Masking Countermeasures Against Side-Channel Attacks on Quantum Computers

Jason T. LeGrow jlegrow@vt.edu
Travis Morrison tmo@vt.edu
Jamie Sikora sikora@vt.edu
Nicolas Swanson nicswanson@vt.edu

May 5, 2024

Abstract

Recent advances in practical quantum computing have led researchers and businesses to develop proprietary quantum algorithms, with the hope that these algorithms will eventually outperform the best classical alternatives. Cloud quantum computing providers like IBM, Microsoft, and Amazon enable users to run these algorithms to reap the benefits of quantum computers without bearing the substantial cost of owning one. However, as with any cloud-based service, there is a security concern: the physical devices are not under the users' control. A malicious insider with physical access to the quantum computer could potentially use side-channel information to learn sensitive information about a quantum algorithm. We propose a modification to the transpiler of a quantum computer to safeguard against such side-channel attacks. We demonstrate that if it is feasible to shield a specific subset of gates from side-channel attacks, then it is possible to conceal all information in a quantum circuit by transpiling it into a new circuit whose depth grows linearly, depending on the quantum computer's architecture. We provide concrete examples of implementing this protection on IBM's quantum computers, utilizing their virtual gates and editing their transpiler.

1 Introduction

Advances in quantum algorithms and quantum information [2, 20, 3] have created demand for accurate, reliable, and secure quantum computers. Presently, cloud-based vendors such as IBM Quantum [12], Amazon Bracket [1], and Microsoft Azure [19] allow anyone to run an algorithm on a noisy intermediate-scale quantum (NISQ) device. As more sensitive algorithms and data are entrusted to cloud-based quantum computing services, the supplier would likely offer a contract promising not to use the data for anything other than the task prescribed by the user. Even so, the cloud-based computing service needs to be careful about the side-channel information leaked when running a quantum algorithm on the data. In general, side-channel information can be thought of as the information leaked via any physical interactions the computer has with its environment. Particularly in the context of cryptography, side-channel attacks have been executed using the acoustics, electromagnetic radiation, timing, and power consumption of the computer. As shown in [25], if quantum algorithms are implemented in their naïve way, then a power-based side-channel attack can usually identify the algorithm being run and even reconstruct most of the algorithm itself. This poses a problem for the user and the cloud-based quantum computing suppliers if the power drawn by the computer is readable from an untrusted third party.

In this paper, we provide information-theoretic tools and demonstrate methods to enhance protection against side-channel attacks on quantum computers. These tools are crafted exclusively for quantum computers and the quantum circuit model, meaning they are not easily transferable to classical computers. We illustrate that virtual gates, a specific subset of gates used by [12], provide resistance to side-channel attacks through our devised technique of "virtual gate masking." In [25], the authors propose a method of substituting real gates with virtual gates. In our paper, we build on this idea by masking all gates in the circuit with virtual realizations, while also concealing information about the placement of gates in a circuit.

There is a regime called "blind quantum computation" [7] that provides an information-theoretic secure way for a client to perform a quantum computation without the cloud computing service even knowing what algorithm is being run. However, these protocols require the existence of a quantum channel between the client and server, as well as a device from the client that has the ability to create single qubit quantum states. Instead, we consider a scenario in which the client does not have a quantum device, trusts the cloud provider, but requires protection from adversaries in proximity to the quantum computer.

In this work, we discuss what it means to leak quantum side-channel information in a quantum algorithm. Our intention is to modify the transpilation process so that the algorithmic behavior does not change, but it masks its gate layout. We start with an overview of quantum side-channel attacks and define various types of information that a side-channel attack might aim to extract from a quantum algorithm in Section 2. In Section 3, we outline a masking process that enables us to conceal all the information in the quantum algorithm, assuming a certain subset of gates cannot be identified. In Section 4, we discuss the concept of a virtual gates, which are used on IBM's superconducting quantum computers. In Section 4.2, we apply our new machinery outlined in Section 3 to provide a new transpilation process, allowing IBM's quantum computers to achieve information-theoretic security against side-channel attacks, assuming that virtual gate information is undetectable. Finally, we present a detailed application of our masking process in Section 5, where we hide the discriminant of a class group on a cloud-based quantum computer. Such computations are relevant in post-quantum cryptography, so it is crucial that their quantum implementable subroutines are not vulnerable to side-channel attacks.

2 Quantum Side-Channel Attacks

Side-channel attacks on any type of computer depend heavily on the hardware and architecture of the computer. This makes side-channel attacks against quantum computers particularly tricky in the age of NISQ devices since their hardware and architecture are constantly evolving [17]. In this paper, our focus is not on how threat models will be physically realized. Instead, we examine the types of information a quantum side-channel attacker can obtain and draw equivalences where helpful.

Unlike classical side-channel attacks, which attempt to recover data used in an algorithm, the ultimate goal of a quantum side-channel attack is *circuit reconstruction* [25]. Using side-channel information to directly recover the computer's qubits, rather than its evolution, would effectively measure parts of its internal quantum state, a process limited by the *no-cloning theorem* which prohibits creating identical copies of an unknown quantum state. This would not only affect the computation of the quantum computer but also suggest that the system's decoherence is significant and unfit for quantum computation. This gives quantum computers a unique advantage over

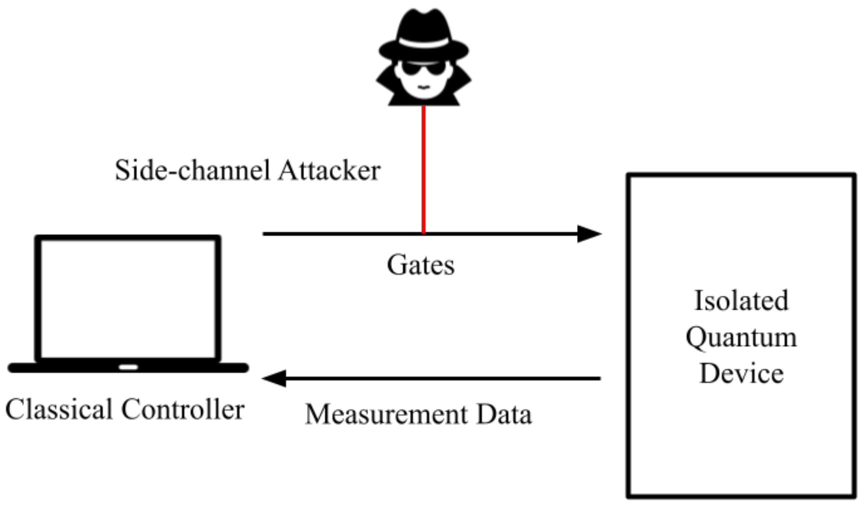

Figure 1: Diagram of assumed side-channel attacker and computer architecture.

classical computers when defending against side-channel attacks. Any attack which reveals (partial information about) the quantum state means the state will (partially) collapse, impacting the output of the algorithm possibly in a detectable manner. In many current quantum computer realizations, quantum algorithms start with n fresh qubits, each in the state $|0\rangle$, which is the quantum analog of a 0 in a traditional computer. These qubits evolve over time to a state which is then measured [18]. The measurement data is information a side-channel attacker could try to obtain, but this information is purely classical, i.e., not quantum, after the measurement is made. We do not explore side-channel attacks that target measurement data in this work, but rather, we focus on attacks that attempt to learn something about the algorithm itself.

