scientific reports

Check for updates

OPEN

# Implementing Grover's on AES-based AEAD schemes

Surajit Mandal[1], Ravi Anand[2,3]✉, Mostafizar Rahman[2]✉, Santanu Sarkar[1] & Takanori Isobe[3]

**Extensive research is currently underway to determine the security of existing ciphers in light of the advancements in quantum computing. Against symmetric key cryptography, Grover's search algorithm is a prominent attack, capable of reducing search costs to the square root. For using Grover's algorithm, it is imperative to embed the target cipher into a quantum circuit. Even so, this area of research is relatively new; it has garnered significant attention from the research community. In this study, we provide the first estimation of the cost of Grover's key search attack against the AES-based AEAD schemes `Rocca-S`, `AEGIS-128`, and `Tiaoxin-346`. Our analysis considers circuit depth restrictions specified in NIST's PQC standardization process. Considering NIST's maximum depth constraints, We present the overall cost of these attacks using gate count and depth-times-width metrics. We observed that for MAXDEPTH $= 2^{40}$, `Rocca-S`, `AEGIS-128`, and `Tiaoxin-346` can be retrieved using Grover's search algorithm with gate count of $1.09 \times 2^{253}$, $1.14 \times 2^{124}$, and $1.22 \times 2^{124}$ respectively. Concerning the current updated values by NIST, these ciphers are secure in terms of the cost of implementing Grover's attack for key recovery. The quantum circuits of these ciphers are implemented using QISKIT, an open-source software development kit (SDK) designed for working with quantum computers running on the IBM Quantum Experience platform.**

**Keywords** Grovers, `Rocca-S`, `AEGIS-128`, `Tiaoxin-346`, QISKIT

The emergence of quantum computers poses a critical challenge for cryptographic security, necessitating a reevaluation of post-quantum defenses in cryptographic systems. Shor's algorithm[1,2] presents a direct threat to public-key encryption methods like RSA, ECDH, and ECDSA by efficiently solving factoring and discrete logarithm problems. Consequently, researchers have focused on crafting public-key cryptographic systems resilient against quantum adversaries. Meanwhile, symmetric encryption schemes like block ciphers and hash functions were traditionally considered immune to quantum threats. However, Grover's algorithm alters this perspective, offering a quadratic speedup in an exhaustive key search for symmetric ciphers and hash functions.

To counter this, a cautious security approach suggests doubling the security parameters by augmenting the key or output size of hash functions, mitigating potential impacts from Grover's algorithm. Yet, assessing the cost implications of applying Grover's algorithm on symmetric ciphers is imperative. Recent studies[3–5] have endeavored to gauge the computational expenses associated with utilizing the Grover search algorithm across different versions of the Advanced Encryption Standard (AES). These investigations have illuminated potential vulnerabilities in symmetric ciphers against quantum attacks, challenging the notion that symmetric-key cryptography is impervious to quantum advancements.

Advancements in quantum cryptanalysis have further demonstrated that symmetric-key cryptography can be efficiently compromised or significantly weakened using the capabilities of quantum computers[6–12]. However, the extent of compromise hinges on the expertise and capabilities of the potential attacker. Mere augmentation of key sizes may not suffice in the long run to ensure communication network security in the presence of quantum computers. This underscores the pressing need for a deeper comprehension of these potential threats and the development of more robust defense mechanisms.

To effectively gauge the security of a quantum encryption system, it becomes essential to construct a quantum circuit for encryption and evaluate the practical expenses associated with implementing a quantum attack. The complexity of such an attack can be quantified in terms of the quantum circuit's width (number of qubits), gate count, and depth. Numerous scholarly articles have concentrated on adapting symmetric ciphers into a quantum framework and estimating the computational resources necessary for executing Grover's key search algorithm on these ciphers. For deeper insights, interested individuals can refer to several key references: SIMON[13], SPECK[14,15], GIFT[16], PRESENT[17], ChaCha[18], RECTANGLE[19], DEFAULT[20]. Determining the precise cost of quantum gates presents challenges due to the early stage of quantum computing. Ongoing research has predominantly focused

[1]Indian Institute of Technology Madras, Chennai, India. [2]Indraprastha Institute of Technology Delhi, New Delhi, India. [3]University of Hyogo, Kobe, Japan. ✉email: ravianandsps@gmail.com; mrahman454@gmail.com

on reducing circuit depth, particularly emphasizing T-depth (defined by the minimum count of steps where T-gates are simultaneously applied in a circuit, allowing for parallel application on distinct qubits). This emphasis holds dual significance as elaborated in references like[5,21].

Firstly, various metrics suggest reduced overall costs by accepting a slightly larger number of qubits in exchange for diminished circuit depth. This trade-off potentially yields computational benefits concerning resource utilization. Secondly, as underscored in[22], the runtime of quantum algorithms predominantly hinges on T-depth rather than gate count, circuit depth, or measurement depth. Therefore, optimizing T-depth emerges as critical for achieving efficient execution.

Certainly, Grover's algorithm faces limitations in effective parallelization. Therefore, the emphasis on reducing depth rather than width (the number of qubits) becomes crucial for efficient resource utilization. NIST[23] has recognized this and introduced the MAXDEPTH constraint, considering the challenges of running excessively prolonged serial computations. NIST recommended a minimum security level for a given cipher to ensure ample security in the post-quantum era. Drawing from available research, likely the sole work at that time by[3], NIST estimated the complexities in[23] as follows: Level 1: $2^{170}$ (applicable to 128-bit key ciphers), Level 3: $2^{233}$ (for 192-bit key ciphers), and Level 5: $2^{298}$ (for 256-bit key ciphers). These complexity bounds were computed by multiplying the total number of decomposed gates ($G$-cost) and the full depth ($D$) required for Grover's key search circuit. Consequently, this research intends to minimize circuit depth and quantum gate usage. By investigating cipher security under NIST's MAXDEPTH limitation, the focus rests on optimizing depth while allowing for a slight increase in qubit count, ultimately aiming for enhanced efficiency.

Inspired by the work of Jean and Nikolic[24], which explores various constructions based on the AES (Advanced Encryption Standard) round function, this cipher's design draws upon their structures as foundational elements. These structures have been instrumental in crafting message authentication codes (MAC) and authenticated encryption with associated data (AEAD) schemes. Notably, ciphers like `AEGIS`[25] and `Tiaoxin-346`[26] are influenced by Jean and Nikolic's constructions. These ciphers were contenders in the CAESAR competition to identify secure and efficient authenticated ciphers. Among the submissions, `AEGIS-128` was chosen as part of the final portfolio for high-performance applications. It's worth noting that the round functions in both `AEGIS` and `Tiaoxin-346` exhibit significant similarities, likely due to the influence of Jean and Nikolic's work.

`AEGIS`, `Tiaoxin-346`, and `Rocca-S` represent encryption schemes rooted in the AES framework. Hence, their architectures are intricately tied to the structure of AES round operations. Notably, there's been a surge in research endeavors focused on AES quantum circuit optimization, surpassing earlier achievements[3,5,21,27–30].

## Our contributions

Our main focus centers on estimating the cost of Grover's key search concerning the `Rocca-S`, `Tiaoxin-346`, and `AEGIS` ciphers. Evaluating a cipher's security against quantum adversaries necessitates the creation of a dedicated quantum circuit. There's a common assumption that T-depth significantly impacts the runtime of fault-tolerant quantum computation. Thus, our efforts have been channeled into minimizing T-depth during the cipher construction. Through this construction approach, we gauge the implementation costs of Grover's algorithm for key search and subsequently compare these expenses with those associated with AES. Our source code will be made available on a public repository soon.

It has been established that Grover's search algorithm reduces the security of symmetric key cryptosystems in half. What we need to consider is the necessary cost. If a huge cost is required to apply the Grover search algorithm, it can be judged that it is resistant to attacks by quantum computers. We show an analysis of the circuit complexity required to mount Grover's search for these ciphers.

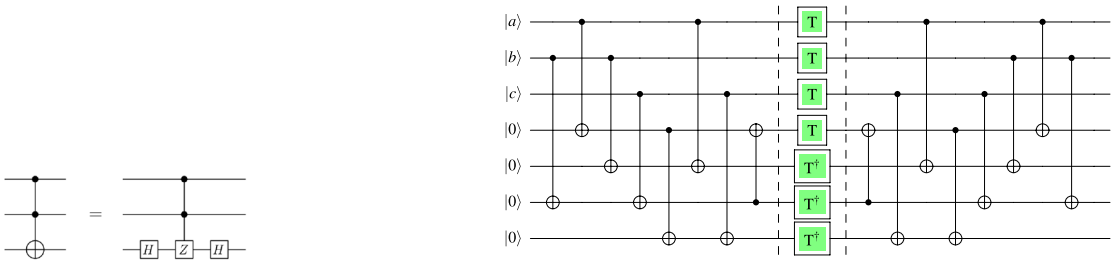
***Remark 1*** Based on the updated NIST recommendations[31], Page 45, denoted by $\text{NIST}_{up}$ in Table 1, our results show that these ciphers are secure in terms of the cost of implementing Grovers attack for key recovery. NIST[31], Page 45 adjusted the threshold for quantum security to $2^{157}$ for AES-128 and $2^{285}$ for AES-256, based on Jaques et al.'s work[32]. However, readers should note that there was an error in the original estimation provided by[32], which has recently been corrected in their revised paper. Despite this correction, the new NIST values have not yet been updated. Nonetheless, the complexities for quantum security mentioned in[31] are still achievable as discussed in[28]. Furthermore, it can also be seen that the ciphers analyzed in this work failed to meet the threshold recommended in the initial draft of the Post-Quantum Cryptography (PQC) standardization organized by the US National Institute of Standards and Technology (NIST)[23], Page 17-18.

## Organization of the paper

We organize the paper as follows.

| **key-size: $k$** | **128** | | | | **256** | | |
|---|---|---|---|---|---|---|---|
| Cipher | `AEGIS-128` | `TIAOXIN` | NIST [23] | $\text{NIST}_{up}$ [31] | `Rocca-S` | NIST [23] | $\text{NIST}_{up}$ [31] |
| $G$-cost × $D$ | $2^{164}$ | $2^{164}$ | $2^{170}$ | $2^{157}$ | $2^{294}$ | $2^{298}$ | $2^{285}$ |

**Table 1.** Comparison of the gate-cost times circuit-depth, represented as $G$-cost × $D$, of the proposed attacks with the NIST's security bound (shown under the column titled 'NIST'). $\text{NIST}_{up}$: NIST adjusted the threshold for quantum security in[31], Page 45, citing Jaques et al.'s work[32].




**(a)** Toffoli gate through alteration of basis

**(b)** Doubly controlled Z-Gate

**Figure 1.** Toffoli gate expressed with T-depth 1 representation[33].

1. We commence by going through some prerequisites and conventions required for the readers, such as Quantum basics, Grover's search algorithm, MAXDEPTH concept, and depth constraints metric for cryptanalysis in Section 2.
2. In Section 3 the quantum resource estimations for AES-128 components are demonstrated. Since the quantum circuit of `Rocca-S`, `Tiaoxin-346`, `AEGIS` depends totally on the quantum circuit of Subbytes, ShiftRows, and MixColumns operations within AES.
3. In Section 4, 5, and 6 we outline the creation of quantum circuits of `Rocca-S`, `Tiaoxin-346`, `AEGIS-128` respectively with a target to decrease the depth of the circuits at the cost of some qubits.
4. Finally, in Section 7, we estimate the cost of implementing Grover's key search algorithm, then find resource estimation for key search under NISTs MAXDEPTH metric.
5. We conclude in Section 8.

