Quantum Opacity, Classical Clarity: A Hybrid Approach to Quantum Circuit Obfuscation

Amal Raj
Singapore Institute of Technology
Punggol, Singapore
amal.raj@singaporetech.edu.sg

Vivek Balachandran
Singapore Institute of Technology
Punggol, Singapore
vivek.b@singaporetech.edu.sg

Abstract

Quantum computing leverages quantum mechanics to achieve computational advantages over classical hardware, but the use of third-party quantum compilers in the Noisy Intermediate-Scale Quantum (NISQ) era introduces risks of intellectual property (IP) exposure. We address this by proposing a novel obfuscation technique that protects proprietary quantum circuits by inserting additional quantum gates prior to compilation. These gates corrupt the measurement outcomes, which are later corrected through a lightweight classical post-processing step based on the inserted gate structure. Unlike prior methods that rely on complex quantum reversals, barriers, or physical-to-virtual qubit mapping, our approach achieves obfuscation using compiler-agnostic classical correction. We evaluate the technique across five benchmark quantum algorithms—Shor's, QAOA, Bernstein-Vazirani, Grover's, and HHL—using IBM's Qiskit framework. The results demonstrate high Total Variation Distance (above 0.5) and consistently negative Degree of Functional Corruption (DFC), confirming both statistical and functional obfuscation. This shows that our method is a practical and effective solution for the security of quantum circuit designs in untrusted compilation flows.

CCS Concepts

• **Security and privacy → Software security engineering**; *Privacy-preserving protocols*; *Pseudonymity, anonymity and untraceability*.

Keywords

Quantum Obfuscation, Classical Deobfuscation, Security

ACM Reference Format:
Amal Raj and Vivek Balachandran. 2025. Quantum Opacity, Classical Clarity: A Hybrid Approach to Quantum Circuit Obfuscation. In *Proceedings of the 2025 Workshop on Research on Offensive and Defensive Techniques in the Context of Man At The End (MATE) Attacks (CheckMATE '25), October 13–17, 2025, Taipei, Taiwan.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3733817.3762698

1 Introduction

Quantum computing is an emerging field that utilizes the principles of quantum mechanics to achieve exponential speed-ups for certain tasks compared to even the most powerful classical computers [1]. By exploiting quantum properties such as superposition and entanglement, quantum bits—or qubits—can exist in multiple states simultaneously, allowing quantum computers to tackle problems that are computationally hard for classical systems, such as factoring large numbers [2], simulating molecular interactions [3], or optimizing complex systems [4]. This potential has led to extensive research in the field, with leading institutions and companies driving quantum computing forward. Recent developments, ranging from hardware advances such as Google's Willow chip [5] and Microsoft's pursuit of topological qubits based on Majorana fermions [6] to security-oriented prototypes like the QIris framework for quantum-accelerated rainbow table attacks using Grover's algorithm [7], reflect growing efforts to make quantum computing practical.

However, the current era of quantum computing, called the Noisy Intermediate-Scale Quantum (NISQ) [8] era, is marked by significant technical challenges. NISQ devices operate with limited numbers of qubits, and are prone to noise and decoherence, meaning their quantum states degrade rapidly over time [9]. This noise causes errors in calculations, limiting the complexity and depth of algorithms that can be run securely. To bridge this gap, developers often depend on third-party quantum compilers to optimize the circuit. Compilers are tools that translate high-level quantum algorithms into machine-readable instructions tailored to a specific hardware. Several companies have developed compilers catering to their hardware platforms, such as IBM's Qiskit [10], Quantinuum's t|ket⟩[11], and Rigetti's Quilc [12], among others.

This reliance on external tools necessitates the need to safeguard sensitive quantum code. When untrusted compilers receive proprietary algorithms, the potential exists for valuable intellectual property (IP) to be reverse-engineered, stolen, or abused, as quantum computing becomes increasingly of commercial interest. To counter these risks, researchers and developers have been investigating obfuscation mechanisms, which seek to hide code structure and purpose without affecting its functionality. Such mechanisms range from circuit-level transformations to higher-level program-control techniques, such as the quantum opaque predicate method [13], which conceals classical control flow in hybrid quantum–classical systems. Addressing these challenges is crucial as quantum computing advances from theoretical promise to practical reality.

1.1 Proposed Idea

We propose a novel quantum obfuscation technique that simplifies the deobfuscation process by leveraging classical post-processing. The technique involves inserting additional quantum gates into

CheckMATE '25, Taipei, Taiwan

ACM ISBN 979-8-4007-1906-6/25/10
https://doi.org/10.1145/3733817.3762698

some or all qubits of the original quantum circuit prior to compilation by an untrusted third-party compiler. These gates can be selected randomly or based on the gates in the original circuit, with the only constraint being the exclusion of Hadamard gates or their controlled versions, as they would introduce superposition which is difficult to correct classically. Once the circuit is compiled and executed, the resulting classical measurement outcomes are significantly corrupted due to the presence of the inserted gates. These outputs are then passed through a classical correction circuit, constructed based on knowledge of the inserted "encryptor" gates, to recover the actual measurement results — effectively reversing the obfuscation classically. A diagrammatic representation of the same is shown in Fig. 1.

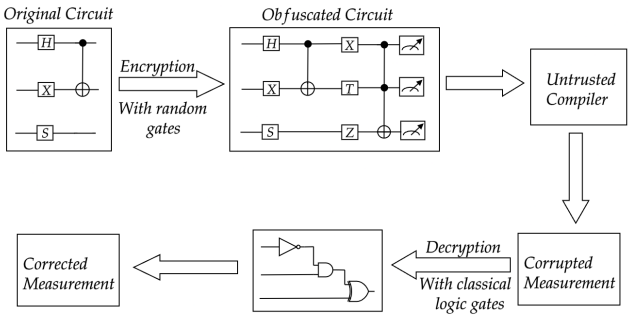

Figure 1: Workflow of our proposed idea

Existing obfuscation techniques for quantum circuits employ various strategies. One approach involves inserting a random quantum circuit into the original design. Its inverse is then compiled using another quantum compiler and inserted at the appropriate location after the original circuit has been compiled [14]. This method implicitly assumes that the obfuscator or developer has access to a personal quantum compiler or quantum computer, which may not always be practical or scalable. Other techniques include inserting dummy gates that are subsequently removed during optimization [15], or applying logic locking methods to restrict functionality without the correct key [16, 17]. These methods often require barriers to mark insertion points or knowledge of the physical-to-virtual (p2v) qubit mappings, making deobfuscation non-trivial and error-prone. By utilizing classical post-processing for decryption, our approach eliminates the need for such indicators or intricate quantum reversals. By avoiding the need to track insertion locations and utilizing classical post-processing for decryption, we reduce overhead and improve practicality when working with untrusted compilers.

