kim[1], Gyeongju Song[1], Hwajeong Seo[1], and Anupam Chattopadhyay[2]

[1] Division of IT Convergence Engineering, Hansung University, Seoul, South Korea

[2] Temasek Laboratories, Nanyang Technological University, Singapore

starj1023@gmail.com, anubhab001@e.ntu.edu.sg, khj1594012@gmail.com, thdrudwn98@gmail.com hwajeong84@gmail.com, anupam@ntu.edu.sg

Abstract. Quantum computing is considered among the next big leaps in computer science. While a fully functional quantum computer is still in the future, there is an ever-growing need to evaluate the security of the secret-key ciphers against a potent quantum adversary. Keeping this in mind, our work explores the key recovery attack using the Grover's search on the three variants of AES (-128, -192, -256). In total, we develop a pool of 14 implementations per AES variant, by taking the state-of-the-art advancements in the relevant fields into account.

In a nutshell, we present the least Toffoli depth and full depth implementations of AES, thereby improving from Zou et al.'s Asiacrypt'20 paper by more than 98 percent for all variants of AES. We show that the qubit count - Toffoli depth product is reduced from theirs by more than 75 percent. Furthermore, we analyze the Jaques et al.'s Eurocrypt'20 implementations in details, fix the bugs (arising from some problem of the quantum computing tool used and not related to their coding) and report corrected benchmarks. To the best of our finding, our work improves from all the previous works (including the Asiacrypt'22 paper by Huang and Sun) in terms of various quantum circuit complexity metrics (such as, Toffoli depth, full depth, Toffoli depth - qubit count product, and so on).

Equipped with the basic AES implementations, we further investigate the prospect of the Grover's search. In that direction, under the MAXDEPTH constraint (specified by NIST), the circuit depth metrics (Toffoli depth, T-depth and full depth) become crucial factors and parallelization for often becomes necessary. We provide the least depth implementation in this respect, that offers the best performance in terms of metrics for circuit complexity (like, depth-squared - gate count product, depth-squared - qubit count product).

Keywords: Quantum Implementation · Grover's Search · AES

We thank Da Lin (Hubei University, Wuhan, PR China) for the kind support. This work presented in this paper won the grand award at the Cryptography Paper Competition (cryptography application and utilization category) organized by the South Korean Government, 2022 (https://kiisc.or.kr/bbs/nn/article/2147).

1 Introduction

In the current situation in the world of cryptography, quantum computers are considered an upcoming major threat. This is due to the innate nature of how the quantum computers can efficiently model and solve certain problems. There is an overlap between the problems efficiently solvable by a functional quantum computer and those act as the backbones to certain cryptographic systems. Those problems are hard to solve by a classical computer, hence considered secure as of now, but the security of those systems may be threatened if quantum computers become viable in the future. It is well-known that there will be severe consequence in the field of public key cryptography [26], still the secret key counterpart will likely not be completely unscathed either. Depending on the structure, a secret key cipher, too, can have severe security flaw against a quantum computer (refer to [21,35])[3].

One serious way for this to manifest arises from the observation that, a lot of the post-quantum ciphers use some secret key ciphers internally as a component in one way or the other (apart from the standalone usage of the secret key ciphers). This is evident from the current portfolio of the Post-Quantum Cryptography (PQC) standardization[4] being organized by the US government's National Institute of Standards and Technology (NIST). While the core components of ciphers are based on a problem presumed to be quantum-safe, due to the usage of secret key ciphers, it may be possible for the attacker to bypass the overall security claim (i.e., by exploiting only the secret key component). In other words, it may just so happen that the secret key component becomes the security bottleneck of the a post-quantum cipher (despite the core components being secure) against a potent quantum computer. Therefore, it is probably a commendable plan to consider the quantum security of the secret key ciphers, to be on the safe side.

Therefore, finding the generic quantum security level for a secret key cipher is among the top research directions (see Section 2.2 for related works). One of the main way an attacker with a functional quantum computer can try to mitigate the security of the secret key ciphers is by running the Grover's search algorithm [25]. As a rule of thumb, it reduces the time complexity of exhaustive key search to nearly the square-root bound (with a high probability).

Our work makes a detailed and systematic attempt to estimate the search complexity on the AES family of block ciphers (AES-128, AES-192 and AES-256) [16], thereafter finding the complexity for the Grover's search [25]. In the process, we revisit recent research works to incorporate state-of-the art advancements in various related areas (including those which are reported recently like [40,41,42,54]). Our objective lies in reducing the cost in various metrics (see Section 2.1 for an overview on the quantum gates); such as qubit count,

[3] However, it is to be mentioned that the quantum computers are the nowhere near to be considered a serious generic threat against the secret key ciphers (due to impractical resource requirement) as of yet, despite the paradigm growing in leaps and bound in the past few years.

[4] https://csrc.nist.gov/projects/post-quantum-cryptography.

gate count, circuit depth (Toffoli depth, full depth) and/or cost-depth trade-off (Toffoli depth × qubit count, full depth × qubit count, among other options). In the process, we carefully weigh and choose from a number of possible options.

Contribution and Organization

The prerequisite for this work is summarized in Section 2. In particular, the quantum gates are briefly described in Section 2.1, and previous literary works in Section 2.2.

We discuss in detail about the considerations/choices that are made during design separately for AES in Section 3 and architecture for combined components in Section 4.

We observe that the implementation by [34] contains some Q# programming issue, which probably results in underestimating the resources for non-linear components; although the linear components are not affected. We patch the issues (such as impossible parallelism and inconsistencies from reported quantum resources) and estimate the correct quantum gates and depth from the number of qubits in Section 5. It is to be noted that the same Q# issue was reported in the Asiacrypt'20 [59], Asiacrypt'22 [28] and Indocrypt'22 [31] papers. Also the authors of [34] confirmed it in a private communication.

Main results are consolidated in Section 6 (cost of the implemented quantum circuits) and Section 7 (cost for running the Grover's search). Comparison of our implementations with respect to the previous works are shown in Table 4 for the three variants of AES. Table 1 shows the overall performance gain of our work with respect to previous AES implementations. It can be seen that we make significant improvement over the Asiacrypt'20 paper [59] (such as our Toffoli depth TD is reduced by over 98% for AES-128) and also the bug-fixed version of the Eurocrypt'20 paper [34]. We also include the two implementations done in [28] for a quick comparison. In [28], the qubit count and Toffoli depth of the AES quantum circuit are determined by the number of parallel S-box implementations which is denoted by p — as p increases, the Toffoli depth decreases, but the number of qubits increases.

We develop multiple quantum implementations of the ciphers in the AES family (AES-128, AES-192 and AES-256), and report the least full depth FD (with moderate number of qubits M and quantum gates G) and cost-depth trade-off (TD-M; FD-M; and G-FD) implementations so-far. By increasing the number of qubits by a less quantity, we reduce the full depth greatly, so that the overall produce is significantly reduced. Moreover, this low depth is highly advantageous for reducing the cost when parallelization is required due to the depth limit in Grover's search (see Appendix B). Optimization is done at three levels, namely individual component level (S-box, MixColumn etc.), architecture level (16 S-boxes to make 1 SubBytes, 4 MixColumn to make 1 MixColumns etc.), and finally by sharing of resources among the modules. We present a pool of three implementations, each optimized for a specific objective (see Section 3.1 for related discussion):

1. The *regular* version uses the least qubit count and FD-M cost in our work, and reduces Toffoli circuit depth compared to the previous works for all the 3 variants. The MixColumn implementation is taken from [54], which allows zero ancilla/garbage qubit and incurs 92 CNOT gates.
2. The *shallow* version runs all parallel-executable parts of AES simultaneously, including reverse operations. The depth of one round only counts SubBytes + MixColumns, which is optimal. The shallow version takes the least qubit count and Toffoli depth product (TD-M cost) with an improved pipeline architecture. According to [59], this is an important notion of circuit complexity. Similar to the regular version, the MixColumn implementation is taken from [54].
3. Further, the *shallow/low depth* version looks for reducing the circuit depth by opting for a low quantum depth implementation of MixColumn (which was found by the authors of [40]). This version is optimal when parallelization of Grover's search is unavoidable under the constraints of depth.

Orthogonal to three architectural choices, we also use two S-box implementations (from [28]) that incur the Toffoli depth of 4 and 3 respectively. On top of that, we present two implementations of the bug-fixed version of Eurocrypt'20 [34] in Section 5 for AES-128, AES-192 and AES-256. These two versions differ based on whether the in-place MixColumn from [34] is used or the Maximov's MixColumn implementation from [43] is used (both were used in [34]). In order to keep the modification at minimum, we reuse the same design choices made in [34]. For this reason, we reuse the S-box implementation as in [34], which was adopted from [13].

In this work, we present 8 distinct implementations for each variant of AES:

1. Regular (MixColumn from [54]).
   (a) 3 Toffoli depth S-box [28].
   (b) 4 Toffoli depth S-box [28].
2. Shallow (MixColumn from [54]).
   (a) 3 Toffoli depth S-box [28].
   (b) 4 Toffoli depth S-box [28].
3. Shallow/low depth (MixColumn from [40]).
   (a) 3 Toffoli depth S-box [28].
   (b) 4 Toffoli depth S-box [28].
4. Bug-fixing [34] (S-box from [13]).
   (a) In-place MixColumn [34].
   (b) Maximov's MixColumn [43].

Further, except the bug-fixed version (for which only the Toffoli gate version is available), rest of the 3 versions are all done for the Toffoli gate as well as the AND gate. Combining all, we present 42 implementations altogether: 3 versions $\times$ 2 type of S-box $\times$ 3 AES variants $\times$ 2 type of gates (Toffoli and AND) + bug-fixed JNRV for 3 AES variants $\times$ 2 type of MixColumn.

About the novelty/new building block introduced in this paper, we would like to note the following points:

1. We optimize the depth of the components by using more ancilla sets (except in-place MixColumns) through parallelization. We reduce the depth while conserving the number of qubits by allowing for many ancilla qubits and reusing them in the next round through reverse operations.
2. We present a new idea for pipelining of operation (Figure 4(b)), which reduces the T-depth and full depth from the previous works (as in Figure 4(a)). This involves combining the previous round's reverse operation with the current round's operation by using two alternate ancilla sets.

Table 1: Performance comparison of AES quantum implementations.

	AES	Toffoli depth (TD)	Qubit count (M)	$TD \times M$	Full depth (FD)	$FD \times M$
128	GLRS [24]	12672 (99.76)	984 (−84.55)	12469248 (97.96)	110799 (99.12)	109026216 (94.29)
	LPS [39]	1880 (98.41)	864 (−86.43)	1624320 (84.32)	28927 (96.62)	24992928 (75.08)
	ZWSLW [59]	2016 (98.51)	512 (−91.96)	1032192 (75.32)	·	·
	HS [28] † 18	820 (95.12)	492 (−92.27)	403440 (30.86)	·	·
	HS [28] † 9	1558 (97.43)	374 (−94.13)	582692 (56.29)	·	·
	❁	2394 (98.33)	1656 (−74.00)	3964464 (93.58)	33320 (97.07)	55177920 (88.71)
	✿	40◎◈	6368◎	254720◎	978◎	6227904◎
192	GLRS [24]	11088 (99.68)	1112 (−83.37)	12329856 (97.34)	96956 (98.79)	107815072 (92.72)
	LPS [39]	1640 (97.81)	896 (−86.60)	1469440 (78.15)	25556 (95.41)	22898176 (65.71)
	ZWSLW [59]	2022 (98.22)	640 (−90.43)	1294080 (75.19)	·	·
	❁	2682 (98.21)	1976 (−70.46)	5299632 (93.94)	37328 (96.86)	73760128 (89.36)
	✿	48◎◈	6688◎	321024◎	1174◎	7851712◎
256	GLRS [24]	14976 (99.72)	1336 (−80.85)	20007936 (98.05)	130929 (98.95)	174921144 (94.51)
	LPS [39]	2160 (98.06)	1232 (−82.34)	2661120 (85.32)	33525 (95.89)	41302800 (76.74)
	ZWSLW [59]	2292 (98.17)	768 (−88.99)	1760256 (77.81)	·	·
	❁	3306 (98.31)	2296 (−67.09)	7590576 (94.85)	46012 (97.01)	105643552 (90.91)
	✿	56◎◈	6976◎	390656◎	1377◎	9605952◎

Parenthesized numbers show % (positive) improvement reported in this work.
†: Choice of p.

☆: Regular version (using Toffoli gate).
◎: Shallow version (using Toffoli gate).
◈: Shallow/low depth version (using Toffoli gate).
✿: S-box with Toffoli depth 4.
❁: Bug-fixed JNRV [34] (using Toffoli gate).

3. We propose two new structures (shallow, shallow/low depth). The shallow/low depth version has the advantage that the ancilla qubits for MixColumn can be taken for free (in the regular version, used in [34], ancilla qubits are not free when the Q# bug is patched).

As a consequence of our analysis, the state-of-the-art bounds of the quantum security level [45] is updated in Section 7. The cost for the Grover's search for each implementation can be observed from Table 7 (Table 7(a) with Toffoli and 7(b) with AND gates). We conclude in Section 8, where we present the other AES related quantum analysis with respect to the updated security level (refer to Figure 6 for a quick view).

Some additional information/discussion can be found in Appendices A (a short discussion on the AES variants), B (the Grover's search algorithm), C (discussion on the requirement/specification by NIST), D (a brief comparison of classical and quantum depths for the in-place XOR/CNOT implementation of linear layers), E (detailed discussion on the Eurocrypt'20 [34] bug), and F (per-round based break-up of quantum resources).

Our source codes are written in ProjectQ[5], which is a Python-based open-source framework for quantum computing. All our relevant source codes can be accessed online as an open-source project[6].

2 Background

2.1 Quantum Gate Basic

Throughout this paper, we use the following shorthand notations: #NOT (reversible NOT gate count), #CNOT (CNOT count), #Toffoli (Toffoli count), TD (Toffoli depth), #T (T-gate count), Td (T-depth), #1qCliff as Clifford gate count, #Measure (Measurement count), G (total gates), FD (full depth) and M (qubit count). The full depth is related to the execution time of circuits [10]. The importance of depth is also noted in NIST's post-quantum security requirements. In estimating the complexity of quantum attacks, NIST used only the number of gates and depth as metrics, not the number of qubits [45].

We optimize AES for quantum computers; keeping an eye on the qubit count, Toffoli depth and full depth. Further, we also consider the Toffoli depth $\times$ qubit count, the TD-M cost, and full depth $\times$ qubit count, the FD-M cost as metrics for trade-off. Our AES quantum circuits attain the least Toffoli and full depths, TD-M and FD-M costs, significantly contributing to the advancement of the state-of-the-art.

It can be stated that the Toffoli gate is decomposed in terms of the Clifford and T gates, the cost and depth of such a decomposition varies based on the method [3, 27, 48]. Further, a Clifford gate can refer to CNOT and 1qCliff gates. Also, the T-depth, an important factor in error correction, is determined by T gates when Toffoli gate is decomposed. After designing the quantum circuit, we need to decompose the Toffoli gates to estimate detailed quantum resources. In this paper, when estimating detailed quantum resources, the Toffoli gate is decomposed into (8 Clifford gates + 7 T gates), T-depth 4, and full depth 8 following one of the methods in [3].