A quantum algorithm can be parameterized by the sequence of gates used to evolve its qubits, which can be written as a quantum circuit. While there are other models for quantum computers, such as the measurement-based model [6], today's largest and most used quantum computers use superconducting qubits and the circuit model [5]. The goal of circuit reconstruction is to identify this sequence of gates. The implementation of a quantum algorithm has details relevant to side-channel attacks that are not needed for theoretical quantum algorithms. Our goal in this section is to explicitly outline the information contained in a physically implemented quantum circuit.

A *universal gate set* is a set S of gates, represented by unitary matrices, that form a generating set for all gates under multiplication and the tensor product. In practical terms, quantum computers apply only the gates in their universal gate set when physically executing a quantum algorithm. For our definition of a universal gate set, we assume, without loss of generality, that it always contains I_2, the 2×2 identity unitary. This is the gate which does nothing, i.e., it leaves the qubits alone.

Definition 1. An n-qubit *quantum circuit* is described by a tuple $C = (U_1, \ldots, U_T)$ where each U_i is a unitary on n qubits with a fixed tensor decomposition $U_i = \bigotimes_{j=1}^{l} V_{i,j}$. Each U_i is called a *time step unitary* and each $V_{i,j}$ is a unitary. An n-qubit *implemented quantum circuit* using a universal gate set S is a quantum circuit where each $V_{i,j}$ is a gate in S that can physically be executed by the quantum computer.

We usually denote a quantum circuit by C and, in particular, express it via its time step unitaries $(U_1, \ldots, U_T)$. There may be many ways to realize a quantum algorithm as a quantum circuit, but when we refer to a quantum circuit C, we assume a choice for $(U_1, \ldots, U_T)$ has been made along with a tensor decomposition for each U_i. When a quantum circuit is implemented we mean the gates in this decomposition are in S. The integer T is called the circuit's *depth*.

Definition 2. A gate V used in a circuit C has four attributes:

1. An *identification* (or *label*): a unitary matrix that encodes the operation that V performs on the qubits it acts on;
2. An *index*: the time step at which it is applied in C. Formally, the index is an integer $ind \in \{1, \ldots, T\}$ such that V is a gate of the time step unitary at index ind;
3. A *wire label*: a list of qubits the gate is applied to. We can label the n qubits $\{1, \ldots, n\}$ and let the wire label be the subset of these indices corresponding to the qubits on which the gate V acts; and,
4. A *size*: the number of qubits V acts on.

In our paper, when we refer to a gate V, we do not only refer to its matrix label, we also imply it is in a quantum circuit with a fixed position. We implicitly assume that V is in the tensor decomposition of some time step unitary U_i when we say V is (used) in C. Note that a gate's size is included in its wire label but its size does not determine its wire label. If a gate has the identification of an identity matrix of any size, we call it an *identity gate*.

Example 3. When specifying a circuit, one must be careful with the tensor decomposition of its time step unitaries. Consider the circuits $C_1 = (U_1) = (I \otimes H)$ and $C_2 = (U_1') = ((I \otimes H))$ pictured in Figure 2. Note that C_1 has two gates: one with identification I, index 1, and wire label 1, and one with identification H, index 1, and wire label 2. However C_2 has one gate with identification $I \otimes H$, index 1, and wire label $\{1, 2\}$. Both these circuits are functionally equivalent, but are technically different circuits.

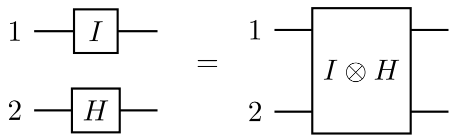

Figure 2: Pictured is C_1 on the left and C_2 on the right from Example 3. These circuits do the same thing, but have different representations in terms of gates.

Example 4. Consider the three circuits in Figure 3 (on the next page). The first circuit is $C_1 = (U_1) = (U)$, where U is a unitary matrix. The second circuit C_2 exhibits a decomposition of the gate U in terms of "simpler" gates more commonly used. The third circuit C_3 is an implemented quantum circuit written in the universal gate set $S = \{I, \text{CNOT}, H, X_{\pi/2}, T\}$. The transformation from C_1 to C_3 or from C_2 to C_3 is an example of a *transpilation process*, a process that takes an arbitrary gate as input and implements it using a fixed gate set.

Note for all of our circuits, we have fixed an ordered labeling of the qubits. A frequently used two-qubit gate is the CNOT gate, which has a control qubit and a target qubit. If you switch the control with the target, you get a completely different gate even though they act on the same two qubits. Thus, these two gates are different and require different matrix identifications even though they are both commonly referred to as CNOT.

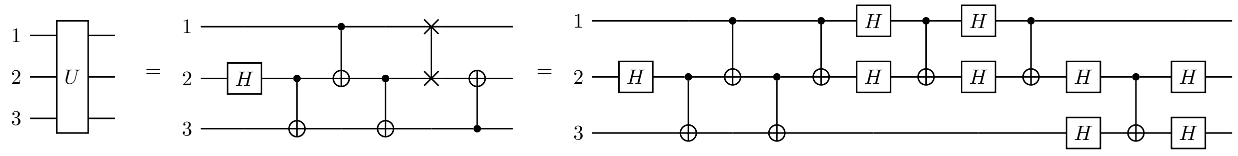

Figure 3: Different circuits for the quantum teleportation algorithm. The circuit C_1 has a single gate with identification U, index 1, wire label $\{1, 2, 3\}$, and size 3; the circuit $C_2 = (U_1, \ldots, U_6)$ where $U_1 = I \otimes H \otimes I$, $U_2 = I \otimes \text{CNOT}$, and so on. The first Hadamard gate on the left of C_2 has identification H, index 1, wire label 2, and size 1. In C_3, the last two gates have identification H, index 12, size 1, and wire labels 2 and 3 respectively.

We now take a more detailed look at the goal of circuit reconstruction of a quantum circuit C.

Definition 5. Types of information about a circuit C:

1. The *total information* of C is the identification, index, and wire label of the gates in C.
2. For a subset $R \subset S$, the *R-absent information* of C is the identification, index, and wire label of all gates not identified by an element of R.
3. The *positional information* of C is the index and wire label information of its gates.
4. For a subset $R \subseteq S$, the *R-positional information* is the *R-absent information* of C together with the index and wire labels of gates in R.

The ideal side-channel attack would be able to recover the total information of C. The side-channel attacks studied in [25] use power consumption and timing information in their attacks. The strongest attack they implement is the *Per-Channel Power Single Trace Attack*, which allows the attacker to measure the power traces of the drive and control channels in the quantum computer separately. This translates to knowledge of the R-absent information of the circuit for a specific subset of gates called virtual gates which we discuss in more detail in section 4.

3 Masking in the Transpiler

3.1 Masking certain gates

The term *transpilation* is used by Qiskit [13] to refer to the process of transforming a high-level description of a quantum algorithm into a circuit using gates that can be physically executed by the quantum computer. One can think of the transpiler as the procedure that decomposes a quantum circuit C into an implemented quantum circuit $B = (U_1, \ldots, U_n)$ where each gate of B is in the universal gate set S. In the following, we outline a modification to the transpiler that gives extra protection against circuit reconstruction under the assumption that a certain subset of gates $R \subseteq S$ are difficult to detect. From here on out, we assume the quantum computer operates on n qubits and we let N be the maximum size of a gate at the time our transpiler modification occurs.