## Preliminaries

### Quantum basics

In quantum computation, qubits, the quantum counterparts of classical binary bits, undergo manipulation by applying quantum gates. A qubit stands as the foundational unit of quantum information. Describing the state of a 2-qubit quantum system involves expressing it as $|\psi\rangle = a|0\rangle + b|1\rangle$, where $a, b$ are complex numbers that meet the condition $| a |^2 + | b |^2 = 1$. More broadly, in using the orthonormal basis $\{|00\ldots00\rangle, |00\ldots01\rangle, \ldots, |11\ldots11\rangle\}$, an n-qubit system's state finds representation as unit vectors within $\mathbf{C}^{2^n}$. Quantum algorithms are formulated by amalgamating quantum gates, visualized as quantum circuits. These circuits delineate the sequential application of specific quantum operations (gates) aimed at processing input and generating the desired output. By scrutinizing the gate sequence within a quantum circuit, we gain insight into the system's behavior and compute the likelihood of measuring a particular output state. Notably, all quantum computations maintain reversibility.

The circuits discussed in this paper utilize qubits and are built using a combination of Clifford and $T$ gates. This gate ensemble comprises the phase gate ($S = T^2$), the Hadamard gate ($H$), the controlled-not gate (CNOT), and unit scalars, constituting a universally acknowledged fault-tolerant gate set. Within the standard notation, the Pauli operators ($X$, $Y$, and $Z$).

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \; Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \; S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \; H = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \; T = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{i} \end{pmatrix}$$

In quantum computing, certain traditional gates from classical computing find their analogs. For instance, the $X$ gate in quantum computing mirrors the classic NOT gate, recognized also as the flip gate. It switches $|1\rangle$ to $|0\rangle$ and $|0\rangle$ to $|1\rangle$. Meanwhile, the CNOT gate, performing the XOR operation classical version, acts on two qubits ($a$ and $b$), transforming $|a, b\rangle$ into $|a, a \oplus b\rangle$. Within the CNOT gate, the control qubit ($a$) undergoes an XOR operation with the target qubit ($b$).

The Toffoli gate, akin to the AND operation of the classical version, operates on three qubits and maps $|a, b, c\rangle$ to $|a, b, c \oplus ab\rangle$. In the Toffoli gate, the control qubits ($a$ and $b$) decide whether the target qubit ($c$) is XORed with the ANDed value of $a$ and $b$ (i.e., $a \cdot b$).

To execute the Toffoli gate, Selinger's method[33] leverages the idea that a doubly-controlled Z-gate, acquired through a basis transformation, can be a substitute. This approach maps a computational basis state $|abc\rangle$ to $(-1)^{abc}|abc\rangle$ using the doubly-controlled Z-gate.

The implementation of a Toffoli gate, illustrated in Fig. 1, achieves a $T$-depth of 1 and a combined depth of 7. It employs seven $T$ gates, sixteen CNOT gates, two single-qubit Clifford gates, and four ancilla qubits to achieve this.

### Grover's search algorithm

Grover's algorithm[34], a cornerstone of quantum computing, stands as a robust quantum search algorithm. However, it doesn't deliver a super-polynomial speedup like another famous quantum algorithm, Shor's algorithm for factorization. Its strength lies in efficiently solving the unstructured search problem, explicitly pinpointing the unique input of a black box function that generates a specific output value. Remarkably, Grover's algorithm achieves this with only $\mathcal{O}(\sqrt{N})$ function evaluations, where $N$ denotes the domain size of the function. This

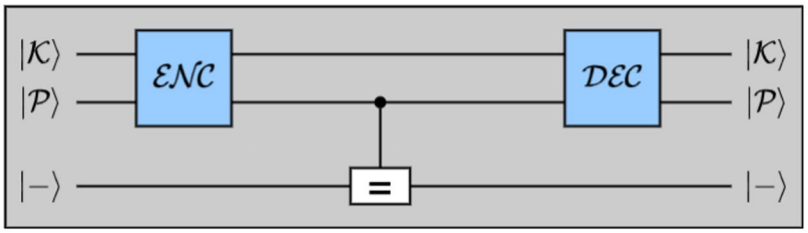



**Figure 2.** The Grover oracle. The (=) operator assesses the correspondence between the $\mathcal{ENC}$ output and the provided ciphertexts, and flips the target qubit in case of equality.

quantum algorithm excels at function inversion, enabling the calculation of $x$ from $y$ when presented with the function $y = f(x)$ on a quantum computer. The Grover's problem addressed by this algorithm can be defined as follows:

**Problem 1** Suppose $f : \{0,1\}^n \to \{0,1\}$ be a function, with oracle access to $f$, search an element $x$ such that $f(x) = 1$

Solving this problem through classical means necessitates an exhaustive search across all potential $x$ values, resulting in a time complexity of $2^n$. Yet, Grover's algorithm, pioneered by Grover[34], presents a remarkable quadratic speedup, significantly enhancing the process. In cases where there are $M$ solutions to the problem, the functioning of Grover's algorithm can be succinctly outlined as follows:

1. The initial state is prepared as:

$$|\phi\rangle = H^{\otimes}|0^n\rangle = \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} |x\rangle$$

   where $H$ is the Hadamard transformation
2. Repeat the Grover iteration:

$$\mathcal{G} = (2|\phi\rangle\langle\phi| - I_{2^n})O_f$$

   $j$ times to amplify the initial probability. Here $O_f$ is defined as $|x\rangle \mapsto (-1)^{f(x)}|x\rangle$
3. Measure the final quantum state to obtain an element $x$ which satisfies $f(x) = 1$

Let's establish $\alpha$ as $M/2^k$, signifying the initial success probability, where $\theta = \arcsin\sqrt{\alpha}$ with $0 < \theta \leq \pi/2$. After conducting $j$ iterations of Grover's algorithm, the probability of achieving a favorable outcome is expressed as $Pr(j) = \sin^2((2j+1)\theta)$. Notably, when $M << 2^k$, $\sin(\theta)$ becomes exceedingly small, enabling us to approximate $\theta \approx \sin(\theta) = \sqrt{M/2^k}$. Consequently, if we opt for the iteration count $j$ to be approximately $\pi/(4\theta) - 1/2$, the amplitude of favorable outcomes $(2j+1)\theta$ converges close to $\pi/2$, guaranteeing a success probability of at least $1-\alpha$.

*Grover oracle* In implementing Grover's search algorithm, constructing an oracle is vital. This oracle encrypts a 256-bit plaintext using a specific key and assesses a Boolean expression to verify if the resulting ciphertext aligns with the provided ciphertext. Termed Grover's oracle, it flips the state of the target quantum bit if an identical match between ciphertexts is detected.

In the context of this paper, constructing the oracle involves devising a circuit denoted as $\mathcal{ENC}$ for the cipher. This circuit encrypts a specified plaintext $P$ to produce a ciphertext. The generated ciphertext is subsequently compared against the provided ciphertext. The visualization of this oracle is illustrated in Fig. 2.

It's important to note that $\mathcal{DEC}$ is crafted by reversing the operations performed by $\mathcal{ENC}$. Essentially, $\mathcal{DEC}$ is formed by uncomputing $\mathcal{ENC}$, where $\mathcal{ENC}$ represents the quantum circuit responsible for initialization and encryption functions.

## Cost of Grover's algorithm

This research delves into two cost metrics put forth by Jaques and Schanck[35]. These metrics include the G-cost, representing the total count of gates, and the DW-cost, denoting the circuit depth and width's product. Leveraging the insights from Jaques et al.'s work[32], we offer a concise overview of the cost analysis concerning Grover's algorithm. This summary encompasses scenarios both with and without depth restrictions. Before delving into this analysis, we provide an overview of the depth constraints and the approach taken to parallelize Grover's algorithm.

*Depth constraints metric for cryptanalysis*

In our research, we operate under the assumption that any quantum adversary is confined by the total depth of its quantum circuit. This limitation is critical in quantum attacks, mainly due to the constraints imposed by the

coherence decay in early quantum computers, restricting them to shallow circuits. To overcome this limitation, NIST introduced the parameter MAXDEPTH, fixing the running time (circuit depth) for quantum attacks. The recommended ranges for MAXDEPTH, accounting for available time and gate speed assumptions, typically span from $2^{40}$ to $2^{96}$. These suggested values align with various considerations: $2^{40}$ approximates the number of gates expected to be serially performed in a year by envisioned quantum computing architectures, $2^{64}$ aligns with the estimated count of gates current classical computing architectures can carry out serially in a decade, and $2^{96}$ represents the estimated count of gates achievable by atomic-scale qubits with speed-of-light propagation over a millennium[23].

Considering NIST's guidelines, we propose that a cipher might be deemed vulnerable if it succumbs to an attack requiring a gate count of $< 2^{130}$ for MAXDEPTH = $2^{40}$, especially concerning ciphers with a key size of 128. This underscores the critical role of circuit depth in both algorithm implementations and quantum attack analysis, aligning with the broader emphasis within the cryptography community.

Furthermore, when the total depth of a quantum algorithm surpasses a certain threshold, the necessity for parallelization becomes apparent.

*Strategy for parallelizing Grover's algorithm*

Kim, Han, and Jeong[36] put forward two parallelization techniques for Grover's algorithm: outer parallelization and inner parallelization. The former involves running multiple algorithm instances concurrently, requiring success in just one instance. On the other hand, inner parallelization partitions the search space into separate subsets, assigning every subset to a parallel machine. Both techniques aim to diminish the required number of iterations.

Zalka's study[37] showcases that employing $M$ parallel Grover oracles can achieve a $\sqrt{M}$ enhancement in the necessary Grover iterations, an asymptotically optimal outcome. However, this parallelization method is comparatively less efficient than classical algorithms, demanding an increase in width by $M$ to reduce depth by $\sqrt{M}$. Both approaches eliminate the need for communication during Grover iterations, yet communication is necessary at the beginning and end for various tasks. In the discussed context, the preference leans toward inner parallelization, and the rationale for this preference is elaborated below.

Consider $M$ parallel machines executing Grover's algorithm for $j$ iterations to recover the key. In the case of a single machine, the probability of success is defined as $p(j) = \sin^2((2j+1)\theta)$. Employing outer parallelization, the probability that at least one machine retrieves the right key is denoted as $p_M(j) = 1 - (1 - p(j))^M$. To achieve a $\sqrt{M}$ improvement, each machine operates for $j_M = \frac{\pi}{4\theta\sqrt{M}}$ iterations.