1.2 Organization of the Paper

The rest of this paper is structured as follows. Section 2 describes background on quantum computing, covering qubits, quantum gates, compilation, and measurement operations. Section 3 is a survey of current works on obfuscating quantum circuits, citing current techniques and their shortcomings. Section 4 is the introduction of the evaluation metrics, Total Variation Distance (TVD) and Degree of Functional Corruption (DFC), used to quantify the effectiveness of obfuscation. Section 5 describes our suggested obfuscation technique, its methodology, and an example application of the QAOA circuit. Section 6 demonstrates the simulation results on a range of quantum algorithms, which verifies the effectiveness of the approach. Section 7 describes the shortcomings of our research and proposes directions for future work. Section 8 summarizes the paper, including key contributions and implications.

2 Background

2.1 Basics of Quantum Computing

2.1.1 Qubits. Quantum bits or qubits are the fundamental units of quantum computing, analogous to bits in classical computing. While a classical bit exists only in one of the two states, 0 or 1, a qubit can exist in either of its basis states, $|0\rangle$ or $|1\rangle$, or as a linear combination of these states, known as superposition. Mathematically, the state of a qubit $|\psi\rangle$ is expressed as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{1}$$

where α and β are complex numbers satisfying the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. Here, $|\alpha|^2$ and $|\beta|^2$ represent the probabilities of measuring the qubit in states $|0\rangle$ and $|1\rangle$, respectively [18].

2.1.2 Quantum Gates. Quantum gates are the building blocks of quantum circuits, analogous to logic gates in classical computing. They are represented by unitary matrices, which ensure reversibility—a key property of quantum computation. A matrix U is said to be unitary if it satisfies the condition $UU^\dagger = I$, where $U^\dagger$ represents the conjugate transpose of the matrix U, and I is the identity matrix. Quantum gates operate on one or more qubits to perform transformations such as bit flips, phase shifts, or entanglement creation. Common gates include:

i. Pauli-X gate: Flips the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice-versa.
$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
ii. Pauli-Y gate: Combines a bit flip and a phase shift, transforming $|0\rangle$ to $i\,|1\rangle$ and $|1\rangle$ to $-i\,|0\rangle$.
$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$
iii. Pauli-Z gate: Introduces a phase change if the qubit is in state $|1\rangle$, and leaves $|0\rangle$ unaffected.
$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$
iv. Hadamard (H) gate: Transforms either basis states to a superposition with equal probabilities of $|0\rangle$ and $|1\rangle$.
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
v. Controlled-NOT (CNOT) gate: A two-qubit gate that flips the target qubit's state if the control qubit is $|1\rangle$, entangling

the qubits if the control is in superposition.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

2.1.3 Compilation Process. Quantum compilation is essentially about translating an arbitrary high-level quantum algorithm into an implementation which can be run over an underlying quantum hardware platform. In an ideal quantum computing paradigm, all qubits can be connected to each other, and in an ideal setup, any quantum gate can be composed in front of any qubit or any combination of qubits. Realistic quantum hardware, especially in the NISQ era, places strong constraints on this ideal notion: they have only a few native gates (called basis gates) and restricted qubit interconnectivity, given by an underlying coupling map [19]. The coupling map is often modeled as a graph, whose vertices represent physical qubits and whose edges represent pairs of qubits that can be directly connected using two-qubit gates. Compilation fills this gap between theoretical algorithms and feasible hardware by using the following well-defined intermediate steps:

i. Gate Synthesis: Complex gates (e.g., controlled-Z) are broken into basis gates like CNOT and single-qubit rotations.
ii. Qubit Mapping: Logical qubits in the algorithm are assigned to physical qubits on the hardware, adhering to the coupling map. This physical-to-virtual (p2v) mapping can be complex, as the initial assignment may not align with connectivity requirements. If a two-qubit gate targets non-adjacent qubits, SWAP gates—each costing three CNOTs—are inserted to reposition qubit states, increasing circuit depth and error potential.
iii. Optimization: Redundant gates are removed, and the circuit is refined to reduce errors.
iv. Native Conversion: The circuit is translated into the device's gate set (e.g., {RZ, X, CZ}).

2.1.4 Measurement Operation. Measurement collapses a qubit's superposition into a classical state, either 0 or 1, with probabilities determined by its amplitudes. For a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the probability of measuring $|0\rangle$ is $|\alpha|^2$, and $|1\rangle$ is $|\beta|^2$. Post-measurement, the qubit loses its quantum properties, becoming a classical bit. For instance, measuring $|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ yields 0 or 1 with equal probability, and the system collapses to the observed state.

3 Related Works

In the past decade, many different obfuscation techniques have been developed for quantum circuits. Our threat model aligns very closely with that of the work discussed in [15], which proposed the inclusion of a dummy CNOT gate for optimal corruption of the functionality of a quantum circuit. Barriers are placed around the dummy gate to facilitate post-compilation removal, and around the other gates to mask its presence. Although efficient, this approach loses circuit optimization since the compilers cannot optimize across barriers, and the barriers may be seen as a sign of the presence of the dummy gate, thereby compromising security.

