Article

Quantum Logic Locking for Security

Rasit Onur Topaloglu [1,†,‡]

[1] IBM, Poughkeepsie, NY; rasit@us.ibm.com

Abstract: Having access to a unique quantum circuit that one wants to protect against use by others is a very likely scenario. However currently users rely on classical computer security schemes, which have known shortcomings. In this chapter we introduce a novel protection scheme along with a survey of other known methods to protect quantum information. In particular we review physically unclonable functions (PUFs), obfuscation, and introduce quantum logic locking.

Keywords: quantum computing, quantum circuit, qubit, quantum security, quantum logic locking, QLL, logic locking, hardware security

1. Introduction

1.1. Quantum Circuits

We assume that reader is already familiar with basic quantum computing information. However one can also refer to [1] for a brief review.

Figure 1 is a quantum circuit that implements Grover's Algorithm from [2], which is a search function. It finds the input that would result in a particular output for a black box function that a user could evaluate. As summarized in [3], this algorithm provides a solution to an NP-complete problem in processing time that is correlated to square root of time. For comparison, a classical algorithm would be correlated to time directly. Here we will not focus on how this algorithm works, but rather use this circuit to explain how quantum circuits operate in general.

There can be more than one way to implement a quantum circuit, using different types of gates for example. For our case, we use the Hadamard and Pauli-Z gate implementation as proposed in [4].

A quantum circuit is how a quantum algorithm can be implemented in hardware. In Figure 1, each line represents a qubit. Processing on a qubit starts from the left side of this line. Gate operations apply on qubits as we move towards the right side. Although not shown explicitly, gates are applied at specific time intervals. The time step is determined per quantum hardware and represents time it would take to apply slowest gate operation plus some buffer time. The first set of Hadamard gates are applied to three qubits in the same time interval. In the second time interval, no operation is applied to the second qubit so its information does not change. Meanwhile, first qubit controls a Pauli-Z gate on the third qubit. Hence first and third qubits end up being entangled. In the third time interval, first qubit proceeds while second qubit controls a Pauli-Z gate on the third qubit. After intervals of Hadamard and Pauli-X, first and second qubits control a Pauli-Z gate on the third qubit. After Pauli-X and Hadamard applications, finally output is measured at the far most right end of each qubit line. The measurement takes us back to classical domain.

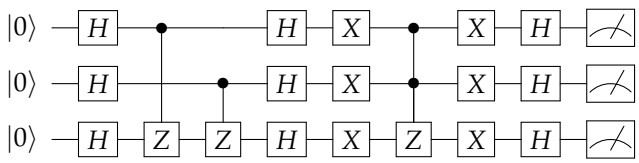

Figure 1. Quantum circuit implementing Grover's algorithm.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2022 doi:10.20944/preprints202212.0130.v1

2. Quantum Physically Unclonable Functions (QPUFs)

Most quantum computers are hosted on the cloud. This means that accessing a quantum computer happens through cloud computing and currently only few companies may have in-house quantum computers. Thus it is difficult to ensure your quantum circuit is running on a particular quantum computer. For example there may be multiple quantum computers served by the same “quantum cloud" provider. Providers usually specify exact quantum computer used these days, but it is feasible that at some point a random one may be assigned suiting certain specifications. Against this potential concern, a quantum PUF (QPUF) can be utilized. The QPUF can be implemented as part of a quantum circuit or as a separate one to be run right before another quantum circuit. Running a QPUF circuit generates a unique signature for an assigned quantum computer that cannot be replicated by another quantum computer. Such uniqueness comes from the errors inherent in qubits, gates, and measurement circuitry for a particular quantum computer.

[5] provides a practical implementation of a quantum PUF as shown in Figure 2. Readers could also refer to [6] for a theoretical treatment of the topic. [7] extends the idea in [5] to have multiple cycles of rotation gates instead of a single cycle as shown in Figure 3. [8] presents yet another implementation via a regular and entangled configuration utilizing X, Z, and CZ gates as shown in Figure 4.

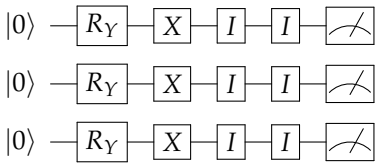

Figure 2. QPUF Implementation from [5]. There can be multiple I stages; only two shown.

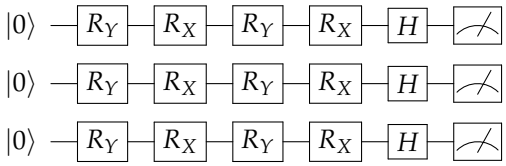

Figure 3. QPUF implementation [7] uses multiple R_Y and R_X stages with different rotation degrees; only two shown.