For $M = 1$, $p_1(j_1) \approx 1$. With $M = 2$, $p_2(j_2) \approx 0.961$. Moving to $M = 3$, $p_3(j_3) \approx 0.945$. As $M$ grows, especially towards infinity, $p_M(j_M) \approx 0.915$. Upon substituting the value of $j_M$, we arrive at:

$$p_M(j_M) \approx 1 - \left(1 - \left(\sin\left(\frac{\pi}{2\sqrt{M}}\right)\right)^2\right)^M.$$

Expanding the sine function, the expression becomes:

$$p_M(j_M) \approx 1 - \left(1 - \left(\frac{\pi^2}{4M} + O\left(\frac{1}{M^2}\right)\right)\right)^M \xrightarrow[M\to\infty]{} 1 - e^{-\frac{\pi^2}{4}} \approx 0.915.$$

Therefore, scaling up the number of parallel machines ($M$) doesn't yield the expected $\sqrt{M}$ improvement in required iterations to approach a success probability near 1. However, inner parallelization concentrates on exploring the search space of a single machine, elevating the probability of success. Precisely, one machine's search space contains the correct key. Conducting $j_M$ iterations on this particular machine enables a near-certain identification of the correct key. Conversely, the remaining machines are assured to avoid identifying the correct key. Consequently, inner parallelization enhances the probability of success using the same number of parallel instances ($M$) and iterations.

*Cost of Grover's algorithm*

Certainly, this part aligns with insights from[32]. Consider the search space size $N = 2^k$. Assuming the utilization of a Grover oracle $\mathcal{G}$, where one Grover iteration costs $G_g$ gates, has a depth of $G_d$, and employs $G_w$ qubits. In the inner parallelization approach, we partition the search space into $M$ disjoint segments, where $M$ represents the number of parallel machines utilized. To attain a specific success probability $p$, determining the necessary iterations involves considering $p \le \sin^2((2j+1)\theta)$, where $j_p = \lceil(\sin^{-1}(\sqrt{p})/\theta - 1)/2\rceil \approx \frac{\sin^{-1}(\sqrt{p})}{2}\sqrt{N/M}$. Let $c_p = \sin^{-1}(\sqrt{p})/2$, then the overall depth of a $j_p$-fold Grover iteration is:

$$D = j_p G_d \approx c_p \sqrt{\frac{N}{M}} G_d = c_p 2^{\frac{n-m}{2}} G_d \tag{1}$$

Each of the $M$ machines uses $j_p G_g \approx c_p \sqrt{\frac{N}{M}}.G_g$ gates, so the total G-cost over all machines is

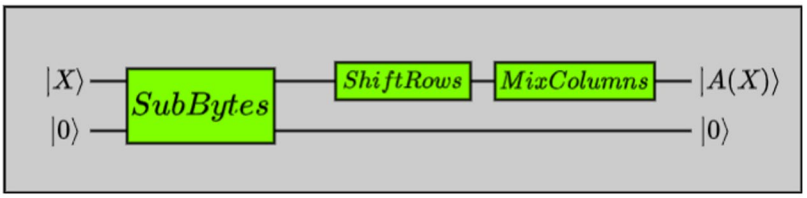



**Figure 3.** Quantum Circuit for $A(X)$.

$$G = M.j_p G_g \approx c_p\sqrt{N.M}.G_g = c_p 2^{\frac{n+m}{2}} G_g \; gates. \tag{2}$$

And the total width is

$$W = M.G_w = 2^m G_w \; qubits. \tag{3}$$

So $DW$ cost is

$$DW \approx c_p\sqrt{N.M} G_d G_w \; qubit - cycles. \tag{4}$$

Based on the mentioned expressions, it is clear that reducing the number of parallel machines ($M = 2^m$) results in a decrease in both the DW and G-cost. Therefore, when there are fixed constraints on depth, width, and the number of gates, the optimal approach for an adversary is to utilize the entire depth budget and minimize parallelization.

*Optimizing under a depth limit*

As previously highlighted, Grover's algorithm doesn't naturally lend itself to efficient parallelization overall. Therefore, the recommended approach often involves specifically parallelizing within the oracle circuit. By reducing the depth of the oracle, it becomes feasible to execute a greater number of iterations within the designated depth limit, thereby minimizing the necessity for parallelization.

Let's assume a fixed maximum depth constraint denoted as $D_{\max}$. Given the oracle's depth $G_d$, we can conduct $j_{\max} = \lfloor \frac{D_{\max}}{G_d} \rfloor$ Grover iterations using the oracle $G$. To achieve a desired success probability $p$, we determine the number $M$ of parallel instances required for this probability within the instance whose keyspace partition encompasses the key. This is formulated as $p = \sin^2\left((2j_{\max}+1)\sqrt{\frac{M}{N}}\right)$.

$$M = \left\lceil \frac{N.arcsin^2(\sqrt{p})}{(2.\lfloor D_{max}/G_d \rfloor + 1)^2} \right\rceil \approx c_p^2 2^k \frac{G_d^2}{D_{max}^2} \tag{5}$$

Applying this in equation (2) produces an overall gate count of

$$G = c_p^2 2^k \frac{G_d G_g}{D_{max}} \; gates \tag{6}$$

As a result, the full $DW$-cost considering the depth constraint is

$$DW = c_p^2 2^k \frac{G_d^2 G_w}{D_{max}^2} \; qubit - cycles \tag{7}$$

Therefore our main goal is to optimize $G_d^2 G_w$.

## Quantum circuits of the AES components

The quantum circuit for the round functions of all these ciphers depends on that of $A(X)$. The circuit for $A(X)$ is shown in Fig. 3.

### ShiftRows circuit resources

*ShiftRows* is a permutation that acts on the entire state. As a permutation, it can be represented entirely using rewiring. Previous studies on Quantum circuits for AES and other ciphers often consider rewiring a cost-free operation.

### MixColumns circuit resources

The *MixColumn* operation can be represented by a $32 \times 32$ binary matrix over $\mathbb{F}_2$. A commonly used approach to implement this matrix is through LUP-type decomposition[38], enabling efficient execution of *MixColumn* under the s-Xor metric. In an s-Xor implementation, outputs are stored directly in input registers, eliminating the need for additional registers. Additionally, simulating an Xor operation under the s-Xor metric can be easily accomplished using a CNOT gate. Numerous quantum circuits for *MixColumn* have been introduced in recent years. Notably, Xiang et al.[39] to construct our quantum circuit for *MixColumn*. In this circuit, 92 *CNOT* gates

| #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|
| 368 | 0 | 0 | 0 | 30 | 128 |

**Table 2.** *MixColumn* Resources.

| #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|
| 1604 | 160 | 546 | 3 | 68 | 278 |

**Table 3.** AES S-box resources.

| #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|
| 51072 | 5056 | 17472 | 6 | 139 | 4448 |

**Table 4.** SubBytes resources.

| #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|
| 51440 | 5056 | 17472 | 6 | 169 | 4448 |

**Table 5.** *A(X)* resources.

are utilized, and the depth is set at 30. As *MixColumn* can be applied to all four columns of the state simultaneously, the depth remains at 30, and the total requirement for *CNOT* gates is $92 \times 4 = 368$. The circuit[39] is more convenient and cost-effective for our specific application. Notably, this identical circuit is also utilized in[20].

The cost analysis of setting up a quantum circuit for *MixColumn* is provided in Table 2.

*Cost of SubBytes*

In our quest to incorporate the AES S-box into our framework, we require a fitting quantum circuit for it. Our primary goal is to opt for an in-place circuit, facilitating efficient utilization within our construction while upholding a low-depth characteristic. After a thorough evaluation, we've opted for the S-box circuit proposed in[21]. This circuit presents two enhanced versions of the S-box circuit highlighted in[5]. One version achieves a Toffoli-depth of 6, condensed to 4 while maintaining the same qubit count. The other version attains a Toffoli-depth of 3, albeit at the expense of certain Clifford gates and qubits. For our paper, focusing on depth reduction, particularly Toffoli depth, we solely utilize the circuit with a Toffoli-depth of 3. It's important to note that in our setup, we employ a decomposition of the Toffoli gate into a circuit with T-depth 1. Thus, referencing a circuit with Toffoli-depth 3 essentially implies a circuit with T-depth 3. The resource requisites for each circuit are detailed in Table 3.

The S-box is implemented using Toffoli gates, precisely Selinger's implementation without measurements. This implementation necessitates 7 T gates, 16 CNOT gates, 2 single-qubit Clifford gates, and 4 ancilla qubits, featuring a T-depth of one and an overall depth of 7. It's noteworthy that while this reduces the T-depth of the circuit, it comes at the cost of 80 ancilla qubits.

In our implementation, the SubBytes operation involves a set of 16 S-boxes. One approach entails sequentially implementing these S-boxes using the same set of ancilla qubits. However, this results in a circuit with significant depth. An alternative approach involves a parallel implementation of the S-boxes, with each S-box allocated its own set of ancilla qubits. Although this parallel implementation increases the qubit requirement, it drastically reduces the circuit depth, making it the preferred choice for the SubBytes operation. As this parallel implementation method has been previously employed in works like[28,40], we'll exclusively present the resource requirements for this implementation in Table 4, without delving into a detailed circuit description. The qubit count aligns with information from[21].

It is now possible to compute the cost of executing $A(X)$ once. The results of these computations are showcased in Table 5.

## Quantum circuit of Rocca-S

The `Rocca-S` authenticated encryption scheme is characterized by its incorporation of four distinct phases: initialization, associated data processing, encryption, and finalization. It takes as inputs a 256-bit key, denoted as $K_0||K_1 \in \mathbb{F}_2^{128} \times \mathbb{F}_2^{128}$, a 128-bit nonce $N$, associated data $AD$, and the message $M$. The key $K_0||K_1$ is formed by concatenating two components, $X$ and $Y$. The resulting output consists of the corresponding ciphertext $C$ and a 256-bit tag $T$. Specifically, if we define $\overline{X}$ as $X||0^l$, where $0^l$ represents a zero string of length $l$ bits, the value of $l$ is the smallest non-negative integer that ensures the length of $|\ \overline{X}\ |$ is a multiple of 256. Breaking down $X$ further,

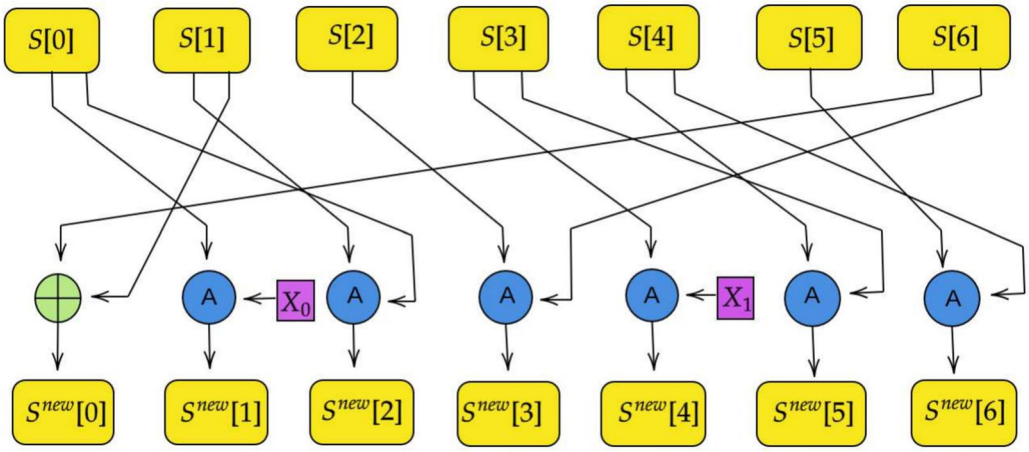



**Figure 4.** The round function of `Rocca-S` (in image A denotes the AES operation).

we express it as $X = X_0||X_1||\ldots||X_{\frac{|X|}{256}-1}$, where each $X_i$ has a length of 256 bits. Additionally, $X_i$ is decomposed into $X_i^0||X_i^1$, with both $X_i^0$ and $X_i^1$ having a length of 128 bits.