Definition 6. A subset of a universal gate set $R \subseteq S$ is called a *covering gate set* if for each $m \in \{1, \dots, N\}$, there is a fixed ordered tuple of gates in S, $(S_1, \dots, S_{r_m})$, where each S_i acts on m qubits and is a tensor product of gates in S. These m-qubit gates must satisfy that any m-qubit gate U can be decomposed as

$$U = \prod_{i=1}^{r_m} R_i S_i \tag{1}$$

for some choice of $(R_1, \dots, R_{r_m})$ where each R_i is a tensor product of gates in R.

Procedure 7. Given a universal gate set S and a covering gate set $R \subseteq S$, we define *R-gate masking* to be the following modification to a deterministic transpilation process:

1. For each m up to N, fix an ordered tuple $(S_1, \dots, S_{r_m})$ given by Definition 6.
2. Given a high-level quantum circuit C, decompose each non-identity gate V in C using Equation (1) with m equal to the number of qubits V acts on.
3. Proceed with transpilation as usual.

Remark. Note that step (1) serves as a preprocessing step, performed prior to receiving an algorithm for transpilation. Step (3) is necessarily vague due to the dependence of hardware and architecture on transpilation, which involves more than just decomposing gates. For example, if the given high-level quantum circuit applies two gates U and V at the same time step, the hardware of the computer may require U and V to act on the same number of qubits. Moreover, the application of U and V may be feasible only when they act on qubits physically separated from each other within the quantum computer. Because of this, the computer may need to apply U before V, or vice versa, resulting in the addition of time steps. We require that the existing transpilation process is deterministic because we want the positional information of a transpiled circuit to be determined by the inputted algorithm.

Theorem 8. *Suppose we have a universal gate set S, covering set R, and two quantum circuits C and C' with the same positional information. If C is transpiled into B and C' is transpiled into B', both with R-gate masking, then the R-positional information of B and B' are equivalent.*

Proof. Two circuits with equal positional information will have transpiled circuits with equal positional information. This is because each m-qubit gate V is transpiled into a sequence of gates containing positional information dependent only on the positional information of V and not its identification. Therefore the positional information of B and B' are equal. Moreover, if we consider the set of gates with identification in $S \setminus R$, they have the same identification in both B and B' since $S_1, \dots, S_r$ is a sequence of gates fixed by the transpilation process and each gate V gets transpiled as $V = \prod_i^{r_m} R_i S_i$. Thus, the only difference in B and B' is the identifications of gates in R. $\square$

One way to interpret the above theorem is that R-gate masking provides a way to hide all identification information in a circuit, given one can hide the identification of the gates in R. Ideally, the set R is large enough so that it is infeasible to simply guess the identification of a gate in a R-masked circuit whose identification is in R. We show an application where R is the set of virtual gates, which is an infinite set. Note that the sequence B still contains the positional information of C, even if the identification of gates in R are unknown. For example, if C first applies a single-qubit gate to wire 1, then a two-qubit gate to wires $\{2, 3\}$, an attacker with R-positional information will still be able to recover this information from B. Luckily, there are ways to mask these properties as well.

3.2 Masking gate positions

We now provide a procedure to mask the positional information of any quantum circuit. This is achieved by creating a subcircuit that applies a gate to every possible wire label.

We first discuss the possible wire labels in a circuit. In a high-level description of a quantum circuit, any two-qubits can share a gate. At the physical hardware level, this is not always the case. In current superconducting quantum computers, there is a notion of a *topology* which refers to the physical layout of the qubits. See Figure 4 as an example. The topology imposes a restriction on which multi-qubit gates can be applied.

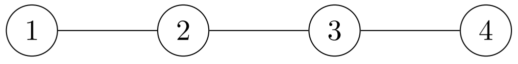

Figure 4: A possible layout of 4 physical qubits, called a 1D array. If two qubits can have a gate applied to them, they are connected with an edge. These are called *nearest neighbor interactions.*

Let W be the set of all possible wire labels up to size N that the topology of the quantum computer allows to be applied. Note sets in W are subsets of qubit indices that represent targets for multi and single gates based on the topology.

We now define a circuit which may seem odd at first, but it will be a building block to mask positional data. Let Y be a circuit consisting solely of identity gates in the following manner. The circuit Y has an identity gate that acts on qubits with indices in w for every $w \in W$. Since the construction of Y is not unique, let us fix one with minimum depth, and denote the depth as p. See Figure 5 for an example of such a Y circuit construction.

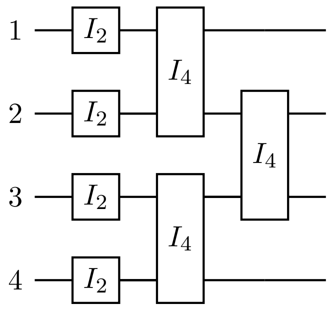

Figure 5: Using the topology in Figure 4, assuming we can apply only single-qubit gates and two-qubit gates, we have $N = 2$ and possible wire-labels $W = \{1\}, \{2\}, \{3\}, \{4\}$, $\{1,2\}$, $\{2,3\}$, and $\{3,4\}$. The circuit Y with minimum depth is shown above with depth $p = 3$.

Now that we have fixed this circuit Y, we show how any time step unitary U_i can be written to have the exact same positional data as Y. This will be achieved by replacing the identity gates in Y with real gates in U_i, forming Y_i. Then, we will then R-gate mask both the identity and real gates in Y_i, effectively hiding the positional information U_i. We give an example of this in Figure 6.

Procedure 9. Fix a hardware topology, a universal gate set S, and a covering gate set $R \subseteq S$. Define *total R-gate masking* to be the following modification to a transpilation process: given a

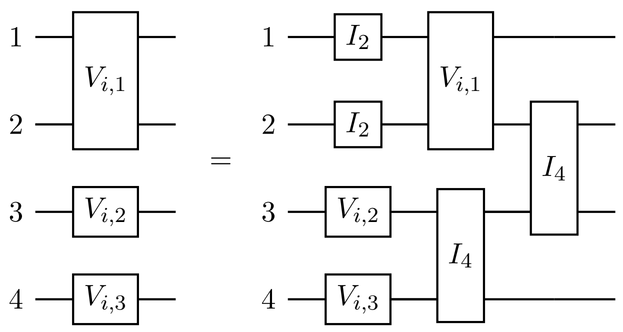

Figure 6: Using the topology in Figure 4, we can replace the time step unitary $U_i := \otimes_j^l V_{i,j}$ on the left-hand side with the functionally equivalent unitary Y_i on the right-hand side. Note that the positional data of Y_i is the same as in Y as seen in Figure 5, regardless of how U_i is defined.

circuit $C = (U_1, \ldots, U_T)$ such that for each time step unitary $U_i = \otimes_{j=1}^l V_{i,j}$, the gate $V_{i,j}$ acts on at most N gates:

1. Define the new circuit B consisting of T consecutive instances of the circuit Y. Let Y_i denote the subcircuit of B corresponding to the ith instance of Y in B.
2. For each time step unitary $U_i = \otimes_{j=1}^l V_{i,j}$, each $V_{i,j}$ acts on qubits with indices equal to some $w \in W$. For each $V_{i,j}$, replace the identity gate in Y_i that acts on w with $V_{i,j}$.
3. Perform R-gate masking on B but also mask all the identity gates, meaning we remove the restriction pertaining to identity gates in step (2) of Procedure 7.

Remark. In step (3), masking does not necessarily need to be applied to all identity gates, only the set of gates that correspond to $w \in W$. Finally, note that forming Y can be done in preprocessing, much like step (1) of R-gate masking.

