Leveraging Quantum Circuit Cutting for Obfuscation and Intellectual Property Protection

George Typaldos
Yale University
New Haven, CT, USA
george.typaldos@yale.edu

Wei Tang
Princeton University
Princeton, NJ, USA
weit@princeton.edu

Jakub Szefer
Yale University
New Haven, CT, USA
jakub.szefer@yale.edu

***Abstract*—Circuit cutting is a novel technique in the field of quantum computing that offers a promising approach for breaking up large quantum circuits into smaller ones that can be executed on smaller quantum computers. Circuit cutting offers a scalable and hybrid computing approach, by combining classical and quantum computers to run large quantum circuits. At the same time, quantum circuits represent the fundamental building blocks of quantum algorithms, and obfuscation techniques are crucial for protecting the sensitive information and intellectual property encoded in these circuits. This paper constitutes the first security analysis of circuit cutting techniques as a means to protect quantum circuits, and not just for cutting bigger quantum circuits to fit on smaller devices. As cloud providers are in control of quantum computers and give access to users, utilizing cloud-based quantum computers for executing quantum circuits involves considerable security risks. This work demonstrates that quantum circuits can be effectively obfuscated by using quantum circuit cutting techniques. If users are worried about an adversary who attempts to reverse-engineer the original quantum algorithm, circuit cutting can be used as a security measure. To further provide configurable obfuscation level, i.e. number of possible circuits that the cloud provider would have to guess, this work proposes a novel dummy-subcircuits technique. This technique introduces dummy subcircuits with additional qubit cut points to create more confusion for the cloud provider and further help to obfuscate the user's original circuits with limited cost.**

***Index Terms*—quantum computing, security, code obfuscation, circuit cutting**

I. Introduction

Cloud-based quantum computers represent a significant leap in computing technology, merging the power of quantum computation with the accessibility and scalability of cloud computing. Many cloud-based quantum computers are available today, from IBM Quantum [1], Amazon Braket [2], and Azure Quantum [3]. In a cloud-based quantum computing model, these providers give access to quantum processing units (QPUs) to remote users who pay based on the amount of time they need to execute their circuits. The QPUs are time-shared among users. This allows researchers, scientists, and businesses to harness the computational potential of quantum systems without the need for extensive on-site infrastructure and cost. Users can access quantum computing resources, run algorithms, and perform complex calculations via the internet, making quantum computing more widely available and practical. Once a user is done, the QPU can be utilized by another user.

The cloud-based approach democratizes access to quantum computing, enabling a broader range of industries and applications to benefit from quantum computational advantages. As this technology continues to evolve, it holds the promise of revolutionizing fields such as cryptography, optimization, drug discovery, and materials science, among others.

While cloud-based quantum computers offer many benefits, they suffer from possible security threats. In particular, untrusted cloud provider is a major security concern; the whole cloud provider could be untrusted, or there could be untrusted insiders working within the cloud provider, which would represent insider attackers [4]. Since the cloud providers, or the insiders, have full access to the quantum circuits executing on the quantum computers, they may be incentivized to copy the user's code, copy results from the user's computation, spy on types of computations performed by the user, etc. At the same time, users have no choice but to use cloud-based quantum computers if they want to take advantage of this technology; as most users do not have resources or the know-how to build their own quantum computers. When using cloud-based quantum computers, users today have limited means to protect themselves. Blind quantum computation, e.g. [5], remains theoretical and cannot be deployed in practice yet.

In this work, we explore a new approach to protecting user's circuits from an untrusted cloud provider by leveraging recent techniques introduced for cutting quantum circuits into smaller subcircuits [6]. Circuit cutting was invented due to the problem that there is so far a lack of large quantum computers, and due to short decoherence times of existing quantum computers. We realize, however, for the first time that quantum circuit cutting can be used as an obfuscation mechanism. The obfuscation is based on the difficult problem that the cloud provider has in reconstructing the original circuit from the subcircuits generated by quantum circuit cutting. Because the cloud provider does not have information on how the subcircuits are composed into a bigger circuit, they cannot effectively reconstruct the original circuit. The cloud provider knows the subcircuits, their inputs, and measurement results. However, we show that this is not sufficient to easily reconstruct the original circuits.

This work was supported in part by National Science Foundation grant 2312754. This work was supported in part by Department of Energy contract DE-SC0012704. This work was done prior to Wei Tang joining Amazon.

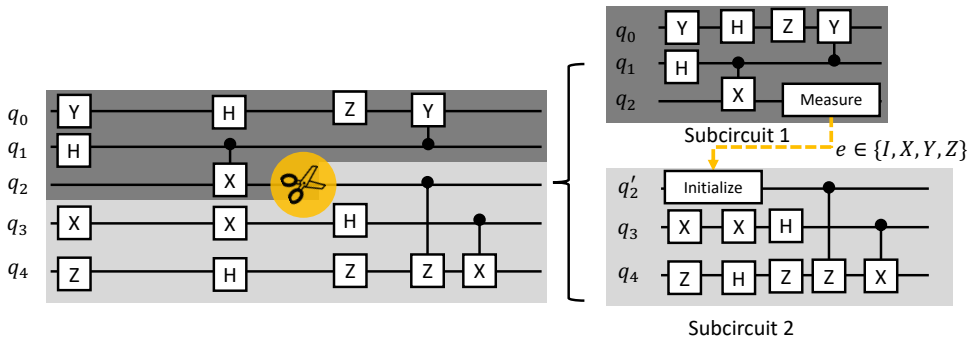


Fig. 1: Example of cutting a 5-qubit quantum circuit using a single cut, resulting in two distinct, smaller subcircuits. (Left) A scissor marks the point where qubits were cut. The darker shaded area represents subcircuit 1, while the lighter shade denotes subcircuit 2. (Right) A dashed arrow traces the route of the cut qubit wire. To perform circuit cutting, the two subcircuits are executed independently on QPUs, and all qubits are measured at the end of each subcircuit. Note that thanks to circuit cutting, 3-qubit QPUs are needed vs. 5-qubit QPUs if no circuit cutting was used. For the cut qubit, at the cut point, the qubit is initialized in the $\{I, X, Y, Z\}$ bases, and subcircuit 2 is executed with each initialization. Resulting data is used to reconstruct the final circuit output by running the circuit cutting algorithm on a classical CPU or GPU.

As we demonstrate, there's a large number of ways that the subcircuits could be composed, which generates significant confusion, and thus significant level of obfuscation. Further, on average, each improper reconstruction results in output probabilities different from what the correct circuit would generate, thus there is also a significant level of confusion for the attacker. Therefore, we show that circuit cutting is an efficient and scalable method for the obfuscation of circuits and their outputs, and it can be already used today.

The insights of this work are the realization that without the information on how to compose the subcircuits into a bigger circuit, the cloud provider has no effective means to find the original circuit, except for a brute-force search. In particular, quantum circuit cutting software can be executed on the user's computer, to generate subcircuits and the metadata about how to compose the subcircuits. Then, the metadata can be kept locally safe by the user, while he or she submits the subcircuits to the cloud-based quantum computer provider. The provider executes each subcircuit, returning the (classical) measurement results to the user. The user can then reconstruct the correct output from the measurement results of all the subcircuits and the metadata. Circuit cutting is explained in more detail in Section II, but at a high level, the cloud provider can guess which qubits in which subcircuits are initial qubits, cut points, or measurements. However, as we demonstrate, even with that information, there is an extremely high number of possible connections, through the cut points. Thus, given only the subcircuits, there is a significantly large number of circuits, which we call candidate circuits, that could feasibly be made from these subcircuits.

