Quantum Circuit Optimization with LLMs under a Structured Guideline

Kyungbae Jang[1], Hyunji Kim[2], Hwajeong Seo[2], and Anupam Chattopadhyay[1]

[1] College of Computing and Data Science, Nanyang Technological University, Singapore
[2] Division of IT Convergence Engineering, Hansung University, Seoul, South Korea
starj1023@gmail.com, khj1594012@gmail.com, hwajeong84@gmail.com, anupam@ntu.edu.sg

Abstract. The cost of quantum cryptanalysis is dominated by the quantum circuit of the target cipher. Estimating the quantum attack cost of a cipher thus requires building that circuit and measuring its qubit count, Toffoli count, and Toffoli depth. This is manual work that needs expert knowledge and must be redone for each cipher and each cost target. Large language models handle ordinary programming well, but their use in constructing quantum circuits for ciphers is still limited. In this work, we collect quantum circuit optimization techniques that apply across many ciphers. We write these techniques into a guideline for a general-purpose LLM. Given this guideline and a single target cipher, the model produces two circuits. One minimizes the qubit count, and the other minimizes the Toffoli depth. Each circuit is verified against the test vectors of the cipher before its resources are estimated. Using this approach, we implement quantum circuits of CRAFT, MANTIS, QARMA, mCrypton, EPCBC, and Pyjamask for which quantum circuit implementations have not previously been reported. We further apply the same approach to ciphers with existing implementations. Without access to prior results, the generated circuits reach resource counts comparable to manually optimized ones.

Keywords: Quantum Computing · Quantum Circuit · Quantum Cryptanalysis · Large Language Model

1 Introduction

Quantum computers change how we judge the security of symmetric ciphers. Grover's algorithm searches a key space of size N in about $\sqrt{N}$ steps, so the real question is how costly that search actually is. NIST set the security levels in its post-quantum standardization from the concrete cost of running Grover's algorithm against AES [7,14]. The largest part of that cost is the quantum circuit of the cipher itself, which sits at the core of the Grover oracle. Implementing these circuits and measuring their cost in qubits, Toffoli gates, and Toffoli depth has thus become an active research topic. Building such a circuit is manual work that needs expert knowledge and must be redone for each cipher and each cost

target. Much of this knowledge is never written down and is reused by hand for every new cipher.

Large language models work well on ordinary programming, but their use for quantum circuit construction is still limited. Most of the existing work [16,6,23] targets variational circuits and OpenQASM generation, and reversible circuits for ciphers have drawn little attention. On its own, a model often returns circuits that are incorrect or circuits that work but use more qubits or gates than needed [23]. Other fields face the same problem and solve it by giving the model clear written instructions instead of retraining it. MetaGPT puts standard operating procedures into the prompts of its agents [8]. An agent skill groups instructions, scripts, and references into a module that is loaded only when needed [24]. A controlled study also shows that guidelines written by domain experts help a lot, while guidelines the model writes for itself do not [18].

We take the same approach for quantum circuit construction. We gather optimization techniques that work for many ciphers rather than just one, and write them up as a guideline for a general-purpose language model. With a reversible circuit simulator and resource counters, the model uses this guideline to build the circuits for target ciphers. For each cipher it produces two versions, one that lowers the qubit count and one that lowers the Toffoli depth. Using this guideline, we implement quantum circuits of symmetric ciphers, CRAFT, MANTIS, QARMA, mCrypton, EPCBC, and Pyjamask. Note that we adopted these ciphers because no quantum circuit for them has been public so far. We also apply the guideline to ciphers that already have hand-optimized implementations, LED, PRESENT, GIFT, and PRINCE [22,12,2], which lets us check the performance of the generated circuits. Note that the model was given no access to the implementations in [22,12,2].