Example 10. Assume again we have $N = 2$ and the topology in Figure 4. In Figure 7, let the

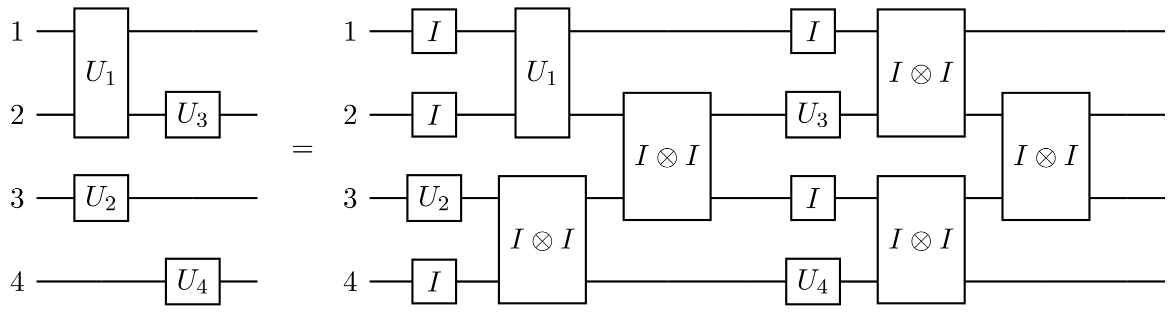

Figure 7: Example of total R-gate masking.

quantum circuit on the left with depth two be C. We have already formed our desired Y time step unitary in Figure 5. We create B from 2 copies of Y and then for each i from 1 to T, we insert the non-identity gates from in U_i into B by replacing the identity gates in Y_i with the same wire label. Note in Figure 7, the identity gates in Y_1 and Y_2 are only shown if they uniquely correspond to a wire label $w \in W$. Only the identity gates that are shown need to be masked in total R-gate masking.

Theorem 11. *Suppose we have a universal gate set S, a covering set R, a fixed topology, and two quantum circuits C and C' of the same depth with gates up to size N. If C is transpiled into the circuit B and C' is transpiled into the circuit B' using total R-gate masking, then the R-positional information of B and B' are equivalent.*

Remark. Note that we can always make the circuits C and C' have the same depth by adding in identity gates.

Proof. Since both circuits have gates of size at most N and the same depth, the intermediate circuits D and D' produced at the end of step (2) of Procedure 9 on input C and on input C' will have the same positional information as the circuit consisting of T consecutive instances of the circuit Y. Therefore the positional information of D and D' are equal. By Theorem 8, the R-positional information of B and B' are equal since the positional information of D and D' are equal. $\square$

Remark. Notice that C and C' no longer have to share the same positional information like they do in Theorem 8. We have successfully hidden the positional information of a circuit given the identification of gates in R are undetectable.

3.3 The depth of masked circuits

Now, we discuss the overhead incurred by these masking transpilation processes in terms of circuit depth.

Theorem 12. *Suppose we have a quantum algorithm $\mathcal{A}$, a universal gate set S, a covering set R, and a fixed topology. Suppose further that there is an existing transpiler that transpiles $\mathcal{A}$ into the quantum circuit C with depth T after considering the topology and universal gate set S. Let B be the circuit of depth T_B transpiled by the same transpiler with R-gate masking and let D be the circuit of depth T_D transpiled using total R-gate masking. Let Y be the circuit of depth p as defined in the previous section. We have*

$$T_B \leq 2rT$$

where $r = \max_{m \in \{1,\ldots,N\}} \{r_m\}$ is given by Procedure 7 and

$$T_D \leq 2rpT$$

where p is the depth of Y.

Remark. A transpiler might consider more than just the topology and the universal gate set when it transpiles. Additional physical implementation restrictions may need to be considered by the computer, adding depth to the circuit. For example, a quantum computer may only allow a single gate to be applied in every time step, increasing the depth by a factor of at most n for every time step in the original circuit. Since the circuit Y does not account for these physical restrictions, we assume the transpiler addresses them after our masking process has been executed. See Section 4.2 for an example. It is also worth noting that the overhead incurred by R-gate masking and total R-gate masking can be made much lower when one is willing to leak partial information about the circuit C. In Section 5, we demonstrate that only a handful of masked gates is enough to hide important information in a circuit.

We now consider the depth of the circuit Y, detailed in the last subsection. In superconducting quantum computers, it is common to represent their topology with a graph $G = (V, E)$ where each vertex is the index of a qubit after an ordering of the qubits has been fixed. An edge between qubits u and v means that the computer can apply a two-qubit gate to u and v. In this case, we assume that the topology only allows for one and two-qubit interactions, meaning $N = 2$ as we cannot apply gates to three or more qubits. Indeed, there exist many universal gate sets which contain only 1 and 2-qubit gates.

The first requirement of the circuit Y is that it must apply a single qubit identity gate to all qubits. Note regardless of the topology, we can always apply a single qubit gate to every qubit in one time step.

To discuss the minimum number of time steps required to apply an identity gate to all possible two-qubit pairs, we need to recall the edge chromatic number of a graph G. The *edge chromatic number*, denoted by $\chi'(G)$, is the minimum number of colors required to color the edges of G in such a way that if two edges share a common vertex, they have different colors. We can trivially color any graph with $|E|$ colors, but $\chi'(G)$ can be strictly less than $|E|$.

Suppose two edges are colored "blue" in a valid edge coloring. This corresponds to two distinct 2-qubit gates with no qubits in common, meaning they can be applied in the same time step in the circuit Y. In fact, all edges corresponding to the same color can be applied in parallel and therefore we can apply all possible two-qubit identity gates in at most $\chi'(G)$ time steps.

Theorem 13. *Suppose $N = 2$ and the topology of the quantum computer is given by a graph G. Let Y be the minimum depth circuit that implements an identity gate for every possible wire label $w \in W$. Then the depth of Y is either $\chi'(G)$ or $\chi'(G) + 1$.*

Proof. We know that we can implement the single-qubit gates in Y with one time step and the two-qubit gates in Y with $\chi'(G)$ time steps. Now, we argue that $\chi'(G)$ is the minimum number of time steps we can implement the two qubit identity gates in Y. Suppose there exists a circuit $C = (U_1, \ldots, U_d)$ of depth $d < \chi'(G)$ that implements a two qubit identity gate for every edge $e \in E$. We will color the graph with $d - 1$ colors, leading to a contradiction. For every two-qubit gate in U_1, color the corresponding edge "color 1". For every two-qubit gate in U_2, color the corresponding edge "color 2," and so on. Doing this, we would eventually color the whole graph with $d < \chi'(G)$ colors, as desired. So we do in fact need at least $\chi'(G)$ many time steps in a circuit C implementing two-qubit gates. We do not need one more time step in the case that we can place single qubit identity gates in C such that every possible qubit is acted on at least once. In this case, the time step consisting of single-qubit identity gates can possibly be distributed amongst the portion of the circuit applying two qubit identity gates, making the depth of Y $\chi'(G)$. Otherwise, the depth of Y is $\chi'(G) + 1$. □

We have now bounded the depth of Y in terms of the edge chromatic number of G, but we can push this further. Denote the maximum degree of the graph G by $\Delta(G)$ and note that $\Delta(G) \leq |V| - 1 = n - 1$ where n is the number of qubits in our quantum computer. A theorem due to Vizing [23] states $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$. This brings us to the following corollary, which can be expressed independent of the graph G.

Corollary 14. If the gates in S act on at most $N = 2$ qubits, the depth of Y is at most $\Delta(G) + 2 \leq n + 1$ where n is the number of qubits.