Das and Ghosh [14] present an obfuscation technique based on the addition of a random quantum circuit composed of gates of the original circuit, to the original circuit before compilation. An inverse of the random circuit is separately compiled and concatenated after compilation in order to restore functionality. While their approach uses quantum reversibility, ours differs by classically correcting corrupted outputs without quantum reversals. Their approach can be vulnerable to information leakage if the two circuits are given to the same or related compilers, and compiler gate library variations can hinder physical-to-virtual (p2v) qubit map continuity, which needs careful insertion. If a random circuit is inserted in the middle of a circuit, barriers would be needed for post-compilation detection, restoring the security-optimization tradeoff. While their enhanced method enhances random circuit insertion for quantum circuits, our approach uniquely relies on classical post-processing to deobfuscate without requiring quantum reversals or barriers.

Logic locking methods, a hardware security technique that embeds a key-dependent transformation to prevent unauthorized use or reverse engineering, have also been explored for quantum circuit obfuscation [20]. The work in [16] proposes the addition of controlled gates or the substitution of existing gates with their controlled counterparts, using ancilla qubits as controls in the $|0\rangle$ (idle gate) or $|1\rangle$ (active gate) state, where the ancilla configuration encodes a binary key. An enhancement is presented in [17], which combines multiple key bits into a single ancilla qubit using a technique called "H-masking," wherein Hadamard gates are inserted during encryption and selectively replaced with Pauli-X gates during decryption to conceal the key. While effective, these methods increase circuit complexity and may exacerbate crosstalk errors [21] in noisy NISQ environments, potentially making them impractical under current hardware constraints.

Rehman, Langford, and Liu [22] propose an obfuscation method that adjusts phase gate angles. Their approach segments the circuit into layers of phase and non-phase gates, sets rotation angles in the phase layers using a binary key, and inserts a dummy phase gate layer—with barriers between layers—to disrupt circuit functionality. However, the method requires the presence of phase gates, limiting its applicability for circuits without them. Furthermore, the mandatory use of barriers hinders circuit optimization and may leave the dummy layer exposed to attackers. In contrast, our approach avoids barriers and phase-specific constraints, offering broader applicability and enabling straightforward deobfuscation via classical correction.

4 Metrics

4.1 Total Variation Distance

Total Variation Distance (TVD) is a statistical measure of the difference between two probability distributions[23]. TVD is used here as a means of quantifying the degradation achieved by the addition of gates to the circuit. It is computed as the sum of absolute differences between the measurement outcomes of original and obfuscated circuits, normalized by the number of shots, as shown in the equation below.

$$\text{TVD} = \frac{\sum_i |x_{\text{i,obfus}} - x_{\text{i,orig}}|}{2 \times \text{Number of shots}} \tag{2}$$

where $x_{\text{i,orig}}$ and $x_{\text{i,obfus}}$ refer to the count of the i^{th} measurement output of the circuits, before and after addition of encryptor gates respectively. A TVD of 0 indicates identical distributions, and 1 represents maximum difference between the distributions. Thus, a higher TVD in our case would indicate that there is significant difference between the outputs of original circuit and encrypted circuit, which is desired.

4.2 Degree of Functional Corruption

Degree of Functional Corruption (DFC) is a metric introduced by Das and Ghosh [14] to quantify the degree of obfuscation. It is defined as the difference between the count of correct output of the obfuscated circuit and the highest incorrect output of the obfuscated circuit, normalized over the number of shots, given by the equation below.

$$\text{DFC} = \frac{\text{Count of correct output} - \text{Highest Incorrect Output}}{\text{Number of shots}} \quad (3)$$

If a circuit is perfectly obfuscated, every measured output would be incorrect. If the circuit produces same incorrect output over all shots, then normalized DFC would be -1. For an unobfuscated circuit, if the correct output is produced over all the shots, DFC would be 1. Thus, DFC lies in the range $[-1, 1]$, with lower values indicating better obfuscation.

5 Proposed Obfuscation Technique

5.1 Methodology

This section describes our obfuscation method for quantum circuits. The technique leverages specific properties of quantum gates and quantum measurement to enable classical decryption of corrupted outputs post-measurement.

The theoretical foundation of our method is based on the following key properties of quantum systems:

i. Every quantum state, upon measurement, collapses into the measured state. This means that once a quantum system is measured, it loses the quantum behavior and collapses into a classical state.
ii. Phase changes do not affect measurement outputs. If a gate is applied to a qubit such that it introduces only a phase change (such as the S-gate, T-gate, Z-gate, and so on), upon measurement, it would be as though the gate was never applied. Consider a qubit,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \quad (4)$$

After applying an arbitrary phase gate, which introduces a phase of angle θ, the state of qubit changes as

$$|\psi'\rangle = \alpha\,|0\rangle + e^{i\theta}\beta\,|1\rangle \quad (5)$$

The probability amplitudes of measuring $|\psi'\rangle$ as a 0 or 1 remain exactly same as that of $|\psi\rangle$, as $|e^{i\theta}| = 1$.
iii. A gate that changes the state of a qubit, from $|0\rangle$ to $|1\rangle$ and vice-versa, along with their controlled versions, introduce a definite bit flip to the measured output (if the gate introduced is not a controlled-gate), or introduce bit flip based on some other bit (if the gate is a controlled gate). The effect introduced by such gates to the measurement output can be reversed classically, by application of appropriate classical gates. A few examples are noted in Fig. 2.
iv. A gate that introduces both phase change and state change (such as Pauli Y-gate) will impact the measurement output as a combination of rules (ii) and (iii) above.