1.1 Contribution

Our contributions are as follows:

- We write a guideline that collects quantum circuit optimization techniques reusable across ciphers. It is packaged as an agent skill, so a general-purpose LLM can use it without fine-tuning. We treat two targets separately, qubit count and Toffoli depth, since the two rarely admit the same circuit.
- We use the guideline to build quantum circuits for CRAFT, MANTIS, QARMA, mCrypton, EPCBC, and Pyjamask. None of these ciphers has a reported quantum circuit implementation. We verify each circuit against the test vectors and report its qubit count, Toffoli count, and Toffoli depth.
- We repeat the process on ciphers that already have hand-optimized implementations: LED, PRESENT, GIFT, and PRINCE [22,12,2]. The model is given no access to these results, yet it achieves performance close to the reported costs.

2 Foundations

2.1 Grover's Exhaustive Key Search

Grover's algorithm provides a quantum speedup for searching the key space of a cipher, cutting the classical search effort by a square-root factor. A k-bit key that costs $O(2^k)$ to search exhaustively on a classical machine can be recovered in about $\sqrt{2^k}$ steps with Grover search on a quantum computer. Given a known plaintext-ciphertext pair, the search for a k-bit key proceeds as *prepare* $\rightarrow$ (*Grover oracle* and *diffusion operator*)$^{\sqrt{2^k}}$ $\rightarrow$ *measure*.

The procedure starts by placing the key in superposition: a Hadamard (H) gate on each of the k qubits spreads the state evenly over all 2^k key values, as in Equation 1.

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

X gates then set the plaintext register according to the known plaintext P_{known}. At the heart of the oracle sits the quantum circuit of the target cipher: it encrypts P_{known} under the k-qubit key $\psi(k)$, yielding a superposition of the ciphertexts produced by every candidate key. The oracle compares each of these with the known ciphertext C_{known}, and whenever they agree (Equation 2) it marks that key by flipping the sign of the corresponding state (Equation 3).

$$f(x) = \begin{cases} 1 \text{ if } Enc_{\psi(k)}(P_{known}) = C_{known} \\ 0 \text{ if } Enc_{\psi(k)}(P_{known}) \neq C_{known} \end{cases} \tag{2}$$

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

The diffusion operator then raises the amplitude of the key marked by the oracle, making the correct key more likely to be observed. Applying the oracle and the diffusion operator in turn, repeated about $\sqrt{2^k}$ times, builds the amplitude of the solution high enough for the key to be measured with high probability. Because the encryption circuit of the cipher is the dominant component inside the oracle, reducing the resources it consumes is the key lever for lowering the overall cost of the Grover key search.

2.2 NIST Post-quantum Security

To measure how well a cipher resists quantum attacks, NIST set resource bounds for several levels [19,20]:

- Level 1: An attack costs about as much as breaking AES-128 ($2^{170} \rightarrow \mathbf{2^{157}}$).
- Level 2: An attack costs about as much as breaking SHA-256/SHA3-256.
- Level 3: An attack costs about as much as breaking AES-192 ($2^{233} \rightarrow \mathbf{2^{221}}$).

– Level 4: An attack costs about as much as breaking SHA-384/SHA3-384.
– Level 5: An attack costs about as much as breaking AES-256 ($2^{298} \rightarrow \mathbf{2^{285}}$).