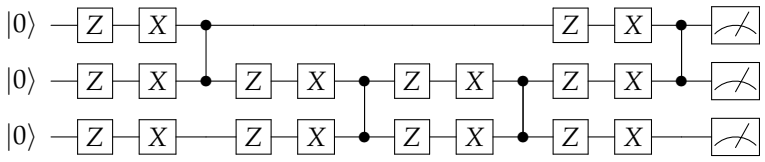

Figure 4. Another QPUF Implementation [8] uses multiple Z, X, and CZ stages ; only three qubit version shown.

3. Quantum Obfuscation

Another potential concern could be that a unique quantum circuit may be stolen by an adversary. Obfuscation can be a solution against this concern. [9] defines obfuscation simply as encryption of functionality. In that work, it is shown that a quantum obfuscator cannot obfuscate a classical circuit. Herein, we focus on quantum obfuscating a quantum circuit. To enable the obfuscation, additional quantum gates can be inserted to obfuscate the original design. Insertion of additional gates corresponds to encrypting the circuit. Then when the quantum circuit is actually run, the newly added gates are removed. This step is decryption. The adversary does not know which gates to remove to obtain the correct circuit even when they have access to the encrypted circuit, and thus would waste exponential time to remove gates by trial and error to identify the correct circuit. In addition

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2022 doi:10.20944/preprints202212.0130.v1

to time lost by such trial and error, the adversary would have exponentially less chances of figuring out which result is correct as only one would be the correct one out of the many an adversary may try. Figuring out if a given output is correct may not be an easy choice.

[10] proposes one such quantum obfuscation example. Here CNOT gates are inserted strategically to ensure that the output of the circuit would be different than what is intended. CNOT insertion is the encryption step. The user or the quantum computer keeps track of where the CNOT gates are inserted and removes them before execution. Hence the input/output relationship is kept intact while obfuscated circuit cannot be directly used. An example of this obfuscation method is shown in Figure 5.

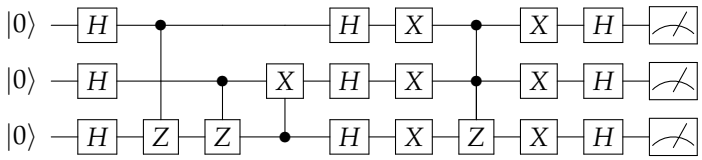

Figure 5. Obfuscated version of Grover's algorithm. CNOT location decided to ensure output is different than original.

4. Quantum Logic Locking

Logic locking has been only proposed for single flux quantum (SFQ) type of circuits so far. These quantum circuits implement classical digital electronic circuits but use superconducting devices. Hence they are not considered universal quantum computers. Thus logic locking for SFQ devices cannot target qubits which rely on transmon, trap-based, or spin implementations. For quantum circuits that rely on aforementioned superconducting qubits, we introduce herein novelties such as new ancillas, replacement of control gates, and insertion of dummy control gates.

4.1. Security Problem

Quantum circuit development for certain applications is sometimes a very difficult step requiring deep theoretical knowledge. For example, a novel quantum algorithm can be invented that targets a specific problem. One may then want to keep the quantum circuit implementation of such algorithm a secret. However an adversary may gain access to it and start using it elsewhere despite the presence of state of the art software security techniques. Having access to the results of such stolen quantum circuit gives the adversary knowing what the circuit does undeserved benefit and causes significant monetary damages to the original owner (defender). Quantum logic locking as proposed herein has the potential to provide a remedy for this.

5. Steps of Quantum Logic Locking

We provide a method where in addition to having access to the circuit, a defender or the quantum computer stores a secret key separately without which the results of such quantum circuit become useless. The secret key in particular controls a set of special added ancilla qubits that we name "lock ancilla qubits".

A user first determines the desired security complexity level for a given quantum application/circuit. Then the user inserts required number of lock ancilla qubits for desired security complexity level. Following this step, we replace select Pauli-X, Y, Z, H, Rotation, or Phase-Rotation gates with controlled versions where control is one of the lock ancillas with value $|1\rangle$. This circuit change is summarized in Figure 6.

Notice that controlled versions of certain gates may not be readily available. For example, a controlled version of Z or Hadamard may be lacking. However we can use a transformation such as in Figure 7 or 8 to enable them.