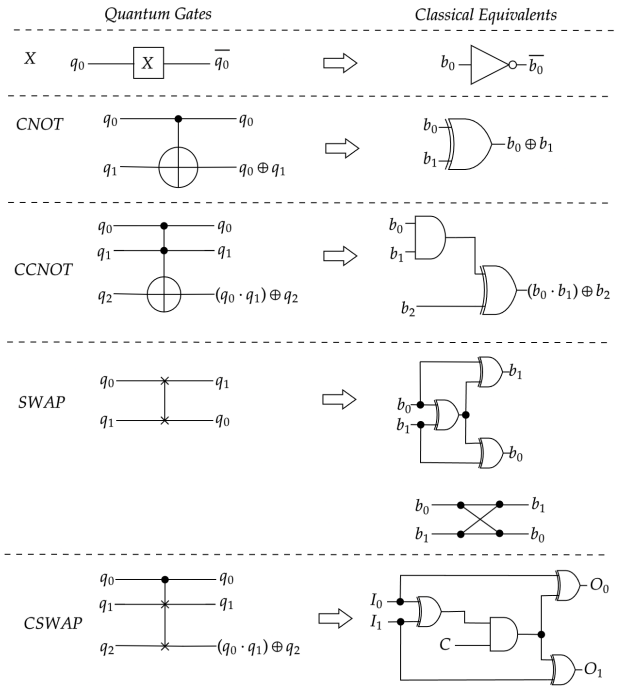

Figure 2: Some quantum gates with their classical equivalents

These properties form the basis for the following stages of our obfuscation technique:

5.1.1 Indexing Gates. The first step for applying our obfuscation scheme is to identify the gates that are to be added to the circuit. These gates can be state-changing gates (such as Pauli-X), phase-changing gates (such as Pauli-Z) or a combination of both (such as Pauli-Y), as well as their controlled versions. Hadamard gates and their controlled variants are deliberately excluded from our encryption process, as they introduce superposition and alter the measurement outcome space. Since our technique relies on classical post-processing to decrypt results, the non-deterministic nature of superposition makes it infeasible to reverse such transformations classically.

Once the set of gates is chosen, each of the individual gates within the set must be assigned a unique index. This index serves as a compact identifier in the key and ensures unambiguous decoding during decryption. The indexing scheme is user-defined, but it should be consistent and fixed for a given execution. Controlled versions of gates do not require their own separate indexing scheme; any gate, controlled or uncontrolled, can be assigned any unique index as long as it is used consistently throughout the encryption

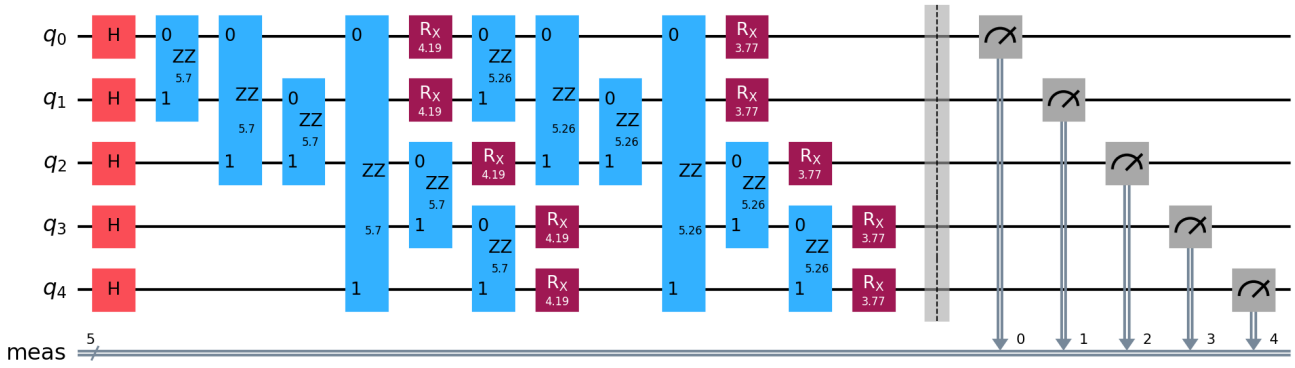

Figure 3: QAOA Circuit

process. This abstraction maintains the key light-weighted and structured even in the case of multiple gates.

5.1.2 Adding Gates and Key Generation. After the gate set has been finalized and indexed, the next step is to append the selected gates to the end of the original quantum circuit. The user determines which gate to add and to which qubit(s) it should be applied. These additions serve as the encryption layer of the obfuscation process.

To generate the key, the index of each gate, along with the corresponding qubit(s) it is applied to, is recorded. The format of the key includes delimiters or markers to separate gate indices from qubit references, as well as to distinguish between multiple gate-qubit entries. Importantly, the decryption process applies corrections in the reverse order of insertion. Therefore, the most recently added gate-qubit pair is placed at the beginning of the key, maintaining a stack-like (LIFO) [24] structure.

Although the selection of gates and their placement is user-defined, this process can also be randomized to enhance the security of the obfuscation. For example, gates can be randomly selected from the valid set and assigned to qubits chosen according to a pseudorandom function, provided that the user retains or stores the corresponding key structure.

5.1.3 Running on a Quantum Computer. After the obfuscation step has been performed, the obfuscated quantum circuit is then compiled and executed on a quantum computer. In practice, this may involve sending the circuit to a third-party compiler or a quantum service provider. The compiled circuit is then executed for a certain number of shots, and the measurement results are fed back to the user.

Since the inserted gates intentionally manipulate the measuring outcome, the outputs produced by the quantum hardware will not represent the real output of the original circuit. However, the encryption is designed such that the corrupted outputs can be rectified classically using the secret key generated during encryption. The next step is applying this key to restore the correct output, as detailed below.

5.1.4 Correcting outputs post measurement. Once the incorrect measurement output is obtained, the key—which is only available to the user—can be used to extract the correct output. The markers set in the key help in separating the gate-qubit sets, and subsequently the gate and qubit(s) from each set. Once the gate and qubit(s) are identified, the output is corrected classically, by applying appropriate classical operations as shown in Fig. 2. Gates that introduce only a phase change to the qubit can be ignored while correcting measurement output, as these gates do not affect measurement value. Classical operations for gates other than those shown in Fig. 2 can also be derived in a similar manner.

5.2 Case Study: QAOA circuit

The obfuscation technique has been demonstrated for the case of the Quantum Approximate Optimization Algorithm (QAOA) circuit. QAOA is a hybrid quantum-classical algorithm used for combinatorial optimization problems [25]. We have considered the implementation of QAOA for the MaxCut problem, demonstrated in IBM Qiskit Learning [26].