To allow for arbitrarily increasing the number of the candidate circuits, i.e. the obfuscation level, we propose an additional technique; a novel *dummy-subcircuits* approach that introduces dummy subcircuits with randomly selected qubit cut points. By submitting the dummy subcircuits along with the subcircuits generated by quantum circuit cutting software, the obfuscation level can be arbitrarily increased. For example, for user circuits of 4 qubits with significantly low obfuscation level by default, adding 60 dummy subcircuits could result in 2^{278} candidate circuits that the cloud provider would have to consider when trying to guess the users' original circuit. This number of candidate circuits, i.e. the obfuscation level, can be arbitrarily increased by adding more dummy subcircuits.

A. Contributions

The contributions of this work are as follows:

- Analysis of quantum circuit cutting as security mechanisms for obfuscation and protection of user circuits from untrusted cloud providers.
- Introduction of *dummy-subcircuits* technique to further enhance obfuscation level, through the insertion of dummy subcircuits with randomized qubit cut points.
- Evaluation of the security level of circuit cutting when used for obfuscation. Among others, we consider the number of possible candidate circuits as well as the difference between the output probabilities of the original circuit vs. candidate circuits.
- Evaluation of the execution overhead for the attacker, associated with generating the candidate circuits and also for the reconstruction process needed to compute the outputs from these circuits.

II. Circuit Cutting Background

Prior quantum computing research mainly focused on making better use of a single quantum computer. The emergence of quantum circuit cutting theory [6] makes it possible to cut a large quantum program into smaller sub-programs, execute the sub-programs on smaller quantum processing units (QPUs), and later reconstruct the sub-results with classical computing.

Figure 1 illustrates the process of cutting a basic quantum circuit, as depicted in the left panel, which represents a 5-qubit quantum circuit. Each qubit wire runs horizontally. Single qubit quantum gates are represented by boxes on individual

qubit wires, while boxes spanning two wires indicate 2-qubit quantum gates. Typically, executing this circuit demands a QPU with at least 5 high-quality qubits to perform all quantum gates before significant error accumulation. However, circuit cutting segments this circuit into smaller, manageable subcircuits. A single cut, marked by a scissor symbol, bisects the quantum circuit into two subcircuits. These subcircuits can be independently executed in parallel by multiple 3-qubit QPUs, without needing quantum interconnections. Thus, circuit cutting involves making perpendicular cuts across qubit wires, allowing a larger quantum circuit to be divided into several smaller subcircuits through multiple cuts.

The underlying physics concept of circuit cutting involves expressing unknown quantum states at cut points as a sum of their Pauli bases components. In this process, QPUs execute four versions of subcircuit 1, each measuring the upstream cut qubit q_2 in one of the $\{I, X, Y, Z\}$ Pauli bases. Notably, measurements in the I and Z bases are performed using identical single-qubit rotations, reducing the necessity to only 3 subcircuits for each upstream cut qubit. In parallel, QPUs conduct four variations of subcircuit 2, initializing the downstream cut qubit q_2' in one of the $|0\rangle, |1\rangle, |+\rangle, |i\rangle$ states. These states then form the basis for constructing the $\{I, X, Y, Z\}$ Pauli bases as linear combinations. Measuring (or initializing) a qubit in varying bases involves adding different single qubit rotations to the end (or beginning) of the subcircuits, which are straightforward operations causing minimal additional complexity in the subcircuits. This method results in four distinct outcomes for subcircuit 1, denoted as p_1^e for each cut edge $e \in \{I, X, Y, Z\}$, and similarly produces four outcomes for p_2^e.

Circuit cutting ultimately achieves the classical reconstruction of quantum state outputs by combining the results of the subcircuits. Prior circuit cutting work in CutQC [7] establishes that the output P of the original, intact circuit is equal to the sum $\sum_e p_1^e \otimes p_2^e$. Circuit cutting extends to multiple subcircuits beyond two by sequentially combining the outputs of all subcircuits across the Pauli decompositions of every cut quantum edge:

$$P = \sum_{e=\{I\}^{|E|}, e \in E}^{Z^{|E|}} \otimes_{i=1}^{n} p_i^{e \in E_i} \quad (1)$$

The state-of-the-art circuit cutting technique [8] employs scalable tensor network strategies for computing the reconstruction problem. The quantum interactions among the subcircuits are therefore substituted by classical post-processing, which is analogous to the communication cost paid in classical parallel computing.

A. Circuit Cutting Workflow

In practical applications, users initially employ local algorithms to identify efficient cuts for a large quantum circuit. These algorithms include options like a Mixed Integer Programming (MIP) solver [7] or a heuristic solver [8]. Once suitable cuts are determined, users divide the large quantum circuit into smaller subcircuits. These subcircuits are then sent to a cloud-based QPU provider for execution. Cloud backends then compile the subcircuits to run on their QPUs. After the subcircuits have been processed in the cloud, users retrieve the results from these cloud backends. The final step involves users utilizing local GPUs to conduct post-processing of tensor networks to reconstruct the results of the initial quantum program.

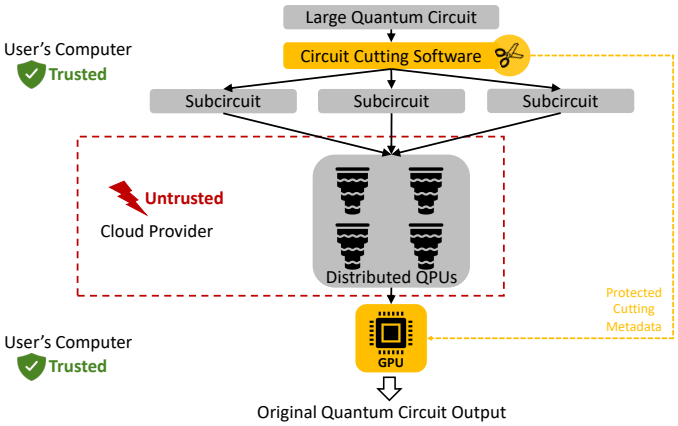


Fig. 2: Circuit cutting workflow: large circuits are broken into smaller subcircuits, then smaller subcircuits are executed on the remote quantum computer, and finally the results of the subcircuits are combined to reconstruct the output of the original, large circuits. The trusted and untrusted parts are highlighted in the figure; they are discussed in detail in the Threat Model, in Section III.

Figure 2 depicts the entire process, from initial cut finding to final result reconstruction. The circuit cutting steps emphasize the collaboration between local computational resources and cloud services in executing complex quantum computations.

B. Circuit Cutting, Subcircuits, and Metadata

Reconstructing the original quantum circuit from the subcircuit outcomes hinges on accurately tracing the connections formed by the cut edges between subcircuits. The correct computation of equation 1 requires precise alignment of the cut edges among subcircuits. Matching wrong pairs of cut qubits produces nonsensical reconstruction results.

Figure 3 (a) shows an example of 3 subcircuits with 2 cuts. The user knows exactly how the cuts connect the subcircuits since they have full knowledge of the cuts being made. However, cloud providers only receive the subcircuits after the cuts are made, and only know which qubits are cut qubits in each subcircuit, Figure 3 (b). If cloud providers attempt to reconstruct the initial quantum circuit, it is possible to come up with wrong but valid cut qubits correspondence that lead to wrong reconstruction outcomes, Figure 3 (c). If *dummy-subcircuit* is introduced, Figure 3 (d), then the number of feasible, but incorrect reconstructions increases, Figure 3 (e).