4 Virtual Gates

A qubit can be interpreted non-uniquely as a point on a unit sphere and single-qubit unitaries as maps from the unit sphere to itself. Single-qubit unitary transformations can therefore be decomposed as rotations around the x, y, and z axes of the sphere. These rotations are denoted X_θ, Y_θ, and Z_θ respectively where θ denotes the angle of rotation about that axis.

Note to transform one point on the sphere to another, we only need to rotate about two of the axes of the sphere. In fact, we can write any single qubit unitary as

$$\begin{aligned} U(\theta,\phi,\lambda) &:= Z_\theta X_\phi Z_\lambda \\ &= Z_{\phi-\pi/2} X_{\pi/2} Z_{\pi-\theta} X_{\pi/2} Z_{\lambda-\pi/2} \end{aligned} \tag{2}$$

for some choice of angle θ, ϕ, and λ. This means that the gate set $\{X_{\pi/2},\, Z_\theta\}_{\theta\in[0,2\pi)}$ is universal for single-qubit gates [18].

We discuss how such a single qubit unitary $U(\theta,\phi,\lambda)$ is applied on IBM's superconducting quantum computer. On IBM's superconducting quantum computer, gates are applied to qubits using pulses that consist of an amplitude $\Omega(t)$, a frequency ω, and a phase γ. The pulse is a microwave generated by an arbitrary waveform generator (AWG), a constant amplitude microwave generator, and an IQ mixer [25]. The collection of these machines allows one to create an arbitrary microwave of the form $\Omega(t)\cos(\omega t-\gamma)$. The AWG is the component that is classically programmable and used for shaping the wave [18]. Assuming a constant amplitude pulse Ω for duration T, the unitary applied by the pulse is

$$A(\gamma,\Omega,T) := e^{-i\frac{\Omega T}{2}(\cos(\gamma)X_\pi+\sin(\gamma)Y_\pi)}.$$

The unitary $A(\gamma,\Omega,T)$ is a rotation around some axis. The key point regarding the virtual gate is that adjusting γ rotates the axis of rotation of A about the z-axis. For example, if $\gamma=0$, the unitary $A(0,\Omega,T)$ is rotation by ΩT around the x axis; likewise if $\gamma=\pi/2$, the unitary $A(\pi/2,\Omega,T)$ is rotation by ΩT around the y axis [18]. This means the unitary $U(\theta,\phi,\lambda)=Z_\theta X_\phi Z_\lambda$ can be applied by adjusting the phase for the pulse of X_ϕ by θ and adjusting the phase of all future gates by λ. Since measurements yield the classical bits 0 or 1 with a probability given by a function of the position along the z axis, a rotation about the z axis of the qubit has no effect on the measurement probabilities.

The physical gate set employed by IBM is composed of four gates: $Z_\theta, X_{\pi/2}$, X and CNOT [15]. This gate set was chosen due to the efficiency of using virtual gates which are given by Z_θ for some angle θ. To minimize error, sending as few pulses as possible to the qubits is ideal. In this paper, we demonstrate the two-fold usefulness of these virtual gates: they not only provide an efficient implementation of gates but also offer resistance to side-channel attacks through our process of *virtual gate masking*. In IBM's universal gate set, only $X_{\pi/2}, X$, and CNOT are able to be detected using side-channel attacks on the pulses of the quantum computer, unless one is able to externally detect the phase of a pulse sent to a qubit. There is certainly a possibility that side-channel attacks with direct access to the AWG or the classical controller could detect the use of a Z_θ gate, as well as θ. In this context, virtual gate masking has the effect of concentrating the side-channel vulnerabilities of the quantum computer to classical devices, which is useful as there is already plenty of literature on protecting against classical side-channel attacks [21, 16].

For example, the researchers in [25] are unable to detect the application of virtual gates using their strongest attempt at a power-based side-channel attack, motivating the following definition.

Definition 15. The *non-virtual information* of a quantum circuit C is the $\{Z_\theta\}_{\theta\in[0,2\pi)}$-absent information in C.

We show that this limitation has significant implications for attempts at circuit reconstruction and identification. The threat model in [25] only takes power-based measurements of the drive and control channels of the quantum computer, which is why virtual gates are undetectable. Their attack could be significantly strengthened by adding probes to the AWG or the classical controller.

4.1 Virtual gates as covering sets

Let us fix the universal gate set to be $S = \{Z_\theta, X_{\pi/2}, X, CNOT\}$ and $R = \{Z_\theta\}_{\theta\in[0,2\pi)}$. We now prove that R is a covering set when $N = 2$ is the maximum size of a gate that can be applied.

Note that each single-qubit gate can already be decomposed using Equation (2), which gives us our single-qubit sequence for free. Specifically, we have $r_1 = 3$ and $S_1 = X_{\pi/2}$, $S_2 = X_{\pi/2}$, and $S_3 = I$.

For two-qubit gates, it was proven in [22] that any two-qubit unitary can be decomposed into eight single-qubit gates and three CNOT gates as shown in Figure 8. Since $(\theta_i, \phi_i, \lambda_i)$ parameterize

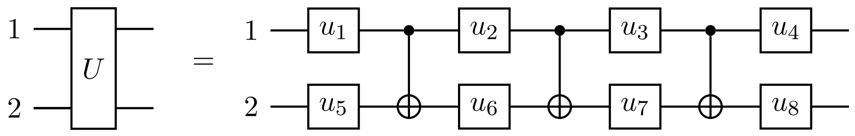

Figure 8: Decomposition of a two-qubit gate into eight single-qubit gates and three CNOT gates.

u_i using Equation (2), we have that $r_2 = 12$ and

$$\begin{aligned} S_1 &= X_{\pi/2} \otimes X_{\pi/2} & S_7 &= X_{\pi/2} \otimes X_{\pi/2} \\ S_2 &= X_{\pi/2} \otimes X_{\pi/2} & S_8 &= X_{\pi/2} \otimes X_{\pi/2} \\ S_3 &= \text{CNOT} & S_9 &= \text{CNOT} \\ S_4 &= X_{\pi/2} \otimes X_{\pi/2} & S_{10} &= X_{\pi/2} \otimes X_{\pi/2} \\ S_5 &= X_{\pi/2} \otimes X_{\pi/2} & S_{12} &= X_{\pi/2} \otimes X_{\pi/2} \\ S_6 &= \text{CNOT} & S_{13} &= I \otimes I. \end{aligned}$$

4.2 Virtual Gate Masking

In this section we discuss implementing our masking procedure using virtual gates on IBM's quantum computers.

Definition 16. We call R-gate masking *virtual gate masking* when R is the set of virtual gates $\{Z_\theta\}_{\theta\in[0,2\pi)}$.