The QAOA circuit used is shown in Fig. 3. The circuit shown is the decomposed version of QAOA Ansatz taken in IBM Qiskit Learning. Upon measurement of the final output, we get the histogram as shown in Fig. 4.

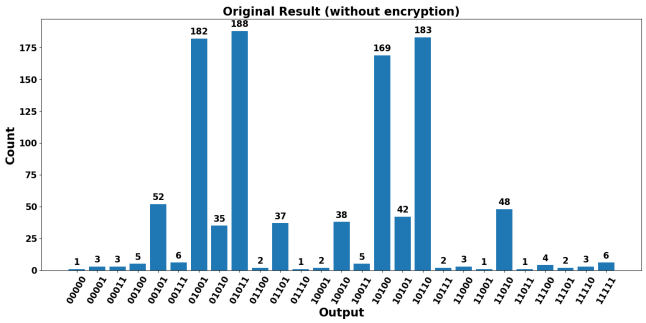

Figure 4: Histogram showing measurement count of the QAOA circuit

Note that 01001 and 01011, along with their complements 10110 and 10100 respectively, exhibit the highest peaks in the histogram.

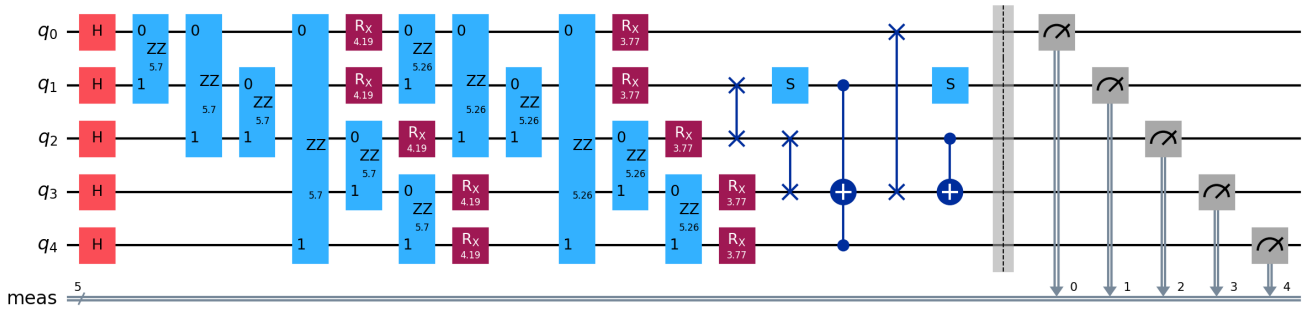

Figure 5: QAOA circuit with newly added gates

Due to Qiskit's bit-order convention (little-endian), these outputs correspond to 10010, 11010, 01101 and 00101 respectively. For 10010, the optimal partition groups vertices 0, 3 together and vertices 1, 2, 4 in the other set. For 11010, the grouping is vertices 0, 1, 3 in one set and vertices 2, 4 in the other.

Now, some extra gates are added to corrupt the measurement outputs. The circuit along with the added gates is shown in Fig. 5.

The addition of the extra gates has totally corrupted the measurement outputs, which is evident in the histogram shown in Fig. 6. It can be seen that peaks in the histogram, now correspond to different bitstrings compared to original results, indicating that the output has been degraded.

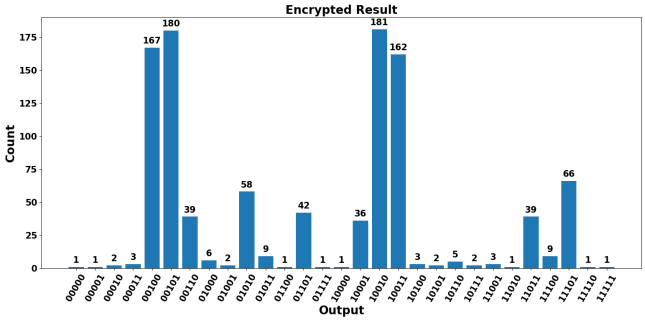

Figure 6: Measurement results after addition of extra gates

The results are now to be corrected using the gate-index mapping generated prior, and the key obtained after the addition of extra gates. For this case, the following information is available:

- The key obtained: *"2#2|1@5#1@2#2|3@3#1|4|3@2#3|0@5#1@1#2|3"*, with "@", "#" and "|" being the demarcators — "@" separating one gate-qubit(s) set from another, "#" separating gate from qubit(s), and "|" separating the qubits.
- From the key, the index-qubit(s) set is obtained to be the following — (2, 2, 1), (5, 1), (2, 2, 3), (3, 1, 4, 3), (2, 3, 0), (5, 1), (1, 2, 3). This is the order that will be used while correcting the output later.
- The gate indexing is the following — {0: "X", 1: "CNOT", 2: "SWAP", 3: "CCNOT", 4: "CSWAP", 5: "S"}.

The above information can now be used to correct the measured output. For each group in the index-qubit(s) set obtained above, apply the classical equivalent of the corresponding quantum gate, as shown in Fig. 2, to the bits indicated by the qubit number. If the quantum gate changes only the phase of a qubit, an identity operation can be considered as its classical equivalent, and the bits remain unaffected. The steps for decryption in this particular case is written below:

i. (2, 2, 1) — Swap the bits 1 and 2.
ii. (5, 1) — Identity operation as the "S" gate is a quantum phase gate.
iii. (2, 2, 3) — Swap the bits 2 and 3.
iv. (3, 1, 4, 3) — Flip the bit 3 if bits 1 and 4 are set, else do nothing.
v. (2, 3, 0) — Swap the bits 3 and 0.
vi. (5, 1) — Identity operation as the "S" gate is a quantum phase gate.
vii. (1, 2, 3) — Flip the bit 3 if bit 2 is set, else do nothing.

The corrected output after performing the above operations is shown in the Fig. 7 below. The peaks now correspond to the same values that were obtained prior to the addition of gates, indicating the incorrect results were correctly decoded to the actual measurement results. GitHub link for the project is available at [27].