Within `Rocca-S`, the state $S$ is composed of seven blocks structured as $S = (S[0], S[1], \ldots, S[6])$, where each $S[i]$ (for $0 \leq i \leq 6$) signifies an individual block. The initial block is denoted as $S[0]$. The function AES(X, Y) is defined as (MixColumns ∘ ShiftRows ∘ SubBytes($X$)) $\oplus$ $Y$. This function incorporates operations akin to those found in AES, where MixColumns, ShiftRows, and SubBytes emulate their respective AES counterparts. Additionally, there exists another operation, A(X), which involves the composition of MixColumns, ShiftRows, and SubBytes (X). The round function $R(S, X_0, X_1)$ (Fig. 4) plays a crucial role in updating the state $S$ within Rocca-S, similar to the mechanism employed in Tiaoxin-346. This round function takes the state $S$ and two blocks, $X_0$ and $X_1$, as inputs and produces an updated state denoted as $S^{new}$ (in short $S'$), symbolized by $R(S, X_0, X_1) \rightarrow S'$. Notably, this update involves the application of constants $Z_0$ and $Z_1$ in the process.

$$
\begin{aligned}
S'[0] &= S[6] \oplus S[1] & S'[1] &= \mathrm{AES}(S[0], X_0)\\
S'[2] &= \mathrm{AES}(S[1], S[0]) & S'[3] &= \mathrm{AES}(S[2], S[6])\\
S'[4] &= \mathrm{AES}(S[3], X_1) & S'[5] &= \mathrm{AES}(S[4], S[3])\\
S'[6] &= \mathrm{AES}(S[5], S[4])
\end{aligned}
$$

*Initialization* During this initialization phase, the state undergoes setup using the nonce $N$, the keys $K_0$ and $K_1$, and two 128-bit constants $Z_0$ and $Z_1$. Specifically, the values of $N$, $K_0$, $K_1$, $Z_0$, and $Z_1$ are loaded into the corresponding substates: $S[1]$, $S[3]$, $S[0]$, $S[2]$, and $S[4]$, respectively. The remaining substates, $S[5]$ and $S[6]$, are initialized with the values $N \oplus K_1$ and 0, respectively. Following the initial setup, the round update function $R(S, Z_0, Z_1)$ is iteratively applied 16 times. Finally, the keys $K_0$ and $K_1$ are XOR-ed with specific substates. Precisely, $K_0$ is XOR-ed with substates $S[0]$, $S[1]$, $S[3]$, and $S[4]$, while $K_1$ is XOR-ed with $S[2]$, $S[5]$, and $S[6]$. This final step completes the initialization process, preparing the state for subsequent operations within the Rocca-S encryption scheme.

*Processing the associated data* In this phase, the process of updating the state is contingent on the length of the associated data $AD$. Initially, padding bits are appended to $AD$ to ensure it is a multiple of 256, resulting in $\overline{AD}$. This representation can be expressed as $\overline{AD} = \overline{AD}_0||\cdots||\overline{AD}_{d-1}$, where each 256-bit block $AD_i$ is constituted of two 128-bit words, denoted as $AD_i^0||AD_i^1$. The state update function $R(S, \overline{AD}_i^0, \overline{AD}_i^1)$ is then iteratively applied for $d$ times, considering each $\overline{AD}_i$. It is important to note that when $AD$ is empty (and consequently $d = 0$), this phase is skipped, as there is no associated data to process.

*Encryption* In this stage, the message $M$ undergoes a transformation to $\overline{M}$ by incorporating additional padding bits, ensuring its length is a multiple of 256. Let $M = M_0||\cdots||M_{m-1}$, where each 256-bit block $M_i$ comprises two 128-bit words, represented as $M_i^0||M_i^1$. Subsequently, depending on the length of the message, the ciphertext is produced, and the state undergoes an update as outlined below:

$$
\begin{aligned}
&C_i^0 = \mathrm{AES}(S[3] \oplus S[5], S[0]) \oplus \overline{M}_i^0,\\
&C_i^1 = \mathrm{AES}(S[4] \oplus S[6], S[2]) \oplus \overline{M}_i^1,\\
&R(S, \overline{M}_i^0, \overline{M}_i^1)
\end{aligned}
$$

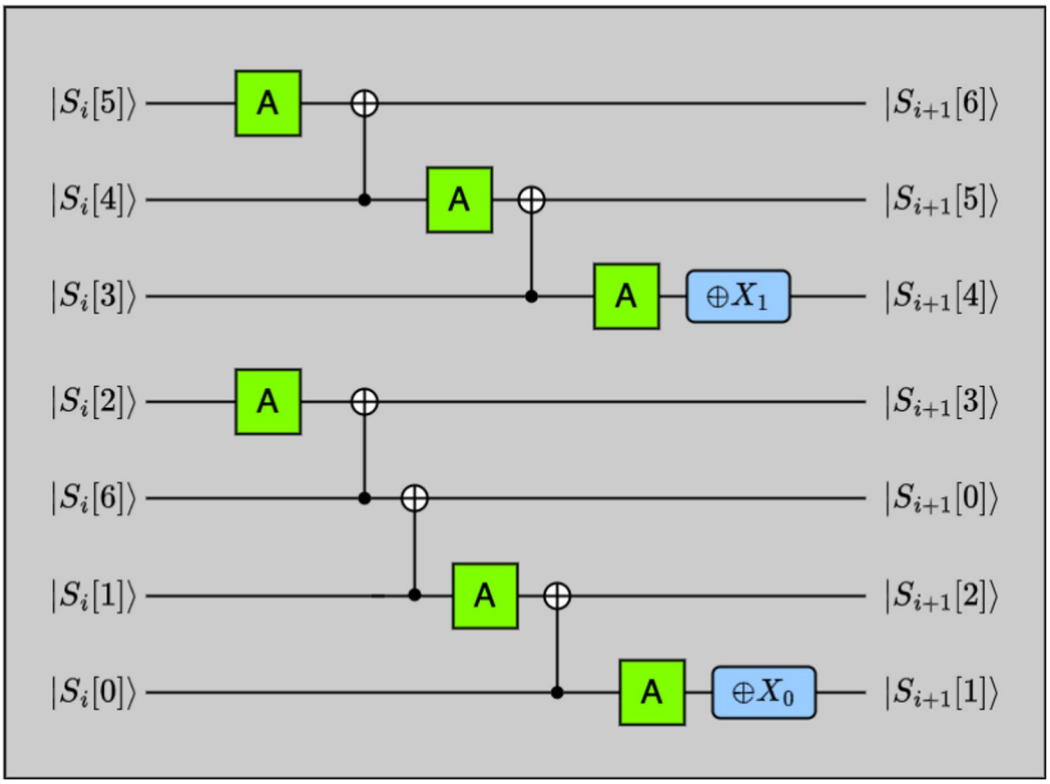



**Figure 5.** `Rocca-S` Round Function Quantum Circuit.

| #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|
| 309280 | 30336 | 104832 | 18 | 510 | 9536 |

**Table 6.** `Rocca-S` round function resources.

where $i = 0$ to $m - 1$, and $m = \frac{|\overline{M}|}{256}$. If the length of the last block of the message is less than 256 bits and $b$ bits are padded. Then the last $b$ bits from the last block of the ciphertext are trimmed ensuring that the ciphertext length equals the message length. Note that, when $|M| = 0$, this phase is skipped.

## In-place circuit of `Rocca-S` round update function

Here, we propose an in-place circuit of `Rocca-S`, composed of six instances of the $A(X)$ function and five bitwise XOR operations involving two 128-bit registers. Implementing a bitwise XOR between two 128-bit registers can be achieved by applying 128 CNOT gates in parallel along with 2 XOR operations using a constant. The circuit is depicted in Fig. 5.

Even though there are six occurrences of $A(X)$ within a single round, only two of these instances are executed simultaneously, as illustrated in Fig. 5. Consequently, the allocation of the necessary ancilla qubits is essential for both of these concurrent implementations. As these ancillae reset to 0 after applying $A(X)$, they can be repurposed for the remaining two $A(X)$ implementations and later reused for the final pair. This approach effectively reduces the required ancilla count to accommodate only two $A(X)$ implementations instead of the original six.

Regarding qubits, the requisite count encompasses those required for the state and the number essential for implementing two parallel $A(X)$ operations. Thus we have,

$$\begin{aligned} \#\text{CNOT gates} &= 6 \times A(X) \#\text{CNOT gates} + 5 \times 128 \\ \#\text{1qCliff gates} &= 6 \times A(X) \#\text{1qCliff gates} \\ \#\text{T gates} &= 6 \times A(X) \#\text{ T gates} \\ \#\text{ancillae} &= 2 \times \text{ ancillae required in } A(X) \end{aligned}$$

and

$$\text{T depth} = 3 \times \text{ T depth of } A(X) \qquad \text{Full depth} = 3 \times \text{ full depth of } A(X) + 3$$

| | #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|---|
| Initialization | 4948480 | 485376 | 1677312 | 288 | 8160 | 9536 |
| Encryption | 103648 | 10112 | 34944 | 6 | 172 | 256 |

**Table 7.** Rocca-S encryption resources.

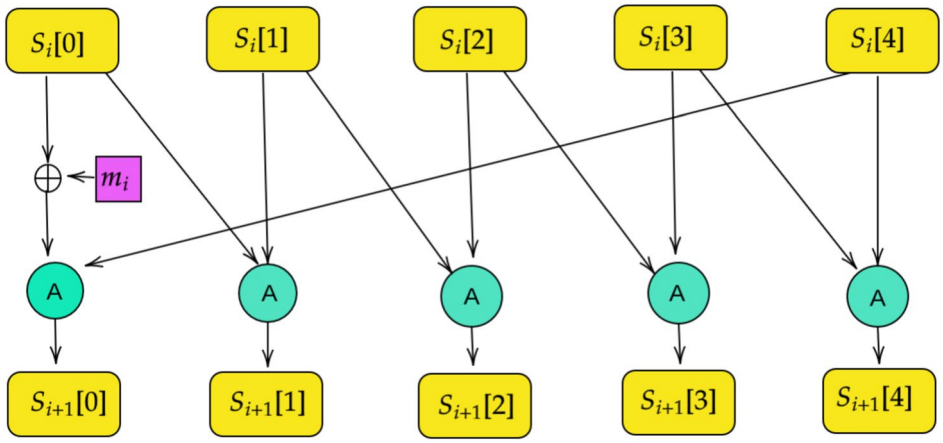



**Figure 6.** AEGIS-128 round function(A denotes the *AES* operation).

## Quantum resource estimates of Rocca-S

Here, we provide detailed cost assessments for the quantum circuits implementing the Rocca-S encryption scheme. All subroutines are implemented using the Qiskit programming language. The specific breakdown of costs for a single round of Rocca-S is detailed in Table 6.