Additionally, we adopt the quantum AND gates from [34]. This AND gate is decomposed into (11 Clifford gates + 4 T gates), T-depth 1, and full depth 8, and requires 1 ancilla qubit. The reverse of the AND gate which does the un-compute operation (i.e., AND† gate) is designed according to the measured value of the target qubit of the AND† gate. This AND† gate is counted as (7 Clifford gates + 1 Measurement gate) in resource estimation. Although not adopted in our work, there is another version of the AND gate [23] that does not require an ancilla qubit, but has a T-depth of 2.

We first use Toffoli gates to verify the simulation results of the implemented quantum circuit. Since ProjectQ allows classical simulation of Toffoli gates, we can verify test vectors for large-scale quantum circuits. A Toffoli gate can be simulated classically and decomposed only when estimating resources. On the

[5] Homepage: https://projectq.ch/.

[6] https://github.com/starj1023/AES_QC.

other hand, classical simulation of AND gates is not supported. Therefore, we adopt a method of verifying the implemented quantum circuit using Toffoli gates and then replacing the top part with AND gates to estimate resources.

2.2 Related Work

Quantum analysis of secret-key ciphers with respect to the Grover's search algorithm is one of the major research direction now-a-days. Some of the prominent examples include, but not limited to, AES [11,34,39,59], SIMON [6], SPECK [5,29], PRESENT and GIFT [32], SHA-2 and SHA-3 [2], FSR-based ciphers [4], ChaCha [9], SM3 [49,57], RECTANGLE and KNOT [7], DEFAULT [30], ARIA [15], few Korean ciphers [33,37], SPECK and LowMC [31].

In comparison with the recent work by Huang and Sun (Asiacrypt'22) [28], we note the following points. Our approaches are mostly disjoint from that of [28]; and when their S-box implementation is used in our implementation, our result outperforms theirs (thus we have the best-known implementation so far). As it can be seen from Table 1, our results are indeed better than those are reported in [28]. Further, we cover optimized quantum implementations of AES-192 and AES-256 as well.

3 Components for Quantum Circuits for AES

3.1 Regular, Shallow and Shallow/Low Depth Versions

Our quantum circuit implementations are divided into regular and shallow versions. The regular version offers high parallelism while taking into account the trade-off of depth-qubit. The regular version has the best performance for FD-M cost (which is the full depth - qubit count product. The shallow version also considers the trade-off of qubit-depth, but further reduces the depth by burdening the use of qubit. The shallow version has the best performance in terms of Toffoli depth and TD-M cost (which is the Toffoli depth - qubit count product). The shallow/low depth version seems to achieve the lowest depth for quantum circuit implementation. The shallow/low version is the optimal choice when parallelization of the Grover's search is essential due to the depth limit.

The regular version focuses on the parallelism within the round. In this version, while the current round awaits, the previous round goes through the reverse (i.e., un-compute) operation. In other words, the next round cannot start until the reverse operation on the current round is complete.

On the other hand, the shallow version manages to parallelize the processing for all the rounds. In this version, the reverse operation of the previous round is run simultaneously with the current round, alternating between the even and the odd rounds (for instance, while the even rounds are at compute operation, the odd rounds are at the un-compute operation). This version uses more qubits, but offers lower depths, because all the rounds of the parallelizable parts of the cipher run simultaneously. As a consequence, it achieves lower circuit depth, as in

this case the bottleneck of the depth is that of the SubBytes plus MixColumns in every round (except for the last round where MixColumns depth is not counted).

That said, one may notice that the depth can be reduced if a different implementation of MixColumn is opted for, though the Toffoli depth is unchanged. In our shallow version, we choose the MixColumn implementation from [54], as it offers in-place implementation. As pointed out in Table 3, it is possible to lower the depth at the expense of more qubits, if the MixColumn implementation from [40] is chosen instead. Thus, everything else being inherited from the shallow version, the shallow/low depth version achieves lower full depth.

Most papers implementing quantum circuits for AES focus on reducing the usage of qubits [1, 24, 28, 39, 52, 59]. However, the serial circuit structure (which aims at reducing the number of qubits) significantly increases the circuit depth. As stated already, our quantum circuits for AES attempt to find the best possible balance between the number of qubits required with its relation to increment of the circuit depth. Thanks to the careful choices, our AES quantum circuits provide arguably the best trade-offs in terms of TD-M cost by varying TD and M, where recall that TD is the Toffoli depth and M is the number of qubits. This product is taken as the trade-off indicator for the quantum circuit in [59]. We also use the depth - qubits count product in estimating the FD-M cost. This metric is also realistic and is used primarily for evaluation. Our AES quantum circuit achieves the best trade-offs even in terms of FD-M cost.

3.2 Implementation of S-box (SubByte)

Table 2 shows the resources required for the implementations found by Boyar-Peralta [12,13] and the resources for the S-boxes used by the previous authors [39, 59]. Resource estimation is performed in ProjectQ and according to the method of [3], one Toffoli gate is decomposed into (8 Clifford gates + 7 T gates), and T-depth of 4, and full depth of 8. For the cost comparison and implementation details in Section 3, we use only the Toffoli gate. If we adopt the AND gate instead of the Toffoli gate, an ancilla qubit is required, but it can be saved depending on the overall structure. Thus, the cost of the AND gate version is estimated in Section 6 by replacing the Toffoli gates at the top of the implemented AES quantum circuits with AND gates.

Note that the S-box implementation in [24] is based on a field inversion technique, while the rest are based on some version of the Boyar-Peralta's algorithm [12, 13]. Apart from these, another method which is a courtesy of Dansarie [17, 18] exists. This is rather generic, as it can find implementation of an arbitrary 8-bit S-box (i.e., unlike [12, 13], this is not specific to the AES S-box), with respect to a user-provided set of logic gates. In total, we found 5 implementations in which the number of lines in the C source files is in the ballpark of 400 (it contains AND, OR and NOT gates; and sometime one line contains more than 1 gate). These are not used in this work due to high quantum cost (see Table 2 for the benchmarks).

If the Boyar-Peralta's S-box implementations [12, 13] are directly ported to quantum, then the version of [13] requires more ancilla qubits (120 ancilla qubits)

Table 2: Comparison of quantum implementations of AES S-box.

Method		#CNOT	#1qCliff	#T	TD	M	Full depth
S-box [24]		1818	124	1792	88	40	951
S-box [12]		358	68	224	8	123	104
S-box [13]	✤	392	72	238	6	136	85
S-box [39]		628	98	367	40	32	514
S-box [59]		437	72	245	55	22	339
S-box [17, 18]	391 lines	1470	670	1218	66	399	640
	406 lines	1507	548	1245	74	414	709
	413 lines	1484	561	1169	62	421	591
	409 lines	1483	574	1190	74	416	693
	400 lines	2244	1006	2254	111	408	998
S-box [28]	✿	418	72	238	4	136	72
	❀	824	160	546	3	198	69

✤: Reused in this work to fix [34].
✿: Used in this work (Toffoli depth 4).
❀: Used in this work (Toffoli depth 3).

than the quantum version of [12] (107 ancilla qubits), but attains lower depth. JNRV adopted the implementation of the S-box of [13] on a quantum circuit [34] as-is.

Recently, Huang and Sun reported an improved quantum implementation for the S-box of [34] in their Asiacrypt'22 paper [28]. They presented two quantum implementations of reduced Toffoli depth with new observations of the classical implementation of the AES S-box as given in [13]. The first version reduced the Toffoli depth without increasing the number of qubits, while the second version used more qubits to further reduce the Toffoli depth.

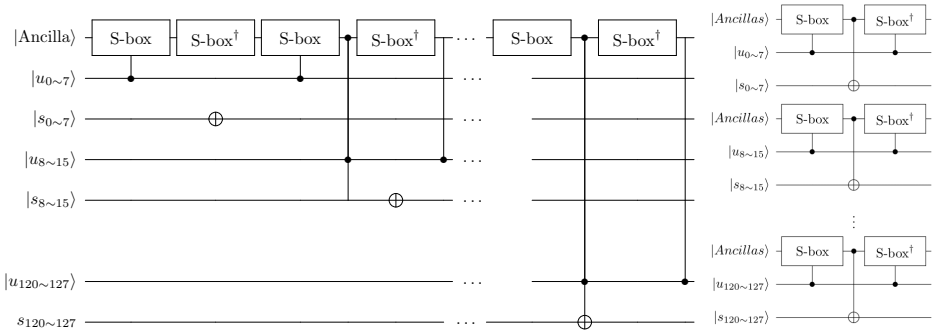
In [39, 59], the authors extended the first S-box implementation by Boyar-Peralta [12] and presented the S-box quantum circuit with a reduced number of qubits. Consequently, it leaves us with a few of ways to choose from.

Considering the trade-off between the circuit depth and the number of qubits required for an S-box implementation, we treat two cases. The first case is when the ancilla qubits have to be allocated per SubBytes, which is indeed sensitive to the number of qubits. The second case is when the initially allocated ancilla qubits can be reused. In this case, there is no need to allocate additional ancilla qubits for the next SubBytes. Therefore, the number of ancilla qubits is maintained, but the depth and number of gates increase due to the reverse operations needed to reuse the ancilla qubits. We choose the second case for our SubBytes implementation, since we believe the benefit of reducing the number of qubits outweighs the cost saved by not performing additional reverse operations. In this case, only the initial allocation is burdened because the ancilla qubits are reused. Thus, we use Huang and Sun's [28] S-box implementations with relatively high qubit count but low depth. That is, we increase the initial burden and use fast (low depth) S-boxes for free (without ancilla qubits) until the end.

One may note that the AES implementation in [59] required the implementation of the inverse S-box. In our case, however, we do not use the inverse S-box.

3.3 Implementation of SubBytes

After we decide upon the implementation of one S-box (SubByte, Section 3.2), this can be used to implement 16 S-boxes (SubBytes). Regarding the implementation of SubBytes in AES, Figure 1(a) shows the method that uses the fewest qubits. In this case, all S-boxes are executed sequentially, which causes a significant increase in depth, as shown in Figure 1(a). On the other hand, we reduce the depth by allocating more ancilla sets initially. The notation S-box† is described in Appendix A.3.



(a) Using one ancilla set.

(b) Using multiple ancilla sets.

Fig. 1: SubBytes implementation in quantum.

In one round, 16 S-boxes in SubBytes and 4 S-boxes in key schedule, a total of 20 S-boxes are operated, simultaneously. Therefore, we allocate 20×120 ancilla qubits for S-boxes with Toffoli depth 4 and 20×182 ancilla qubits for S-boxes with Toffoli depth 3 to run all S-boxes simultaneously. Figure 1(b) shows 16 S-boxes operation in parallel using multiple ancilla sets. After S-box operations, ancilla qubits are not in a clean state (i.e., not all ancilla is 0). Initialization with 16 S-boxes† operation (i.e., returning to 0) is performed in parallel for the next round. Thanks to this, we can reuse the initialized ancilla qubits in the next round of SubBytes. Of course, these reverse operations save qubits, but increase depth. However, if we allocate ancilla qubits each time by skipping reverse operations, it is an abuse of qubits. We consider these trade-offs carefully and the shallow version offsets this depth overhead from reverse operations (this will be described in Section 4.2).

In [59], 16 S-boxes of SubBytes were implemented in parallel using residual ancillas, but key schedule was not implemented in parallel with SubBytes.

3.4 Implementation of Key Schedule

In the key schedule of AES, SubWord operates on rearranged 32-qubit. Out of the $20 \times (120 \text{ or } 182)$ ancilla qubits previously decided to use (refer to Section 3.3), $4 \times (120 \text{ or } 182)$ ancilla qubits are used to simultaneously operate S-boxes for 32-qubit in the key schedule (16×120 or 16×182 ancilla qubits are used in SubBytes of round). For rearranging the 32 qubits, quantum resources are not used by using logical swap that only changes the index of the qubits.

In SubBytes, the outputs of S-boxes are stored in new qubits. On the other hand, in the key schedule, no additional qubits are allocated because the outputs of the S-boxes are XORed (using CNOT gates) inside the key. Since SubWord for 32-qubit operates in parallel with SubBytes of round, there is no depth overhead in our AES quantum circuit implementation. This approach is already utilized in [34]. XORing the 8-bit round constant (RC) is implemented by performing X gates to $|k_{120\sim127}\rangle$ according to the positions where the bit value of the round constant is 1. Lastly, the CNOT gates inside the key are performed. Figure 2 shows the quantum circuit for the AES-128 key schedule (see Appendix A.3 for description of Rotation† and SubWord†).

All the S-boxes in key schedule and round function are designed to operate in parallel. That is, the depth is the same as operating an 8-bit S-box once. Quantum implementation for S-box is required for key schedule and SubBytes, and S-box occupies the most resources in AES quantum circuit. In [24], GLRS used Itoh–Tsujii inversion to implement S-box of AES, which requires a lot of quantum resources. Recently, the hardware design for AES has been adopted to implement an efficient S-box quantum circuit. In particular, S-box implementation techniques [12, 13] proposed by Boyar-Peralta were frequently used. In [39], Langenberg et al. adopted the S-box implementation of [12] and converted it to suit their purpose of reducing qubits. The S-box implementation of [12] was adopted and improved in [58]. ZWSLW [59] also used the S-box^{-1} implementation in designing a new architecture for AES that reduced number of qubits. For the key schedule, an on-the-fly approach is adopted, and our AES quantum circuit implementation executes the key schedule simultaneously with SubBytes in the round function.

In most implementations of AES quantum circuits, the full depth and Toffoli depth of AES-128 are higher [24, 34, 39] or similar [59] to those of AES-192. Although AES-128 has fewer rounds, this is due to differences in key schedule. AES-128 requires 16 S-boxes for SubBytes and 4 S-boxes for key schedule in every round. On the other hand, some rounds of AES-192 require only 16 S-boxes for SubBytes, since SubWord in the key schedule are not required. As a result, AES-128 has a higher depth than AES-192.

Another interpretation of this is that there is a depth overhead for key schedule in implementing AES quantum circuits. However, in our AES quantum circuits there is no depth overhead for key schedule (there is overhead for gates and

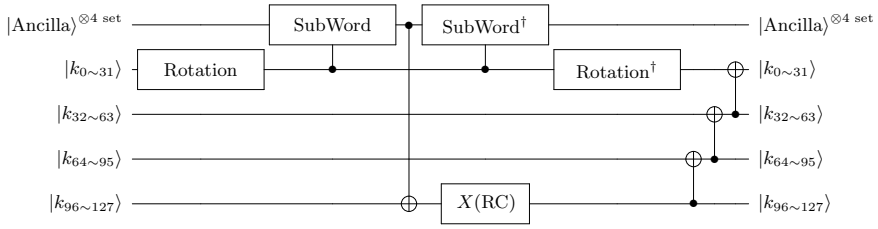


Fig. 2: AES-128 key schedule in quantum.

ancilla qubits). Our AES quantum circuit runs the key schedule in complete parallel, so we achieve the same depth as if the key schedule was omitted. As a result, unlike other implementations, the quantum resources required for our AES-128, -192, and -256 quantum circuits are strictly dependent on the number of rounds.

3.5 Implementation of AddRoundKey and ShiftRows

The AddRoundKey operation, which XORs a 128-qubit round key, can be implemented simply by using 128 CNOT gates. In the case of ShiftRows, it can be implemented using swap gates, but quantum resources are not used through logical swap that changes the index of qubits. Since no special implementation technique is applied for AddRoundKey and ShiftRows, this approach is mostly used in quantum circuit implementations.