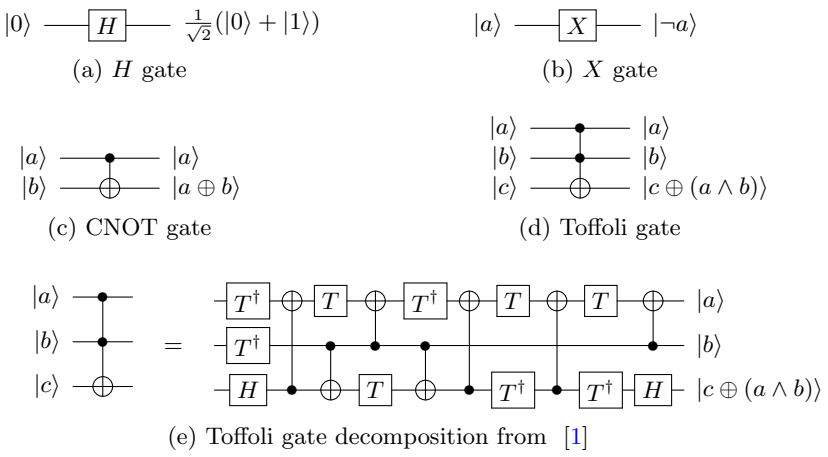
For Levels 1, 3, and 5, which correspond to the three AES variants, NIST took the attack complexities to be 2^{170}, 2^{233}, and 2^{298}, obtained as the total gate count multiplied by the depth of the Grover search, following the cost estimates of Grover key search on AES reported by Grassl et al. [7]. These figures in [19] therefore come from the PQCrypto'16 results [7]. Quantum circuits for AES have been refined steadily since then, and the cost of the attack has dropped considerably in recent years [15,25,10,9]. NIST itself notes that these per-level complexities are relative, precisely because quantum circuits for AES keep improving (see page 17 in [19]); a more efficient attack would call for the benchmarks to be revised. NIST has since updated the AES bounds [20] in light of the Eurocrypt 2020 results [15], where the quantum attack costs for AES-128, -192, and -256 fell to 2^{157}, 2^{221}, and 2^{285}, matching the revised values in [20].

NIST MAXDEPTH Even with Grover search, an exhaustive key search is still far out of reach for current quantum computers. The square-root reduction in security is only a theoretical bound. A working attack would have to run a quantum circuit of enormous depth. To capture this, NIST fixes a maximum allowable depth for a quantum attack (see page 46 of [20]). This limit is called MAXDEPTH and is set between 2^{40} and 2^{96}. Once an attacker exceeds MAXDEPTH, the Grover key search must be carried out in parallel by either an outer or an inner method [17]. However, parallelization of Grover's search is generally inefficient. This is because the reduction in depth is small compared with the loss in success probability. The product of depth and qubit count is a common measure of how good a quantum circuit is. But lowering the depth of the Grover search by a factor of S requires running S^2 instances in parallel [17,15]. The relevant cost for a parallel Grover search is therefore the depth^2 times qubit count rather than the depth times qubit count. For this reason, keeping the depth low becomes the more useful goal under parallelization.

2.3 Quantum Gates

Cryptographic algorithms are built in quantum circuits from a small set of gates, mainly the X (NOT), CNOT, and Toffoli (CCNOT) gates. The X gate is the quantum form of the classical NOT and inverts a qubit ($\mathrm{X}(x) = \sim x$). The CNOT gate acts on two qubits, a control and a target, and flips the target when the control is 1 and leaves it unchanged when the control is 0 ($\mathrm{CNOT}(x, y) = (x, x \oplus y)$). This is exactly an XOR of the control into the target, so CNOT plays the role of the classical XOR. The Toffoli gate acts on three qubits, two controls and one target, and flips the target only when both controls are 1 ($\mathrm{Toffoli}(x, y, z) = (x, y, z \oplus xy)$).

This XORs the AND of the two controls into the target, so the Toffoli gate plays the role of the classical AND. With these gates we can implement encryption algorithms on a quantum computer, replacing the classical NOT, XOR, and AND operations.



(a) H gate (b) X gate (c) CNOT gate (d) Toffoli gate

(e) Toffoli gate decomposition from [1]

Fig. 1: Common quantum gates for quantum circuit implementations.

Optimizing a quantum circuit centers on reducing the number of Toffoli gates, since they are the most expensive to implement. A Toffoli gate is realized from T gates, which determine the T-depth, together with Clifford gates. Several ways to decompose the Toffoli gate exist, and the term *full depth* denotes the circuit depth after this decomposition. In this work we estimate resources using a decomposition with 7 T gates and 8 Clifford gates, which gives a T-depth of 4 and a full depth of 8 for one Toffoli gate [1].