In Figure 9, we discuss IBM's Qiskit standard transpiler [14] and our modification to it to protect against quantum side-channel attacks. The edit we make in both our masking processes is to pass 5, 'Translate to Basis Gates,' which is the step that decomposes arbitrary gates into gates in S. Thanks to pass 2, we can assume that each gate is at most size two in the stage, meaning that $N = 2$ is the maximum gate size in the circuits we are considering during our

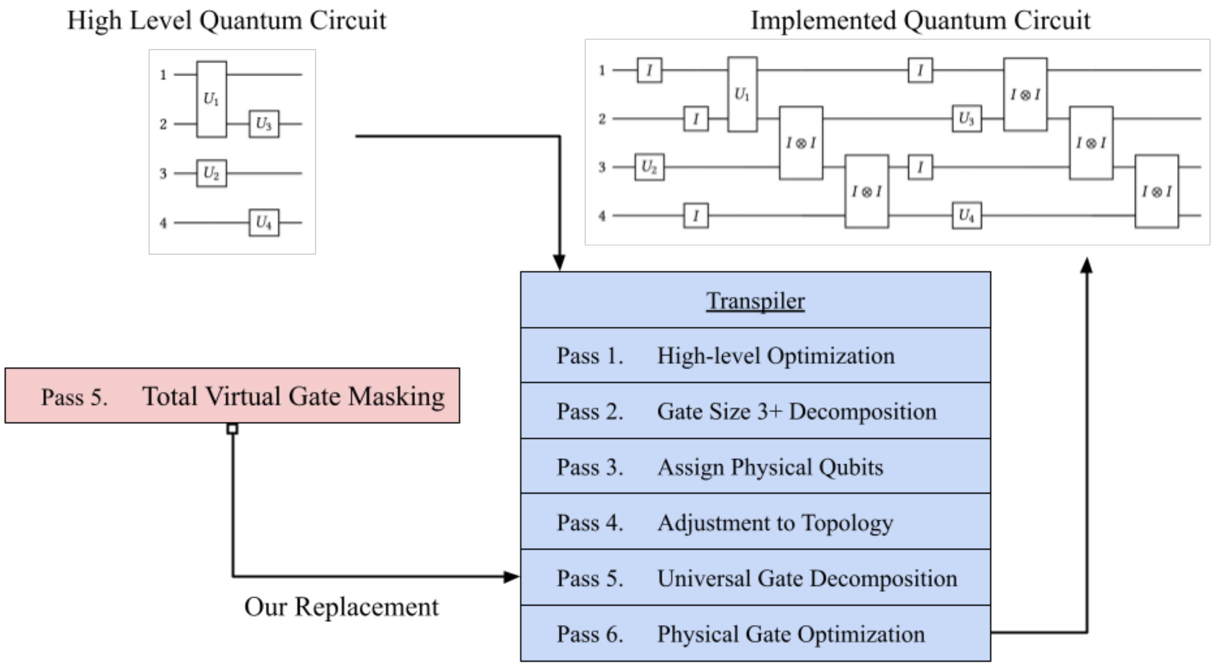
High Level Quantum Circuit
Implemented Quantum Circuit
1
2
3
4
U_1
U_2
U_3
U_4
I
$I \otimes I$
Transpiler
Pass 1. High-level Optimization
Pass 2. Gate Size 3+ Decomposition
Pass 3. Assign Physical Qubits
Pass 4. Adjustment to Topology
Pass 5. Universal Gate Decomposition
Pass 6. Physical Gate Optimization
Pass 5. Total Virtual Gate Masking
Our Replacement

Figure 9: Our modification to IBM's Standard Transpiler on Qiskit.

masking process. The preprocessing needed for virtual gate masking has been done in the previous subsection. Pass 5, 'Translate to Basis Gates,' will be modified so that size 1 gates are decomposed using Equation (2) and size 2 gates are decomposed using the equivalence depicted in Figure 8, with the single-qubit gates u_i still being decomposed with Equation (2). Note that when a size 1 or 2 gate is applied, an attacker that cannot see the identification of a virtual gate will see the sequence $(A, X_{\pi/2}, A, X_{\pi/2}, A, I)$ or $(A \otimes A, X_{\pi/2} \otimes X_{\pi/2}, A \otimes A, X_{\pi/2} \otimes X_{\pi/2}, A \otimes A, A \otimes A, \text{CNOT}, \dots)$ for some gates A with unknown identification whenever a gate is applied.

Note that virtual gate masking always translates a single-qubit gate into 6 gates in S, three of which are virtual. One can remove $S_{r_1} = I$ to make this 5 instead. A single-qubit gate is always implemented with 5 gates during virtual gate masking, so the extra depth added to the circuit is increased by at most 4 gates for every single-qubit gate. Similarly, a two-qubit gate is always implemented with a depth of 24 which can be made into 23 be removing $S_{r_2} = I \otimes I$. This means the extra depth added to the circuit is at most 23 for every two-qubit gate.

We need to do additional preprocessing for total virtual gate masking. In particular, we need to create the circuit Y from Section 3.2 dependent on the topology of our quantum computer. When modifying the transpiler for total virtual gate masking, we add the additional modification to pass 5 outlined in Procedure 9. Note in total virtual gate masking, a gate on each wire is masked for every time step in the original circuit so that an attacker blind to the virtual gate identifications does not know which wire label has a gate that is not the identity.

The overhead incurred by total virtual gate masking is discussed in Section 3.3. Note that the algorithm that is output in Figure 9 is not the same as the one in Figure 7. This is because pass 6 of the transpiler may make some additional modifications to our circuit. For example, in Figure 9 it is assumed that two gates are not applied in the same time step if any of the qubits the gates act on are adjacent to each other in the computer's topology.

5 Example: Class Group Computations

Our technique of virtual gate masking could find application in a scenario in which both the output of the quantum algorithm along with the algorithm itself must be protected from an adversary. For example, a corporation booking time on a quantum computer in order to develop new fertilizers, pharmaceuticals, or solar panels might view both the product and the recipe—the quantum algorithm itself—as proprietary information. As another example, in a "post-post-quantum" world when a large-scale quantum computer exists but is perhaps not widely available, a quantum computer might be useful for generating parameters for some cryptographic protocols. In both scenarios, it is reasonable that some part of the quantum algorithm is known or safe to reveal but other parts must be secured. In this case, a modified transpilation process can have significantly less overhead. We show this with the example in this section.

We examine a specific example where the quantum algorithm computes the relation lattice of the ideal class group $\mathrm{cl}(-D)$ for an imaginary quadratic field $\mathbb{Q}(\sqrt{-D})$, considering squarefree, positive integer D. We demonstrate how to protect sensitive parts of the algorithm from side-channel attacks with little overhead. Quantum computers offer a near exponential acceleration to algorithms for computing the class group, while a cryptographic application would necessitate side-channel resistance. We show that one can mask critical parts of the quantum algorithm with small overhead.

Ideal class groups of imaginary quadratic fields have been recently proposed as "groups of unknown order" [24, 4, 10], which have numerous cryptographic applications, such as time-lock puzzles, (verifiable) delay functions, and cryptographic accumulators. A closely related notion is the *trapdoor* group of unknown order—essentially a group whose order is known only to one person. These naturally lead to *trapdoor* delay, i.e., functions that require a prescribed number of sequential steps to evaluate, unless a secret piece of information is known. In the case of the ideal class group $\mathrm{cl}(-D)$ with a fixed generating set $g_1, g_2, \ldots, g_n$, the trapdoor is the *relation lattice*: a group K given by

$$K = \{\vec{x} \in (\mathbb{Z}/q\mathbb{Z})^n : g_1^{x_1} \cdots g_n^{x_n} = 1\}.$$

This group K satisfies $\mathrm{cl}(-D) \cong \mathbb{Z}_q^n/K$, and knowledge of its structure allows a user to easily compute reduced representations of elements of the form

$$g_1^{z_1} g_2^{z_2} \cdots g_n^{z_n}$$

even for doubly-exponentially-large integers $z_1, z_2, \ldots, z_n$.

While there is no known efficient classical algorithm to compute the relation lattice, any finite abelian group's relation lattice can be computed using the quantum abelian hidden subgroup solver [8], depicted in Figure 10.