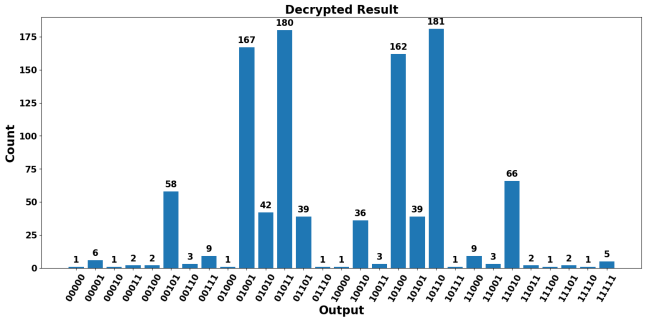

Figure 7: Corrected measurement results

6 Evaluation

This section presents the experimental evaluation of our proposed obfuscation scheme. All simulations were carried out using IBM's Qiskit framework on a noise-free AerSimulator backend. The evaluation is divided into two parts: the implementation setup used to apply encryption, and the performance results based on statistical metrics across various quantum algorithms.

6.1 Implementation Setup

All simulations were performed locally on a laptop with an AMD Ryzen 5 5500U processor (2.10 GHz), 8 GB RAM, and running Windows 11 Home. Five standard quantum algorithms were selected for evaluation: Shor's algorithm [2], the Quantum Approximate Optimization Algorithm (QAOA) [25], the Bernstein-Vazirani (BV) algorithm [28], Grover's algorithm [29], and the Harrow-Hassidim-Lloyd (HHL) algorithm [30]. These cover a broad range of application fields—ranging from optimization, factorization, and linear systems—to enable us to gauge the overall applicability of the proposed obfuscation method.

The quantum circuits varied in size, ranging from 4, 5, or 10 qubits depending on the algorithm. Each algorithm was executed 100 times, and each execution was done for 1,024 shots under identical, noise-free conditions. This ensured that all differences in output distribution could be attributed solely to the encryption process rather than noise or hardware-induced variability.

The encryptor gates belonged to a gate pool shared across all algorithms. Gates were chosen at random from this pool and applied to randomly chosen qubits in each experiment. In spite of the fact that this gate pool was shared among all algorithms, the method still realized good obfuscation effects. This suggests that even generic encryption gate sets can be broadly effective. Future work may investigate the benefits of algorithm-specific gate sets to further optimize obfuscation performance.

6.2 Evaluation Metrics

The obfuscation quality achieved by our technique was quantitatively evaluated using Total Variation Distance (TVD) and Degree of Functional Corruption (DFC) metrics. These metrics were computed for every execution of the five algorithms, results of which are summarized in the Table 1 below. Distribution of metrics across the various executions can be observed through the graphs shown in Figs. 8–12.

Table 1: Median TVD and DFC across the quantum algorithms

Algorithm Tested	Median TVD	Median DFC
Shor's	0.5381	-0.2583
QAOA	0.7817	-0.1123
BV	1	-1
Grover's	0.9097	-0.8169
HHL	0.8047	-0.5522

It can be observed that the median TVD value is consistently above 0.5 for all the tested algorithms. This statistical divergence indicates that the obfuscated circuits produce outputs that are distinct from their original counterparts, thereby effectively concealing the underlying functionality.

Similarly, the median DFC scores are negative for all five algorithms, confirming that the encryption scheme corrupts functional

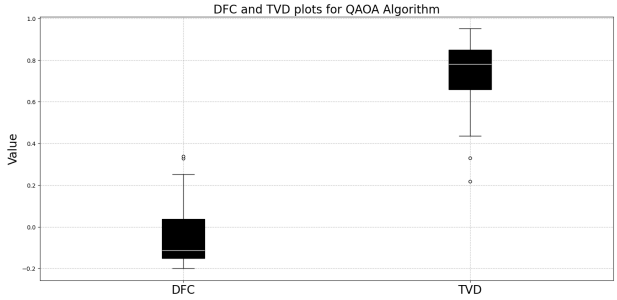

Figure 8: Distribution of TVD and DFC for QAOA implementation. QAOA circuit was obfuscated 100 times by addition of random gates each time, and the TVD and DFC values were computed each time.

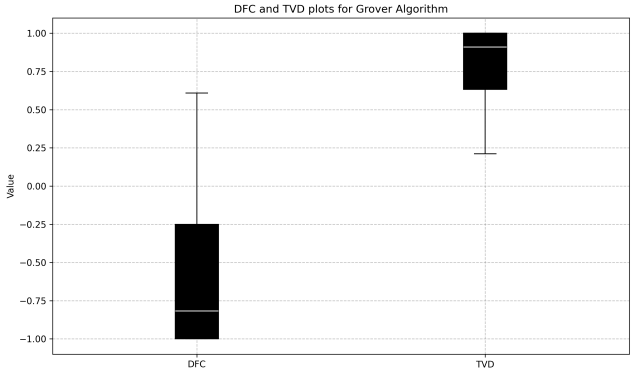

Figure 9: Distribution of TVD and DFC for Grover's algorithm implementation

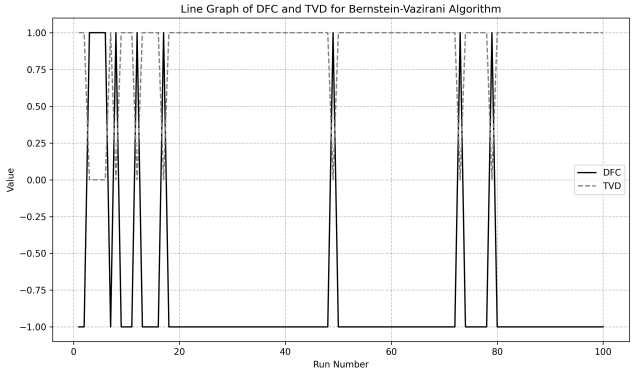

Figure 10: Distribution of TVD and DFC for Bernstein-Vazirani algorithm implementation