3.6 Implementation of MixColumn

In [54], Xiang et al. presented a novel heuristic search algorithm to optimize the implementation of linear layers based on factorization of binary matrices. When applied to the MixColumn of AES, their algorithm resulted in an implementation using 92 XOR gates (with classical depth 6) in a classical circuit. A different implementation costing 92 XOR gates (with classical depth 6) was reported previously by [43]. These two were the least cost implementations in classical circuits, until another implementation with 91 XOR gates (with classical depth 7) was found by [41]. Recently, a new implementation of AES MixColumn was found thanks to [42], which managed to reduce the classical depth to 3 with 103 XOR gates (cf. 103 XOR/3 classical depth implementation from [8]). However, this work came as a tie with another implementation from [40], albeit the latter required 105 XOR gates.

When it comes to quantum implementation, one may observe that the following implementations operate in-place (i.e., of the form $a \leftarrow a \oplus b$ and require only 32 qubits):

(a) PLU factorization in some form (used in [1, 24, 34, 59]);
(b) 92 XOR implementation reported in [54] (used in [28]).

Note from Table 3 that, the implementation by [54] requires the least number of XOR/CNOT gates. This hugely improves from the previous in-place implementations based on the PLU factorization [24,34,59][7]. In contrast, implementations like that of [41,42,43], do not work in-place, due to the require usage of temporary variables (i.e., ancilla/garbage qubits) and/or depth (due to cleaning up qubits). On a different direction, the implementation from [40] appears to have lower depth than that of [42] when converted to quantum circuits. Related discussion can be found in [47].

We port the implementation of MixColumn in [54] to quantum and use it in our AES quantum circuit. This implementation is used in the regular and shallow versions. Additionally, in order to minimize the circuit depth, we also use the MixColumn implementation from [40] in our shallow/low depth version.

Table 3: Comparison of quantum implementations of AES MixColumn.

Method		#CNOT	#qubit (M)	Depth
MixColumn (Naïve)	GF(2^8)	184	64	25
	GF(2)			52
MixColumn [24,59]†		277	32	39
MixColumn [38]		194	129	15
MixColumn [1]†		275	32	200
MixColumn [43]✛		188	126	13
MixColumn [34]✚†		277	32	111
MixColumn [50]		188	126	17
MixColumn [54]☆◎ †		92	32	30
MixColumn [41]✳		182	123	16
MixColumn [42]✶		206	135	13
MixColumn [8]	103 XOR/3 depth	206	135	11
	95 XOR/6 depth	190	127	15
MixColumn [40]✶◈		210	137	11

✛✚: Reused in this work to fix [34] ❄.
☆◎: Used in regular and shallow versions; in [28].
✳: Least XOR count in classical circuit.
✶: Least depth in classical circuit.
◈: Used in shallow/low depth version.
†: In-place implementation.

The authors of [8] presented two implementations (103 XOR/3 classical depth, and 95 XOR/6 classical depth). If taken as-is, the 103 XOR/3 classical depth implementation yields 206 CNOT gates, 135 #qubits, with 11 quantum depth when ported. Thus, it is in theory possible to slightly improve our shallow/low depth version by switching to this implementation. Further, if the 95 XOR/6 classical depth implementation is ported as-is; it incurs 190 CNOT gates with 127

[7] In the Eurocrypt'20 paper [34], the authors remarked that they could not reproduce the result from [24].

#qubits with depth 15; however we could not verify the results (probably due to an encoding issue). Second, an implementation of 108 XOR count is mentioned in [22, Footnote 3/Page 42], but it is not clear to us so far.

Apart from the specialized MixColumn implementations just narrated, it is perhaps worth noting that the naïve quantum implementation (i.e., directly porting the matrix to quantum circuit) was seemingly never studied. With our implementations, one as a 4×4 matrix over GF(2^8), and the other as a 32×32 binary matrix; we notice from Table 3, the CNOT count being the same, that the depth varies.

3.7 Implementation of MixColumns

For the 128-bit MixColumns operation (i.e., 4 MixColumn operations), the MixColumn implementation can be scaled up directly. As the MixColumn used in the regular and the shallow versions work in-place, we do not have to consider the impact of ancilla qubits. This, however, is more complicated in case of the shallow/low depth version, as described next.

In the shallow/low depth version, we need to account for the ancilla qubits (since the implementation [40] is not in-place). This observation although hints that we need extra qubits (to work as ancilla), here we show how this is not the case. Recall from the implementation of SubBytes (Section 3.2 and Section 3.3) the S-box implementation is also not in-place, requiring ancilla qubits (20×120 or 20×182). Those ancilla qubits are initialized as 0 after one SubBytes operation (to use in the next round). Therefore, during the MixColumns operations those qubits are idle. As we only need 64 qubits to implement the MixColumn from [40] (32 as input plus 32 as output qubits), those idle qubits are reused. Thus, even though the MixColumn implementation is not in-place, at the end, we do not need any extra qubit. So, the qubit count does not increase when SubBytes is counted within the scope.

In other words, the total number of qubit requirement is 64 for any implementation in Table 3 (save for the in-place implementations [1,54] where it is 32) when the non-standalone implementation of MixColumns (in which MixColumn does not operate in-place) is considered. However, when the combined SubBytes and MixColumns is considered, because of efficient resource sharing, the total qubit count does not increase, also the full depth does not increase.

4 Architecture of AES Quantum Circuits

A combined description of the AES quantum circuits for all the 3 versions is presented here. There are several architectures for designing quantum circuits of AES. The architectures differ in how they store the 128-qubit output generated from SubBytes in each round. In [1,24,39], the basic zig-zag architecture (Figure 3(a)) was adopted that uses 4 lines to save qubits by performing reverses on rounds. In [59], an improved zig-zag architecture that requires only 2 lines of qubits (Figure 3(b)) was presented. By using a quantum circuit of S-box^{-1}, they were

able to achieve an improved architecture using fewer qubits. The basic pipeline architecture allocates 128-qubits every round and does not need reverses of rounds. Simply put, the zig-zag architecture requires reverse operations on rounds to save qubits, significantly increasing depth and gates. The pipeline architecture allocates new qubits per round, but does not require reverse operations, reducing depth and gates. It is a trade-off issue, but in a sense, a generic pipeline is probably the most efficient architecture for implementing AES quantum circuits. We believe that it is much more efficient to allocate a new 128-qubits per round than doubling the gates, depth by performing reverse operations on the rounds to save qubits.

In our approach, where we have allocated many ancilla qubits already, the overhead of increasing the number of qubits according to the architecture is relatively low. Therefore, for our implementation, rather than reducing the number of qubits with the zig-zag method, a pipeline architecture that can reduce the depth by omitting the reverses is more suitable. Figure 4(a) shows the pipeline architecture of our AES-128 quantum circuit in more detail for the regular version, and Figure 4(b) shows the same for the shallow and shallow/low depth versions. To be more precise, each $R_{1\sim10}$ in Figure 3 represents the full round, but each $R_{1\sim10}$ in Figure 4 does not contain SubBytes.

Table 4: Comparison of quantum resources required for variants of AES.

	AES		#CNOT	#NOT	#Toffoli	TD	#qubits	TD-M cost ($TD \times M$)	Full depth	TD^2-M cost ($TD^2 \times M$)
128	GLRS [24]		166548	1456	151552	12672	984	12469248	110799	158010310656
	ASAM [1]		192832	1370	150528	·	976	·	·	·
	LPS [39]		107960	1570	16940	1880	864	1624320	28927	3053721600
	ZWSLW [59]		128517	4528	19788	2016	512	1032192	·	2080899072
	HS [28]	† 18	126016	2528	17888	820	492	403440	·	330820800
		† 9	126016	2528	17888	1558	374	582692	·	907834136
	☆		84120	800	12920	76	3936	299136	1364	22734336
	◎	✿	81312	800	12240	40	6368	254720	978	10188800
	◈		90816	800	12240	40	7520	300800	799	12032000
	☆		138080	800	29640	57	5176	295032	1307	16816824
	◎	❀	132432	800	28080	30	8848	265440	948	7963200
	◈		141936	800	28080	30	10000	300000	769	9000000
192	GLRS [24]		189432	1608	172032	11088	1112	12329856	96956	136713443328
	LPS [39]		125580	1692	19580	1640	896	1469440	25556	2409881600
	ZWSLW [59]		152378	5128	22380	2022	640	1294080	·	2616629760
	☆		96112	896	14688	92	4256	391552	1627	36022784
	◎	✿	92856	896	14008	48	6688	321024	1174	15409152
	◈		104472	896	14008	48	8096	388608	955	18653184
	☆		157456	896	33696	69	5496	379224	1558	26166456
	◎	❀	151360	896	32136	36	9168	330048	1138	11881728
	◈		162976	896	32136	36	10576	380736	919	13706496
256	GLRS [24]		233836	1943	215040	14976	1336	20007936	130929	299638849536
	LPS [39]		151011	1992	23760	2160	1232	2661120	33525	5748019200
	ZWSLW [59]		177645	6103	26774	2292	768	1760256	·	4034506752
	☆		117704	1103	18088	108	4576	494208	1907	53374464
	◎	✿	113744	1103	17408	56	6976	390656	1377	21876736
	◈		127472	1103	17408	56	8640	483840	1118	27095040
	☆		193248	1103	41496	81	5816	471096	1826	38158776
	◎	❀	186448	1103	39936	42	9456	397152	1335	16680384
	◈		200176	1103	39936	42	11120	467040	1076	19615680

†: Choice of p.

☆: Regular version (using Toffoli gate).
◎: Shallow version (using Toffoli gate).
◈: Shallow/low depth version (using Toffoli gate).

✿: S-box with Toffoli depth 4.
❀: S-box with Toffoli depth 3.

4.1 Regular Version

In our parallel design, the key schedule operates simultaneously with SubBytes and MixColumn operates simultaneously with SubBytes†. Therefore, the circuit depth is determined by the number of serial operations of SubBytes and SubBytes†.

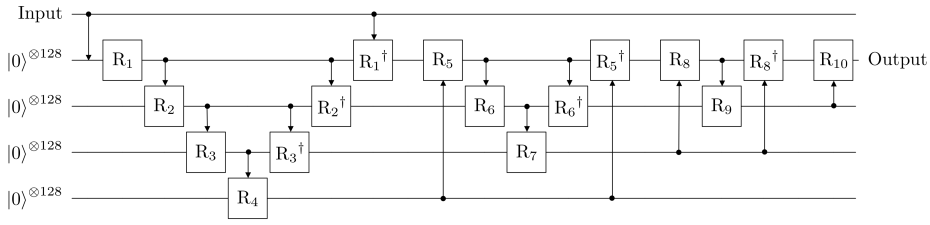
As shown in Figure 4(a), SubBytes generates 128-qubit output and SubBytes† cleans the ancilla qubits. In total, SubBytes runs 10 times and SubBytes† runs 9 times (as it is redundant to clean the last round SubBytes) serially, 19 times in total. Similarly, AES-192 operates 23 times (12 SubBytes plus 11 SubBytes†) and AES-256 operates 27 times (14 SubBytes plus 13 SubBytes†).

In SubBytes, S-boxes operate simultaneously. The depth of SubBytes is 72 equal to the depth of S-box (with Toffoli depth 4) once. Finally, when S-box with Toffoli depth 4 is used, our AES quantum circuits provide a depth of 1364 (about 72×19) for AES-128, 1627 (about 72 × 23) for AES-256, and 1907 (about 72 × 27) for AES-256.

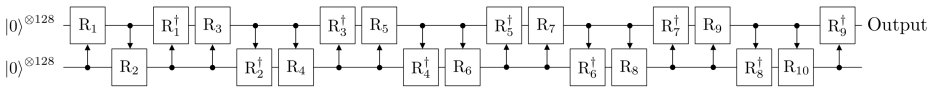
4.2 Shallow Version and Shallow/Low Depth Version

Further, we propose a shallow version in which all possible parts of AES quantum circuits operate, simultaneously. When S-box with Toffoli depth 4 is used, this can be achieved by using 2 sets of 20 × 120 ancilla qubits. In the shallow version, the first SubBytes in Figure 4(b) uses the first 20 × 120 ancilla qubits. The second SubBytes uses the second 20 × 120 ancilla qubits, and at the same time SubBytes† cleans the first 20 × 120 ancilla qubits. That is, SubBytes† operates simultaneously with the SubBytes of the next round. Conceptually, this can be thought as all SubBytes† in Figure 4(a) are pushed one space to the right. This is possible because SubBytes and SubBytes† do not share any ancilla qubit. The shallow version counts the depth for one round as SubBytes (72) + MixColumns (30), which is the ideal depth. The circuit depth of AES-128 is 978 (about 9 rounds × 102 + 72), that of AES-192 is 1174 (about 11 rounds × 102 + 72), and the same for AES-256 is 1377 (about 13 rounds × 102 + 72). In the shallow version, up to SubBytes† operates concurrently within one round, providing maximum parallelism. Finally, the shallow and shallow/low depth versions offer the least Toffoli depth of the S-box's Toffoli depth × rounds and Toffoli depth × qubit count.

The shallow/low depth version replaces only the MixColumn implementation from the shallow version to a MixColumn which is a courtesy of [40]. The low depth version counts the depth for one round as SubBytes (72) + MixColumns (11). The low depth version of AES offers the least Toffoli depth and full depth.



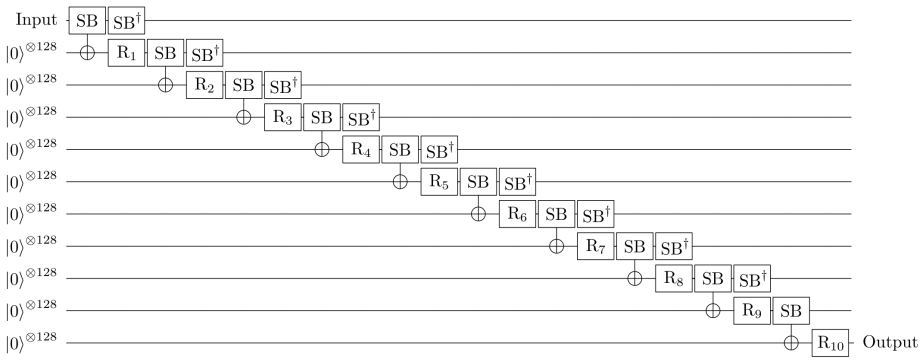
(a) Basic (GLRS, LPS, ASAM).



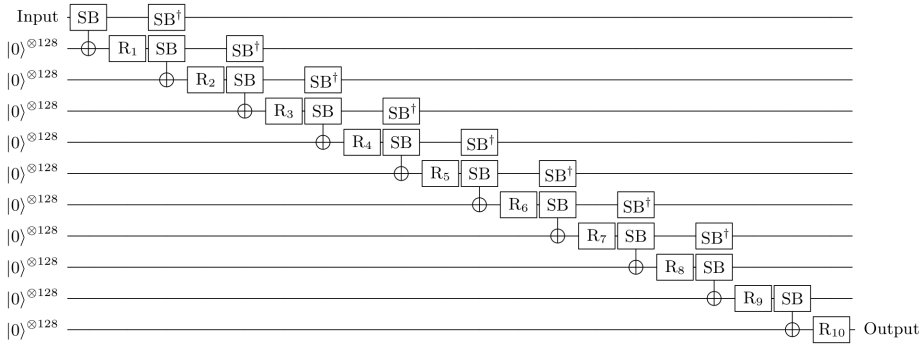
(b) Modified (ZWSLW).

Fig. 3: Zig-zag architecture for AES-128 quantum circuit.

SB: SubBytes. SB†: Clean ancilla qubits used in preceding SubBytes.