It is crucial to note that the count of single-qubit Clifford gates for the AddConstants subroutine may vary, depending on the number of ones present in the round constants. A comprehensive breakdown of costs for the entire Rocca-S encryption circuit can be found in Table 7.

## Quantum circuit of AEGIS

AEGIS[25] is constructed based on the AES encryption round function, excluding the last round. In the case of AEGIS-128L, eight AES round functions are employed to process a 256-bit message block in a single step. Five AES round functions are utilized for AEGIS-128, which handles a 128-bit message block, and AEGIS-256 uses six AES round functions. The focus of this paper is specifically on AEGIS-128.

AEGIS-128 employs a 128-bit key and a 128-bit initialization vector for encrypting and authenticating a message. The lengths of associated data and plaintext are both less than $2^{64}$ bits. Additionally, the length of the authentication tag is less than or equal to 128 bits.

*The state update function of* AEGIS-128: The state update function transforms the 640-bit state $S_i$ with a 128-bit message block $m_i$. The representation of state update function (Fig. 6) $S_{i+1} = R(S_i, m_i)$ is given below:

$$S_{i+1}[0] = AES(S_i[4], S_i[0] \oplus m_i) \qquad S_{i+1}[1] = AES(S_i[0], S_i[1])$$
$$S_{i+1}[2] = AES(S_i[1], S_i[2]) \qquad S_{i+1}[3] = AES(S_i[2], S_i[3])$$
$$S_{i+1}[4] = AES(S_i[3], S_i[4])$$

*Initialization:* While initializing, the state is set up using $IV$, and the key $K$ in combination with the constants $c_1$ and $c_0$, and the cipher is iterated ten times with $K$ and $IV$ used as the message.

1. The substate $S_{-10}[0]$ is initialized by $K_{128} \oplus IV_{128}$. The constants $c_0$ and $c_1$ is loaded into the substates $S_{-10}[1]$ and $S_{-10}[2]$ respectively. The remaining substates $S_{-10}[3]$ and $S_{-10}[4]$ are initialized by $c_0 \oplus K_{128}$ and $c_1 \oplus K_{128}$ accordingly.
2. $K$ and $IV$ used as the message as follows: $m_{2i} = K_{128}$; $m_{2i+1} = K_{128} \oplus IV_{128}$ when $i = -5$ to $-1$.
3. Now the cipher is iterated 10 times, for $i = -10$ to $-1$, $S_{i+1} = R(S_i, m_i)$.

*Processing the associated data* The associated data is absorbed in the state updation process in this phase. Initially, padding bits are appended to the associated data $AD$ to ensure it aligns as a multiple of 128, resulting in $\overline{AD}$. Let $adlen = |AD|$. For $i$ ranging from 0 to $\lceil \frac{adlen}{128} \rceil$, the state is updated as follows:

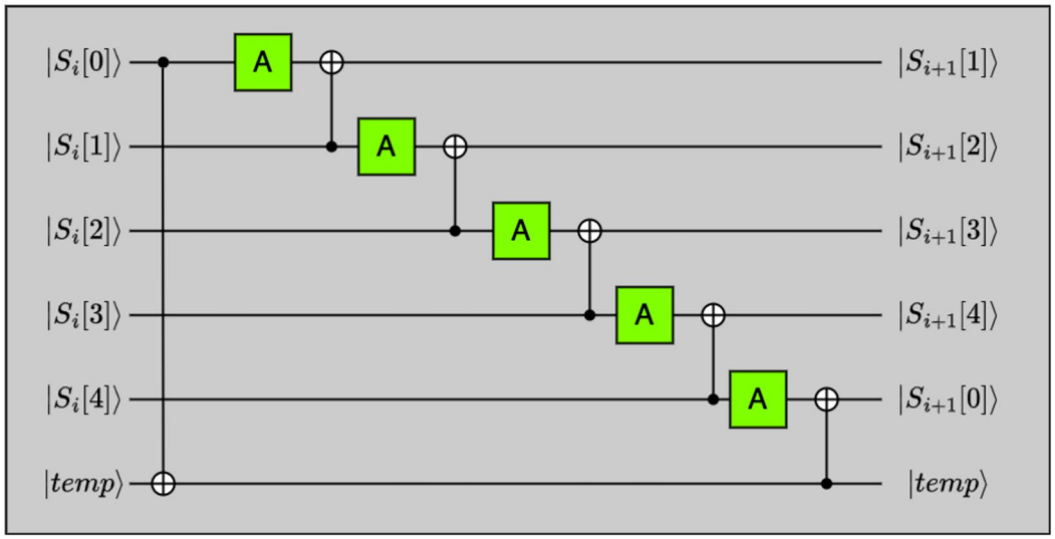



**Figure 7.** AEGIS-128 Round Function Quantum Circuit.

| #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|
| 257968 | 25280 | 87360 | 30 | 851 | 6240 |

**Table 8.** AEGIS-128 round function resources.

$$S_{i+1} = R(S_i, \overline{AD}_i).$$

Here, $\overline{AD} = \overline{AD}_0 || \cdots || \overline{AD}_{\lceil \frac{adlen}{128} \rceil}$, where each $\overline{AD}_i$ represents a 128-bit block.

*Encryption:* Following the processing of the associated data, during each encryption step, each 128-bit plaintext blocks undergoes encryption to yield ciphertexts. Consider a plaintext $P$ which is appended with padding bits (in order to ensure it aligns as a multiple of 128) to obtain $\overline{P}$. Let $\overline{P} = \overline{P}_0 || \cdots || \overline{P}_{\lceil \frac{msglen}{128} \rceil}$ where $|P| = msglen$

.

Let $u = \lceil \frac{adlen}{128} \rceil$ and $v = \lceil \frac{msglen}{128} \rceil$. For $i = 0$ to $v - 1$, encryption is performed, and the state is updated as follows:

$$\begin{aligned} C_i &= \overline{P}_i \oplus S_{u+i}[1] \oplus S_{u+i}[4] \oplus (S_{u+i}[2] \& S_{u+i}[3]); \\ S_{u+i+1} &= R(S_{u+i}, \overline{P}_i). \end{aligned}$$

## Quantum circuit and resource estimation of AEGIS-128 round update function

In the circuit illustrated in Fig. 7 for the round function, there are five occurrences of the $A(X)$ function and 6 bitwise XOR operations between two 128-bit registers. Implementing a bitwise XOR between two 128-bit registers can be done by applying 128 CNOT gates in parallel.

Each $A(X)$ application is executed sequentially, allowing for the reuse of ancillae from the prior $A(X)$ in the subsequent one. Consequently, the ancillae employed in the previous $A(X)$ instances can be repurposed for subsequent applications. The required qubit count encompasses those needed for the state and the qubits necessary for implementing a single $A(X)$ operation. Additionally, the circuit will necessitate 1280 ancillae for the $|temp\rangle$ register.

Henceforth we have,

$$\begin{aligned} \#\text{ CNOT gates} &= 5 \times \#\text{CNOT in } A(X) + 6 \times 128 \\ \#\text{1qCliff gates} &= 5 \times \#\text{1qCliff in } A(X) \\ \#\text{T gates} &= 5 \times \#\text{T in } A(X) \\ \#\text{ancillae} &= \text{ ancillae in } A(X) + 1280 \end{aligned}$$

and

$$\text{T depth} = 5 \times \text{ T depth of } A(X) \qquad \text{Full depth} = 5 \times \text{ full depth of } A(X) + 6$$

| | #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|---|
| Initialization | 2579680 | 252800 | 873600 | 300 | 8510 | 6240 |
| Encryption | 2304 | 256 | 896 | 1 | 9 | 256 |

**Table 9.** Cost of implementing AEGIS-128 encryption.

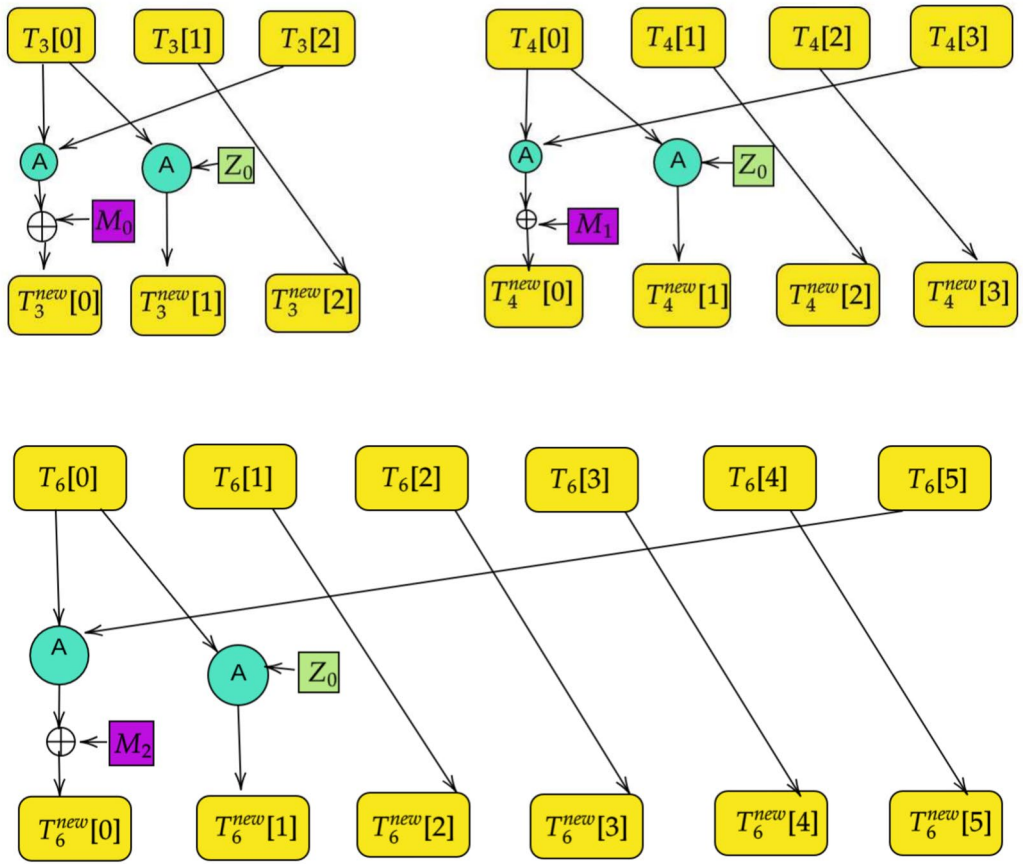



**Figure 8.** The round function of Tiaoxin-346 (A denotes the AES operation).

## Quantum resource estimates of AEGIS-128

This section offers comprehensive cost assessments for the quantum circuits of AEGIS-128, meticulously crafted using the Qiskit programming language. Table 8 outlines the granular cost breakdown for a single round of AEGIS-128. Notably, these estimations take into account variations in the number of single-qubit Clifford gates for AddConstants, contingent upon the count of ones in the round constants.

Furthermore, Table 9 provides an overall view, encapsulating the total cost estimates for the entire encryption circuit of AEGIS-128. It's important to emphasize that these calculations assume an empty associated data set and encryption for a 256-bit plaintext, a choice made to suit Grover's key search requirements.