III. Threat Model

This work focuses on the threat of an untrusted cloud provider (equivalently, untrusted insider working within the cloud provider) who may be trying to reverse engineer what

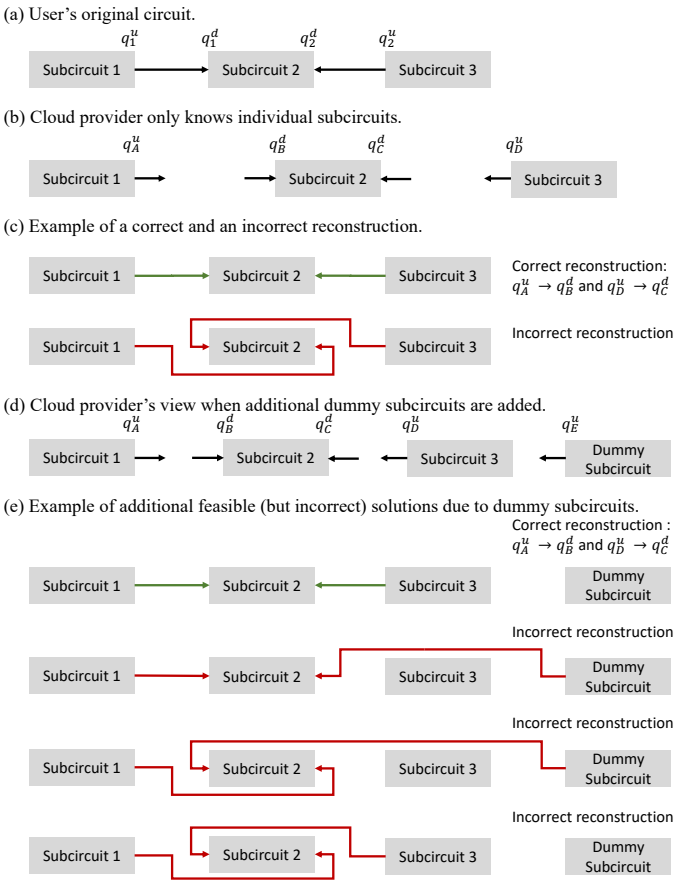


Fig. 3: Without knowing the exact cut edge connectivity among the subcircuits, it is possible to come up with valid but wrong subcircuit reconstructions. (a) The user view of the subcircuits. Users know both the cut qubits and their pairing. q_i^u indicates upstream cut qubits, and q_i^d indicates the downstream cut qubits. (b) Cloud provider only knows which qubits are upstream and downstream cuts, but not their pairing. (c) Example of a correct and incorrect solution; the incorrect solution is still a feasible (but wrong) one; a reconstruction is valid when all the upstream and downstream qubits are matched, but still it does not represent the user's original circuit. (d) Cloud provider's view when *dummy-subcircuit* is introduced by the user. (e) Example of multiple feasible (but incorrect) solutions when *dummy-subcircuit* is introduced.

circuit the user is executing on the cloud provider's quantum computers. We assume that the user compiles their program locally, and that classical reconstruction operations are also performed locally on a trusted computer. We assume the user securely retains circuit cutting metadata that specifies how the subcircuits should be composed into their original circuit. The subcircuits themselves are submitted to the cloud provider who has full access to them, but does not have the metadata. Finally, the results are collected by the user, and the user locally computes the result of their computation from the quantum computer results and the metadata that the user has retained. The trusted and untrusted parties are also shown in Figure 2.

We assume the cloud provider knows the cut points (the upstream and downstream qubits) among the subcircuits. We further assume that the cloud provider knows which qubits and subcircuits involve initial qubits, because they are set into $|0\rangle$ state, and which are the final qubits, because they are measured with regular measurement operations only in one basis (meanwhile for qubits at the cut points, they are measured in multiple bases).

We assume that the cloud provider is not able to use additional knowledge, e.g., what type of circuit is being executed, to help his or her search. We leave as future work analysis of whether subcircuits' structure can reveal information about the nature of the original circuit.

A. Additional Obfuscation with Dummy-Subcircuits

As a further additional obfuscation measure, we propose that users submit dummy subcircuits alongside the correct subcircuits. We assume the structure of the subcircuits, gate distributions, etc. are such that the cloud provider is not able to determine whether a subcircuit is a real subcircuit or a dummy one. Therefore, we can create additional dummy circuits, submitted with the original subcircuits to obfuscate the reconstruction process. Evaluation of profiling of subcircuits and the ability of the cloud provider to distinguish real from dummy subcircuits is left as future work.

IV. Leveraging Quantum Circuit Cutting for Obfuscation of User's Circuits

In this section, we discuss the obfuscation capabilities of quantum circuit cutting. Current quantum devices and technology require the provider to have full access to the user's circuits to properly transfer them to the quantum device for execution. This exposes the submitted quantum circuits and algorithms to security risks. With circuit cutting, instead of submitting original circuits, users submit the subcircuits generated by the circuit cutting software. Now, an adversary with access to these subcircuits could try to recreate the original circuit. This section analyzes the feasibility of such a recreation process and describes a classical algorithm that calculates the total number of possible recreated circuits, which we also call candidate circuits, given a set of subcircuits. Later, this section also examines how the recreation complexity is altered when introducing the *dummy-subcircuits* techniques.

A. Computing Number of Qubits in User's Original Circuit

The first step to recreate quantum circuits, with a set of subcircuits, is to calculate their size. As the subcircuits include measurements in $\{I, X, Y, Z\}$ Pauli bases that give out information about whether a qubit has been cut, an adversary could utilize this to obtain the size of the original circuit. Assuming we have n distinct subcircuits with n_i qubits each and n_c cuts in total (the number of cuts is equal to the number of upstream or downstream cuts), then combining these subcircuits results in a quantum circuit with size n_r, where n_r is calculated by subtracting the number of cuts from the sum of n_i qubits, as shown in Equation 2 below.

$$n_r = \sum_{i=1}^{n} n_i - n_c \tag{2}$$

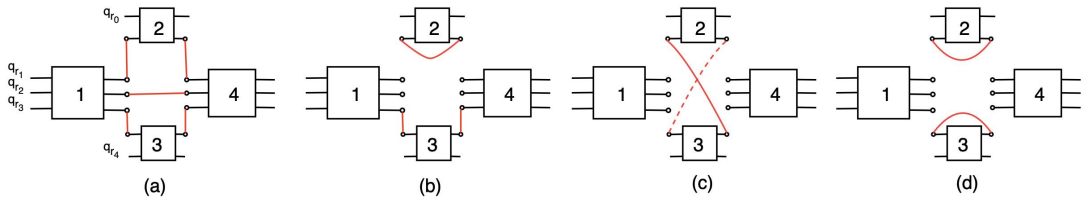


Fig. 4: Examples of subcircuit connectivity and valid and invalid recreations. (a) All subcircuits are connected via upstream and downstream cuts and form a valid recreated circuit with 5 qubits (from Equation 2 we have $n_r = 10 - 5$). (b) Circuits 1 and 4 can be connected through their remaining two upstream/downstream qubits (2! combinations). However, circuit 2 forms a circular connection, as its upstream cut qubit is connected to its downstream cut qubit. This recreation is therefore invalid. (c) Circuits 1 and 4 can be connected through their upstream/downstream qubits (3! combinations), but circuits 2 and 3 are interconnected. Thus, this case is also invalid. (d) Similarly, circuits 1 and 4 can be connected via 3! combinations, but circuits 2 and 3 are intra-connected as their upstream qubit are connected to their downstream qubit, resulting in an invalid recreation.