(a) Regular version (JNRV).



(b) Shallow and shallow/low depth versions (Ours).

Fig. 4: Pipeline architecture of AES-128.

5 Bug-fixing JNRV (Eurocrypt'20) AES Implementation

In this part, we take a deeper look at the AES implementation and resource estimation by Jaques, Naehrig, Roetteler and Virdia in Eurocrypt'20 [34]. It is already well-known the resource estimation in their paper was incorrect due to some problem in Q# (unrelated to the coding of [34]), as already noted by at least two previous works [28,59] as well as acknowledged by the Eurocrypt'20 authors[8] themselves. Also, one may refer to Appendix E for supplementary discussion on this topic. We fix the Q# bugs and report the corrected benchmarks for the resource requirement of [34] by porting their codes to ProjectQ.

[8] See https://github.com/microsoft/qsharp-runtime/issues/1037 and https://github.com/sam-jaques/grover-blocks/tree/sjaques-version-update#issue-with-estimating-resources.

5.1 Issues with Q#

For a clearer context, we give a brief description of the cases where Q#'s `ResourcesEstimator` issues arise and how those issues affect the quantum benchmarks given in the Eurocrypt'20 paper [34]. This was discovered when we tried to cross-check their publicly available source codes[9]. Indeed, this was also noted in [59] as a bug; and this apparently led to underestimation of gate count, qubit count and depth reported in [34] for the non-linear components (namely the S-box and S-box† of AES).

To our understanding, some problems arise if the qubits are allocated by the `using` command in Q# (and it affects the non-linear components). However more experiments are to be carried out in order to be completely certain about it.

Non-parallelizable SubBytes In their implementation, the S-box of [12] is adopted and ported to the quantum domain. The quantum resources required for the S-box quantum circuit reported in the Eurocrypt'20 paper [34, Table 1] are only correct for the stand-alone S-box (except for T-depth, this will be described in Section 5.1). However, in the case of SubBytes operating with 16 S-boxes, incorrect quantum resources are reported. This is a major cause of their resource estimation issues.

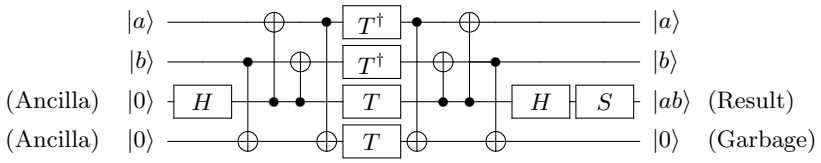
According to the reported number of required qubits, only one ancilla set is used in their SubBytes implementation. In other words, 16 S-boxes share one ancilla set. Thus, the arrangement of qubits in their SubBytes quantum circuit is the serial structure of Figure 1(a). Since 16 S-boxes generate each output using one ancilla set, all S-boxes in a limited space (one ancilla set) must be operated sequentially. However, in their report, the depth of the SubBytes is the same as the depth for a stand-alone S-box (meaning all S-boxes operate in parallel). That is, it is an impossible quantum circuit structure and the lower-bound depth is reported. The same error applies to the SubWord implementation of key schedule.

Issue with AND Gate This issue is also found in their use of AND gates. Suppose that 5 Toffoli gates are operated in parallel during the S-box process. Toffoli gates (the method used in [3]) operate in parallel without any additional work, providing one Toffoli depth and full depth for one Toffoli gate. On the other hand, in the AND gate of Figure 5(a), one garbage qubit (bottom line in Figure 5(a)) is used. Thus, if replaced with AND gates, 5 garbage qubits for 5 AND gates must be allocated for parallel operation. Note that, the garbage qubit of the AND gate is initialized to 0 after operation and can be reused in the next AND gate, but a sequential operation is forced.

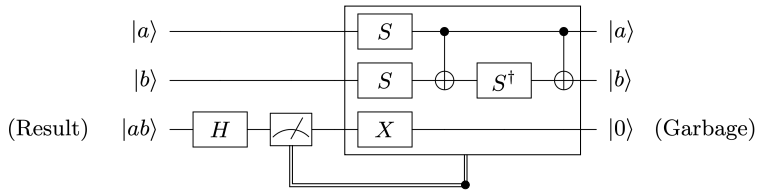
In a nutshell, in their S-box (out of 137 qubits, 136 qubits for the S-box and 1 qubit for the AND gate application), only one ancilla qubit is used for one AND gate. However, quantum resources for parallel operations are reported. Technically speaking, the ancilla qubits required for the AND gates can be replaced with

[9] https://github.com/microsoft/grover-blocks.

idle state qubits in the S-box operation, but this was not considered in their implementation.



(a) AND gate.



(b) AND† gate.

Fig. 5: Quantum AND and AND† gates in JNRV (Eurocrypt'20).

Inconsistency and Underestimation of Full Depth In their AES quantum circuits using Maximov's MixColumn, the AES-192 quantum circuit offers the lowest full depth (see Table 5(b)), although the number of rounds of AES-192 (12 rounds) is higher than that of AES-128 (10 rounds). This case is observed in the zig-zag architecture [24,39] since the number of key schedules is less in AES-192. However, as a result of analyzing their quantum circuit design (e.g., pipeline and parallel structure) and quantum resources, the full depth should depend on the number of rounds because the key schedule operates in parallel with SubBytes. In other words, in their AES quantum circuits, the full depth should be independent of the number of key schedules. However, their AES-192 quantum circuit has a lower full depth than their AES-128 quantum circuit. Moreover, their AES-256 (14 rounds) quantum circuit has a lower full depth than their AES-128 quantum circuit.

Also, the full depth of their AES-192 and 256 quantum circuits cannot be derived. By analyzing full depth with the quantum resources required for their SubBytes and MixColumn, we believe their report is underestimated. Let us assume the following two things to estimate the full depth for their AES quantum circuit. All S-Boxes of SubBytes operate in parallel (in this case the full depth of SubBytes is 101, see Table 5(a)) and the full depth of round is counted only for SubBytes. Then, about 1212 (12 rounds $\times$ 101) should be the full depth of the AES-192 quantum circuit, and the full depth of the oracle where the AES quantum circuit is operated twice should be about 2424 (12 rounds $\times$ 101 $\times$ 2). Even with these optimistic assumptions, the full depth of the oracle they estimate

for AES-192 (1879 in Table 5(b)) cannot be derived. This underestimation also applies to the full depth of the oracle for AES-256, where they estimated 1951 in Table 5(b) $\neq$ about 2828 (14 rounds $\times$ 101 $\times$ 2).

This inconsistency is also observed in AES quantum circuits using in-place MixColumn (full depth is 111, as shown in Table 5(a)). To take one case, the full depth of oracle for AES-256 is 3353 (Table 5(b)). In the AES-256 quantum circuit, MixColumns operates for 13 rounds excluding the last round. Then, even counting only MixColumns, the full depth of oracle for AES-256 is 2886 (13 rounds $\times$ 111 $\times$ 2) even though SubBytes are not counted. If we consider the full depth with SubBytes included (cannot be operated in parallel with MixColumns), the full depth 3353 is lower than expected (i.e., underestimated in the reports of [34]).

5.2 Corrected Report

To our understanding, some problems arise if the qubits are allocated by the `using` command in Q# (and it affects the non-linear components). However more experiments are to be carried out in order to be completely certain about it.

The `using` command automatically disposes when the function ends. If ancilla qubits to implement AES S-box are allocated with the `using` command, the consistency between depth and qubits is lost. When 16 S-boxes are executed in SubBytes, the ancilla qubits allocated by the `using` are counted only for the first S-box and not after. Also counts the depth for executing 16 S-boxes simultaneously. In order to derive the correct result, the number of qubits or depth must be increased. Q#'s `ResourcesEstimator` tries to find its own lower bound for depth and qubit. That is, to achieve the qubits of the lower bound, the depth may have to be increased, and to achieve the depth of the lower bound, the qubits may have to be increased.

Another problem is inconsistencies between quantum resources. We observe underestimation when cross-checking the full depth of oracles, S-box and MixColumn they report. We could not pinpoint the exact cause, but we suspect the problems were caused by the `using` command and the AND gate. As noted, these problems effectively construct quantum circuits that are impossible. To patch, we contribute in three major directions:

1. We reflect on the increasing depth in their number of qubits using only one ancilla set. As shown in Figure 1(a), since the ancilla set is shared, not only SubBytes but also S-boxes of SubWord of the key schedule are operated sequentially.
2. We correct the implementation of MixColumns where the same issue occurs. In Eurocrypt'20 paper [34], two MixColumn implementations were presented. The in-place method of MixColumn implementation (which uses PLU decomposition, and derived by the authors themselves [34]) does not cause this issue. On the other hand, similar to S-box, the same issue applies to the MixColumn implementation by Maximov [43], which requires ancilla qubits, so this is also solved in the same way as the S-box.

3. We have modified the quantum circuits (SubBytes, key schedule and MixColumns) done by [34] and re-implemented their algorithm on ProjectQ to bypass the Q# bug. To avoid confusion, we estimate quantum resources using Toffoli gates (using the method from [3]), rather than applying AND gates (which could lead to some coding-related issues).

One way to correct the error is to estimate the correct depth by fixing the erroneous parallelism based on the number of qubits reported. Another way is to increase the number of qubits to satisfy the excessively estimated parallelism. We adopt the first approach and report the modified depth while keeping the reported number of qubits.

Our results with the bug-fixed Eurocrypt'20 implementation can be found in Table 5. Table 5(a) shows quantum resources for S-box and MixColumns reported in the Eurocrypt'20 paper. Quantum resources in Table 5(a) include cleaning up of used ancilla qubits. Table 5(b) shows corrected estimates for the quantum resources for AES oracles reported in the Eurocrypt'20 paper. Quantum resources are reported for an oracle rather than a single AES quantum circuit.

In the oracle, since the AES quantum circuit operates twice, the estimation of quantum resources for a single AES quantum circuit can be counted in half except for the number of qubits in Table 5(b). Table 5(c) shows the estimated resources (corrected) for SubBytes, key schedule, MixColumns, and one round where the issue occurs. The difference for the corrected MixColumns is relatively small, but the depth estimated as the lower-bound for SubBytes is corrected high. The resources estimated in Table 5(c) include a reverse operation to clean ancilla qubits. At the end, Table 5(d) shows the corrected quantum resources for AES quantum circuits, and it is confirmed that the depth increases significantly when maintaining the number of qubits.

Table 5: Corrected benchmarks for JNRV (Eurocrypt'20) implementation of AES.
(a) AES-128 gate costs.

Method	S-box	MixColumns	
	(SubByte)	In-place [34]	Maximov [43]
#CNOT	654	1108	1248
#1qCliff	184	0	0
#T	136	0	0
#Measure	34	0	0
T-depth	6	0	0
#qubits (M)	137	128	318
Full depth	101	111	22

(b) Oracles.

Method	In-place MixColumn [34]			Maximov's MixColumn [43]		
	AES-128	AES-192	AES-256	AES-128	AES-192	AES-256
#CNOT	292313	329697	404139	294863	332665	407667
#1qCliff	84428	94316	116286	84488	94092	116062
#T	54908	61436	75580	54908	61436	75580
#Measure	13727	15359	18895	13727	15359	18895
T-depth	121	120	126	121	120	126
#qubits (M)	1665	1985	2305	2817	3393	3969
Full depth	2816	2978	3353	2086	1879	1951

(c) AES-128 Modules.

Method	#CNOT	#1qCliff	#T	T-depth	#qubit	Full depth
SubBytes	12000	1220	7328	768	376	2672
Key schedule	3096	355	1832	192	248	669
MixColumns [43]	1248	0	0	0	318	88
One round†	16472	1507	9160	960	632	3417

†: One typical round (that includes MixColumn).

6 Performance of Quantum Circuits

In this part, we present the performance of our implementations of AES quantum circuits. We use the open-source quantum programming tool ProjectQ to implement and simulate the quantum circuits. An internal library, `ClassicalSimulator`, simulates quantum circuits and verifies test vectors. Quantum resources required to implement quantum circuits are estimated using another library, `ResourceCounter`.

As for the results, Table 4 shows the quantum resources required to implement our AES quantum circuits and previous AES quantum circuits. Although various decompositions exist for the Toffoli gate, Table 4 enables consistent comparison with NCT (NOT, CNOT, Toffoli) level analysis. Table 4 only covers the

(d) Summary.

AES	#CNOT	#1qCliff	#T	T-depth	#qubit	Full depth
128	161982	14400	91380	9576	1656	33320
192 ✚	182774	16128	102372	10728	1976	37328
256	224214	19871	126188	13224	2296	46012
128	163242	14994	91380	9576	2808	33914
192 ✛	184314	16854	102372	10728	3384	38054
256	226034	20729	126188	13224	3960	46870

✚: In-place MixColumn [34].
✛: Maximov's MixColumn [43].

version using the Toffoli gate, not the version using the AND gate. In [1, 24], the Itoh–Tsujii-based inversion is implemented on a quantum circuit, so many resources are used for SubBytes. In [39, 59], more efficient quantum circuits are implemented by extending the S-box of [12], but the circuit depth is increased due to the serial execution of S-boxes by concentrating on saving qubits. On the other hand, our implementation focuses on minimizing circuit depth while considering the trade-offs for using qubits. In [59], the TD-M cost metric (where TD is the Toffoli depth, and M is the number of qubits) was used to measure the trade-off of quantum circuits. The TD-M cost evaluates the performance of the quantum circuit alone, but in practice, due to depth limitations under the Grover's search, parallelization is necessary. The TD^2-M complexity metric in Table 4 demonstrates that in the trade-off of parallelization under Grover's search, the depth metric becomes significantly more important (this is discussed in more detail in the next Section 7). In this work, all AES quantum circuits with reduced depth and quantum gates using a reasonable number of qubits offer the best trade-off.

In [34], the quantum resources required to implement quantum circuits for AES were also estimated. However, there seem to be some issues with Q#'s `ResourcesEstimator`[10] used in their work, specially in implementing quantum circuits for SubBytes. Therefore, the results of [34] are not used here. In the NCT level analysis, replacing Toffoli gates with AND gates does not make much sense. As decomposition-based estimation is meaningful, we compare the required quantum resources by decomposing Toffoli and AND gates. Similar to [59, Table 10], Table 6 shows the detailed quantum resources by decomposing Toffoli gates (Table 6(a)) and AND gates taken from [34] (Table 6(b)) for the AES quantum circuits implemented in this work. The Toffoli gate is decomposed into (8 Clifford gates + 7 T gates), and T depth 4, and full depth 8 following to one of the methods (described in Section 3.2) in [3]. The AND gate requires 1 ancilla qubit and is decomposed into (11 Clifford gates + 4 T gates), and T depth 1, and full depth 8; and the AND† gate (Figure 5(b)) is decomposed into (7 Clifford gates + 1 Measurement gate), and incurs full depth of 6.

To replace the AES quantum circuits that use the Toffoli gate with the AND gate, a number of ancilla qubits equal to the maximum number of AND gates operating in parallel is required. However, the number of ancilla qubits needed for AND gates can be minimized by utilizing idle ancilla qubits that are already allocated for S-boxes. As a result, for the AND gate version using the S-box with Toffoli depth 4, only 4 ancilla qubits are needed for replacement; while for the version using the S-box with Toffoli depth 3, an additional 432 ancilla qubits are allocated for replacement.

7 Performance of Quantum Key Search

In this part, the corresponding costs for applying Grover's search algorithm to exhaustive key search are estimated based on the proposed quantum circuits for

[10] https://github.com/microsoft/qsharp-runtime/issues/192.