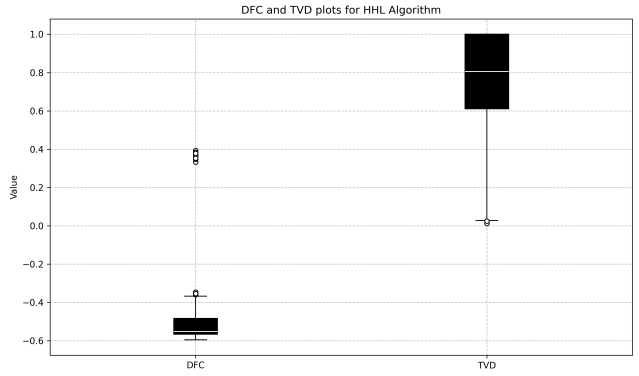

Figure 11: Distribution of TVD and DFC for HHL algorithm implementation

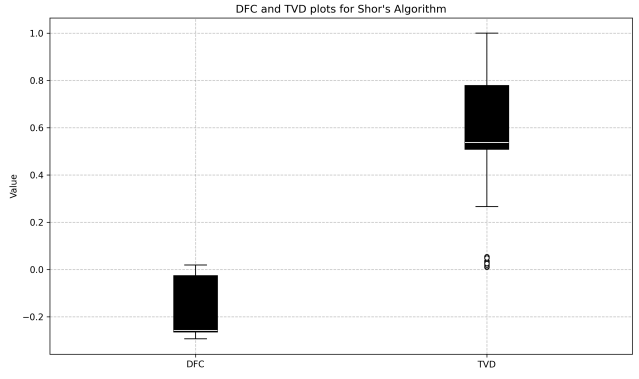

Figure 12: Distribution of TVD and DFC for Shor's algorithm implementation

behavior. This means that, aside from statistical divergence, the obfuscation also disrupts the functional correctness of the circuits such that the outputs become unreliable in the absence of the decryption key. While Bernstein-Vazirani algorithm and Grover's algorithm demonstrate strong obfuscation, with DFC values approaching -1, QAOA and Shor's algorithm exhibit more moderate DFC values. This variation could be due to the differences in structure of the algorithms, or more likely, due to the uniform gate pool applied across all circuits. Future work could explore algorithm-specific encryption strategies to further enhance functional disruption where needed.

7 Limitations and Future Works

While our proposed quantum obfuscation technique yields promising results in protecting hidden quantum circuits, it is restricted by some limitations worthy of greater investigation. First, the current framework adopts random injection of quantum gates (excluding Hadamard gates) to compromise measurement outputs. While randomness can readily conceal the behavior of the circuit, under other circumstances it does not always completely optimize corruption. A more targeted approach, where gates are selected based on their type and the structural context of the circuit rather than purely at random, may improve corruption metrics such as Total Variation Distance (TVD) and Degree of Functional Corruption (DFC). Research into heuristic or algorithmic selection of gates can lead to safer obfuscation. Second, the current framework is designed for circuits with measurements performed in a single basis per execution, even if that basis is transformed (e.g., measuring in the X-basis using Hadamard gates). However, the technique faces limitations when applied to quantum algorithms that require simultaneous or repeated measurements in multiple bases—such as in variational quantum algorithms that involve Hamiltonians composed of terms across different Pauli bases (e.g., $XX + ZY + YI$). In such scenarios, a single obfuscation layer may not support all measurement contexts without affecting the correctness or requiring significantly more complex post-processing.

We plan to overcome these disadvantages in future work by developing procedures for the intelligent selection of encryptor gates to maximize corruption at the expense of not compromising classical post-processing simplicity. Based on the connectivity and gate composition of the original circuit, we will devise procedures to strategically insert gates to maximize obfuscation performance. Additionally, we intend to extend the current correction mechanism to support circuits that require measurements in multiple bases within a single execution. This will involve adapting the standard correction procedure to support measurements in other bases, perhaps through the inclusion of basis transformations or generalized correction circuits. These advancements will broaden the applicability of our method to more quantum algorithms and make it more valuable for quantum intellectual property protection in untrusted compilation environments. Besides, we plan to experiment with the scheme in practical NISQ hardware conditions, incorporating noise models to assess its performance and robustness in real-world settings.

8 Conclusion

In this work, we introduced a novel quantum circuit obfuscation technique designed to defend commercial quantum algorithms within the NISQ era, where third-party compiler trust can be an intellectual property vulnerability. By introducing quantum gates, with the exception of Hadamard gates, into the circuit before compilation in a carefully constructed form, our approach is able to corrupt measurement outcomes, which are subsequently corrected by a simple classical post-processing operation. Conversely, with other methods depending on complicated quantum reversals, barriers, or physical-to-virtual qubit mapping concerns, our method reduces deobfuscation complexity and enhances practicality and security. Our technique was evaluated across multiple quantum algorithms and consistently demonstrated high obfuscation strength, both in terms of statistical divergence and functional corruption. These results confirm that the method is broadly applicable across a range of circuit structures, offering a lightweight yet effective layer of security. Overall, this work takes a significant step toward securing quantum computations in untrusted environments, offering a practical and scalable obfuscation solution for real-world quantum applications.

References

[1] IBM. 2025. *Quantum Computing*. https://www.ibm.com/think/topics/quantum-computing Accessed: Apr. 9, 2025.