3 Quantum Circuit Optimization with LLMs

In this section, we present our method for deriving optimized quantum circuits with LLMs. Figure 2 illustrates the proposed workflow for building optimized quantum circuits with LLMs. First, we create guideline for LLMs how to implement correct quantum circuits and how to optimize.

3.1 Guideline for LLMs

In the prompt, we include the guideline `Guideline.md`. Using it, the LLM generates a reversible quantum circuit for the encryption function of the cipher. The guideline covers various aspects of quantum implementation and its optimization, including verification. Note that it is designed for a wide range of ciphers rather than a specific one.

Verification of Quantum Circuits First, the guideline specifies what makes a quantum circuit valid for Grover based key search. Without this, we observed that the LLM sometimes reduces the resources of its implementation by fixing the key to a constant value, which makes the circuit invalid for quantum key search.

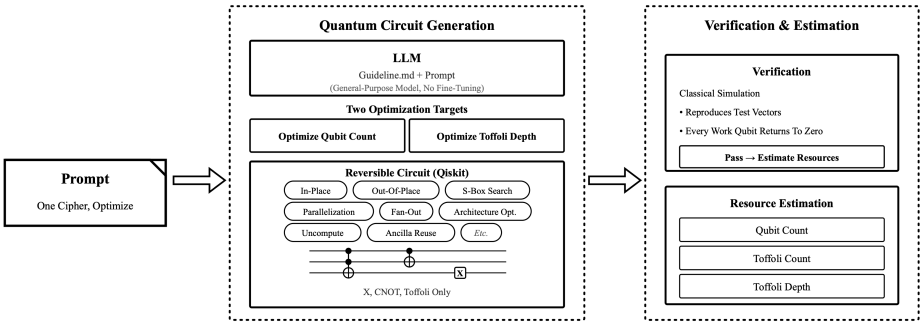


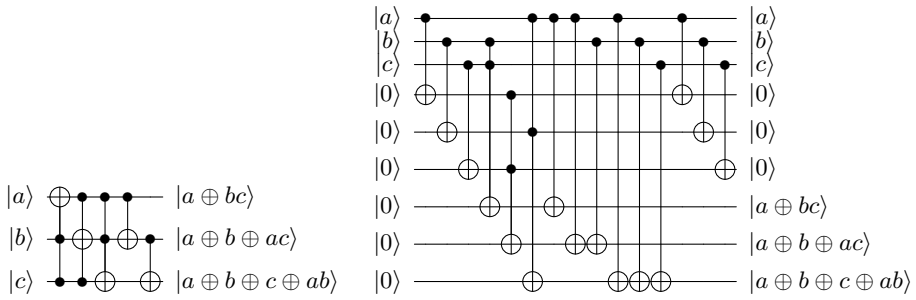
Fig. 2: Overview of the proposed LLM-guided quantum circuit generation

Following the guideline, the input plaintext and the input key are allocated as quantum registers. The generated circuit must then be verified against test vectors. Once it passes, the quantum resources it uses are estimated in terms of qubit count, Toffoli count, and Toffoli depth.

Optimization Strategy In our guideline, two optimization goals are specified and a separate circuit for each is generated by the LLM. One minimizes the qubit count, and the other minimizes the Toffoli depth. Depending on the optimization strategy, the LLM can select from the pool of techniques in the guideline those best suited to that goal. The guideline contains several techniques that can be applied when optimizing quantum circuits for cryptographic algorithms. Importantly, these techniques are generic, since they apply to a wide range of ciphers rather than to a single one.

In-Place and Out-of-Place Implementations In-place and out-of-place approaches are two common options for implementing quantum circuits. The in-place method computes the output on the input qubits, so it requires no separate output qubits. On the other hand, the out-of-place method allocates new output qubits and computes the result on them. Figure 3 shows an example of in-place and out-of-place implementations for same function. As expected, the in-place method suits the qubit count optimization, whereas the out-of-place method targets the Toffoli depth optimization by enabling greater parallelization. Beyond Figure 3, in-place and out-of-place implementations come in many forms. For example, among quantum adders, the Takahashi's adder [21] is in-place and suited to the qubit count optimization, whereas the Draper's out-of-place adder [5] is suited to the Toffoli depth optimization.