Table 6: Quantum circuit resources required for variants of AES (this work).
(a) Using Toffoli gate.

AES		#CNOT	#1qCliff	#T	T-depth (Td)	#qubit (M)	Full depth (FD)	Td-M cost ($Td \times M$)	FD-M cost ($FD \times M$)
128	☆	161640	14400	90440	304	3936	1364	1196544	5368704
	◎ ✿	154752	14400	85680	160	6368	978	1018880	6227904
	◈	164256	16832	85680	160	7520	799	1203200	6008480
	☆	315920	32000	207480	228	5176	1307	1180128	6765032
	◎ ❀	300912	32000	196560	120	8848	948	1061760	8387904
	◈	310416	33248	196560	120	10000	769	1200000	7690000
192	☆	184240	16400	102816	368	4256	1627	1566208	6924512
	◎ ✿	176904	16400	98056	192	6688	1174	1284096	7851712
	◈	188520	19440	98056	192	8096	955	1554432	7731680
	☆	359632	36464	235872	276	5496	1558	1516896	8562768
	◎ ❀	344176	36464	224952	144	9168	1138	1320192	10433184
	◈	355792	38024	224952	144	10576	919	1522944	1522944
256	☆	226232	19871	126616	432	4576	1907	1976832	8726432
	◎ ✿	218192	19871	121856	224	6976	1377	1562624	9605952
	◈	231920	23519	121856	224	8640	1118	1935360	9659520
	☆	442224	44159	290472	324	5816	1826	1884384	10620016
	◎ ❀	426064	44159	279552	168	9456	1335	1588608	12623760
	◈	439792	46031	279552	168	11120	1076	1868160	11965120

☆: Regular version.
◎: Shallow version.
◈: Shallow/low depth version.
✿: S-box with Toffoli depth 4.
❀: S-box with Toffoli depth 3.

the three variants of AES. We estimate the cost of oracle, which accounts for the largest portion of Grover's search algorithm. The overhead for diffusion operator is negligible compared to oracle and is not difficult to implement. For this reason, it is common to estimate the cost for oracle excluding the diffusion operator [5,24,39]. In the oracle, the target cipher's quantum circuit encrypts a known plaintext with the key in the superposition state. The generated ciphertext in the superposition state is compared with the known ciphertext and a reverse operation is performed for Grover's iterations. For comparison, an n-multi controlled NOT gate is used to check that the generated ciphertext (n-qubit) is a known ciphertext. In Grassl et al. [24] and Langenberg et al.'s AES paper [39], the authors added $32n - 84$ T-gates to their estimate for the n-multi controlled NOT gate [53]. If we estimate the cost of a 128-multi control NOT gate, only 4012 ($= 128 \times 32 - 84$) T-gates increase. However, the total number of gates to operate our AES-128 circuit in the oracle is already 532960 (the number of T gates is 180880). However, there is no significant change in the number of gates. In contrast, the T-depth overhead is relatively high. However, the increase in depth was also ignored in [24,39]. Also in [34], the estimation of the n-multi controlled NOT gate was totally ignored. So, for the n-multi controlled NOT gate, we estimate the number of T gates to

(b) Using AND gate.

AES		#CNOT	#1qCliff	#T	#Measure	T-depth (Td)	#qubit (M)	Full depth (FD)	Td-M cost ($Td \times M$)	FD-M cost ($FD \times M$)
128	☆	147160	39560	27200	6120	76	3940	1071	299440	4219740
	◎ ✿	142992	37520	27200	5440	40	6372	928	254880	5913216
	◈	152496	39952	27200	5440	40	7524	749	300960	5635476
	☆	289760	89720	62400	14040	57	5608	1123	319656	6297784
	◎ ❀	280992	85040	62400	12480	30	9280	908	278400	8426240
	◈	290496	86288	62400	12480	30	10432	729	312960	7604928
192	☆	167152	45232	30464	7072	92	4260	1270	391920	5410200
	◎ ✿	162536	43192	30464	6392	48	6692	1114	321216	7454888
	◈	174152	46232	30464	6392	48	8100	895	388800	7249500
	☆	328336	102608	69888	16224	69	5928	1334	409032	7907952
	◎ ❀	319120	97928	69888	14664	36	9600	1090	345600	10464000
	◈	330736	99488	69888	14664	36	11008	871	396288	9587968
256	☆	205216	55367	37536	8704	108	4580	1486	494640	6805880
	◎ ✿	199896	53327	37536	8024	56	6980	1307	390880	9122860
	◈	213624	56975	37536	8024	56	8644	1048	484064	9058912
	☆	403752	125591	86112	19968	81	6248	1562	506088	9759376
	◎ ❀	393832	120911	86112	18408	42	9888	1279	415296	12646752
	◈	407560	122783	86112	18408	42	11552	1020	485184	11783040

☆: Regular version.
◎: Shallow version.
◈: Shallow/low depth version.
✿: S-box with Toffoli depth 4.
❀: S-box with Toffoli depth 3.

be $(32n - 84)$ according to the decomposition method in [53] and T-depth is maintained.

In quantum exhaustive key search, to recover a unique key, not a spurious key, Grassl et al. in [24] estimated the attack cost for r known (plaintext, ciphertext) pairs ($r = 3$, $r = 4$ and $r = 5$, respectively). Later in [39], Langenberg et al. explained that $r = \lceil k/n \rceil$ (key size/block size) is sufficient to successfully recover a unique key. The authors in [34] also estimated the cost for the same r (plaintext, ciphertext) pairs in [39] through detailed computations. Following this approach, we also estimate the cost of recovering a unique key for $r = \lceil k/n \rceil$ (plaintext, ciphertext) pairs. When $r = 1$, the quantum circuit of the target block cipher is serially executed twice in oracle. Thus, the cost of the oracle is twice that required to implement a quantum circuit, excluding qubits. When $r \geq 2$, r target block quantum circuits are executed twice in parallel, and the following should be considered in cost estimation. Although $r \geq 2$ plaintexts are used, only one input key is used, so the cost for key schedule should be estimated only once. Finally, the complexity of quantum exhaustive key search for the target block cipher is roughly the cost of oracle $\times \lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$ (where k is the key size). The complexity figures are estimated at the (Clifford + T) level and computed as the number of total decomposed gates $\times$ full depth.

We show the cost of quantum key search by the Grover's algorithm for AES-128, AES-192, AES-256; with the two S-boxes (i.e., with Toffoli depth of 4 and 3) in Table 7(a) (using Toffoli gate) and Table 7(b) (using AND gate). Based on

Table 7: Quantum resources required for Grover's search on AES (this work).
(a) Using Toffoli gate.

AES		r	#qubit (M)	Total gates (G)	Full depth (FD)	FD-G cost ($FD \times G$)	FD-M cost ($FD \times M$)	Cost under MAXDEPTH		
								FD^2-G	FD^2-M	Td^2-M
128	☆	1	3,937	$1.609 \cdot 2^{82}$	$1.046 \cdot 2^{75}$	$1.683 \cdot 2^{157}$	$\mathbf{1.005 \cdot 2^{87}}$	$1.76 \cdot 2^{232}$	$1.051 \cdot 2^{162}$	$1.668 \cdot 2^{157}$
	◎ ✿		6,369	$1.539 \cdot 2^{82}$	$1.501 \cdot 2^{74}$	$1.155 \cdot 2^{157}$	$1.167 \cdot 2^{87}$	$1.734 \cdot 2^{231}$	$1.752 \cdot 2^{161}$	$1.495 \cdot 2^{156}$
	◈		7,521	$1.611 \cdot 2^{82}$	$1.226 \cdot 2^{74}$	$\mathbf{1.974 \cdot 2^{156}}$	$1.126 \cdot 2^{87}$	$1.21 \cdot 2^{231}$	$1.38 \cdot 2^{161}$	$1.765 \cdot 2^{156}$
	☆		5,177	$1.670 \cdot 2^{83}$	$1.002 \cdot 2^{75}$	$1.674 \cdot 2^{158}$	$1.266 \cdot 2^{87}$	$1.677 \cdot 2^{233}$	$1.269 \cdot 2^{162}$	$1.235 \cdot 2^{157}$
	◎ ❀		8,849	$1.592 \cdot 2^{83}$	$1.454 \cdot 2^{74}$	$1.158 \cdot 2^{158}$	$1.571 \cdot 2^{87}$	$1.684 \cdot 2^{232}$	$1.142 \cdot 2^{162}$	$1.166 \cdot 2^{156}$
	◈		10,001	$1.625 \cdot 2^{83}$	$1.18 \cdot 2^{74}$	$1.916 \cdot 2^{157}$	$1.441 \cdot 2^{87}$	$1.13 \cdot 2^{232}$	$1.7 \cdot 2^{161}$	$1.317 \cdot 2^{156}$
192	☆	2	7,841	$1.694 \cdot 2^{115}$	$1.248 \cdot 2^{107}$	$1.057 \cdot 2^{223}$	$1.195 \cdot 2^{120}$	$1.319 \cdot 2^{330}$	$1.491 \cdot 2^{227}$	$1.22 \cdot 2^{223}$
	◎ ✿		12,225	$1.622 \cdot 2^{115}$	$1.801 \cdot 2^{106}$	$1.460 \cdot 2^{222}$	$1.344 \cdot 2^{120}$	$1.315 \cdot 2^{329}$	$1.21 \cdot 2^{227}$	$1.032 \cdot 2^{222}$
	◈		15,041	$1.71 \cdot 2^{115}$	$1.465 \cdot 2^{106}$	$1.252 \cdot 2^{222}$	$1.345 \cdot 2^{120}$	$\mathbf{1.834 \cdot 2^{328}}$	$\mathbf{1.97 \cdot 2^{226}}$	$1.27 \cdot 2^{222}$
	☆		10,073	$1.758 \cdot 2^{116}$	$1.195 \cdot 2^{107}$	$1.051 \cdot 2^{224}$	$1.469 \cdot 2^{120}$	$1.256 \cdot 2^{331}$	$1.755 \cdot 2^{227}$	$1.759 \cdot 2^{222}$
	◎ ❀		16,689	$1.694 \cdot 2^{116}$	$1.746 \cdot 2^{106}$	$1.479 \cdot 2^{223}$	$1.779 \cdot 2^{120}$	$1.291 \cdot 2^{330}$	$1.553 \cdot 2^{227}$	$\mathbf{1.589 \cdot 2^{221}}$
	◈		19,505	$1.733 \cdot 2^{116}$	$1.41 \cdot 2^{106}$	$1.222 \cdot 2^{223}$	$1.679 \cdot 2^{120}$	$1.723 \cdot 2^{329}$	$1.184 \cdot 2^{227}$	$1.856 \cdot 2^{221}$
256	☆	2	8,417	$1.018 \cdot 2^{148}$	$1.463 \cdot 2^{139}$	$1.489 \cdot 2^{287}$	$1.503 \cdot 2^{152}$	$1.089 \cdot 2^{427}$	$1.099 \cdot 2^{292}$	$1.8 \cdot 2^{287}$
	◎ ✿		12,737	$1.967 \cdot 2^{147}$	$1.056 \cdot 2^{139}$	$1.039 \cdot 2^{287}$	$1.642 \cdot 2^{152}$	$1.097 \cdot 2^{426}$	$1.734 \cdot 2^{291}$	$1.461 \cdot 2^{286}$
	◈		16,065	$1.036 \cdot 2^{148}$	$1.715 \cdot 2^{138}$	$1.776 \cdot 2^{286}$	$1.682 \cdot 2^{152}$	$\mathbf{1.523 \cdot 2^{425}}$	$\mathbf{1.442 \cdot 2^{291}}$	$1.843 \cdot 2^{286}$
	☆		10,649	$1.058 \cdot 2^{149}$	$1.401 \cdot 2^{139}$	$1.481 \cdot 2^{288}$	$1.821 \cdot 2^{152}$	$1.037 \cdot 2^{428}$	$1.276 \cdot 2^{292}$	$1.28 \cdot 2^{287}$
	◎ ❀		17,201	$1.021 \cdot 2^{149}$	$1.024 \cdot 2^{139}$	$1.045 \cdot 2^{288}$	$1.075 \cdot 2^{153}$	$1.07 \cdot 2^{427}$	$1.101 \cdot 2^{292}$	$\mathbf{1.107 \cdot 2^{286}}$
	◈		20,529	$1.0444 \cdot 2^{149}$	$1.65 \cdot 2^{138}$	$1.724 \cdot 2^{287}$	$1.034 \cdot 2^{153}$	$1.422 \cdot 2^{426}$	$1.706 \cdot 2^{291}$	$1.322 \cdot 2^{286}$

☆: Regular version.
◎: Shallow version.
◈: Shallow/low depth version.
✿: S-box with Toffoli depth 4.
❀: S-box with Toffoli depth 3.

Table 7, we can determine the optimal strategy for implementing the Grover's search algorithm for each AES variant while adhering to the depth constraint. For AES-128 (full depth $\leq 2^{96}$), parallelization is not essential since it does not fall under the MAXDEPTH limit. Thus, without considering parallelization, the shallow/low depth version using S-box with Toffoli depth 4 has the lowest attack complexity (circuit size). However, when considering the more realistic metric of FD-M cost, the regular version using S-box with Toffoli depth 4 shows the highest efficiency. If the T-depth metric for error correction takes priority (i.e., Td-M cost), then the shallow version using S-box with Toffoli depth 4 is the optimal choice (although it is not shown in Tables 7(a) and 7(b), it can be found in Tables 6(a) and 6(b)). In contrast to AES-128, AES-192 and AES-256 require parallelization of the Grover' search due to the MAXDEPTH limitation. As specified in Appendix C, parallelizing Grover's search is highly inefficient, and in such cases, we should minimize FD^2-G, FD^2-M, and Td^2-M costs (i.e., Cost under MAXDEPTH in Table 7). Therefore, under the MAXDEPTH limit, the shallow/low depth version using S-box with Toffoli depth 4 is the most efficient in terms of attack complexity (FD^2-G cost) and the realistic metric (FD^2-M). If we consider the T-depth (Td^2-M), then the shallow version using S-box with Toffoli depth 3 is the optimal choice here.

(b) Using AND gate.