## Quantum circuit of Tiaoxin-346

Tiaoxin-346 functions as a nonce-based authenticated encryption scheme, employing a design reliant on stream ciphers. The input parameters encompass a 128-bit key, $K$, a 128-bit nonce, message $M$, and associated data, $AD$, which ranges in size from 0 to $2^{128} - 1$ bits.

The output generated by Tiaoxin-346 is the ciphertext, $C$, aligned with the same byte count as $M$, along with an authentication tag, Tag, which spans a maximum of 128 bits. The scheme operates using three distinct states: $T_3, T_4$, and $T_6$, each constituted by a series of words denoted as $T_m = (T_m[0], T_m[1], ....., T_m[m-1])$, where $T_m[i]$ represents individual words ($i = 0, 1, ...., m-1$), with $T_m[0]$ signifying the initial word.

The round function Tiaoxin-346, described in Fig. 8, $R(T_3, T_4, T_6, M_0, M_1, M_2)$, incorporates the states $T_3, T_4$, and $T_6$, including three extra words, $M_0, M_1$, and $M_2$.

Now, $R(T_3, T_4, T_6, M_0, M_1, M_2) \rightarrow (T_3^{new}, T_4^{new}, T_6^{new})$ is defined as

$$T_3^{new}[0] = \text{AES}(T_3[2], T_3[0]) \oplus M_0 \qquad T_4^{new}[0] = \text{AES}(T_4[3], T_4[0]) \oplus M_1$$
$$T_3^{new}[1] = \text{AES}(T_3[0], Z_0) \qquad T_4^{new}[1] = \text{AES}(T_4[0], Z_0)$$
$$T_3^{new}[2] = T_3[1] \qquad T_4^{new}[2] = T_4[1];\ T_4^{new}[3] = T_4[2]$$

$$T_6^{new}[0] = \text{AES}(T_6[5], T_6[0]) \oplus M_2$$
$$T_6^{new}[1] = \text{AES}(T_6[0], Z_0)$$
$$T_6^{new}[2] = T_6[1]$$
$$T_6^{new}[3] = T_6[2]$$
$$T_6^{new}[4] = T_6[3]$$
$$T_6^{new}[5] = T_6[4]$$

where, $\text{AES(X,Y)} = (\text{MixColumns} \circ \text{ShiftRows} \circ \text{SubBytes}(X)) \oplus Y$, and $Z_0 = 428a2f98d728ae227137449123ef5cd$

*Initialization* During the initialization phase, the state undergoes setup using the key $K$, nonce $IV$, along with two constants $Z_0$ and $Z_1$, where $Z_1$ is a constant word like $Z_0$ is defined as $Z_1 =$ `b5c0fbcfec4d3b2fe9b5d-ba58189dbbc`. Specifically, the value of $K$ is loaded into substates $T_3[0]$, $T_3[1]$, $T_4[0]$, $T_4[1]$, $T_6[0]$ and $T_6[1]$, the value of $IV$ is loaded into substates $T_3[2]$, $T_4[2]$, $T_6[2]$, the value of $Z_0$ and $Z_1$ is loaded into the substates $T_4[3]$ and $T_6[3]$ respectively. The remaining substates $T_6[4]$, $T_6[5]$are initialized with 0. and the states go through 15 rounds, i.e., iterate $R(T_3, T_4, T_6, M_0, M_1, M_2) \rightarrow (T_3^{new}, T_4^{new}, T_6^{new})$ for $i = 0$ to 15.

*Processing the associated data* The associated data $AD$ is partitioned into blocks, each containing 256 bits. If the final block of $AD$ is not complete, then that block is padded with zero bytes. Assume that padded $AD$ has $k$ blocks: $AD = AD_1, \ldots, AD_k$, where each block looks like $AD_i = AD_i^0 || AD_i^1$, i.e composed of two words. Processing associated data is given below:

for $i = 1$ to $k$
  $R(T_3, T_4, T_6, AD_i^0, AD_i^1, AD_i^0 \oplus AD_i^1)$
end for

*Encryption* Suppose that the padded message is divided into $d$ blocks: $M = M_1, \ldots, M_d$, each block is made up of two words i.e. $M_i = M_i^0 || M_i^1$. In the encryption process, each block $M_i$ undergoes a single round, producing a ciphertext block $C_i = C_i^0 || C_i^1$. The encryption is carried out in the following manner:

for $i = 1$ to $d$
  $R(T_3, T_4, T_6, M_i^0, M_i^1, M_i^0 \oplus M_i^1)$;
  $C_i^0 = T_3[0] \oplus T_3[2] \oplus T_4[1] \oplus (T_6[3] \& T_4[3])$;
  $C_i^1 = T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus (T_6[5] \& T_3[2])$;
end for

## Circuit of `Tiaoxin-346` round update function

The circuit shown in Fig. 9 illustrates the round function in `Tiaoxin-346`, involving six occurrences of the $A(X)$ function, three bit-wise XOR operations on two 128-bit registers, and three XORs with a constant.

Implementing the bit-wise XOR operation between two 128-bit registers involves applying 128 CNOT gates in parallel, while XORing with a constant can be achieved by employing an adequate number of NOT gates based on the positions where the bit value of the constant is 1. Presently, the count of NOT gates is excluded from the calculation of the round function's implementation cost for simplicity.

Although a round comprises six $A(X)$ functions, only three of them are executed in parallel, as indicated in Fig. 9. Therefore, the requisite number of ancillae is allocated for these three implementations. Since these ancillae reset to 0 after the execution of $A(X)$ , they can be repurposed for the remaining three instances of $A(X)$ . Consequently, the necessary number of ancillae is equivalent to that required for three implementations of $A(X)$ , rather than six. Additionally, the number of qubits needed corresponds to those required for the state and for implementing three parallel $A(X)$ operations.

Thus we have,

$$\begin{aligned} \#\text{CNOT gates} &= 6 \times \#\text{CNOT in } A(X) + 3 \times 128 \\ \#\text{1qCliff gates} &= 6 \times \#\text{1qCliff in } A(X) \\ \#\text{ T gates} &= 6 \times \#\text{ T in AES} \\ \#\text{ ancillae} &= 3 \times \text{ ancillae in } A(X) \end{aligned}$$

and

$$\text{T depth} = 2 \times \text{ T depth of } A(X) \qquad \text{Full depth} = 2 \times \text{ full depth of } A(X) + 1$$

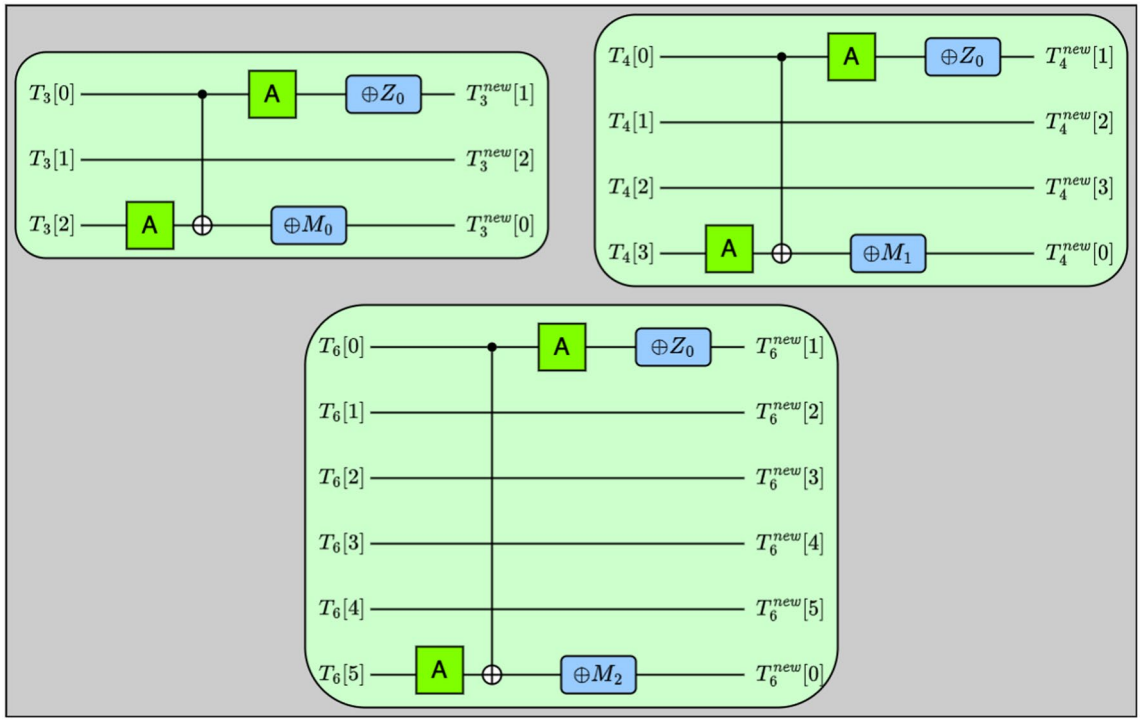



**Figure 9.** Circuit for `Tiaoxin-346` round update function.

| #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|
| 309024 | 30336 | 104832 | 12 | 339 | 14624 |

**Table 10.** Cost of implementing `Tiaoxin-346` round function.

| | #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|---|
| Initialization | 4635360 | 455040 | 1572480 | 180 | 5085 | 14624 |
| Encryption | 2816 | 256 | 896 | 1 | 10 | 256 |

**Table 11.** Cost of implementing `Tiaoxin-346` encryption function.

### Quantum resource estimates of `Tiaoxin-346`

The cost estimates for one round of `Tiaoxin-346` can be found in Table 10.

The total cost estimates for the full `Tiaoxin-346` encryption circuit are provided in Table 11.

## Grover's search algorithm resource estimates

In this section, we delve into the implementations of complete Grover oracles for three specific ciphers: `Rocca-S`, `AEGIS-128`, and `Tiaoxin-346`. Our tool of choice is Qiskit, which conveniently offers cost estimates for these Grover oracles. Our primary focus here is to estimate the quantum resources required for conducting full key search attacks via Grover's algorithm. This estimation process considers NIST's MAXDEPTH limit, following the approach previously utilized by Jaques et al. It involves evaluating the algorithmic costs through inner parallelization, achieved by partitioning the search space.