Recycling Work Qubits To reduce the qubit count, ancilla qubits are often reused through uncomputation. In the qubit count optimization, we uncompute the circuit to reset the ancilla qubits to $|0\rangle$ and then reuse them for later operations. Figure 4 illustrates this concept: once the result has been copied out, applying the $f^\dagger$ operation uncomputes the f operation and restores the work qubit



(a) In-place. (b) Out-of-place.

Fig. 3: In-place and shallow out-of-place implementations of the same function.

so that it can be reused. The technique is central to qubit optimization and is applied broadly across quantum implementations. Note that uncomputation roughly doubles the circuit depth.

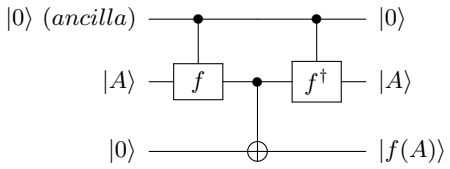
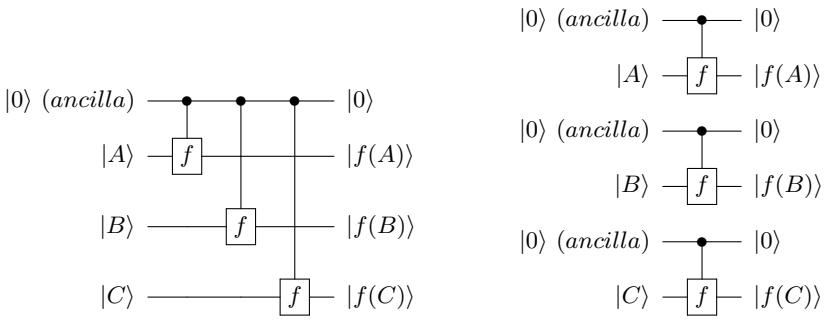


Fig. 4: Cleaning a qubit through uncomputation ($f^\dagger$).

Balancing Qubit Count and Depth Optimizing too heavily for one side often fails to achieve balanced performance. Suppose that one ancilla qubit is required for each f operation and that a single round needs 3 f operations. To minimize the qubit count, we can reuse a single ancilla qubit across all 3 operations. Figure 5(a) illustrates this approach. However, this forces the operations to run serially and increases the depth substantially. In such a case, it is better to parallelize the f operations by allocating 3 ancilla qubits and reusing them across rounds. Figure 5(b) shows this strategy. The right choice therefore depends on the target, and a good implementation balances qubit count against depth rather than optimizing either in isolation[3]. Thus, the guideline advises that when a relatively small number of additional qubits can improve parallelization effectively, those qubits should be allocated.

S-box Synthesis The S-box dominates the Toffoli cost and largely determines the overall optimization of the quantum circuit. We search over $\{X, \mathrm{CNOT}, \mathrm{CCX}\}$

[3] As a side note, this trade-off has already been discussed for the SubBytes implementation of AES quantum circuits in [11, Section 3.3].



(a) Sequential operation (b) Parallelization

Fig. 5: Reuse of a single ancilla versus multiple ancillas.

for the ancilla-free in-place implementation with the fewest Toffoli gates, which gives a provably minimal result[4]. An exact search matters here because approximate methods do not guarantee a minimal Toffoli count and may introduce multi-controlled gates (C^nX, $n \geq 3$) that fall outside this gate set. For our Toffoli depth optimization, a minimal Toffoli count does not imply a minimal depth. An in-place implementation shares wires and runs its Toffoli gates serially, so reducing the depth requires an out-of-place implementation.