AES		r	#qubit (M)	Total gates (G)	Full depth (FD)	FD-G cost ($FD \times G$)	FD-M cost ($FD \times M$)	Cost under MAXDEPTH		
								FD^2-G	FD^2-M	Td^2-M
128	☆	1	3,941	$1.331 \cdot 2^{82}$	$1.644 \cdot 2^{74}$	$1.093 \cdot 2^{157}$	**$1.582 \cdot 2^{86}$**	$1.797 \cdot 2^{231}$	$1.3 \cdot 2^{161}$	$1.663 \cdot 2^{153}$
	◎ ✿		6,373	$1.289 \cdot 2^{82}$	$1.424 \cdot 2^{74}$	$1.836 \cdot 2^{156}$	$1.108 \cdot 2^{87}$	$1.307 \cdot 2^{231}$	$1.578 \cdot 2^{161}$	$1.461 \cdot 2^{152}$
	◈		7,525	$1.361 \cdot 2^{82}$	$1.149 \cdot 2^{74}$	**$1.564 \cdot 2^{156}$**	$1.055 \cdot 2^{87}$	$1.797 \cdot 2^{230}$	$1.212 \cdot 2^{161}$	$1.725 \cdot 2^{152}$
	☆		5,609	$1.372 \cdot 2^{83}$	$1.723 \cdot 2^{74}$	$1.182 \cdot 2^{158}$	$1.180 \cdot 2^{87}$	$1.018 \cdot 2^{233}$	$1.017 \cdot 2^{162}$	$1.325 \cdot 2^{153}$
	◎ ❀		9,281	$1.327 \cdot 2^{83}$	$1.394 \cdot 2^{74}$	$1.849 \cdot 2^{157}$	$1.579 \cdot 2^{87}$	$1.289 \cdot 2^{232}$	$1.101 \cdot 2^{162}$	$1.222 \cdot 2^{152}$
	◈		10,433	$1.359 \cdot 2^{83}$	$1.118 \cdot 2^{74}$	$1.520 \cdot 2^{157}$	$1.424 \cdot 2^{87}$	$1.699 \cdot 2^{231}$	$1.592 \cdot 2^{161}$	$1.374 \cdot 2^{152}$
192	☆	2	7,845	$1.398 \cdot 2^{115}$	$1.948 \cdot 2^{106}$	$1.362 \cdot 2^{222}$	$1.865 \cdot 2^{119}$	$1.327 \cdot 2^{329}$	$1.817 \cdot 2^{226}$	$1.212 \cdot 2^{219}$
	◎ ✿		12,229	$1.360 \cdot 2^{115}$	$1.709 \cdot 2^{106}$	$1.162 \cdot 2^{222}$	$1.276 \cdot 2^{120}$	$1.986 \cdot 2^{328}$	$1.09 \cdot 2^{227}$	$1.026 \cdot 2^{218}$
	◈		15,045	$1.448 \cdot 2^{115}$	$1.373 \cdot 2^{106}$	$1.988 \cdot 2^{221}$	$1.261 \cdot 2^{120}$	**$1.365 \cdot 2^{328}$**	**$1.731 \cdot 2^{226}$**	$1.261 \cdot 2^{218}$
	☆		10,825	$1.44 \cdot 2^{116}$	$1.023 \cdot 2^{107}$	$1.474 \cdot 2^{223}$	$1.352 \cdot 2^{120}$	$1.508 \cdot 2^{330}$	$1.383 \cdot 2^{227}$	$1.882 \cdot 2^{218}$
	◎ ❀		17,441	$1.4 \cdot 2^{116}$	$1.672 \cdot 2^{106}$	$1.17 \cdot 2^{223}$	$1.780 \cdot 2^{120}$	$1.956 \cdot 2^{329}$	$1.488 \cdot 2^{227}$	**$1.63 \cdot 2^{217}$**
	◈		20,257	$1.439 \cdot 2^{116}$	$1.336 \cdot 2^{106}$	$1.923 \cdot 2^{222}$	$1.652 \cdot 2^{120}$	$1.285 \cdot 2^{329}$	$1.104 \cdot 2^{227}$	$1.894 \cdot 2^{217}$
256	☆	2	8,421	$1.678 \cdot 2^{147}$	$1.14 \cdot 2^{139}$	$1.912 \cdot 2^{286}$	$1.172 \cdot 2^{152}$	$1.09 \cdot 2^{426}$	$1.336 \cdot 2^{291}$	$1.791 \cdot 2^{283}$
	◎ ✿		12,741	$1.635 \cdot 2^{147}$	$1.002 \cdot 2^{139}$	$1.639 \cdot 2^{286}$	$1.558 \cdot 2^{152}$	$1.642 \cdot 2^{425}$	$1.561 \cdot 2^{291}$	$1.427 \cdot 2^{282}$
	◈		16,069	$1.739 \cdot 2^{147}$	$1.607 \cdot 2^{138}$	$1.398 \cdot 2^{286}$	$1.576 \cdot 2^{152}$	**$1.123 \cdot 2^{425}$**	**$1.266 \cdot 2^{291}$**	$1.8 \cdot 2^{282}$
	☆		11,401	$1.73 \cdot 2^{148}$	$1.198 \cdot 2^{139}$	$1.037 \cdot 2^{288}$	$1.667 \cdot 2^{152}$	$1.242 \cdot 2^{427}$	$1.997 \cdot 2^{291}$	$1.37 \cdot 2^{283}$
	◎ ❀		17,953	$1.688 \cdot 2^{148}$	$1.962 \cdot 2^{138}$	$1.655 \cdot 2^{287}$	$1.075 \cdot 2^{153}$	$1.624 \cdot 2^{426}$	$1.055 \cdot 2^{292}$	**$1.131 \cdot 2^{282}$**
	◈		21,281	$1.734 \cdot 2^{148}$	$1.564 \cdot 2^{138}$	$1.357 \cdot 2^{287}$	$1.016 \cdot 2^{153}$	$1.061 \cdot 2^{426}$	$1.589 \cdot 2^{291}$	$1.341 \cdot 2^{282}$

☆: Regular version.
◎: Shallow version.
◈: Shallow/low depth version.
✿: S-box with Toffoli depth 4.
❀: S-box with Toffoli depth 3.

Additionally, a quick comparison of NIST's security level (under the Grover's search) of our work together with the previous works is given in Table 8. As it can be seen, when compared with the current state-of-the-art security bounds, we reduce the quantum complexity for running the Grover's search on the AES family, thereby setting up a new benchmark for the NIST security levels. The complexity is calculated in terms of the product of decomposed (Clifford and T) gate count and full depth. Also, the MAXDEPTH constraint (see Appendix C) is not considered in the computation. For instance, the figure of $2^{156.97}$ corresponding to the shallow/low version in Table 8 is computed as the product of the total number of decomposed gates and the full depth for 2^{64} (i.e., square-root bound of the exhaustive case) searches (required to run Grover's search). If the MAXDEPTH constraint is to be considered, one has scale down the complexity figures by dividing by the MAXDEPTH constant.

8 Conclusion

In this work, we collate multiple research contributions, including the up-to-date optimizations on the building blocks of the ciphers in one place; whence significantly reducing the quantum circuit complexity for the AES family of block ciphers. Among other results, we show the least Toffoli depth and full depth implementations of all variants of AES (more than 98% and 95% improvement from [59] and [28] respectively). At the same time, we improve the Toffoli depth -

Table 8: Comparison of NIST security levels based on AES variants.

Level (AES)	GLRS [24]	NIST [45]	LPS [39]	This work			
				☆	◎	◈	❄
1 (128)	$2^{168.6683}$	2^{170}	$2^{162.6093}$	✿: $2^{157.1283}$	✿: $2^{156.8766}$	✿: $\mathbf{2^{156.6452}}$	✚: $2^{162.3577}$
				❀: $2^{158.2412}$	❀: $2^{157.8867}$	❀: $2^{157.6041}$	✢: $2^{162.5641}$
3 (192)	$2^{233.4645}$		$2^{227.6491}$	✿: $2^{222.4457}$	✿: $2^{222.2166}$	✿: $\mathbf{2^{221.9913}}$	✚: $2^{227.5867}$
				❀: $2^{223.5597}$	❀: $2^{223.2265}$	❀: $2^{222.9434}$	✢: $2^{227.6260}$
5 (256)	$2^{298.3467}$		$2^{292.3100}$	✿: $2^{286.9351}$	✿: $2^{286.7128}$	✿: $\mathbf{2^{286.4834}}$	✚: $2^{292.1520}$
				❀: $2^{288.0524}$	❀: $2^{287.7268}$	❀: $2^{287.4404}$	✢: $2^{292.1900}$

☆: Regular version (using AND gate).
◎: Shallow version (using AND gate).
◈: Shallow/low depth version (using AND gate).
✿: S-box with Toffoli depth 4.
❀: S-box with Toffoli depth 3.
❄: Bug-fixed JNRV [34] (using S-box from [13]; using Toffoli gate).
✚: In-place MixColumn [34]. ✢: Maximov's MixColumn [43].

qubit count product by more than 75% and 30%, and more than 84% and 99% in the Toffoli depth-square - qubit count product compared to the respective papers. A bird's-eye view can be seen from Figure 6, where we show our work contributes in lowering the quantum circuit complexity (in terms of qubit count and full depth) compared to GLRS [24] and LPS [39]. In total, we present 14 implementations per variant of AES (including bug-fixing of [34]), each incorporating a special design idea/optimization.

Most recent papers about AES quantum implementations focus on reducing the number of qubits, but do not appear to give much consideration on depth reduction of the circuit [1, 24, 39, 51, 52, 59]. In our work, one of the major ways we lower the depth metrics is by allowing a relatively higher number of qubits, so that the product terms (i.e., when the number of qubits is multiplied with the circuit depth metrics or decomposed gate counts) becomes smaller. Having a lower circuit depth also makes it easier to maximize the number of iterations (required to run the Grover's search algorithm) and thus is a crucial factor in reducing the cost of evaluating the overall quantum search complexity for exhaustive key search a cipher.

Finding optimizations for the cipher building blocks can be considered among the top priorities for the future research works. Besides, the idea in [19] can be used on top of our implementations to further reduce the cost for AES-192 and AES-256 (i.e., when $r > 1$); this is kept as a follow-up work. Similarly, other decompositions of the Toffoli gate (e.g., [48]) can also be considered in the future scope.

References

1. Almazrooie, M., Samsudin, A., Abdullah, R., Mutter, K.N.: Quantum reversible circuit of AES-128. Quantum Information Processing **17**(5) (may 2018) 1–30

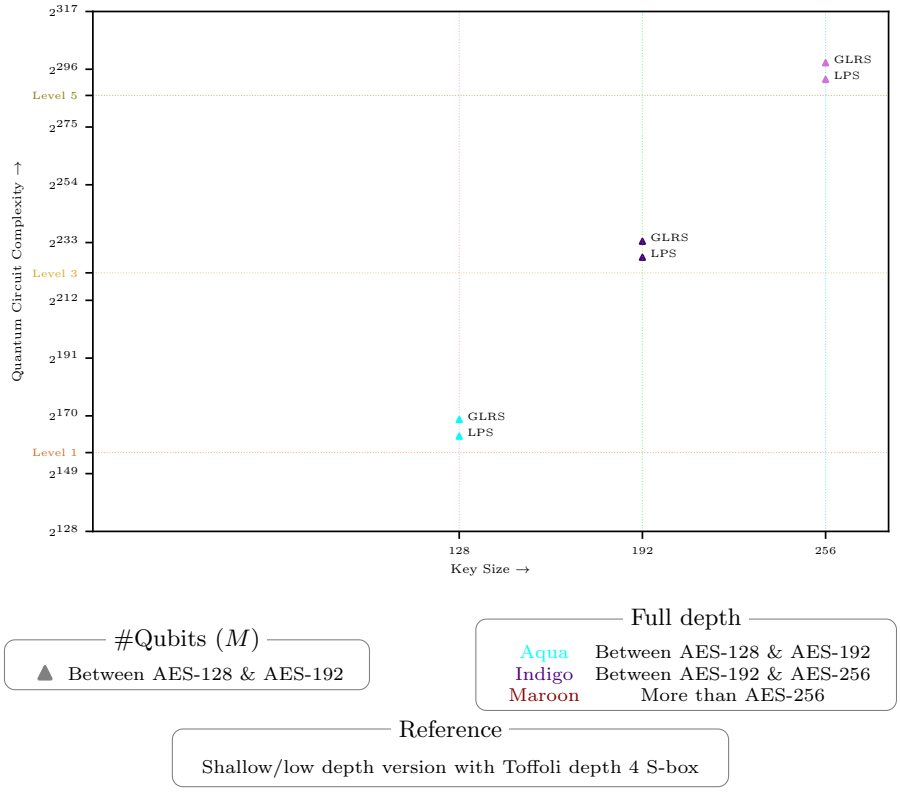


Fig. 6: Comparison of quantum circuit complexities for AES variants.

2. Amy, M., Di Matteo, O., Gheorghiu, V., Mosca, M., Parent, A., Schanck, J.: Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In Avanzi, R., Heys, H., eds.: Selected Areas in Cryptography – SAC 2016, Cham, Springer International Publishing (2017) 317–337
3. 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) (Jun 2013) 818–830
4. Anand, R., Maitra, A., Maitra, S., Mukherjee, C.S., Mukhopadhyay, S.: Quantum resource estimation for FSR based symmetric ciphers and related Grover's attacks. In Adhikari, A., Küsters, R., Preneel, B., eds.: Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings. Volume 13143 of Lecture Notes in Computer Science., Springer (2021) 179–198
5. 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, Cham, Springer International Publishing (2020) 395–413

6. Anand, R., Maitra, A., Mukhopadhyay, S.: Grover on SIMON. Quantum Information Processing **19**(9) (Sep 2020) http://dx.doi.org/10.1007/s11128-020-02844-w.
7. Baksi, A., Jang, K., Song, G., Seo, H., Xiang, Z.: Quantum implementation and resource estimates for rectangle and knot. Quantum Information Processing **20**(12) (dec 2021)
8. Banik, S., Funabiki, Y., Isobe, T.: Further results on efficient implementations of block cipher linear layers. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **104-A**(1) (2021) 213–225
9. Bathe, B.N., Anand, R., Dutta, S.: Evaluation of Grover's algorithm toward quantum cryptanalysis on ChaCha. Quantum Inf. Process. **20**(12) (2021) 394
10. Bhattacharjee, D., Chattopadhyay, A.: Depth-optimal quantum circuit placement for arbitrary topologies. arXiv preprint arXiv:1703.08540 (2017)
11. Bonnetain, X., Naya-Plasencia, M., Schrottenloher, A.: Quantum security analysis of AES. IACR Transactions on Symmetric Cryptology **2019**(2) (Jun. 2019) 55–93
12. Boyar, J., Peralta, R.: A new combinational logic minimization technique with applications to cryptology. In Festa, P., ed.: Experimental Algorithms, Berlin, Heidelberg, Springer Berlin Heidelberg (2010) 178–189
13. Boyar, J., Peralta, R.: A depth-16 circuit for the AES S-box. Cryptology ePrint Archive, Report 2011/332 (2011) https://eprint.iacr.org/2011/332.
14. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik **46**(4-5) (Jun 1998) 493–505
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 (2020) 238–258
16. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Information Security and Cryptography. Springer (2002)
17. Dansarie, M.: Cryptanalysis of the SoDark family of cipher algorithms. PhD thesis, Naval Postgraduate School, Dudley Knox Library (2017) https://calhoun.nps.edu/handle/10945/56118.
18. Dansarie, M.: sboxgates: A program for finding low gate count implementations of S-boxes. Journal of Open Source Software **6**(62) (2021) 2946
19. Davenport, J.H., Pring, B.: Improvements to quantum search techniques for block-ciphers, with applications to aes. In Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C., eds.: Selected Areas in Cryptography, Cham, Springer International Publishing (2021) 360–384
20. de Wolf, R.: Quantum Computing: Lecture Notes. (2019) https://arxiv.org/pdf/1907.09415v1.pdf.
21. Dong, X., Dong, B., Wang, X.: Quantum attacks on some feistel block ciphers. Des. Codes Cryptogr. **88**(6) (2020) 1179–1203
22. Ekdahl, P., Johansson, T., Maximov, A., Yang, J.: A new snow stream cipher called snow-v. IACR Transactions on Symmetric Cryptology **2019**(3) (Sep. 2019) 1–42
23. Gidney, C.: Halving the cost of quantum addition. Quantum **2** (2018) 74
24. 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 (2016) 29–43
25. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. (1996) 212–219
26. Grumbling, E., Horowitz, M.: Quantum Computing: Progress and Prospects. The National Academies Press, Washington DC (2019) https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects.