### Resource estimates of implementing Grover's oracle

To determine the total count of required `CNOT`, `1qCliff`, and $T$ gates within the oracle, we analyze the cipher instances and the encryption function. Specifically, the sum of `CNOT` and `1qCliff` gates is derived from those necessary for the cipher instances. This computation can be formulated as:

$$2 \cdot \texttt{\#CNOT}(\mathscr{E}\ \mathscr{NC}\ ) \text{ and } 2 \cdot \texttt{\#1qCliff}(\mathscr{E}\ \mathscr{NC}\ )$$

The Grover oracle, responsible for comparing the cipher instances with the provided ciphertexts, relies on $(k \cdot r)$ -controlled `CNOT` gates, where $k$ represents the key size. Our estimation for the required $T$ gates to implement

| Cipher | #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width |
|---|---|---|---|---|---|---|
| AEGIS-128 | 5163968 | 506112 | 1753004 | 602 | 17038 | 6496 |
| Rocca-S | 10104256 | 990976 | 3432620 | 588 | 16664 | 9793 |
| Tiaoxin-346 | 9276352 | 910592 | 3150764 | 362 | 10190 | 14880 |

**Table 12.** Cost of Grover's oracle.

| cipher | #CNOT | #1qCliff | #T | T-Depth | Full-Depth | Width | *G*-cost | *DW*-cost |
|---|---|---|---|---|---|---|---|---|
| AEGIS-128 | $1.93 \cdot 2^{85}$ | $1.52 \cdot 2^{82}$ | $1.31 \cdot 2^{84}$ | $1.85 \cdot 2^{72}$ | $1.63 \cdot 2^{77}$ | 6496 | $1.39 \cdot 2^{86}$ | $1.30 \cdot 2^{90}$ |
| Rocca-S | $1.89 \cdot 2^{150}$ | $1.48 \cdot 2^{147}$ | $1.29 \cdot 2^{149}$ | $1.80 \cdot 2^{136}$ | $1.60 \cdot 2^{141}$ | 9793 | $1.36 \cdot 2^{151}$ | $1.91 \cdot 2^{154}$ |
| Tiaoxin-346 | $1.74 \cdot 2^{86}$ | $1.36 \cdot 2^{83}$ | $1.18 \cdot 2^{85}$ | $1.11 \cdot 2^{72}$ | $1.95 \cdot 2^{76}$ | 14880 | $1.25 \cdot 2^{87}$ | $1.77 \cdot 2^{90}$ |

**Table 13.** Cost estimates for Grover's algorithm with $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ Grover oracle iterations for attacks with high success probability, without a depth restriction.

these controlled CNOT gates aligns with the expression $(32 \cdot t - 84)$[41]. The total count of $T$ gates is then approximated as

$$(32 \cdot k - 84) + 2 \cdot \#\text{T gates for}\{(\mathscr{E} \quad \mathscr{NC} \quad )\}$$

In determining the full depth and $T$-depth, our focus is on the depths of the cipher instances. Assuming parallelism, both cipher instances can be executed simultaneously using separate qubit sets. Hence, the oracle's depth equals

$$2 \cdot (\text{Depth of } \mathscr{E} \quad \mathscr{NC} \quad )$$

For a comprehensive overview of the resources required for constructing complete Grover oracles across all ciphers, please refer to Table 12. The oracle's construction is illustrated in Fig. 2.

## Resource estimates for key search

Utilizing the cost estimates outlined in Table 12 for the Grover oracles, we can offer a comprehensive assessment of the expenses for full key search attacks. These estimations are independent of any depth limit or parallelization prerequisites. In Table 13, we present cost approximations for the exhaustive execution of Grover's algorithm when $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ iterations of the Grover oracle are executed without parallelization, with $k$ denoting the key size. The $G$-cost denotes the cumulative count of gates, aggregating the first three columns in the table, representing the numbers of 1-qubit Clifford and CNOT gates, T gates, and measurements. Conversely, the $DW$-cost symbolizes the product of the entire circuit depth and width, as illustrated in columns 6 and 7 of the table.

## Resource estimates for key search under NISTs MAXDEPTH metric

NIST has established security requirements for AES based on computational resources, listed in order of increasing security strength[23].

NIST conservatively estimated the $G$-cost $\times$ $D$ (total gates $\times$ depth) as $2^{170}$, $2^{233}$, and $2^{298}$ for AES-128, AES-192, and AES-256, respectively. NIST has since then adjusted the threshold for quantum security in[31], Page 45 by referring to Jaques et al.'s work[32] as: $2^{157}$ and $2^{285}$ for AES-128 and AES-256, respectively.

Given that AEGIS-128 and Tiaoxin-346 have a key size of 128 bits, they fall under security category 1, while Rocca-S, with a key size of 256 bits, falls under security category 3.

Now we compare the security of the ciphers with the security strength of NIST, as shown in Table 14 and we present the gate counts for Grover search under the constraint of MAXDEPTH

Table 15 outlines the cost projections for executing Grover's algorithm against the ciphers within a specified depth limit. The table showcases outcomes achieved by imposing this depth threshold, requiring the parallelization of Grover's algorithm through inner parallelization, aligning with our analysis assumptions. For the Grover oracle, a single 256-bit plaintext-ciphertext pair suffices for key recovery. Following the Grover search, each measured potential key undergoes classical verification against one plaintext-ciphertext pair.

## Conclusion

In this study, we conducted a thorough analysis of the security of three AES-based AEAD ciphers, namely Rocca-S, AEGIS-128, and Tiaoxin-346, against quantum adversaries. Quantum computing introduces new attack possibilities that were not feasible in the classical era. Among these attacks, Grover's search algorithm stands out as a powerful technique to reduce the search complexity to the square root. To apply Grover's algorithm, it is necessary to implement the target cipher as a quantum circuit. Therefore, we developed quantum circuit constructions for these ciphers, specifically focusing on optimizing the circuit's T-depth. Using these constructions, we estimated the costs associated with implementing Grover's exhaustive key search algorithm for the studied ciphers, we have shown that estimation of the $G$-cost $\times$ $D$ (total gates $\times$ depth) of Rocca-S,

| $k$ | | MAXDEPTH= $2^{40}$ | MAXDEPTH= $2^{64}$ | MAXDEPTH= $2^{96}$ | $G$-cost × $D$ |
|---|---|---|---|---|---|
| 128 | AEGIS-128 | $1.14 \cdot 2^{124}$ | $1.14 \cdot 2^{100}$ | $1.39 \cdot 2^{86*}$ | $1.14 \cdot 2^{164}$ |
| | TIAOXIN | $1.22 \cdot 2^{124}$ | $1.22 \cdot 2^{100}$ | $1.25 \cdot 2^{87*}$ | $1.22 \cdot 2^{164}$ |
| | NIST[23] | $2^{130}$ | $2^{106}$ | $2^{74}$ | $2^{170}$ |
| | $NIST_{up}$[31] | $2^{117}$ | $2^{93}$ | $2^{84}$ | $2^{157}$ |
| 256 | Rocca-S | $1.09 \cdot 2^{253}$ | $1.09 \cdot 2^{229}$ | $1.09 \cdot 2^{197}$ | $1.08 \cdot 2^{294}$ |
| | Rocca[40] | $1.56 \cdot 2^{251}$ | $1.56 \cdot 2^{227}$ | $1.56 \cdot 2^{195}$ | $1.56 \cdot 2^{291}$ |
| | NIST[23] | $2^{258}$ | $2^{234}$ | $2^{202}$ | $2^{298}$ |
| | $NIST_{up}$[31] | $2^{245}$ | $2^{221}$ | $2^{189}$ | $2^{285}$ |

**Table 14.** Cost of Grover search on the ciphers under MAXDEPTH. Note that $*$ denotes a special case as the attack does not require any parallelization.

| Scheme | MD | $r$ | $M$ | $\log_2$ (MKP) | $D$ | $W$ | $G$-cost | $DW$-cost |
|---|---|---|---|---|---|---|---|---|
| (a) The depth cost metric is the full depth D only. | | | | | | | | |
| Aegis-128 | $2^{40}$ | 1 | $1.33 \cdot 2^{75}$ | −75.42 | $1.00 \cdot 2^{40}$ | $1.06 \cdot 2^{88}$ | $1.14 \cdot 2^{124}$ | $1.06 \cdot 2^{128}$ |
| Aegis-128 | $2^{64}$ | 1 | $1.33 \cdot 2^{27}$ | −27.42 | $1.00 \cdot 2^{64}$ | $1.06 \cdot 2^{40}$ | $1.14 \cdot 2^{100}$ | $1.06 \cdot 2^{104}$ |
| Aegis-128 | $2^{96}$ | 1 | $1.00 \cdot 2^{0}$ | −0.66 | $1.63 \cdot 2^{77}$ | $1.59 \cdot 2^{12}$ | $1.39 \cdot 2^{86}$ | $1.30 \cdot 2^{90}$ |
| Rocca-S-256 | $2^{40}$ | 1 | $1.28 \cdot 2^{203}$ | −203.35 | $1.00 \cdot 2^{40}$ | $1.53 \cdot 2^{216}$ | $1.09 \cdot 2^{253}$ | $1.53 \cdot 2^{256}$ |
| Rocca-S-256 | $2^{64}$ | 1 | $1.28 \cdot 2^{155}$ | −155.35 | $1.00 \cdot 2^{64}$ | $1.53 \cdot 2^{168}$ | $1.09 \cdot 2^{229}$ | $1.53 \cdot 2^{232}$ |
| Rocca-S-256 | $2^{96}$ | 1 | $1.28 \cdot 2^{91}$ | −91.35 | $1.00 \cdot 2^{96}$ | $1.53 \cdot 2^{104}$ | $1.09 \cdot 2^{197}$ | $1.53 \cdot 2^{200}$ |
| Tiaoxin-128 | $2^{40}$ | 1 | $1.91 \cdot 2^{73}$ | −73.93 | $1.00 \cdot 2^{40}$ | $1.73 \cdot 2^{87}$ | $1.22 \cdot 2^{124}$ | $1.73 \cdot 2^{127}$ |
| Tiaoxin-128 | $2^{64}$ | 1 | $1.91 \cdot 2^{25}$ | −25.93 | $1.00 \cdot 2^{64}$ | $1.73 \cdot 2^{39}$ | $1.22 \cdot 2^{100}$ | $1.73 \cdot 2^{103}$ |
| Tiaoxin-128 | $2^{96}$ | 1 | $1.00 \cdot 2^{0}$ | −0.66 | $1.95 \cdot 2^{76}$ | $1.82 \cdot 2^{13}$ | $1.25 \cdot 2^{87}$ | $1.77 \cdot 2^{90}$ |
| **Scheme** | MD | $r$ | $M$ | $\log_2$ (MKP) | $D$ | $W$ | $G$-cost | $T$-$DW$-cost |
| (b) The depth cost metric is the T depth T-D only. | | | | | | | | |
| Aegis-128 | $2^{40}$ | 1 | $1.71 \cdot 2^{65}$ | −65.77 | $1.00 \cdot 2^{40}$ | $1.35 \cdot 2^{78}$ | $1.28 \cdot 2^{119}$ | $1.35 \cdot 2^{118}$ |
| Aegis-128 | $2^{64}$ | 1 | $1.71 \cdot 2^{17}$ | −17.77 | $1.00 \cdot 2^{64}$ | $1.35 \cdot 2^{30}$ | $1.28 \cdot 2^{95}$ | $1.35 \cdot 2^{94}$ |
| Aegis-128 | $2^{96}$ | 1 | $1.00 \cdot 2^{0}$ | −0.66 | $1.85 \cdot 2^{72}$ | $1.59 \cdot 2^{12}$ | $1.39 \cdot 2^{86}$ | $1.46 \cdot 2^{85}$ |
| Rocca-S-256 | $2^{40}$ | 1 | $1.63 \cdot 2^{193}$ | −193.70 | $1.00 \cdot 2^{40}$ | $1.95 \cdot 2^{206}$ | $1.23 \cdot 2^{248}$ | $1.95 \cdot 2^{246}$ |
| Rocca-S-256 | $2^{64}$ | 1 | $1.63 \cdot 2^{145}$ | −145.70 | $1.00 \cdot 2^{64}$ | $1.95 \cdot 2^{158}$ | $1.23 \cdot 2^{224}$ | $1.95 \cdot 2^{222}$ |
| Rocca-S-256 | $2^{96}$ | 1 | $1.63 \cdot 2^{81}$ | −81.70 | $1.00 \cdot 2^{96}$ | $1.95 \cdot 2^{94}$ | $1.23 \cdot 2^{192}$ | $1.95 \cdot 2^{190}$ |
| Tiaoxin-128 | $2^{40}$ | 1 | $1.23 \cdot 2^{64}$ | −64.30 | $1.00 \cdot 2^{40}$ | $1.12 \cdot 2^{78}$ | $1.39 \cdot 2^{119}$ | $1.12 \cdot 2^{118}$ |
| Tiaoxin-128 | $2^{64}$ | 1 | $1.23 \cdot 2^{16}$ | −16.30 | $1.00 \cdot 2^{64}$ | $1.12 \cdot 2^{30}$ | $1.39 \cdot 2^{95}$ | $1.12 \cdot 2^{94}$ |
| Tiaoxin-128 | $2^{96}$ | 1 | $1.00 \cdot 2^{0}$ | −0.66 | $1.11 \cdot 2^{72}$ | $1.82 \cdot 2^{13}$ | $1.25 \cdot 2^{87}$ | $1.01 \cdot 2^{86}$ |