Optimization at the Architecture Level The optimizations described so far act on individual operations. A higher level of optimization looks beyond a single operation and considers how the rounds of the cipher are composed as a whole. Inspired by the *shallow* architecture of [11], we generalize this architecture-level optimization. We argue that this optimization applies generically to any cipher with a round-based structure. Figure 6 shows the quantum circuit of the shallow architecture. The uncomputation of one round (i.e., $f^\dagger$) is run in parallel with the normal computation (i.e., f) of the next round. To enable this overlap two sets of ancilla qubits (i.e., $ancilla_0$ and $ancilla_1$) are maintained, one for each of the two rounds that are active at the same time. Put simply, one ancilla set is used for odd rounds and another ancilla set is used for even rounds.

One Cipher per Prompt. Asking the LLM to implement multiple ciphers in a single prompt highly degrades its output. This effect is consistent with prior reports that batched or multi-task prompting degrades LLM performance [4,3]. For quantum circuit optimization, the degradation appears far more severe.

4 Results

In this section we report the quantum circuits produced under the guideline. The model receives the guideline together with a prompt naming a single target

[4]This approach does not scale; for larger S-boxes the space is too large to enumerate, and heuristic synthesis becomes necessary.

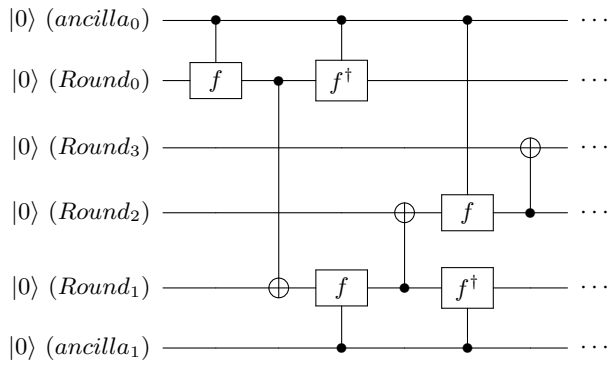


Fig. 6: Generic form of the shallow architecture in [11].

cipher and returns two circuits: one minimizing the qubit count (*Q opt*) and one minimizing the Toffoli depth (*D opt*). Each circuit is simulated classically and is accepted only when it reproduces the test vectors of the cipher. For an accepted circuit we then estimate the qubit count, the Toffoli count, and the Toffoli depth.

We adopt `GPT-5.5` as the general-purpose language model, with no fine-tuning. To keep the runs independent and reproducible, we generate every circuit in a temporary chat session with memory disabled, so that no context from earlier conversations carries over into the current generation.

This work focuses on ciphers for which no quantum circuit implementation has previously been publicly available: CRAFT, MANTIS, QARMA, mCrypton, EPCBC, and Pyjamask. Table 1 reports these results. For the Q opt version, the generated circuits use the optimal number of qubits, equal to the sum of the block, tweak, and key sizes. For the D opt version, the circuits allocate additional qubits to reduce the Toffoli depth.

Table 2 compares the circuits generated by our workflow with previously reported implementations of the same ciphers, LED, PRESENT, GIFT, and PRINCE [22,12,2]. In the qubit-optimized setting the results are nearly identical. Lightweight block ciphers have simple round structures, and the qubit count is determined mainly by the in-place S-box construction. The minimum is therefore close to fixed and leaves little room for divergence between independent implementations. We do not claim an improvement here. The agreement does indicate that the generated circuits reach the same resource figures as manually optimized ones. To rule out the possibility that this comes from recall of published circuits, we prepared the generation environment so that no prior implementation, paper, or reported resource count was accessible to the model. Each circuit was derived only from the cipher specification and our implementation guideline

4.1 Discussion

Our guideline based approach performs well on lightweight ciphers with relatively simple round structures. For these ciphers the generic techniques in the guideline

Table 1: Quantum resource estimation for lightweight block ciphers without prior full-cipher quantum implementations.