[2] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (1997), 1484–1509. doi:10.1137/S0097539795293172

[3] Neil Gershenfeld and Isaac L. Chuang. 1998. Quantum Computing with Molecules. *Scientific American* 278, 6 (1998), 66–71. https://www.jstor.org/stable/26057857

[4] Akshay Ajagekar and Fengqi You. 2019. Quantum computing for energy systems optimization: Challenges and opportunities. *Energy* 179 (2019), 76–89. doi:10.1016/j.energy.2019.04.186

[5] Google Quantum AI and Collaborators. 2025. Quantum error correction below the surface code threshold. *Nature* 638 (2025), 920–926. doi:10.1038/s41586-024-08449-y

[6] Microsoft Azure Quantum, Morteza Aghaee, Alejandro Alcaraz Ramirez, et al. 2025. Interferometric single-shot parity measurement in InAs–Al hybrid devices. *Nature* 638 (2025), 651–655. doi:10.1038/s41586-024-08445-2

[7] Lee Jun Quan, Tan Jia Ye, Goh Geok Ling, and Vivek Balachandran. 2024. QIris: Quantum Implementation of Rainbow Table Attacks. In *International Conference on Information Systems Security*. Springer, 213–222. doi:10.1007/978-3-031-80020-7_12

[8] John Preskill. 2018. Quantum Computing in the NISQ era and beyond. *Quantum* 2 (Aug. 2018), 79. doi:10.22331/q-2018-08-06-79

[9] Abdullah Ash Saki, Mahabubul Alam, Koustubh Phalak, Aakarshitha Suresh, Rasit Onur Topaloglu, and Swaroop Ghosh. 2021. A Survey and Tutorial on Security and Resilience of Quantum Computing. In *2021 IEEE European Test Symposium (ETS)*. 1–10. doi:10.1109/ETS50041.2021.9465397

[10] IBM Quantum. 2024. *Qiskit: An open-source framework for quantum computing*. https://qiskit.org Accessed: Dec. 2024.

[11] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan. 2020. t|ket⟩: A retargetable compiler for NISQ devices. *Quantum Science and Technology* 6, 1 (Nov. 2020), 014003. doi:10.1088/2058-9565/ab8e92

[12] Rigetti Computing. 2025. *Quilc: The Quil compiler*. https://rigetti.com Accessed: April. 2025.

[13] Vivek Balachandran. 2021. Quantum Obfuscation: Quantum Predicates with Entangled qubits. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy* (Virtual Event, USA) *(CODASPY '21)*. Association for Computing Machinery, New York, NY, USA, 293–295. doi:10.1145/3422337.3450317

[14] Subrata Das and Swaroop Ghosh. 2023. Randomized Reversible Gate-Based Obfuscation for Secured Compilation of Quantum Circuit. arXiv:2305.01133 [quant-ph] https://arxiv.org/abs/2305.01133

[15] Aakarshitha Suresh, Abdullah Ash Saki, Mahabubul Alam, Rasit Onur Topaloglu, and Swaroop Ghosh. 2022. Short Paper: A Quantum Circuit Obfuscation Methodology for Security and Privacy. In *Proceedings of the 10th International Workshop on Hardware Architecture Support for Security and Privacy (HASP '21)*. Virtual, CT, USA, Article 6. doi:10.1145/3505253.3505260

[16] Rasit Onur Topaloglu. 2023. Quantum Logic Locking for Security. *J* 6, 3 (2023), 411–420. doi:10.3390/j6030027

[17] Yuntao Liu, Jayden John, and Qian Wang. 2025. *E-LoQ: Enhanced Locking for Quantum Circuit IP Protection*. arXiv:2412.17101 [quant-ph] https://arxiv.org/abs/2412.17101

[18] Phillip Kaye, Raymond Laflamme, and Michele Mosca. 2006. *An Introduction to Quantum Computing*. Oxford University Press. doi:10.1093/oso/9780198570004.001.0001

[19] Bochen Tan and Jason Cong. 2020. Optimal layout synthesis for quantum computing. In *Proceedings of the 39th International Conference on Computer-Aided Design* (Virtual Event, USA) *(ICCAD '20)*. Association for Computing Machinery, New York, NY, USA, Article 137, 9 pages. doi:10.1145/3400302.3415620

[20] Jarrod A. Roy, Farinaz Koushanfar, and Igor L. Markov. 2008. EPIC: Ending Piracy of Integrated Circuits. In *2008 Design, Automation and Test in Europe*. 1069–1074. doi:10.1109/DATE.2008.4484823

[21] Abdullah Ash Saki, Mahabubul Alam, and Swaroop Ghosh. 2021. Impact of noise on the resilience and the security of quantum computing. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*. 186–191. doi:10.1109/ISQED51717.2021.9424258

[22] Anees Rehman, Vincent Langford, Jayden John, and Yuntao Liu. 2025. OPAQUE: Obfuscating Phase in Quantum Circuit Compilation for Efficient IP Protection. In *2025 26th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 1–6. doi:10.1109/isqed65160.2025.11014313

[23] Sergio Verdú. 2014. Total variation distance and the distribution of relative information. In *2014 Information Theory and Applications Workshop (ITA)*. San Diego, CA, USA, 1–6. doi:10.1109/ITA.2014.6804281

[24] Alfred V. Aho, John E. Hopcroft, and Jeffrey Ullman. 1983. *Data Structures and Algorithms* (1st ed.). Addison-Wesley Longman Publishing Co., Inc., USA.

[25] Jaeho Choi and Joongheon Kim. 2019. A Tutorial on Quantum Approximate Optimization Algorithm (QAOA): Fundamentals and Applications. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. 1264–1266. doi:10.1109/ICTC46691.2019.8939749

[26] IBM Quantum. 2025. *Quantum Approximate Optimization Algorithm*. https://learning.quantum.ibm.com/tutorial/quantum-approximate-optimization-algorithm Accessed: Apr. 16, 2025.

[27] Amal Raj and Vivek Balachandran. 2025. *Quantum-Classical Obfuscation*. https://github.com/vivekianity/quantum-classical-obfuscation

[28] Moein Naseri, Tulja Varun Kondra, Suchetana Goswami, Marco Fellous-Asiani, and Alexander Streltsov. 2022. Entanglement and coherence in the Bernstein-Vazirani algorithm. *Physical Review A* 106, 6 (2022), 062429. doi:10.1103/PhysRevA.106.062429

[29] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC '96)*. Association for Computing Machinery, New York, NY, USA, 212–219. doi:10.1145/237814.237866

[30] Anika Zaman, Hector Jose Morrell, and Hiu Yung Wong. 2023. A Step-by-Step HHL Algorithm Walkthrough to Enhance Understanding of Critical Quantum Computing Concepts. *IEEE Access* 11 (2023), 77117–77131. doi:10.1109/ACCESS.2023.3297658