**Table 15.** Sizes of circuits for parallel Grover key search against Aegis are detailed, employing *inner* parallelization as discussed in Section 2.3. Here, $r$ represents the number of plaintext-ciphertext pairs used in the Grover oracle, *MKP* indicates the probability of spurious keys in the subset containing the target key, MD denotes MAXDEPTH, $M$ is the number of subsets into which the key-space is divided, and D and W are the depth and qubit width of the full circuit, respectively. DW represents the full circuit cost in terms of depth × width, while T-DW signifies the T-depth × width circuit cost.

AEGIS-128, and Tiaoxin-346 are $1.08 \times 2^{294}$, $1.14 \times 2^{164}$, and $1.22 \times 2^{164}$ respectively, which were higher than those required for implementing Grover's algorithm on AES. Based on the reported results, we conclude that these ciphers are secure against quantum adversaries when considering Grover's attack.

As an immediate follow-up, similar analyses can be conducted on various other AEAD schemes to address security concerns in the quantum adversarial model. Another interesting direction would be to analyze these schemes when the underlying AES is replaced with different block ciphers. Additionally, developing a tool to automatically analyze these schemes against Grover's attack, with the underlying block cipher as an input parameter, would be highly beneficial. The proposed model can be extended for other such AES-based AEAD ciphers. Furthermore, future research on different quantum error correcting schemes might yield substantially optimized outcomes in practice. Identifying improvements in the cipher building blocks may be regarded as a key focus for upcoming research efforts.

Received: 14 December 2023; Accepted: 1 August 2024
Published online: 10 September 2024

## References


1. Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, 124–134, https://doi.org/10.1109/SFCS.1994.365700 (IEEE Computer Society).
2. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**, 1484–1509. https://doi.org/10.1137/S0097539795293172 (1997).
3. Grassl, M., Langenberg, B., Roetteler, M. & Steinwandt, R. Applying grover's algorithm to aes: quantum resource estimates. In *Post-Quantum Cryptography: 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings 7*, 29–43 (Springer, 2016).
4. Langenberg, B., Pham, H. & Steinwandt, R. Reducing the cost of implementing AES as a quantum circuit. *IACR Cryptol. ePrint Arch.* **2**, 854 (2019).
5. 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*, vol. 12106 of *Lecture Notes in Computer Science*, 280–310, https://doi.org/10.1007/978-3-030-45724-2_10 (Springer).
6. Kuwakado, H. & Morii, M. Security on the quantum-type even-mansour cipher. In *Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012, Honolulu, HI, USA, October 28-31, 2012*, 312–316 (2012).
7. Ulitzsch, V. & Seifert, J. Breaking the quadratic barrier: Quantum cryptanalysis of milenage, telecommunications' cryptographic backbone. *IACR Cryptol. ePrint Arch.* **6**, 733 (2022).
8. Hosoyamada, A. & Sasaki, Y. Quantum demiric-selçuk meet-in-the-middle attacks: Applications to 6-round generic feistel constructions. In Catalano, D. & Prisco, R. D. (eds.) *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, vol. 11035 of *Lecture Notes in Computer Science*, 386–403, https://doi.org/10.1007/978-3-319-98113-0_21 (2018).
9. Hosoyamada, A. & Sasaki, Y. Cryptanalysis Against Symmetric-Key Schemes with Online Classical Queries and Offline Quantum Computations. In Smart, N. P. (ed.) *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, vol. 10808 of *Lecture Notes in Computer Science*, 198–218, https://doi.org/10.1007/978-3-319-76953-0_11 (2018).
10. Kaplan, M. Quantum attacks against iterated block ciphers. *CoRR***abs/1410.1434** (2014). 1410.1434.
11. Kaplan, M., Leurent, G., Leverrier, A. & Naya-Plasencia, M. Quantum differential and linear cryptanalysis. *IACR Trans. Symmetric Cryptol.***2016**, 71–94, https://doi.org/10.13154/tosc.v2016.i1.71-94 (2016).
12. Kaplan, M., Leurent, G., Leverrier, A. & Naya-Plasencia, M. Breaking symmetric cryptosystems using quantum period finding. In Robshaw, M. & Katz, J. (eds.) *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, vol. 9815 of *Lecture Notes in Computer Science*, 207–237, https://doi.org/10.1007/978-3-662-53008-5_8 (2016).
13. Anand, R., Maitra, A. & Mukhopadhyay, S. Grover on simon. *Quant. Inf. Process.* **19**, 340. https://doi.org/10.1007/s11128-020-02844-w (2020).
14. 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 - 21st International Conference on Cryptology in India, Bangalore, India, December 13-16, 2020, Proceedings*, vol. 12578 of *Lecture Notes in Computer Science*, 395–413, https://doi.org/10.1007/978-3-030-65277-7_18 (Springer).
15. Jang, K., Choi, S., Kwon, H. & Seo, H. Grover on SPECK: Quantum resource estimates. *IACR Cryptol. ePrint Arch.* **8**, 640 (2020).
16. Jang, K., Kim, H., Eum, S. & Seo, H. Grover on GIFT. *IACR Cryptol. ePrint Arch.* **9**, 1405 (2020).
17. Jang, K. *et al.* Efficient implementation of present and gift on quantum computers. *Appl. Sci.* **11**, 4776 (2021).
18. Bathe, B. N., Anand, R. & Dutta, S. Evaluation of grover's algorithm toward quantum cryptanalysis on chacha. *Quant. Inf. Process.* **20**, 394. https://doi.org/10.1007/s11128-021-03322-7 (2021).
19. Baksi, A., Jang, K., Song, G., Seo, H. & Xiang, Z. Quantum implementation and resource estimates for rectangle and knot. *Quant. Inf. Process.* **20**, 395. https://doi.org/10.1007/s11128-021-03307-6 (2021).
20. Jang, K., Baksi, A., Breier, J., Seo, H. & Chattopadhyay, A. Quantum implementation and analysis of DEFAULT. *IACR Cryptol. ePrint Arch.* **3**, 647 (2022).
21. 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*, vol. 13793 of *Lecture Notes in Computer Science*, 614–644, https://doi.org/10.1007/978-3-031-22969-5_21 (Springer).
22. Fowler, A. G. Time-optimal quantum computation. arXiv preprint arXiv:1210.4626 (2012).
23. Nist. *NIST Post-Quantum Cryptography*https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf *(2016).*
24. Jean, J. & Nikolić, I. Efficient design strategies based on the aes round function. In *Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers 23*, 334–353 (Springer, 2016).
25. Wu, H. & Preneel, B. Aegis: a fast authenticated encryption algorithm (v1. 1) caesar competition.
26. Nikolic, I. Tiaoxin-346. *Submission to the CAESAR competition***1** (2014).
27. Almazrooie, M., Samsudin, A., Abdullah, R. & Mutter, K. N. Quantum reversible circuit of AES-128. *Quant. Inf. Process.* **17**, 112. https://doi.org/10.1007/s11128-018-1864-3 (2018).
28. Jang, K. *et al.* Quantum analysis of AES. *IACR Cryptol. ePrint Arch.* **4**, 683 (2022).
29. Langenberg, B., Pham, H. & Steinwandt, R. Reducing the cost of implementing the advanced encryption standard as a quantum circuit. *IEEE Trans. Quant. Eng.* **1**, 1–12 (2020).
30. 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 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, vol. 12492 of *Lecture Notes in Computer Science*, 697–726, https://doi.org/10.1007/978-3-030-64834-3_24 (Springer).
31. Nist. *NIST Post-Quantum Cryptography*https://doi.org/10.6028/NIST.FIPS.205.ipd *(2016).*
32. Jaques, S., Naehrig, M., Roetteler, M. & Virdia, F. Implementing grover oracles for quantum key search on aes and lowmc. In *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30*, 280–310.
33. Selinger, P. Quantum circuits of t-depth one. *Phys. Rev. A* **87**, 042302 (2013).
34. Grover, L. K. A fast quantum mechanical algorithm for database search. In Miller, G. L. (ed.) *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, 212–219, https://doi.org/10.1145/237814.237866 (ACM).

35. Jaques, S. & Schanck, J. M. Quantum cryptanalysis in the ram model: Claw-finding attacks on sike. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39*, 32–61 (Springer, 2019).
36. Kim, P., Han, D. & Jeong, K. C. Time-space complexity of quantum search algorithms in symmetric cryptanalysis: Applying to AES and SHA-2. *Quant. Inf. Process.* **17**, 339. https://doi.org/10.1007/s11128-018-2107-3 (2018).
37. Zalka, C. Grover's quantum searching algorithm is optimal. *Phys. Rev. A* **60**, 2746 (1999).
38. Trefethen, L. N. & Bau, D. *Numerical Linear Algebra*, vol. 181 (2022).
39. Xiang, Z., Zeng, X., Lin, D., Bao, Z. & Zhang, S. Optimizing implementations of linear layers. *IACR Trans. Symmetric Cryptol.***2020**, 120–145, https://doi.org/10.13154/tosc.v2020.i2.120-145 (2020).
40. Anand, R. & Isobe, T. Quantum security analysis of rocca. *Quant. Inf. Process.* **22**, 164 (2023).
41. Wiebe, N. & Roetteler, M. Quantum arithmetic and numerical analysis using repeat-until-success circuits. *Quant. Inf. Comput.* **16**, 134–178 (2016).

## Acknowledgements

Takanori Isobe is supported by JST, PRESTO Grant Number JPMJPR2031. This research was in part conducted under a contract of "Research and development on new generation cryptography for secure wireless communication services" among "Research and Development for Expansion of Radio Wave Resources (JPJ000254)", which was supported by the Ministry of Internal Affairs and Communications, Japan.

## Author contributions

R.A. and M.R. conceived of the presented idea. S.M. developed the theory and performed the computations along with M.R. and R.A.. T.I. and S.S. verified and supervised the results. All authors discussed the results and contributed to the final manuscript. All authors reviewed the manuscript.

## Competing interests

The authors declare no competing interests.

## Additional information

**Correspondence** and requests for materials should be addressed to R.A. or M.R.

**Reprints and permissions information** is available at www.nature.com/reprints.

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.