Cipher	Version	Qubits	Toffoli count	Toffoli depth
CRAFT-64/128	Q opt	256	1984	124
	D opt	2720	7680	62
MANTIS5-64/128	Q opt	256	768	48
	D opt	1600	3168	24
MANTIS7-64/128	Q opt	256	1024	64
	D opt	1856	4320	32
QARMA64/128	Q opt	256	1280	80
	D opt	1888	4800	32
QARMA128/256	Q opt	512	3840	120
	D opt	4800	14720	48
mCrypton-64/64	Q opt	148	2072	182
	D opt	1456	3760	38
EPCBC-48/96	Q opt	144	4608	256
	D opt	4080	11904	64
Pyjamask-96/128	Q opt	224	1344	42
	D opt	1952	2496	14
Pyjamask-128/128	Q opt	256	1792	56
	D opt	2688	3328	28

are sufficient, and the LLM produces correct circuits whose qubit count, Toffoli count, and Toffoli depth are competitive. However, it does not yet match highly optimized implementations for ciphers with more complex structures. AES is a representative example. Starting from the first quantum resource estimates of Grassl et al. [7], a long line of work has steadily refined the AES quantum circuit through increasingly dedicated optimizations [14,9,10], reaching a very high level today. Our guideline focuses on techniques that generalize across many ciphers. Extending it toward the high-level optimizations used in well-studied implementations is left as future work.

5 Conclusion

We study whether a general purpose LLM can build optimized quantum circuits for symmetric ciphers when it is handed a written guideline instead of being fine tuned. The guideline collects optimization techniques that the model can follow on its own. With this single artifact and no model training, the LLM produces circuits for CRAFT, MANTIS, QARMA, mCrypton, EPCBC, and Pyjamask, none of which have a public quantum implementation. Note that every circuit passes verification before its resources are estimated.

Table 2: Comparison of the generated circuits with previously reported implementations.

Cipher	Source	Qubits	Toffoli count	Toffoli depth
LED-64	Wang et al. [22]	128	2048	128
	This paper (Q opt)	128	2048	128
	This paper (D opt)	2912	10912	64
PRESENT-80	Jang et al. [13]	144	2108	124
	This paper (Q opt)	144	2108	124
	This paper (D opt)	5628	11220	62
GIFT-64	Jang et al. [13]	192	1792	112
	This paper (Q opt)	192	1792	112
	This paper (D opt)	2688	6912	56
PRINCE	Bonnetain and Jaques [2]	192	1152	72
	This paper (Q opt)	192	960	60
	This paper (D opt)	2176	4768	24

The guideline treats the two optimization objectives separately. For each cipher the model returns one circuit that minimizes the qubit count and one that minimizes the Toffoli depth. Overall, this approach produces correct and competitive circuits for target ciphers. However, it does not yet reach the highly optimized implementations developed through extensive prior research, as the AES case shows. In this context, closing this gap by adding high level and cipher specific techniques to the guideline remains future work.

6 Acknowledgment

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. RS-2019-II190033, Study on Quantum Security Evaluation of Cryptography based on Computational Quantum Complexity, 50%) and this work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2025-23523436, 50%).

References

1. Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **32**(6), 818–830 (Jun 2013). https://doi.org/10.1109/tcad.2013.2244643, http://dx.doi.org/10.1109/TCAD.2013.2244643 5
2. Bonnetain, X., Jaques, S.: Quantum period finding against symmetric primitives in practice. arXiv preprint arXiv:2011.07022 (2020) 2, 9, 11