Next, we can also insert dummy control gates to select active qubit lines where control line is one of the lock ancillas with value $|0\rangle$ and controlled line is one of the active qubits. This is illustrated in Figure 9. Then we mix the order of lock ancilla qubits randomly

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2022 doi:10.20944/preprints202212.0130.v1

Figure 6. **(a)** Original gate in quantum circuit. **(b)** A controlled version of the same gate. Control signal determines whether or not the gate is applied.

Figure 7. If a native controlled version does not exist, a conversion is usually feasible. For example a controlled version of the Z gate in **(a)** can be implemented with Hadamard and CNOT gates as in **(b)**.

and store $|0\rangle$ and $|1\rangle$ order of them separately from the quantum circuit. During regular operation of circuit, we apply stored lock ancilla qubit signals, perhaps from a separate memory storage, before running the quantum circuit.

A compact representation of the quantum logic locking (QLL) scheme is shown in Figure10. Figure 11 illustrates how a logic locked version of the Grover circuit may look like.

6. Determining Desired Security Complexity Level

Depending on the security level, amount of lock ancillas can be adjusted. Each lock ancilla qubit increases security level exponentially. Adding only one lock ancilla qubit indicates quantum circuit should be executed by an adversary twice as many times as having no lock ancilla qubits. That is because adversary should first assume the lock ancilla has value $|0\rangle$ and in the second execution assume $|1\rangle$. Notice that for each choice, there can be thousands of executions on the quantum computers to average out noise inherent in the system.

10 lock ancilla qubit indicates quantum circuit should be executed by an adversary 2^{10} times the original amount! Depending on how much time a defender desires an adversary to waste, number of lock ancillas is increased. In addition to such execution amount increase, adversary also loses track of which solution is the correct one.

When replacing select gates with controlled versions, one should first determine which gates have a controlled version available in target quantum system. If a direct version is not available, alternative implementations can be considered as discussed previously. From

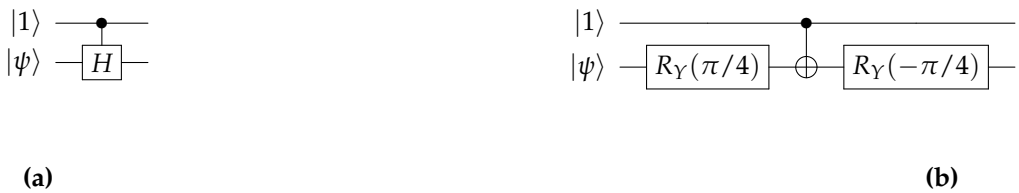

Figure 8. Even a controlled version of common Hadamard gate in **(a)** is feasible as shown in **(b)** using R_Y gates and a CNOT.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2022 doi:10.20944/preprints202212.0130.v1

Figure 9. A dummy control gate can also be inserted. By keeping the hidden key as $|0\rangle$, the dummy gate is not applied.

Figure 10. (a) Original circuit. **(b)** Secret key qubits added and some gates are replaced with controlled versions.

available ones, user selects one of each kind from various qubit lines until target security complexity level is met.

Inserting dummy controlled gates is handled by determining which qubit lines have more slack (empty time steps where no operation or idle operation takes place). After determining exact time slots available, we find out available controlled gates version in target quantum system. This can be done simply via a list comparison for example.

For each randomly picked available time slot, we insert one kind of the available controlled gate version until target security goal is met. A quantum read-only memory can be used to store secret keys and information on chip. A practical implementation is given in [11]. Alternatively keys can be stored in classical but separate storage vs. where the circuit is kept.

7. Experimental Results

We conduct our experiments on a real quantum computer. In particular we use ibmq_belem for our experiments in fairshare mode using 1024 shots. This is a 5 qubit quantum computer that supports a quantum volume of 16 with 2.5k circuit layer operations per second. At the time of execution, it displayed a T1 of 88.62μs, T2 of 91.89μs, CNOT error of 1.34e-2, readout error of 2.61e-2, frequency of 5.22GHz, and gate time of 539ns. We transpile the circuit into R_Z, CNOT, and square root NOT gates.

Figure 12 shows two runs, one with and one without the correct lock ancilla inputs applied. Results in histogram show that users would get a significantly different result by running the quantum circuit if they do not know the correct keys. For example incorrect keys would give '100' and '111' as output whereas correct output is '101' and '110.'