Using the n_r size, the recreation problem turns into a combinatorial one, where we need to calculate the total number of valid recreations. The connections between the subcircuits need to be made via upstream and downstream cut qubits. While one could easily conclude that with n_c cuts we would have $n_c!(= 1 \times 2 \times \cdots \times n_c)$ total recreations, we also have to take into account several cyclic cases, as explained in the following section.

B. Computing Possible Combinations of Candidate Circuits

The process of connecting subcircuits to produce a recreated quantum circuit involves combining the upstream cut qubits with the downstream cut qubits; qubits with no cuts are used directly in the recreated circuit. Figure 4a showcases an example where four subcircuits with five cuts (five upstream and five downstream qubits) are connected; subcircuits 1 and 4 have three upstream and three downstream cuts respectively, whereas subcircuits 2 and 3 have one of each. If we just start connecting upstream to downstream cuts, there would eventually be some cases where the upstream qubit of circuit 2 (or 3) would be connected to its downstream qubit, like in Figures 4b and 4d. Similarly, the upstream qubit of circuit 2 could be connected to the downstream qubit of circuit 3, while the upstream qubit of circuit 3 could be connected to the downstream qubit of 2, as shown in Figure 4c.

Having identified the possibility of cyclic connections, we need a way to calculate and subtract them from the $n_c!$ term. This will give us the total number of possible recreated circuits. To formulate the problem using the examples in Figure 4 we identify the number of cuts, n_c, to 5 (total upstream or downstream qubits). Additionally, we need the number of qubits with solely upstream cuts, $n_u = 3$, i.e. the qubits of circuit 1, the number of qubits with solely downstream cuts, $n_d = 3$, i.e. the qubits of circuit 4, and the number of qubits with both upstream and downstream cuts, $n_{ud} = 2$, i.e. one qubit from circuit 2 and one from circuit 3.

Beginning with Figure 4b, we see that circuit 3 can be connected to either of the 3 upstream, or downstream qubits of circuits 1 and 4 respectively, thus a total of $\binom{n_u}{1} \cdot \binom{n_d}{1}$ combinations, where $\binom{n}{k} = \frac{n!}{k!(n-k)!}$. The remaining qubits of circuits 1 and 4 can be connected with factorial combinations, thus $(n_u - 1)!$. Finally, instead of circuit 3, we could use circuit 2, thus we have to multiply the previous terms with $\binom{n_{ud}}{1}$. Considering all the above and the fact that $n_u = n_d$ for all generated subcircuits, we obtain a total number of $\binom{n_u}{1}^2 \cdot (n_u - 1)! \cdot \binom{n_{ud}}{1} = \frac{n_u!}{1!} \cdot \binom{n_u}{1} \cdot \binom{n_{ud}}{1}$ invalid recreations for case 4b.

In cases 4c and 4d, we see that circuits 1 and 4 can be connected through $n_u! = \frac{n_u!}{0!} \cdot \binom{n_u}{0}$ combinations. Circuits 2 and 3 can either be interconnected, meaning the upstream qubit of one is connected to the other's downstream qubit like in 4c, or intraconnected, meaning their upstream qubit is connected to their downstream qubit like in 4d. For these cases, we have a total of $\binom{n_{ud}}{1} \cdot 1 = \binom{n_{ud}}{1} \cdot \binom{n_{ud}}{2}$ combinations. Therefore, for the cases 4c and 4d, we obtain a total of $\frac{n_u!}{0!} \cdot \binom{n_u}{0} \cdot \binom{n_{ud}}{1} \cdot \binom{n_{ud}}{2}$ invalid recreations.

Combining these results and subtracting them from the factorial of total cuts, we produce the following approximate formula that calculates the number of feasible recreated circuits, e.g. the number of valid combinations between given subcircuits using the number n_c of cuts, the number n_u of qubits with solely upstream cuts and the number n_{ud} of qubits with both upstream and downstream cuts:

$$n_{\text{valid}} = n_c! - \sum_{x=0}^{n_{ud}-1} \frac{n_u!}{x!} \cdot \binom{n_u}{x} \cdot \left[\binom{n_{ud}}{1} \cdot \binom{n_{ud}}{2} \cdots \binom{n_{ud}}{n_{ud} - x} \right] \tag{3}$$

This formula is specific for the structure of our benchmark circuits, i.e. subcircuits where the number of qubits with solely upstream cuts equals to the number of qubits with solely downstream cuts. This formula overestimates the number possible reconstructions. After extended analysis, we conclude that a universal formula for any circuit type and structure, may be impossible to obtain, due to the various possible cut combinations and subcircuits variations. However, we leave as future work the design of an efficient algorithm that can calculate the feasible recreated circuits with minimal execution cost.

C. Recreation Algorithm

Each possible, valid inter-connection of the subcircuits represents one candidate circuit. Each candidate circuit is a circuit that the malicious cloud provider would guess could be the original circuit. One important note is that the qubit order in the recreated circuits does not make any difference, as an attacker would be interested in the probability distribution of the quantum algorithm, rather than the absolute state-vector values. Even if the original quantum circuit produces results with a qubit ordering of $q_n q_{n-1} \dots q_1 q_0$ and a recreated candidate circuit with an order of $q_{n-1} \dots q_1 q_0 q_n$ (or any other random ordering), sorting the probabilities of each outcome would produce the same distribution, from which an attacker could potentially identify the nature of the quantum algorithm used.

To recreate the candidate circuits from a set of subcircuits, we designed an algorithm using Python and IBM's Qiskit SDK. The first step is to produce a mapping for every combination, from the subcircuit qubits and their connections via upstream and downstream qubits, to the n_r qubits of the candidate circuits Since we do not need to worry about the qubit ordering, after obtaining the possible combinations using Equation 3 and as we already know the size of the candidate circuits from Equation 2, we just assign the subcircuit combination to a random qubit. This mapping allows us to append the gates from the subcircuits in the right order to construct the candidate circuit. For example, qubit q_{r_3} from the candidate circuit is mapped to connection $1 \rightarrow 3 \rightarrow 4$ from Figure 4a. The other $n_r - 1$ qubits will either be mapped to the remaining connections, e.g. q_{r_2} to $1 \rightarrow 4$ and q_{r_1} to $1 \rightarrow 2 \rightarrow 4$, or to subcircuit qubits with no cuts, e.g. q_{r_0} to qubit q_{2_0} and q_{r_4} to q_{3_1}.