This class group computation algorithm could be used in a near-future scenario, where NISQ quantum computers are available only through cloud providers, with substantial lead time. However, for trapdoor delay functions instantiated in this way to be secure against side-channel attacks, we must be able to mask the sensitive input to the quantum algorithm: the integer D (and derived information, like $|\mathrm{cl}(-D)|$ and the relation lattice).

For the purposes of our analysis, we assume that the quantum computer is supplied through a trusted cloud service. The attacker has the ability to perform various side-channel attacks on the

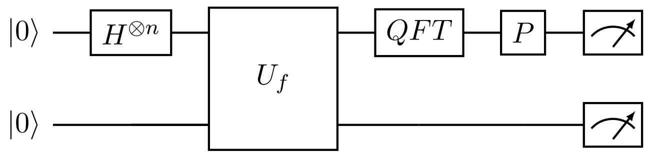

Figure 10: The abelian hidden subgroup solver of [8]. In this algorithm, some amount of $|0\rangle$ qubits are fed into the circuit, and the final operation is a standard measurement. The specifics of the gates are not that important for this discussion, but we mention that U_f is a gate that depends on the function $f(x_1, \ldots, x_n) = g_1^{x_1} \cdots g_n^{x_n}$, and that f hides the relation lattice K.

quantum computer running our algorithm, and they have access to their own quantum computer. However, we can securely send the algorithm we want to run, specific to the *discriminant* D, to the cloud-based vendor.

The class group of an imaginary quadratic field $\mathrm{cl}(-D)$ is isomorphic to the set of reduced binary quadratic forms where the operation is Gauss composition. We can uniquely represent an element in $\mathrm{cl}(-D)$ by a triple $(a, b, c) \in \mathbb{Z}^3$ such that $\gcd(a, b, c) = 1$, $b^2 - 4ac < 0$, $a > 0$ and either ($-a < b < a \le c$ or $0 \le b \le a = c$). There are efficient $O(\mathrm{poly}\log|\mathrm{cl}(-D)|)$ time classical algorithms for finding the generators of $\mathrm{cl}(-D)$, $g_1, \ldots, g_n$, and q the maximum order of a generator assuming the generalized Riemann hypothesis [11]. There is also an efficient classical algorithm for performing Gauss composition [9]. We discuss this algorithm, and two required subroutines, next, to highlight how they depend on D.

```
Algorithm 1: Norm
   Input  : A cl(−D) representative $(a, b, c) \in \mathbb{Z}^3$.
   Output: Another representative of the same element $(a', b', c')$.
 1 $\ell \leftarrow \lfloor \frac{a-b}{2a} \rfloor$
 2 $(a', b', c') \leftarrow (a, 2a\ell + b, a\ell^2 + b\ell + c)$
 3 return $(a', b', c')$
```

```
Algorithm 2: Reduction
   Input  : A cl(−D) representative $(a, b, c) \in \mathbb{Z}^3$.
   Output: The unique reduced representative of $(a, b, c)$ in cl(−D).
 1 $(a, b, c) \leftarrow$Norm$(a, b, c)$
 2 if $a < \frac{\sqrt{D}}{2}$ then
 3     return $(a, b, c)$
 4 end
 5 else if $a < \sqrt{D}$ and ($a \ge c$ or ($b \ge 0$ and $c \ne 0$)) then
 6     return Norm$(c, -b, a)$
 7 end
 8 else
 9     return Reduction$(c, -b, a)$
10 end
```

Algorithm 3: Gauss composition

```
    Input  : Two $\mathrm{cl}(-D)$ reduced representatives $(a_1, b_1, c_1), (a_2, b_2, c_2) \in \mathbb{Z}^3$.
    Output: $(a_3, b_3, c_3)$ a reduced representative of $(a_1, b_1, c_1) \cdot (a_2, b_2, c_2)$.
1   $s \leftarrow \frac{b_1+b_2}{2}$
2   $u', v', d' \leftarrow \gcd(a_1, a_2)$
3   $m, w, d \leftarrow \gcd(d', s)$
4   $u, v \leftarrow u'm, v'm$
5   $a_3 \leftarrow \frac{a_1 a_2}{d^2}$
6   $b_3 \leftarrow b_2 + \frac{2a_2}{d}(v(s - b_2) - wc_2)$
7   $c_3 \leftarrow \frac{b_3^2 - D}{4a_3}$
8   return Reduction$(a_3, b_3, c_3)$
```

Remark. The only gate in the circuit that contains information about $\mathrm{cl}(-D)$, or even D, is U_f. Note that f is a repeated application of Algorithm 3, possibly with a square and multiply subroutine. We notice D is encoded directly in the algorithm for Gauss composition as seen in lines 2 and 4 in Algorithm 2 and line 7 of Algorithm 3. By tracing through the algorithm, the circuit level implementation of U_f will need to consist of quantum-quantum adders (QQA), multipliers (QQM) and dividers (QQM); classical-quantum adders (CQA), multipliers (CQM) and dividers (CQD); and controlled versions of these gates.

If total virtual gate masking is applied to the entire algorithm we know that a side-channel attack attempting to glean non-virtual gate positional information, like a per-channel power trace attack, could not possibly learn information about $\mathrm{cl}(-D)$. However, for this particular cryptographic application, we do not need to mask every gate in the quantum circuit, only the gates that contain information about $\mathrm{cl}(-D)$ or D. We now discuss exactly which parts of the algorithm needs masking and demonstrate how to do it on this example. This serves as a template and could easily be used on other examples as well and illustrates the simplicity and robustness of our masking algorithm.

When we implement line 7 of algorithm 3, inside U_f we need to implement subtraction by D. This can be achieved by the circuit depicted in Figure 11.

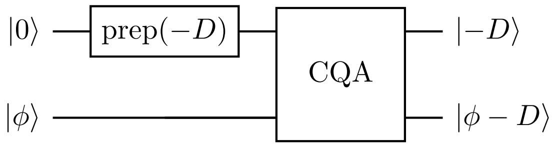

Figure 11: Implementation of subtraction of D used in Algorithm 3. Here, $|\phi\rangle$ is an arbitrary set of qubits and prep$(-D)$ satisfies prep$(-D)\,|0\rangle = |-D\rangle$ where $|-D\rangle$ is $-D$ written out in bits, encoded as a qubit.

Using IBM's gate set, one can implement prep$(-D)$ in one time step by looking at the binary decomposition of $-D = \sum_{i=0}^{n} B(i)2^n$ and using an $X = X_\pi$ gate on each wire i where $B(i) = 1$ and an I_2 gate elsewhere. Using Equation (2) we write

$$X = Z_{3\pi/2} X_{\pi/2} Z_0 X_{\pi/2} Z_{3\pi/2}$$
$$I_2 = Z_{3\pi/2} X_{\pi/2} Z_\pi X_{\pi/2} Z_{3\pi/2}$$

and implement them as such, meaning a per-channel power trace attacker will not be able to distinguish between the use of an X gate and an I_2 gate in prep$(-D)$. This implementation of prep$(-D)$ requires 4 more time steps than the naïve implementation, and up to $4n$ more gates total. However to an attacker that can only measure the power consumption caused from pulses sent to qubits, prep$(-D)$ will look identical for any $D \in \mathbb{Z}$, as desired.

The only other place information of cl$(-D)$ is contained is in lines 2 and 4 of Algorithm 2.