3. Chen, J., Pilehvar, M.T., Camacho-Collados, J.: Understanding llm performance degradation in multi-instance processing: The roles of instance count and context length. arXiv preprint arXiv:2603.22608 (2026) 8
4. Di Maio, F., Gozzi, M.: Degradation of multi-task prompting across six nlp tasks and llm families. Electronics **14**(21), 4349 (2025) 8
5. Draper, T.G., Kutin, S.A., Rains, E.M., Svore, K.M.: A logarithmic-depth quantum carry-lookahead adder. arXiv preprint quant-ph/0406142 (2004) 6
6. Fu, Z., Chen, F., Jiang, L.: Qagent: An llm-based multi-agent system for autonomous openqasm programming. arXiv preprint arXiv:2508.20134 (2025) 2
7. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: Quantum resource estimates. In: Takagi, T. (ed.) Post-Quantum Cryptography. pp. 29–43. Springer International Publishing, Cham (2016) 1, 4, 10
8. Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Yau, S., Lin, Z., Zhou, L., et al.: Metagpt: Meta programming for a multi-agent collaborative framework. In: International Conference on Learning Representations. vol. 2024, pp. 23247–23275 (2024) 2
9. Huang, Z., Sun, S.: Synthesizing quantum circuits of AES with lower T-depth and less qubits. Cryptology ePrint Archive, Report 2022/620 (2022), https://eprint.iacr.org/2022/620 4, 10
10. Jang, K., Baksi, A., Kim, H., Song, G., Seo, H., Chattopadhyay, A.: Quantum analysis of aes. Cryptology ePrint Archive, Paper 2022/683 (2022), https://eprint.iacr.org/2022/683, https://eprint.iacr.org/2022/683 4, 10
11. Jang, K., Baksi, A., Kim, H., Song, G., Seo, H., Chattopadhyay, A.: Quantum analysis of aes. IACR Communications in Cryptology **2**(1) (2025) 7, 8, 9
12. 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), 4776 (2021) 2, 9
13. 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), https://www.mdpi.com/2076-3417/11/11/4776 11
14. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and LowMC. Cryptology ePrint Archive, Report 2019/1146 (2019), https://eprint.iacr.org/2019/1146 1, 10
15. Jaques, S., Naehrig, M., Roetteler, M., Virdia, F.: Implementing grover oracles for quantum key search on AES and lowmc. In: Canteaut, A., Ishai, Y. (eds.) Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12106, pp. 280–310. Springer (2020). https://doi.org/10.1007/978-3-030-45724-2_10, https://doi.org/10.1007/978-3-030-45724-2_10 4
16. Jern, L., Uotila, V., Yu, C., Zhao, B.: Agent-q: fine-tuning large language models for quantum circuit generation and optimization. In: 2025 IEEE International Conference on Quantum Computing and Engineering (QCE). vol. 1, pp. 1621–1632. IEEE (2025) 2
17. Kim, P., Han, D., Jeong, K.C.: Time–space complexity of quantum search algorithms in symmetric cryptanalysis: applying to aes and sha-2. Quantum Information Processing **17**, 1–39 (2018) 4
18. Li, X., Chen, W., Liu, Y., Zheng, S., Chen, X., He, Y., Li, Y., You, B., Shen, H., Sun, J., et al.: Skillsbench: Benchmarking how well agent skills work across diverse tasks. arXiv preprint arXiv:2602.12670 (2026) 2

19. 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 3, 4
20. NIST.: Stateless hash-based digital signature standard (2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.ipd.pdf 3, 4
21. Takahashi, Y., Tani, S., Kunihiro, N.: Quantum addition circuits and unbounded fan-out (2009), https://arxiv.org/abs/0910.2530 6
22. Wang, S., Jang, K., Baksi, A., Chakraborty, S., Lee, B., Chattopadhyay, A., Seo, H.: New results in quantum analysis of led: Featuring one and two oracle attacks. In: International Conference on Cryptology in India. pp. 339–364. Springer (2025) 2, 9, 11
23. Yu, C., Uotila, V., Deng, S., Wu, Q., Shi, T., Jiang, S., You, L., Zhao, B.: Quasar: Quantum assembly code generation using tool-augmented llms via agentic rl. arXiv preprint arXiv:2510.00967 (2025) 2
24. Zhang, B., Lazuka, K., Murag, M.: Equipping agents for the real world with agent skills. Anthropic Engineering Blog (2025) 2
25. Zou, J., Wei, Z., Sun, S., Liu, X., Wu, W.: Quantum circuit implementations of AES with fewer qubits. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020. pp. 697–726. Springer International Publishing, Cham (2020) 4