Having figured out a mapping for each possible combination, the next step is to append the appropriate gates from the subcircuits to a newly created quantum circuit. These gates and their qubit dependencies (for multi-qubit gates), are obtained by parsing the subcircuits and retrieving information about the instructions (gate, qargs) per node (sc_i, q_{sc_i}), the number of gates for each node (its steps), the connectivity between multi-qubit gates, and finally, the total instructions (number of gates) for all subcircuits. For each mapping, the algorithm first creates an empty quantum circuit with size n_r. Then it initiates a breadth-first appending approach, where each qubit q_r is advanced by one step (one additional gate). In the case of a single-qubit gate, the algorithm just appends the gate to the appropriate q_r, and advances the q_r step, while subtracting one instruction from the total remaining ones. If the gate is a multi-qubit gate, then all q_{r_i} qubits for this gate must be in the same step, for it to be appended and the appropriate counters to be modified. This process continues until no more instructions remain. Whereas the mappings handled by the recreation algorithm would be valid, according to Equation 3, there may be cases where a mapping creates a topological error and no more gates can be appended. This happens when a multi-qubit gate has to be appended between qubit q_r and

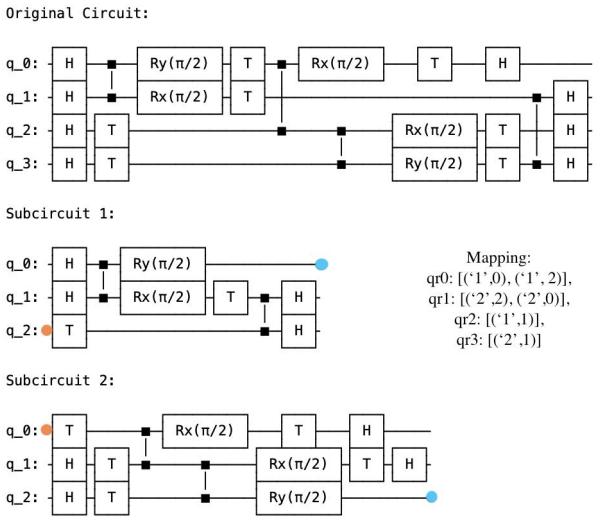


Fig. 5: Topological error in a supremacy-type circuit. Upstream cut qubits are highlighted on the right side of subcircuits 1,2 and downstream qubits on the left. Qubit q_{r_3} is mapped to qubit q_1 of subcircuit 2, i.e., node ('2',1). Qubit q_{r_1} is mapped to node ('2',2) which is connected to node ('2',0). At step 2, q_{r_1} needs to append a CZ gate with node ('2',1), i.e. qubit q_{r_3}. However, at the same step, q_{r_3} has to previously append a CZ gate with node ('2',0), which comes after node ('2', 2). Thus, a topological error arises as neither q_{r_1} nor q_{r_3} can append their gates and the specific mapping cannot result in a valid recreated circuit.

q_{sc_i}, but the q_{sc_i} cannot reach the required step, due to the specific connectivity. As seen in Figure 5, a possible mapping is q_{r_1} : ('2',2) $\rightarrow$ ('2',0) and q_{r_3} : ('2',1), where node ('2',1) corresponds to qubit q_1 from subcircuit 2. At step 2, q_{r_1} needs to append a CZ gate with node ('2',1), i.e. qubit q_{r_3}. However, at the same step, q_{r_3} has to previously append a CZ gate with node ('2',0), which comes after node ('2', 2), therefore, creating a topological error where no qubit can append their required gate. To prevent this and discard such mappings, the algorithm checks if after an iteration round (all q_r qubits have been examined), no qubit has advanced its step. In that case, the process for this mapping stops and the recreated circuit is discarded. An overview of the circuit construction process is shown in Algorithm 1.

D. Obfuscation with Dummy-Subcircuits Technique

As we will see in the evaluation section, there may be cases where the recreated circuits are minimal and not adequate to provide an efficient obfuscation level, when only default circuit cutting is used. This can happen either due to the small number of cut qubits or to the topological structure of the circuit, which will cause many mappings to be discarded, as previously mentioned. To further increase the desirable obfuscation level, we introduce dummy subcircuits, submitted with the original ones, that induce further recreation complexity, as more mappings between subcircuits are produced. Two factors must be considered when adding dummy subcircuits; how will these circuits look like and how many will be added?

Algorithm 1: Recreation Algorithm

Input: List of quantum subcircuits.
Output: List of candidate quantum circuits

$n_c \leftarrow$ number of cuts
$n_r \leftarrow$ number of candidate circuits
$n_u \leftarrow$ qubits with solely upstream cuts
$n_d \leftarrow$ qubits with solely downstream cuts
$n_{ud} \leftarrow$ qubits with up- and downstream cuts
$combinations \leftarrow$ combine subcircuits
$mappings \leftarrow$ mapping for each combination
// Qubits q_{r_i} from the candidate circuit are mapped to qubits q_i from the given subcircuits
$instructions \leftarrow \{(sc_i, q_{sc_i}) : (\text{gate}, qargs)\}$
// Total number of gates, steps (number of gates) per node, and connectivity between multi-qubit gates of each node
for *mapping in mappings* **do**
  $recreated \leftarrow$ QuantumCircuit(n_r)
  $steps \leftarrow$ [0 for qb in range(n_r)]
  while $total\_inst > 0$ **do**
    advanced_steps $\leftarrow 0$ **for** q_r *in* n_r **do**
      $step \leftarrow steps[q_r]$
      $gate, qargs \leftarrow node\_inst[q_r, step]$
      if $qargs > 1$ **then**
        $conn\_list \leftarrow connected\_qubits$
        if $steps[q_c]$ *is* $step$, $\forall\ q_c$ *in* $conn\_list$ **then**
          $recreated.append(gate, qargs)$
          $steps[q_c] \leftarrow steps[q_c] + 1, \forall\ q_c$
          $total\_inst \leftarrow total\_inst - 1$
          advanced_steps $\leftarrow$ advanced_steps $+ 1$
        end
        $recreated.append(gate, qargs)$
        $steps[q_r] \leftarrow steps[q_r] + 1$
        $total\_inst \leftarrow total\_inst - 1$
        advanced_steps $\leftarrow$ advanced_steps $+ 1$
      end
    end
    if *advanced_steps is* 0 **then**
      break and discard recreated
    end
  end
end

1) Structure of Dummy Subcircuits: Adding a random subcircuit may result in subcircuits that have different gates, gate distributions, etc., from the real subcircuits. As a straightforward solution with the lowest computation cost for the user, we choose to copy existing subcircuits and submit them multiple times. To avoid submitting the exact subcircuits, we randomize the gates on the created dummy circuits, and the added gates are chosen from the total gates in the original subcircuits. Note that in the threat model, we assume the cloud provider is not using information about subcircuit structure to help in the recreation of the original circuit, thus, we assume they will not eliminate the dummy subcircuits.

A more important factor in the selection of dummy subcircuits is to conserve the total number of upstream and downstream cut qubits in the submitted circuits. The number of total upstream cut qubits must always be equal to that of the downstream cut qubits. Therefore, when introducing additional dummy circuits, we should introduce an equal number of upstream and downstream cuts, so that a valid recreation exists. To maximize the produced candidate circuits, we choose to copy subcircuits with upstream and downstream cuts, therefore each time adding an even number of dummy subcircuits. This configuration leads to a significantly increased number of candidate circuits and also higher demanded computation time, as the combinations of the various subcircuit mappings (including the dummy ones added after a copied subcircuit) can be extremely high. For this reason, in the evaluation section, we propose an estimation method used to decide the number of added dummy circuits, based on the desired level of protection, without needing to compute the actual recreated circuits.

2) Number of Dummy Subcircuits: Regarding the number of added subcircuits, there are arbitrary choices that one can make, depending on the complexity they want to add. As long as the total number of upstream cut qubits is equal to the number of downstream ones, an arbitrarily high number of dummy circuits can be used. The recreation process could potentially be made computationally unfeasible if multiple dummy circuits are added. Note that the user can distinguish the dummy subcircuits from the real subcircuits, thus they can easily ignore computation results from dummy subcircuits and only use the results from the original ones to finish the circuit cutting reconstruction on their machine.

TABLE I: Local computer and software tools used in the evaluation.