27. He, Y., Luo, M.X., Zhang, E., Wang, H.K., Wang, X.F.: Decompositions of n-qubit toffoli gates with linear circuit complexity. International Journal of Theoretical Physics **56**(7) (2017) 2350–2361
28. Huang, Z., Sun, S.: Synthesizing quantum circuits of AES with lower t-depth and less qubits. In Agrawal, S., Lin, D., eds.: Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III. Volume 13793 of Lecture Notes in Computer Science., Springer (2022) 614–644
29. Jang, K., Choi, S., Kwon, H., Kim, H., Park, J., Seo, H.: Grover on Korean block ciphers. Applied Sciences **10**(18) (2020)
30. Jang, K., Baksi, A., Breier, J., Seo, H., Chattopadhyay, A.: Quantum implementation and analysis of default. Cryptology ePrint Archive, Paper 2022/647 (2022) https://eprint.iacr.org/2022/647.
31. Jang, K., Baksi, A., Kim, H., Seo, H., Chattopadhyay, A.: Improved quantum analysis of SPECK and lowmc. In Isobe, T., Sarkar, S., eds.: Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings. Volume 13774 of Lecture Notes in Computer Science., Springer (2022) 517–540
32. 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)
33. Jang, K., Song, G., Kim, H., Kwon, H., Kim, H., Seo, H.: Parallel quantum addition for Korean block cipher. IACR Cryptol. ePrint Arch. (2021) 1507
34. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and lowmc. In Canteaut, A., Ishai, Y., eds.: Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. Volume 12106 of Lecture Notes in Computer Science., Springer (2020) 280–310
35. Kaplan, M., Leurent, G., Leverrier, A., Naya-Plasencia, M.: Breaking symmetric cryptosystems using quantum period finding. In: CRYPTO, Springer (2016) 207–237
36. Kim, P., Han, D., Jeong, K.C.: Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to aes and sha-2. Quantum Information Processing **17**(12) (2018) 1–39
37. kn, K., Song, G., Kwon, H., Uhm, S., Kim, H., Lee, W.K., Seo, H.: Grover on pipo. Electronics **10**(10) (2021) 1194
38. Kranz, T., Leander, G., Stoffelen, K., Wiemer, F.: Shorter linear straight-line programs for mds matrices. IACR Transactions on Symmetric Cryptology **2017**(4) (Dec. 2017) 188–211
39. 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** (01 2020) 1–12
40. Li, S., Sun, S., Li, C., Wei, Z., Hu, L.: Constructing low-latency involutory mds matrices with lightweight circuits. IACR Transactions on Symmetric Cryptology (2019) 84–117
41. Lin, D., Xiang, Z., Zeng, X., Zhang, S.: A framework to optimize implementations of matrices. In Paterson, K.G., ed.: Topics in Cryptology - CT-RSA 2021 - Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17-20, 2021, Proceedings. Volume 12704 of Lecture Notes in Computer Science., Springer (2021) 609–632

42. Liu, Q., Wang, W., Fan, Y., Wu, L., Sun, L., Wang, M.: Towards low-latency implementation of linear layers. IACR Transactions on Symmetric Cryptology **2022**(1) (Mar. 2022) 158–182
43. Maximov, A.: AES MixColumn with 92 XOR gates. Cryptology ePrint Archive, Report 2019/833 (2019) https://eprint.iacr.org/2019/833.
44. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information (10th Anniversary Edition). Cambridge University Press (2010)
45. NIST.: Submission requirements and evaluation criteria for the post-quantum cryptography standardization process (2016) https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.
46. Perriello, S.: Design and development of a quantum circuit to solve the information set decoding problem. (2019)
47. Roy, S., Baksi, A., Chattopadhyay, A.: Quantum implementation of ascon linear layer. Cryptology ePrint Archive, Paper 2023/617 (2023) https://eprint.iacr.org/2023/617.
48. Selinger, P.: Quantum circuits of t-depth one. Physical Review A **87**(4) (2013) 042302
49. Song, G., Jang, K., Kim, H., Lee, W., Hu, Z., Seo, H.: Grover on SM3. IACR Cryptol. ePrint Arch. (2021) https://eprint.iacr.org/2021/668.
50. Tan, Q.Q., Peyrin, T.: Improved heuristics for short linear programs. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020**(1) (2020) 203–230
51. Wang, Z.G., Wei, S.J., Long, G.L.: A quantum circuit design of aes requiring fewer quantum qubits and gate operations. Frontiers of Physics **17**(4) (2022) 1–7
52. Wang, Z., Wei, S., Long, G.: A quantum circuit design of AES (2021) https://arxiv.org/abs/2109.12354.
53. Wiebe, N., Roetteler, M.: Quantum arithmetic and numerical analysis using repeat-until-success circuits (2014)
54. Xiang, Z., Zeng, X., Lin, D., Bao, Z., Zhang, S.: Optimizing implementations of linear layers. IACR Trans. Symmetric Cryptol. **2020**(2) (2020) 120–145
55. Zalka, C.: Grover's quantum searching algorithm is optimal. Physical Review A **60**(4) (1999) 2746
56. Zhu, C., Huang, Z.: Optimizing the depth of quantum implementations of linear layers. In: Information Security and Cryptology - 18th International Conference, Inscrypt 2022, Beijing, China, December 11-13, 2022, Revised Selected Papers. Volume 13837 of Lecture Notes in Computer Science., Springer (2022) 129–147
57. 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) (2022) 1–38
58. Zou, J., Liu, Y., Dong, C., Wu, W., Dong, L.: Observations on the quantum circuit of the SBox of AES. Cryptology ePrint Archive, Report 2019/1245 (2019) https://eprint.iacr.org/2019/1245.
59. 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, Cham, Springer International Publishing (2020) 697–726

A Concise Description of AES Variants

The Advanced Encryption Standard (AES) [16] is an SPN block cipher family with a block of 128 bits. The state of AES is arranged as a 4×4 matrix of bytes. AES contains three specific variants denoted as AES-128, AES-192 and AES-256 according to the key size. Schematic diagrams of AES-128 round function and key schedule can be found in Figure 7.

A.1 Round Function

The round function of AES consists of AddRoundKey $\circ$ MixColumns $\circ$ ShiftRows $\circ$ SubBytes, except for the last round which misses the MixColumns operation.

SubBytes This operation substitutes each element by a predefined 8×8 S-box.

ShiftRows This operation cyclically rotates the r^{th} row of state to the left by i places, for $i = 0, 1, 2, 3$.

MixColumns The MixColumn operation pre-multiplies each of the state column with the right circulant matrix $(\mathtt{02}, \mathtt{03}, \mathtt{01}, \mathtt{01})$, over $\mathrm{GF}(2^8)[x]$ with modulus x^4+1. Since the MixColumn operates on the state based on an entire column, it can also be represented as a matrix over $\mathbb{F}_2$ with dimension 32×32.

AddRoundKey The sub-key of each round is generated by the Key Expansion algorithm. Each call of AddRoundKey XORs the 128-bit sub-key to the state.

The encryption procedure for different instances of AES family are somewhat similar, except the number of round varies. For AES-128, AES-192 and AES-256, the round numbers are 10, 12, 14 respectively and all round functions are identical except that there is no MixColumns operation in the last round. Note that there is an extra key addition before the first round (also known as whitening).

A.2 Key Schedule

Similar to the state, the master key of AES is allocated to a $4 \times l$ grid of byte in order, where $l = 4, 6$ or 8 for AES-128, AES-192 and AES-256, respectively. Generally, the generation of the round sub-keys are based on *word* (the entire column in the grid) with the operations RotWord (cyclically rotating the bytes in a word to the left by one byte), SubWord (operating the SubBytes of round function on each bytes in a word) and the XOR of $\mathrm{Rcon}[r]$ (the r^{th} 32-bit round constant).

The master key is loaded to the grid $W_0, W_1, \cdots, W_i$; where i is 3, 5 and 7 for AES-128, AES-192 and AES-256 respectively. In order to guarantee the encryption, 40, 46 and 52 words need to be provided by key expansion for those three AES instances, respectively.

For AES-128, the word W_i is generated by

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

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

For AES-192, the word W_i is generated by

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

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

For AES-256, the word W_i is generated by

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

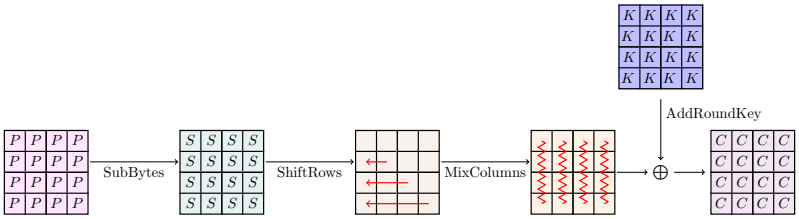
where $i = 8, 9, \cdots, 59$.

A.3 Notes

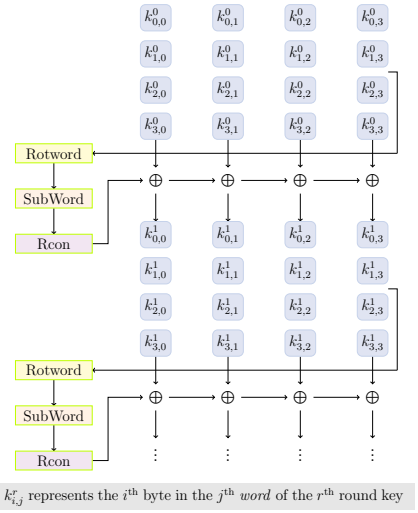
Singular Form and Plural Form The AES state is represented as a 4×4 matrix and the operation on one column of the matrix is denoted here as MixColumn. As described earlier, MixColumn corresponds to a matrix multiplication over $\text{GF}(2^8)$, which can equivalently be expressed as multiplication by a matrix of dimension 32×32 over $\mathbb{F}_2$. In the AES round function, the MixColumns operates on the whole block by applying MixColumn to every four bytes in the state (i.e., one column in the 4×4 matrix). Thus, one MixColumns operation is equivalent to $4\times$ MixColumn operations on different columns in the matrix. Denoting the binary matrix corresponding to MixColumn as M with size 32×32, MixColumns can be represented as the diagonal matrix (M, M, M, M) of dimension 128×128 over $\mathbb{F}_2$.

The bytes in each row of the matrix will be cyclically shifted to the left in each round and the shift operation on the bytes in one row is denoted here as ShiftRow, in the step of ShiftRows, the ShiftRow will be operated on all the rows in the matrix and shift the bytes in the ith row to the left by i bytes, where $i = 1, 2, 3$. Thus, one ShiftRows operation is equivalent to $4\times$ ShiftRow operations on different rows in the 4×4 matrix with the shift parameter varies from 0 to 3.

The SubBytes in the round function updates every byte in the 4×4 matrix in the same way. The process of applying the S-box to one byte in the AES state is denoted here as SubByte. In each round, the SubBytes updates all the bytes in the 4×4 matrix by replacing each byte by another one according to the predefined nonlinear map. Thus, one SubBytes operation is equivalent to 16 SubByte operations on the bytes of the 4×4 matrix.



(a) Round function of encryption (except last round which skips MixColumns).



(b) Key schedule for encryption.

Fig. 7: Schematic of AES construction.

S-box and S-box† in Quantum S-box in quantum denotes before storing values from ancilla qubits to output qubits. Denote the reverse operation of S-box as S-box† and uses input qubits to clean up ancilla qubits.

SubBytes and SubBytes† in Quantum SubBytes of AES in quantum denotes parallel operation for 16 S-boxes. Denote the reverse operation of SubBytes as SubBytes† and cleans up all used ancilla qubits in 16 S-boxes.

Rotation and Rotation† in Quantum Rotation of AES in quantum denotes the same RotWord. The reverse operation of Rotation is denoted as Rotation†.

SubWord and SubWord† in Quantum SubWord of AES in quantum denotes parallel operation for 4 S-boxes. We denote the reverse operation of SubWord as SubWord† (and clean up all used ancilla qubits in 4 S-boxes).

B Quantum Key Search using Grover's Algorithm

For a secret-key cipher using an k-bit key, 2^k queries are required for the exhaustive key search. The Grover's search [25] is a well-known quantum algorithm that recovers the key with a high probability in about $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$ queries. The procedure can be briefly described as follows (some basic familiarity with the quantum notations/terminology is assumed, one may refer to, e.g., [20, 44] for a more detailed description):

1. A k-qubit key (K) is prepared in superposition $|\psi\rangle$ by applying the Hadamard gates. All states of qubits have the same amplitude:

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

2. The cipher (Enc) is implemented as a quantum circuit and placed in oracle. In oracle $f(x)$, the plaintext (p) is encrypted with the key in the superposition state. As a result, the ciphertexts for all key values are generated. The sign of the solution key is changed to a negative by comparing it with the known ciphertext. The condition ($f(x) = 1$) changes the sign to negative and applies to all states. For this phase flip, an n-qubit controlled Z gate is utilized (n is the length of the ciphertext, c).

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

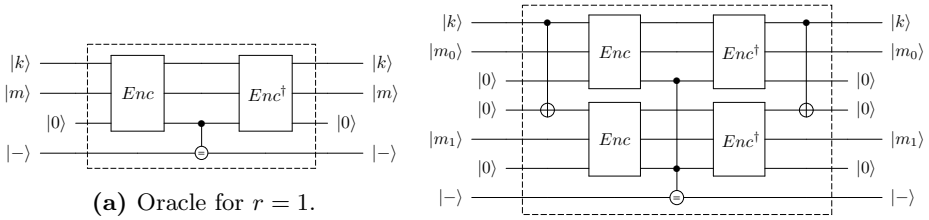
$$U_f(|\psi\rangle |-\rangle) = \frac{1}{2^{k/2}} \sum_{x=0}^{2^k-1} (-1)^{f(x)} |x\rangle |-\rangle \tag{3}$$

3. Lastly, the diffusion operator[11] amplifies the amplitude of the negative sign state. Diffusion operator is implemented with the following (H gates layer $\rightarrow$ X gates layer $\rightarrow$ k-qubit controlled Z gate $\rightarrow$ X gates layer $\rightarrow$ H gates layer). In [46], a simple technique was introduced by which a constant number of X gates are used for the diffusion operator. If a constant number of X gates are applied before the Hadamard gates in Step 1, the diffusion operator is implemented as (H gates layer $\rightarrow$ k-qubit controlled Z gate $\rightarrow$ H gates layer).

The Grover's search executes Equations (2), (3) and diffusion operator in a series to sufficiently increase the amplitude of the solution and observes it at the end. For an k-bit key, the optimal number of iterations of the Grover's search algorithm is roughly $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$ [14], which is about $\sqrt{2^k}$. In the process, an exhaustive key search that requires 2^k queries in a classic computer is reduced to roughly $\sqrt{2^k}$ queries in a quantum computer (this works with a high probability).

[11] Since the diffusion operator is usually generic, it does not require any special technique for implementation.

In the exhaustive key search, $r = \lceil k/n \rceil$ (plaintext, ciphertext) pairs are needed to recover a unique key that is not a spurious key (see Section 7 for details). Figure 8 shows the Grover's oracle of exhaustive key search. Encryption† is defined as the reverse operation of encryption, which reverts to the state before encryption.



(a) Oracle for $r = 1$.

(b) Oracle for $r = 2$.

Fig. 8: Schematic architecture for key search using Grover's algorithm.