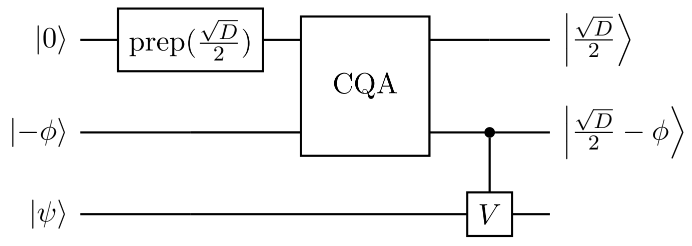
$|0\rangle$
prep$(\frac{\sqrt{D}}{2})$
CQA
$\left|\frac{\sqrt{D}}{2}\right\rangle$
$|-\phi\rangle$
$\left|\frac{\sqrt{D}}{2} - \phi\right\rangle$
$|\psi\rangle$
V

Figure 12: Quantum implementation of the If statement from Algorithm 2. Here, $|-\phi\rangle$ and $|\psi\rangle$ are qubits evolved from previous steps in the algorithm and the CQA and V gates are independent of D.

In the circuit of Figure 12, we implement our if-statement in line 2 of Algorithm 2, $a < \frac{\sqrt{D}}{2}$, by checking if the qubits representing the variable a, namely $|\phi\rangle$, satisfies $\left|\frac{\sqrt{D}}{2} - \phi\right\rangle \neq 0$. The gate V represents the operations that should be applied to another state $|\psi\rangle$ given our if-statement has passed. The circuit implementation of line 4 of algorithm 2 is nearly identical. Notice again that the only gate that contains information about D is the prep$(\frac{\sqrt{D}}{2})$ gate, which we have already shown can be shielded from attacks attempting to obtain non-virtual gate information.

6 Conclusions and Future Work

Total virtual gate masking provides us with information-theoretic protection against any side-channel attack on a quantum computer, assuming it is possible to hide virtual gate angles from the attacker. Virtual gates are programmed and executed in a classical controller, affecting computation only outside of the classical controller—internally inside the arbitrary waveform generator (AWG) and in the phase of microwave pulses sent to qubits. If one is willing to call the AWG classical and assume that the phase of microwave pulses are physically undetectable, then we effectively reduce side-channel attacks on quantum computers to side-channel attacks on the classical components of the quantum computer. Utilizing total virtual gate masking offloads the side-channel security of the quantum computer to just a few classical components, enabling us to employ previously well-studied classical methods to protect against side-channel attacks [21, 16]. Moreover, our procedure could easily be implemented on most current gate-based quantum computing vendors. We explicitly outline how to do so in Qiskit [14] in Section 4.2.

At the start of our analysis, we assumed that our quantum computer has an "isolated quantum device" that the side-channel attacker is unable to tamper with, because any measurement of the evolving quantum state would introduce error into the quantum computation rendering the computer and the attack useless. That being said, one can envision a very aggressive threat model where an adversary is able to make partial measurements to the quantum state, introducing only small amounts of error into the computation undetected. This is a more typical threat model, where a technical analysis of the trade-off between information gained by the attacker and error introduced to the computation. For example, the attacker could attempt to read side-channel information from the the measurement channels of the quantum computer. To be clear, this model is more about learning the state or outcome of the computation, and less about the evolution of the computation, i.e., the algorithm itself. It would be interesting to see if these distinct threat models can be combined to access information previously thought secure.

Acknowledgements

This research was funded in part by the Commonwealth of Virginia's Commonwealth Cyber Initiative (CCI), an investment in the advancement of cyber R&D, innovation, and workforce development. For more information about CCI, visit `www.cyberinitiative.org`.

References

[1] Amazon. Amazon Web Services, Amazon Braket, 2023. `https://aws.amazon.com/braket/`.

[2] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. Noisy intermediate-scale quantum algorithms. *Rev. Mod. Phys.*, 94:015004, Feb 2022.

[3] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, page 893–902, USA, 2016. Society for Industrial and Applied Mathematics.

[4] Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Paper 2018/712, 2018. `https://eprint.iacr.org/2018/712`.

[5] Sergey Bravyi, Oliver Dial, Jay M. Gambetta, Darío Gil, and Zaira Nazario. The future of quantum computing with superconducting qubits. *Journal of Applied Physics*, 132(16):160902, 10 2022.

[6] Hans J Briegel, David E Browne, Wolfgang Dür, Robert Raussendorf, and Maarten Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5(1):19–26, 2009.

[7] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pages 517–526, Atlanta, Georgia, USA, oct 2009. IEEE.

[8] Kevin K. H. Cheung and Michele Mosca. Decomposing finite abelian groups, 2001.

[9] Henri Cohen. *A course in computational algebraic number theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[10] Samuel Dobson, Steven Galbraith, and Benjamin Smith. Trustless unknown-order groups. *Mathematical Cryptology*, 1(2):25–39, 2022.

[11] Sean Hallgren. Quantum algorithms for class group of a number field. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 694–696. Springer US, Boston, MA, 2008.

[12] IBM. IBM Quantum, 2023. `https://quantum-computing.ibm.com/`.

[13] IBM. Qiskit, 2024. `https://qiskit.org/`.

[14] IBM Quantum. Qiskit's transpiler documentation, 2024. `https://docs.quantum.ibm.com/api/qiskit/transpiler#transpiler`.

[15] Abhijith J., Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Daniel O'malley, Diane Oyen, Scott Pakin, Lakshman Prasad, Randy Roberts, Phillip Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter J. Swart, James G. Wendelberger, Boram Yoon, Richard Zamora, Wei Zhu, Stephan Eidenbenz, Andreas Bärtschi, Patrick J. Coles, Marc Vuffray, and Andrey Y. Lokhov. Quantum algorithm implementations for beginners. *ACM Transactions on Quantum Computing*, 3(4), jul 2022.

[16] Xiaoxuan Lou, Tianwei Zhang, Jun Jiang, and Yinqian Zhang. A survey of microarchitectural side-channel vulnerabilities, attacks, and defenses in cryptography. *ACM Computing Surveys (CSUR)*, 54(6):1–37, 2021.

[17] Thomas Lubinski, Sonika Johri, Paul Varosy, Jeremiah Coleman, Luning Zhao, Jason Necaise, Charles H. Baldwin, Karl Mayer, and Timothy Proctor. Application-oriented performance benchmarks for quantum computing. *IEEE Transactions on Quantum Engineering*, 4:1–32, 2023.

[18] David C. McKay, Christopher J. Wood, Sarah Sheldon, Jerry M. Chow, and Jay M. Gambetta. Efficient Z gates for quantum computing. *Physical Review A*, 96(2), aug 2017.

[19] Microsoft. Microsoft Azure Quantum, 2023. `https://azure.microsoft.com/en-us/resources/development-kit/quantum-computing`.

[20] Mario Motta and Julia E. Rice. Emerging quantum computing algorithms for quantum chemistry. *WIREs Computational Molecular Science*, 12(3):e1580, 2022.

[21] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Computing Surveys*, 55(11):1–35, 2023.

[22] G. Vidal and C. Dawson. A universal quantum circuit for two-qubit transformations with three cnot gates. *Physical Review A*, 69, 01 2004.

[23] Vadim G Vizing. Critical graphs with given chromatic class (in russian). *Metody Discret. Analiz.*, 5:9–17, 1965.

[24] Benjamin Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33(4):2113–2147, oct 2020.

[25] Chuanqi Xu, Ferhat Erata, and Jakub Szefer. Exploration of power side-channel vulnerabilities in quantum computer controllers. In *CCS '23: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 579–593, Copenhagen, Denmark, 2023. Association for Computing Machinery.