CPU	Apple M1
RAM	8 GB
Python	3.8.18
Qiskit	0.45.1

V. Evaluation Setup

This section gives details of our evaluation setup, as well as metrics used and the benchmarks upon which our techniques were evaluated.

A. Execution Environment

To develop and execute the recreation algorithm we used a commercial Macbook Air with a Python environment and IBM's Qiskit SDK. We plan to open-source the code developed for this paper. The specifications are shown in Table I.

B. Circuit Cutting Software

We used CutQC software version from June 2022 to perform the circuit cutting and generate the subcircuits. The configura-

tions used for cutting a quantum circuit with size n are shown in Table II.

TABLE II: CutQC configuration used in circuit cutting.

Max. subcircuit width	n
Max. subcircuit cuts	10
Subcircuit size imbalance	2
Max. cuts	10
Desired subcircuit number	2, 3, 4, 5

C. Dummy Subcircuit Generation

We extended circuit cutting with our new techniques, especially dummy subcircuit generation. We developed simple Python software to generate the dummy subcircuits. A CutQC-produced subcircuit is chosen at random, is copied, its gates are randomized and is appended as a dummy circuit, according to the specifications mentioned in the previous section.

D. Benchmarks

To evaluate the complexity of the recreation process, i.e. the effort by the attacker, we used the following circuit types:

1) *Bernstein-Vazirani (BV).* The Bernstein-Vazirani quantum circuits efficiently uncover a hidden binary string within a single query [9]. They showcase quantum parallelism to achieve a quadratic speedup over classical approaches and are a prominent example of quantum supremacy over classical counterparts.
2) *Supremacy*. The supremacy quantum circuits have dense probability output and were used by Google to demonstrate quantum advantage [10]. Due to the structure of these circuits (2-D with the two dimensions differing by up to 2 qubits) not all qubit numbers are valid. Thus, for the evaluation of all circuit types we use circuits of 4, 6, 8, or 12 qubits, to match the supremacy circuits' size.
3) *Quantum Fourier Transform (QFT).* Quantum Fourier Transform circuits are integral components in quantum algorithms [11]. These circuits perform a quantum version of the classical Fourier Transform and are crucial in algorithms like Shor's algorithm for integer factorization. They showcase quantum advantage in parallelism, and can efficiently solve problems related to periodicity and superposition, offering significant speedup over classical algorithms.

The above benchmark circuits, which were also used to evaluate CutQC [7],nserve as a standard collection for gate-based quantum computing platforms and promising near-term applications.

E. Metrics – Number of Generated Recreations

For all aforementioned circuit types, the first step is to utilize CutQC to generate a set of subcircuits for each. These subcircuits are then used by our algorithm to calculate all possible mappings and initiate the recreation process. The number of produced results depends on the original circuit's size and the structure of each connected subcircuit, as mentioned in the previous sections.

F. Metrics – Number of Estimated Recreations

When the algorithm starts recreating a quantum circuit for a given produced mapping, the structure and dependencies of the subcircuits could lead to a topological error, as showcased in Figure 5. Therefore, the final number of recreated circuits can be lower than the one calculated with Equation 3, as this formula is an overestimation of the recreated circuits. For this reason, we plot the number of estimated recreations, i.e. the result of Equation 3 for a given set of subcircuits along the number generated recreations that did not have topological errors. Although Equation 3 can overestimate the number of recreations, we can make predictions for the desired obfuscation level, as shown in 8b, without having to compute all the recreated circuits by using this formula.

G. Metrics – Output Probabilities of Candidate Circuits

The primary objective of circuit obfuscation is to generate output probability distributions that significantly deviate from the original, thereby preventing attackers from extracting useful information. We use a custom Mean Absolute Percentage Error (MAPE) metric to quantify the average percentage difference between two quantum state distributions. We first sort both the ground truth and the reconstruction distributions. Then, we compute the MAPE based on the following equation:

$$MAPE \equiv \frac{1}{2^n} \sum_{i=0}^{2^n-1} \frac{|P_i^r - P_i^g|}{P_i^g} \tag{4}$$

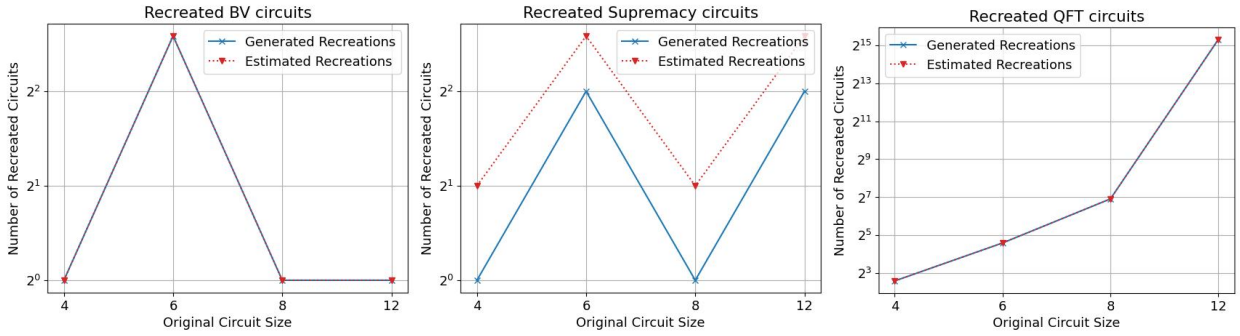
where P^g is the ground truth distribution, P^r is the distribution of a candidate circuit, and n is the number of qubits of the benchmark.

VI. Evaluation Results

We begin the evaluation process by computing the number of recreated circuits for the aforementioned circuit types and varying sizes of the original circuits. As seen in Figure 6, the different circuit types exhibit different behavior, which depends on the subcircuit size. The two important characteristics are the number of generated recreations and whether that number follows the estimation by Equation 3. In the following sections, we analyze the results from Figure 6 for each circuit type.

A. Bernstein-Vazirani (BV)

Employing circuit cutting for BV circuits, using the CutQC software, results in an odd number of subcircuits with an even number of total cuts. For an original circuit with a size of 4, 8, or 12 qubits, CutQC produces three subcircuits with two total cuts. Due to this characteristic, the recreation algorithm generates only a single recreated circuit, i.e. the original one. When the original circuit has a size of 6 qubits, 5 subcircuits with a total of 4 cuts are produced, resulting in 6 recreations. In both cases, it is obvious that circuit-cutting does not provide the required protection to the user, as a malicious cloud provider could easily reconstruct the original circuit and gain access to the algorithm submitted.



(a) (b) (c)

Fig. 6: Recreation results for different circuit types. (a) Bernstein-Vazirani circuits with sizes of 4 up to 12 qubits. The generated circuits are minimal, due to the structure of the produced subcircuits, with one subcircuit having only upstream cuts, one having only downstream cuts and one having both. The estimated recreations are calculated by using (3) and showcase that this type of circuit does not face topological errors during the recreation algorithm. (b) Supremacy circuits with size of 4 up to 12 qubits. While the recreated circuits are minimal, as only two subcircuits are generated, the estimated recreations disclose the topological errors produced during the recreation algorithm, as seen in Figure 5. (c) Quantum Fourier Transform circuits with sizes of 4 up to 12 qubits. The subcircuits generated from circuit-cutting, include many upstream and downstream cuts, thus, they induce multiple possible recreations. Similarly, the estimated recreation results, depict that no topological errors exist during the circuit recreation.