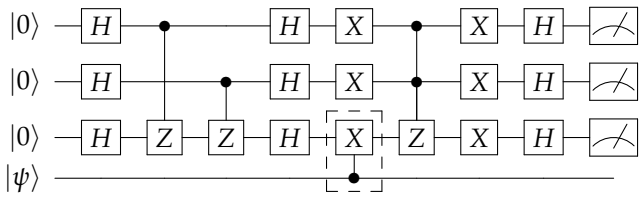

Figure 11. Logic Locked version of Grover's algorithm.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 December 2022 doi:10.20944/preprints202212.0130.v1

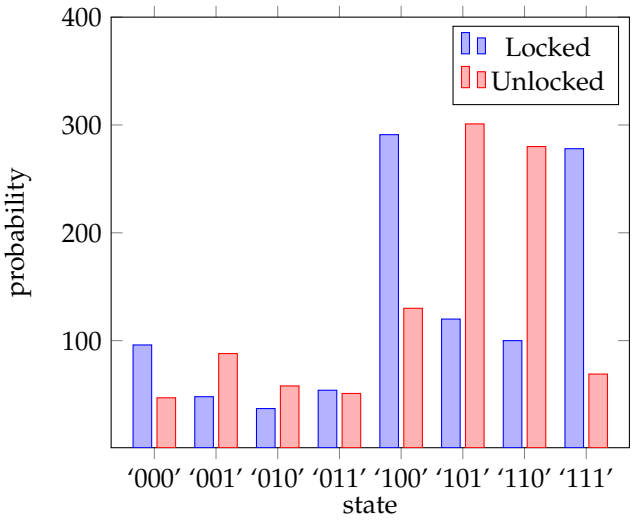

Figure 12. Experimental results. Unlocked results show the correct output from the circuit, whereas locked results show what happens if input keys are not applied.

8. Conclusion

We have reviewed existing methods in quantum hardware security, namely QPUFs and obfuscation. We have also introduced quantum logic locking for gates used in universal quantum computing. While QPUFs can identify a given quantum computer and obfuscation enables encryption of a given quantum circuit, quantum logic locking presents an alternative security measure by reserving only few additional qubits.

References

1. Topaloglu, R.O., Fast Introduction to Quantum Computers. In *Wiley Encyclopedia of Electrical and Electronics Engineering*; John Wiley Sons, Ltd, 2022; pp. 1–6, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/047134608X.W8442]. https://doi.org/https://doi.org/10.1002/047134608X.W8442.
2. Grover, L.K. A fast quantum mechanical algorithm for database search. In Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC), 1996, pp. 212–219.
3. Bennett, C.H.; Bernstein, E.; Brassard, G.; Vazirani, U. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing* **1997**, *26*, 1510–1523, [https://doi.org/10.1137/S0097539796300933]. https://doi.org/10.1137/S0097539796300933.
4. Figgatt, C.; Maslov, D.; Landsman, K.A.; Linke, N.M.; Debnath, S.; Monroe, C. Complete 3-Qubit Grover search on a programmable quantum computer. *Nature Communications* **2017**, *8*, 1918.
5. Phalak, K.; Saki, A.A.; Alam, M.; Topaloglu, R.O.; Ghosh, S. Quantum PUF for Security and Trust in Quantum Computing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **2021**, *11*, 333–342. https://doi.org/10.1109/JETCAS.2021.3077024.
6. Arapinis, M.; Delavar, M.; Doosti, M.; Kashefi, E. Quantum Physical Unclonable Functions: Possibilities and Impossibilities. *Quantum* **2021**, *5*, 475. https://doi.org/10.22331/q-2021-06-15-475.
7. Pirnay, N.; Pappa, A.; Seifert, J.P. Learning classical readout quantum PUFs based on single-qubit gates. *Quantum Machine Intelligence* **2022**, *4*. https://doi.org/10.1007/s42484-022-00073-1.
8. Kumar, N.; Mezher, R.; Kashefi, E. Efficient Construction of Quantum Physical Unclonable Functions with Unitary t-designs, 2021. https://doi.org/10.48550/ARXIV.2101.05692.
9. Alagic, G.; Fefferman, B. On Quantum Obfuscation, 2016. https://doi.org/10.48550/ARXIV.1602.01771.
10. Suresh, A.; Saki, A.A.; Alam, M.; Onur Topaloglu, R.; Ghosh, S. A Quantum Circuit Obfuscation Methodology for Security and Privacy. In Proceedings of the Workshop on Hardware and Architectural Support for Security and Privacy; Association for Computing Machinery: New York, NY, USA, 2022; HASP '21. https://doi.org/10.1145/3505253.3505260.
11. Phalak, K.; Alam, M.; Ash-Saki, A.; Topaloglu, R.O.; Ghosh, S. Optimization of Quantum Read-Only Memory Circuits, 2022. https://doi.org/10.48550/ARXIV.2204.03097.