C NIST Security Levels

The following security levels were defined by NIST [45] to assess the post-quantum security:

① Level 1: Cipher is at least as hard to break as AES-128.
② Level 2: Cipher is at least as hard to break as SHA-256.
③ Level 3: Cipher is at least as hard to break as AES-192.
④ Level 4: Cipher is at least as hard to break as SHA-384.
⑤ Level 5: Cipher is at least as hard to break as AES-256.

NIST recommended that a given cipher should achieve some minimum security level to provide sufficient security in the post-quantum era. Based on the research available back then (probably the only such work was due to [24]), NIST estimated used in [45] the following complexities: Level 1: 2^{170}, Level 3: 2^{233}, Level 5: 2^{298} (on a closer look, however, it seems that complexity estimated in [24] for Level 1 was close to 2^{169}). The complexity bounds were calculated as the product of total number of decomposed gates and full depth required for the Grover's key search circuit.

With the passage of time, as more research works on the AES family have been being reported, the complexity for the security levels (1, 3 and 5) have been gradually reduced. A comprehensive synopsis of the notable works can be seen from Table 8, where we show the impact on our work in reshaping the security levels. In particular, the following new bounds are achieved (see also Table 7(b)):

- ☞ Level 1: $\mathbf{2^{156.6452}}$; with total Clifford, T and measurement gates = $2^{82.4447}$; full depth = $2^{74.2004}$.
- ☞ Level 3: $\mathbf{2^{221.9913}}$; with total Clifford, T and measurement gates = $2^{115.5341}$; full depth = $2^{106.4573}$.
- ☞ Level 5: $\mathbf{2^{286.4834}}$; with total Clifford, T and measurement gates = $2^{147.7983}$; full depth = $2^{138.6844}$.

Along with this, NIST proposed a parameter called MAXDEPTH to impose a limit on circuit depth. The bounds for MAXDEPTH are not clearly stated, rather it is speculated that the following figures can be taken as good indicators: 2^{40}, 2^{64} and 2^{96}; judging by the expected computation power of a quantum computer – in a year, or a decade, or a millennium. Keeping that in mind, one would expect the depth of the quantum circuit for the Grover's search is not higher than 2^{96} (i.e., the highest bound estimated for MAXDEPTH[12]). However, if it turns out that the depth restriction is not within the stipulated bound, then the following approaches can be undertaken [36]:

1. *Outer parallelization:* Restrict depth at the $\leq 2^{96}$ at the expense of lower success probability of key recovery.
2. *Inner parallelization:* Split the search space into multiple subspaces with shallow depth, where each circuit measures the secret key with a lower success probability.
3. Cost is calculated as-is without considering MAXDEPTH (see, e.g., [36, Table 2]). It is worth noting that the previous implementations like [1, 28, 39, 59] also did not appear to consider the MAXDEPTH limit.

The outer and inner parallelization methods lower the probability of measuring a solution by reducing the number of iterations for the Grover oracle. Outer parallelization halts the Grover iterations at the depth limit, leading to the measurement of suboptimal solutions with lower probabilities. Inner parallelization reduces the number of Grover iterations by reducing the search space, which also lower the probablility of discovering a solution. However, parallelizing the Grover's search is highly inefficient due to the poor performance resulting from the analysis in [55], which indicates that only a $\sqrt{S}$ depth reduction can be achieved with S instances (operating in parallel) of the Grover oracle. Thus, the optimal method is to perform as many iterations of the Grover oracle as possible within a limited depth. According to the analysis in [34, Section 3.4], to minimize the FD-G (full depth and gate count product), TD-M (Toffoli depth and qubit count product), and FD-M (Toffoli depth and qubit count product) costs under the parallelization of Grover's search, it is necessary to minimize the FD^2-G, TD^2-M and FD^2-M costs. This is because reducing the depth by $\sqrt{S}$ requires S instances of the Grover oracle, leading to an increase in the total number of

[12]As the Grover's search makes the circuit depth greater than $2^{k/2}$ for k-bit key (the quantum depth for the cipher implementation $\times \lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ required for Grover's search), the quantum depth is trivially greater two smaller MAXDEPTH values for AES variants.

gates and qubits required for parallelization. Therefore, when parallelization due to depth limitation is inevitable, the primary objective should be to minimize the depth.

As of now, we remark that the depths of our AES quantum circuits are the lowest when compared to other quantum circuits available in the literature [1, 28, 39, 59]. Table 9 displays a quick view where the related works (namely, GLRS [24] and LPS [39]) are compared with respect to our implementations in terms of full depth. Note that, only AES-128 satisfies the MAXDEPTH criterion (i.e., $\leq 2^{96}$).

One may further note that the depth of quantum attack on AES-128 (i.e., Level 1) is within the permitted MAXDEPTH limit ($2^{74.2388}$ using the AND gate; the same using the Toffoli gate is $2^{74.2940}$). However, the same cannot be stated for AES-192 and -256, since the full depth figures are respectively $2^{106.4957}$ and $2^{138.7225}$ (using the AND gate; the same using the Toffoli gate are $2^{106.5509}$ and $2^{138.7782}$, respectively). In this work, we adopt the 3rd approach for the sake of brevity and report the cost with considering the MAXDEPTH limit. So, we can identify the optimal parallelization strategy that strikes a balance between adjusted cost – success probability trade-offs (Section 6). As described, the circuit depth metrics are the primary factor determining performance in general.

Table 9: Summary of AES implementations with respect to MAXDEPTH.

AES	GLRS [24]	LPS [39]	This work				MAXDEPTH ($\leq 2^{96}$)
			☆	◎	◈	❁	
128	$2^{81.2141}$	$2^{79.4751}$	✿: $2^{75.0649}$	✿: $2^{74.5859}$	✿: $2^{74.2940}$	✚: $2^{79.6576}$	✓
			❀: $2^{75.0029}$	❀: $2^{74.5400}$	❀: $\mathbf{2^{74.2388}}$	✢: $2^{79.7011}$	
192	$2^{113.4114}$	$2^{111.2987}$	✿: $2^{107.3196}$	✿: $2^{106.8488}$	✿: $2^{106.5509}$	✚: $2^{111.8395}$	✗
			❀: $2^{107.2570}$	❀: $2^{106.8041}$	❀: $\mathbf{2^{106.4957}}$	✢: $2^{111.8673}$	
256	$2^{145.6508}$	$2^{143.6871}$	✿: $2^{139.5489}$	✿: $2^{139.0786}$	✿: $2^{138.7782}$	✚: $2^{144.1412}$	✗
			❀: $2^{139.4865}$	❀: $2^{139.0342}$	❀: $\mathbf{2^{138.7225}}$	✢: $2^{144.1679}$	

☆: Regular version (using AND gate).
◎: Shallow version (using AND gate).
◈: Shallow/low depth version (using AND gate).
✿: S-box with Toffoli depth 4.
❀: S-box with Toffoli depth 3.
❁: Bug-fixed JNRV [34] (using S-box from [13]; using Toffoli gate).
✚: in-place MixColumn [34]. ✢: Maximov's MixColumn [43].

D Depth of Sequential XOR: Classical vs. Quantum

One may note from Table 3 that the depth for quantum circuit corresponding to the implementation by [54] is 30, whereas the same for the classical circuit is 6. Although this implementation operates in-place, it still reuses one variable multiple times. In other words, the same variable appears multiple times in the

right hand side. For example, one may check that x_{31} appears more than once: $x_{16} \leftarrow x_{16} \oplus x_{31}$ (Line 15), $x_4 \leftarrow x_4 \oplus x_{31}$ (Line 29), $x_0 \leftarrow x_0 \oplus x_{31}$ (Line 56), and so on. This does not account for extra depth in a classical circuit (as multiple fan-outs are allowed). However, in a quantum circuit where there is exactly one fan-out, this situation causes increase of quantum depth. Relevant discussion on quantum depth can be found in [56].

E Discussion about Q# Bug in JNRV (Eurocrypt'20)

Continuing from Section 5, we detail more about the Q# bug which affected the Eurocrypt'20 implementation [34]. We encountered two issues. First (non-parallelizable) and second (issue with AND gate) problems analyzed in Section 5.1 can be solved by adjusting the number of qubits. If many ancilla qubits are used, over-parallelized depth may be possible. However, the third problem (inconsistency and underestimation of full depth) in that Section 5.1 cannot be solved that way. A well-observed case of this error is the depth of AES-256 using in-place MC reported in JNRV [34]. Only 234 should be derived as depth for SubBytes $\times$ 14 rounds. This depth margin, therefore, cannot be derived even with excessive parallelization.

The Q# compiler finds non-trivial parallelism in the circuit, but according to our examples, this parallelism is excessive in the Eurocrypt'20 paper [34]. In our case also, the estimated depth of the circuit is slightly reduced, rather than being exactly equal to the product of the round number and the depth (which would indicate trivial parallelism). MixColumns requires the result from SubBytes (i.e., it operates sequentially like this: SubBytes $\rightarrow$ MixColumns $\rightarrow$ SubBytes $\rightarrow$ MixColumns), so it cannot be estimated in parallel. There is a small degree of overlap between the MixColumn operation in the current round and the SubBytes operation in the following round. However, as demonstrated by our example, this overlap is excessive. The reported depth still seems impossible because the depth of each round has to be counted independently (only slight reduction possible with trivial parallelization).

A well-observed case of this error is the depth of AES-256 using in-place MixColumn reported in [34]. The full depth of their AES-256 (in-place MixColumn) oracle is 3353. Then about 1677 (half) would be the full depth of the AES-256 circuit. However, the full depth of the in-place MixColumn is 111, so 13 rounds (excluding the last round) $\times$ 111, the full depth is already 1443. Then only 234 ($= 1677 - 1443$) should be derived as depth for SubBytes $\times$ 14 rounds. Therefore, the full depth derived from Sbox in each round should be only about 17 ($= 234 \div 14$), which cannot be derived even with excessive parallelization or omitting cleaning of ancilla qubits.

Additionally, if the full depth is estimated assuming all parallelization with bugs, the full depth for the AES variants should depend on the number of rounds. However, the full depth of AES-192, -256 (Maximov's MixColumn [43]) reported in JNRV [34] is even lower than AES-128. The lower depth of AES-192 is due to fewer key schedules (corresponding to the zig-zag structure). However, if complete

parallelism is assumed, depth should depend on the number of rounds, since key schedule works in parallel with rounds (like ours).

F Further Result

Similar to [59, Table 6], we show the per-round benchmark for our implementations of the AES family in Table 10 (using the S-box implementation with Toffoli depth 3 and 4 in Table 10(a) and 10(b), respectively).

Table 10: Quantum resources required per round for variants of AES (this work).
(a) Using S-box with Toffoli depth 4.

AES		#CNOT			#NOT	#Toffoli		TD	
	Round	☆	◎	◈	☆◎◈	☆	◎◈	☆	◎◈
128	1≀	8960	5064	6120	79	1360	680	8	4
	2	8832	8960	10016	79	1360	1360	8	4
	3	8832	8960	10016	81	1360	1360	8	4
	4	8832	8960	10016	81	1360	1360	8	4
	5	8832	8960	10016	81	1360	1360	8	4
	6	8832	8960	10016	79	1360	1360	8	4
	7	8832	8960	10016	79	1360	1360	8	4
	8	8832	8960	10016	81	1360	1360	8	4
	9	8832	8960	10016	80	1360	1360	8	4
	10	4504	4568	4568	80	680	680	4	4
192	1≀	9024	9056	10112	79	1360	1360	8	4
	2	8896	8992	10048	79	1360	1360	8	4
	3	7088	7152	8208	64	1088	1088	8	4
	4	8896	8928	9984	81	1360	1360	8	4
	5	8896	8992	10048	81	1360	1360	8	4
	6	7088	7152	8208	64	1088	1088	8	4
	7	8896	8928	9984	81	1360	1360	8	4
	8	8896	8992	10048	79	1360	1360	8	4
	9	7088	7152	8208	64	1088	1088	8	4
	10	8896	8928	9984	79	1360	1360	8	4
	11	8896	5032	6088	81	1360	680	8	4
	12	3552	3552	3552	64	544	544	4	4
256	1≀	7216	4048	5104	64	1088	544	8	4
	2	8832	8040	9096	79	1360	1224	8	4
	3	8832	8832	9888	80	1360	1360	8	4
	4	8832	8832	9888	79	1360	1360	8	4
	5	8832	8832	9888	80	1360	1360	8	4
	6	8832	8832	9888	81	1360	1360	8	4
	7	8832	8832	9888	80	1360	1360	8	4
	8	8832	8832	9888	81	1360	1360	8	4
	9	8832	8832	9888	80	1360	1360	8	4
	10	8832	8832	9888	81	1360	1360	8	4
	11	8832	8832	9888	80	1360	1360	8	4
	12	8832	8832	9888	79	1360	1360	8	4
	13	8832	8832	9888	80	1360	1360	8	4
	14	4504	4504	4504	79	680	680	4	4

≀: Including initial key XOR.

☆: Regular version.

◎: Shallow version.

◈: Shallow/low depth version.

(b) Using S-box with Toffoli depth 3.

AES	Round	#CNOT ☆	#CNOT ◎	#CNOT ◈	#NOT ☆◎◈	#Toffoli ☆	#Toffoli ◎◈	TD ☆	TD ◎◈
128	1≀	14640	7904	8960	79	3120	1560	6	3
	2	14512	14640	15696	79	3120	3120	6	3
	3	14512	14640	15696	81	3120	3120	6	3
	4	14512	14640	15696	81	3120	3120	6	3
	5	14512	14640	15696	81	3120	3120	6	3
	6	14512	14640	15696	79	3120	3120	6	3
	7	14512	14640	15696	79	3120	3120	6	3
	8	14512	14640	15696	81	3120	3120	6	3
	9	14512	14640	15696	80	3120	3120	6	3
	10	7344	7408	7408	80	1560	1560	3	3
192	1≀	14704	14736	15792	79	3120	3120	6	3
	2	14576	14672	15728	79	3120	3120	6	3
	3	11632	11696	12752	64	2496	2496	6	3
	4	14576	14608	15664	81	3120	3120	6	3
	5	14576	14672	15728	81	3120	3120	6	3
	6	11632	11696	12752	64	2496	2496	6	3
	7	14576	14608	15664	81	3120	3120	6	3
	8	14576	14672	15728	79	3120	3120	6	3
	9	11632	11696	12752	64	2496	2496	6	3
	10	14576	14608	15728	79	3120	3120	6	3
	11	14576	7872	8928	81	3120	1560	6	3
	12	5824	5824	5824	64	1248	1248	3	3
256	1≀	11760	6320	7376	64	2496	1248	6	3
	2	14512	13152	14208	79	3120	2808	6	3
	3	14512	14512	15568	80	3120	3120	6	3
	4	14512	14512	15568	79	3120	3120	6	3
	5	14512	14512	15568	80	3120	3120	6	3
	6	14512	14512	15568	81	3120	3120	6	3
	7	14512	14512	15568	80	3120	3120	6	3
	8	14512	14512	15568	81	3120	3120	6	3
	9	14512	14512	15568	80	3120	3120	6	3
	10	14512	14512	15568	81	3120	3120	6	3
	11	14512	14512	15568	80	3120	3120	6	3
	12	14512	14512	15568	79	3120	3120	6	3
	13	14512	14512	15568	80	3120	3120	6	3
	14	7344	7344	7344	79	1560	1560	3	3

≀: Including initial key XOR.

☆: Regular version.

◎: Shallow version.

◈: Shallow/low depth version.

G Response to Crypto'23 Reviewers

An earlier version of our paper was submitted to Crypto 2023. In this part, we address all the points raised by the reviewers.