The key to overcoming this problem is to utilize the *dummy-subcircuits* technique. In Figure 8a, we examine the obfuscation level of a BV circuit with 4 qubits, when adding dummy subcircuits. For 6 added subcircuits, the generated recreations have increased from a single result, when no dummy subcircuits are used, to 5,040 candidate circuits. Therefore, a malicious attacker would have to examine a significantly larger amount of circuits to figure out the correct algorithm submitted by the user. As we keep increasing the number of added dummy circuits, the recreation algorithm would have to process a considerably greater amount of mappings to produce the candidate circuits. Due to this limitation and to calculate the required number of dummy circuits for the desired obfuscation level, we can use Equation 3 to estimate the number of recreated circuits with an arbitrary number of added dummy circuits. As the BV circuits do not exhibit topological errors during the recreation process, therefore no generated mappings have to be discarded as shown in Figure 6a, the n_{valid} number from Equation 3 would match the real number of recreated circuits. We can thus estimate that for 60 added dummy circuits, the number of candidate circuits can reach up to 2^{278}, providing an adequate obfuscation level and making it almost impossible for a malicious attacker to guess the user's circuit.

B. Supremacy

Employing circuit cutting for Supremacy circuits results in 2 produced subcircuits with a varying number of total cuts; 2 cuts for sizes 4 and 8 and 3 cuts for sizes 8 and 12. Therefore, for a supremacy circuit with 4 or 8 qubits, only a single recreated circuit is produced, whereas for a circuit with 6 or 12 qubits, 4 candidate circuits are generated after default circuit cutting. We have already showcased in Figure 5 that this type of circuit encounters topological errors during the recreation process, therefore Equation 3 overestimates the produced recreated circuits, as seen in Figure 6b. When executing random circuits that could generate topological errors, this estimation cannot be employed as it would overestimate the produced results. We leave the examination of such cases for future work.

C. Quantum Fourier Transform (QFT)

Employing circuit cutting for QFT circuits results in 2 subcircuits with multiple cuts, that range from 3 cuts for 4 qubits, to 8 cuts for 12 qubits. Due to this large number of cuts, the QFT circuits have the most generated recreations, which reach up to 40,320 circuits for the case of 12 qubit size. Similarly to BV type, these circuits do not exhibit topological errors during the recreation algorithm, thus we could use Equation!3 to estimate the provided obfuscation level for a varying number of added dummy subcircuits.

D. Probability Distribution of Recreated Circuits

Figure 7 displays the sorted MAPE scores (Equation 4) for 100 randomly sampled candidate circuits of a 16-qubit Supremacy benchmark. Most of the candidate circuits register MAPE scores exceeding 10, indicating that the recreated amplitudes are more than $10\times$ different from the original. This substantial divergence in amplitudes, not just structural differences, highlights the significant alterations in the quantum states of the recreated circuits compared to the original. This demonstrates substantial obfuscation effectiveness, highlighting their potential for robust protection against reverse engineering from attackers.

VII. Time Overhead of Recreation Algorithm for the Attacker

In this section, we evaluate the execution overhead faced by the attacker, i.e. the cloud provider. The two main time-consuming processes are the generation of candidate circuits

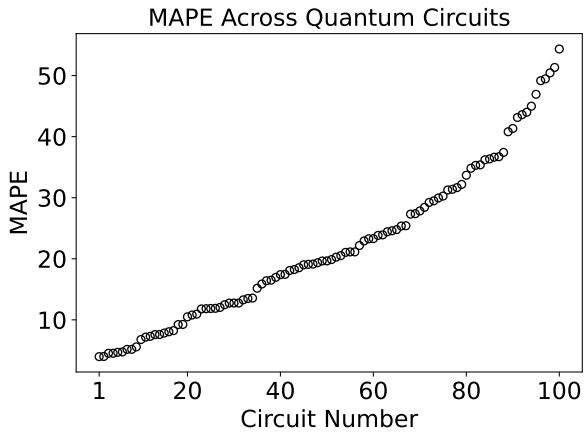


Fig. 7: Sorted MAPE scores for 100 randomly sampled candidate circuits for a 16-qubit Supremacy benchmark circuit without dummy subcircuits.

using the recreation algorithm and the reconstruction of the subcircuit execution results, to retrieve the outcome for each recreated circuit.

In Figure 8b, we compute the time (in seconds) required to execute the recreation algorithm for a BV circuit with 4 qubits and added dummy subcircuits. While additional dummy subcircuits can significantly increase the required time, because a large number of circuits must be constructed as shown in Figure 8a, the most computationally intensive part is the probability generation. Using the execution results from the submitted subcircuits, CutQC reconstructs the original quantum circuit's outcome using the appropriate metadata. An attacker can utilize the same process to compute the outcome for every recreated circuit, just by executing the given subcircuits once and appropriately constructing the metadata, without having to execute every candidate circuit. However, even this process can be extremely time-consuming with large numbers of recreated circuits, as seen in Figure 8c. While for no added dummy subcircuits and a single candidate circuit, this process requires around 8 seconds, when 6 dummy circuits are added and 5,040 candidate circuits are produced, it requires almost half a day. Therefore, the computation complexity for the attacker can be significantly high.

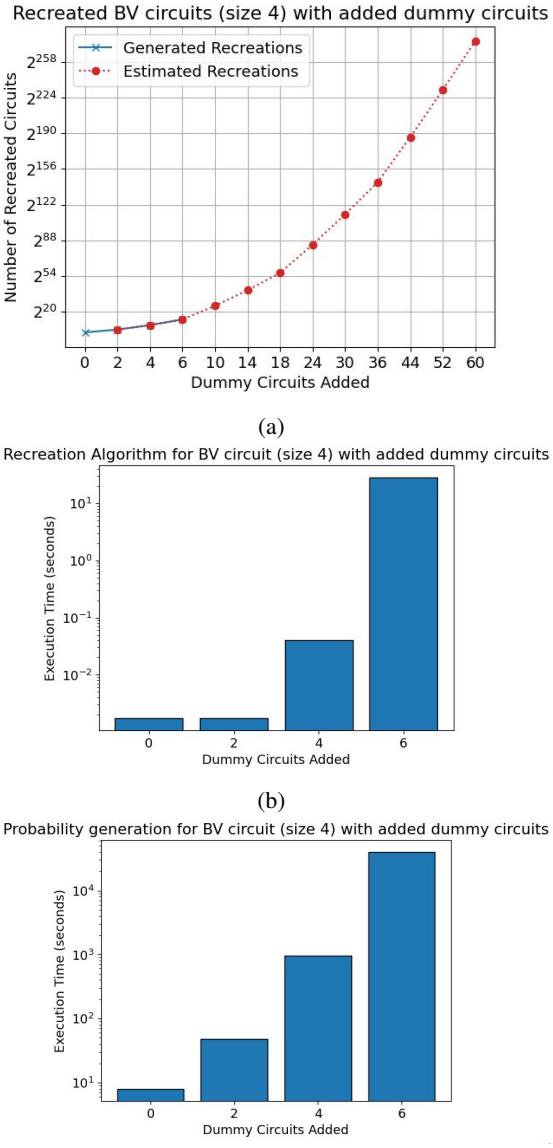


Fig. 8: Recreation results for BV circuits with 4 qubits, after adding dummy subcircuits. (a) By adding an increasing number of dummy subcircuits, more recreated circuits are produced. Due to no topological errors in this circuit type, Equation 3 can be used to calculate the total number of circuits that would be generated. After adding 6 dummy subcircuits, the computational complexity is increased significantly, so estimated results are used to showcase the provided obfuscation levels. (b) Time overhead of the recreation process for up to 6 dummy circuits. (c) Time overhead of CutQC's reconstruction process that generates the outcomes of the original quantum circuit, by combining subcircuit results.

VIII. Related Work

To help protect from untrusted quantum computer cloud providers or insider attackers, a number of researchers have focused on developing delegated quantum computation, blind quantum computation, and similar ideas [5], [12]–[24]. These methods are designed to enable a client to conduct quantum computations on a remote quantum computer while maintaining the confidentiality of the user's code. Majority of blind quantum computation protocols rely on the existence of quantum networking for communication between user's trusted quantum computer and the untrusted, remote quantum client and the server. These approaches are not feasible in cases where the user does not have a (trusted) quantum computer.

An alternative approach to blind quantum computation is the use of some trusted hardware within the quantum computer to make it infeasible for cloud provider to learn the computation being performed on the computer [25]. They proposed to protect the computation by adding decoy control pulses

into the circuits submitted to the cloud provider. Then, on the (trusted) quantum computer hardware, the decoy pulses are removed, i.e. attenuated, before they reach the quantum computer's qubits. While the approach is promising, it requires modification to hardware of the quantum computers.

As an alternative to these approaches, our work presented a new method to obfuscate the quantum circuits by leveraging quantum circuit cutting. It does not require user to have trusted quantum computer, nor quantum networking, nor non-colluding cloud providers, nor does it require hardware changes to quantum computers. Our method can be deployed today and can support arbitrarily high obfuscation level, i.e. arbitrarily high number of candidate circuits that the cloud provider would have to guess, based on the number of dummy subcircuits that users submit along with the subcircuits generated by quantum circuit cutting software.

IX. Conclusion

This paper demonstrated for the first time how quantum circuit cutting can help protect users against untrusted providers of cloud-based quantum computers. To provide a configurable obfuscation level, this work proposed a novel *dummy-subcircuits* technique, which introduces dummy subcircuits with additional cut points. This technique results in multiple generated candidate circuits, creates further confusion for the cloud provider, and helps obfuscate the user's original circuits with limited cost. Additionally, it creates a crucial execution overhead for the attacker, should he or she attempt to recover the outcomes for each recreated candidate circuit. The evaluation of the produced circuits showcased the significant alterations in the quantum states compared to the original quantum circuit, thus, achieving robust protection. Circuit-cutting, combined with *dummy-subcircuits*, efficiently protects against malicious attackers, by providing a configurable level of obfuscation, and can be deployed using current quantum computing infrastructure. As future work, we leave the analysis of random quantum circuits, which could potentially exhibit topological errors during the recreation process. In that case, we need to properly estimate the achieved obfuscation level when using circuit-cutting and additional *dummy-subcircuits*.

References

[1] "Ibm quantum," https://quantum-computing.ibm.com/.

[2] "Amazon braket," https://aws.amazon.com/braket/.

[3] "Azure quantum," https://azure.microsoft.com/en-us/products/quantum.

[4] S. J. Stolfo, S. M. Bellovin, S. Hershkop, A. D. Keromytis, S. Sinclair, and S. W. Smith, *Insider attack and cyber security: beyond the hacker*. Springer Science & Business Media, 2008, vol. 39.

[5] A. M. Childs, "Secure assisted quantum computation," sep 2005, https://arxiv.org/pdf/quant-ph/0111046.pdf. [Online]. Available: https://doi.org/10.26421%2Fqic5.6

[6] T. Peng, A. W. Harrow, M. Ozols, and X. Wu, "Simulating large quantum circuits on a small quantum computer," *Physical Review Letters*, vol. 125, no. 15, p. 150504, 2020.

[7] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi, "Cutqc: using small quantum computers for large quantum circuit evaluations," in *Proceedings of the 26th ACM International conference on architectural support for programming languages and operating systems*, 2021, pp. 473–486.

[8] W. Tang and M. Martonosi, "Scaleqc: A scalable framework for hybrid computation on quantum and classical processors," *arXiv preprint arXiv:2207.00933*, 2022.

[9] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1411–1473, 1997. [Online]. Available: https://doi.org/10.1137/S0097539796300921

[10] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and et al., "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, pp. 595–600, 2018. [Online]. Available: https://doi.org/10.1038/s41567-018-0124-x

[11] C. James W. and T. John W., "An algorithm for the machine calculation of complex fourier series," *Math. Comp. 19 (1965),*, vol. 19, pp. 297–301, 1965. [Online]. Available: https://doi.org/10.1090/S0025-5718-1965-0178586-1

[12] A. Broadbent, J. Fitzsimons, and E. Kashefi, "Universal blind quantum computation," in *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009, pp. 517–526, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5438603.

[13] D. Aharonov, M. Ben-Or, and E. Eban, "Interactive proofs for quantum computations," 2008, https://arxiv.org/pdf/0810.5375.pdf.

[14] T. Morimae, V. Dunjko, and E. Kashefi, "Ground state blind quantum computation on aklt state," 2011, https://arxiv.org/pdf/1009.3486.pdf.

[15] V. Dunjko, E. Kashefi, and A. Leverrier, "Blind quantum computing with weak coherent pulses," *Physical Review Letters*, vol. 108, no. 20, may 2012, https://arxiv.org/pdf/1108.5571.pdf. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.108.200502

[16] T. Morimae and K. Fujii, "Blind topological measurement-based quantum computation," *Nature Communications*, vol. 3, no. 1, sep 2012, https://arxiv.org/pdf/1110.5460.pdf. [Online]. Available: https://doi.org/10.1038%2Fncomms2043

[17] ——, "Blind quantum computation protocol in which alice only makes measurements," *Physical Review A*, vol. 87, no. 5, may 2013, https://arxiv.org/pdf/1201.3966.pdf. [Online]. Available: https://doi.org/10.1103%2Fphysreva.87.050301

[18] J. F. Fitzsimons and E. Kashefi, "Unconditionally verifiable blind quantum computation," *Physical Review A*, vol. 96, no. 1, jul 2017, https://arxiv.org/pdf/1203.5217.pdf. [Online]. Available: https://doi.org/10.1103%2Fphysreva.96.012303

[19] T. Morimae, "Continuous-variable blind quantum computation," *Physical Review Letters*, vol. 109, no. 23, dec 2012, https://arxiv.org/pdf/1208.0442.pdf. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.109.230502

[20] T. Sueki, T. Koshiba, and T. Morimae, "Ancilla-driven universal blind quantum computation," *Phys. Rev. A*, vol. 87, p. 060301, Jun 2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.87.060301

[21] T. Morimae and T. Koshiba, "Composable security of measuring-alice blind quantum computation," 2013, https://arxiv.org/pdf/1306.2113.pdf.

[22] V. Giovannetti, L. Maccone, T. Morimae, and T. G. Rudolph, "Efficient universal blind quantum computation," *Physical Review Letters*, vol. 111, no. 23, dec 2013, https://arxiv.org/pdf/1306.2724.pdf. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.111.230501

[23] A. Mantri, C. A. Pé rez-Delgado, and J. F. Fitzsimons, "Optimal blind quantum computation," *Physical Review Letters*, vol. 111, no. 23, dec 2013, https://arxiv.org/pdf/1306.3677.pdf. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.111.230502

[24] T. Morimae, "Verification for measurement-only blind quantum computing," 2014, https://arxiv.org/pdf/1208.1495.pdf.

[25] T. Trochatos, C. Xu, S. Deshpande, Y. Lu, Y. Ding, and J. Szefer, "Hardware architecture for a quantum computer trusted execution environment," 2023.