SoteriaQ: Securing Quantum Circuits

Theodoros Trochatos, Sanjay Deshpande, Jakub Szefer

Yale University

*This work was supported in part by NSF grants 2312754 and 2245344.

CASLAB.io

Project Overview

We present a first-of-a-kind quantum computer trusted execution environment [1]. The cloud-based environments in which today's and future quantum computers will operate, raise concerns about the security and privacy of user's intellectual property. Quantum circuits submitted to cloud-based quantum computer providers represent sensitive or proprietary algorithms developed by users, and these circuits need protection. To help protect users' circuits and data from possibly malicious quantum computer cloud providers or insider attackers, this work presents SoteriaQ, the first hardware architecture for a trusted execution environment for quantum computers. To protect the user's circuits and data, the quantum computer control pulses are obfuscated with decoy pulses.

Threat Model

Our work considers the threat model of an honest-but-curious cloud provider. In the honest-but-curious scenario, the cloud provider could outright aim to spy on users to learn their intellectual property or data contained in the quantum circuits, or, if the cloud provider as a whole may not be malicious, one of the employees, the so called malicious insider, could try to spy on the operation of the quantum computer, as shown in Figure 1. In honest-but-curious the cloud providers are untrusted, they can spy on operation of the quantum computers, but do not modify them.

SoteriaQ Workflow

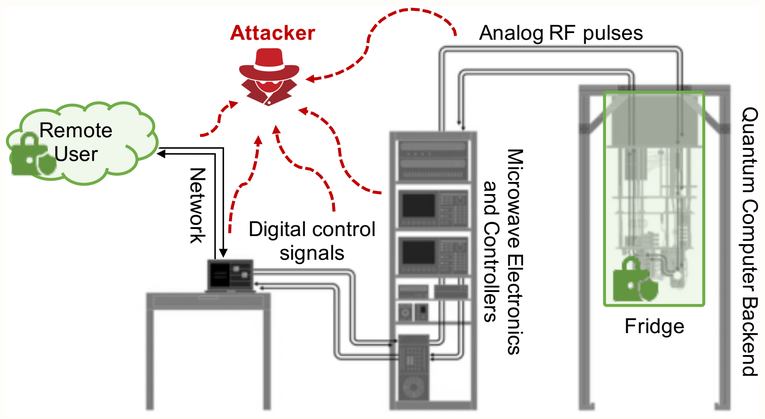

Figure 1: Schematic of a typical superconducting quantum computer, showing a cloud provider attempting to spy on the control pulses.

Figure 3 shows the workflow of cloud-based quantum computation after SoteriaQ architecture is integrated into it. The main principle of operation is to mix the control pulses, which define the user's input quantum circuit, with decoy pulses. In parallel, an (encrypted) input bitmap is generated to identify the location of the decoy control pulses within the transpiled circuit sent for execution. The input bitmap is encrypted with the public key of the backend, i.e. quantum computer, where the circuit will execute. It is also digitally signed by the user. The above steps are represented by the steps in Figure 3. A simplified illustration of addition, and later attenuation, of the decoy pulses in shown in Figure 2 (a) to (c) and an illustration example of the input bitmap is shown at the bottom of Figure 2 (b).

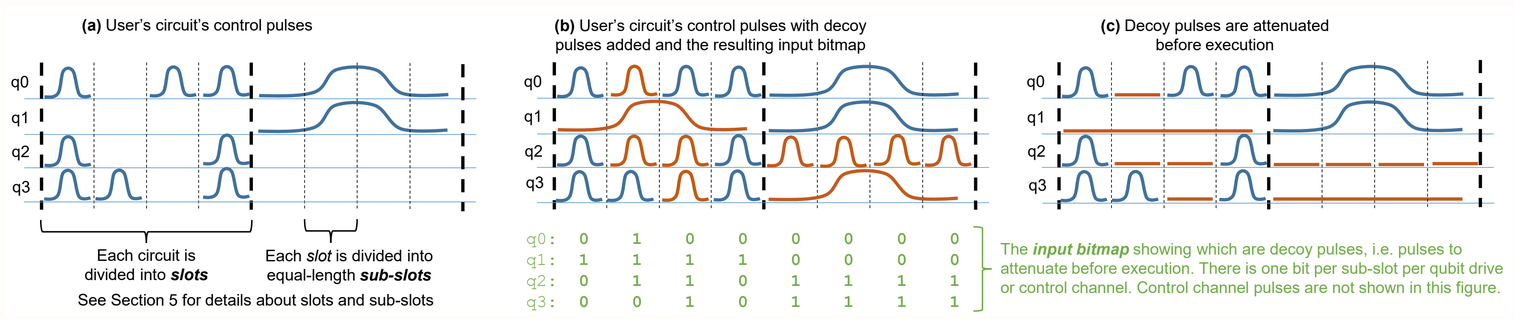

Figure 2: Example illustration of how control pulses are obfuscated. (a) Control pulses correspond to the input circuit of the user. (b) Control pulses with decoy pulses added, along with the input bitmap. (c) Original control pulses are executed after the attenuation of decoy pulses.

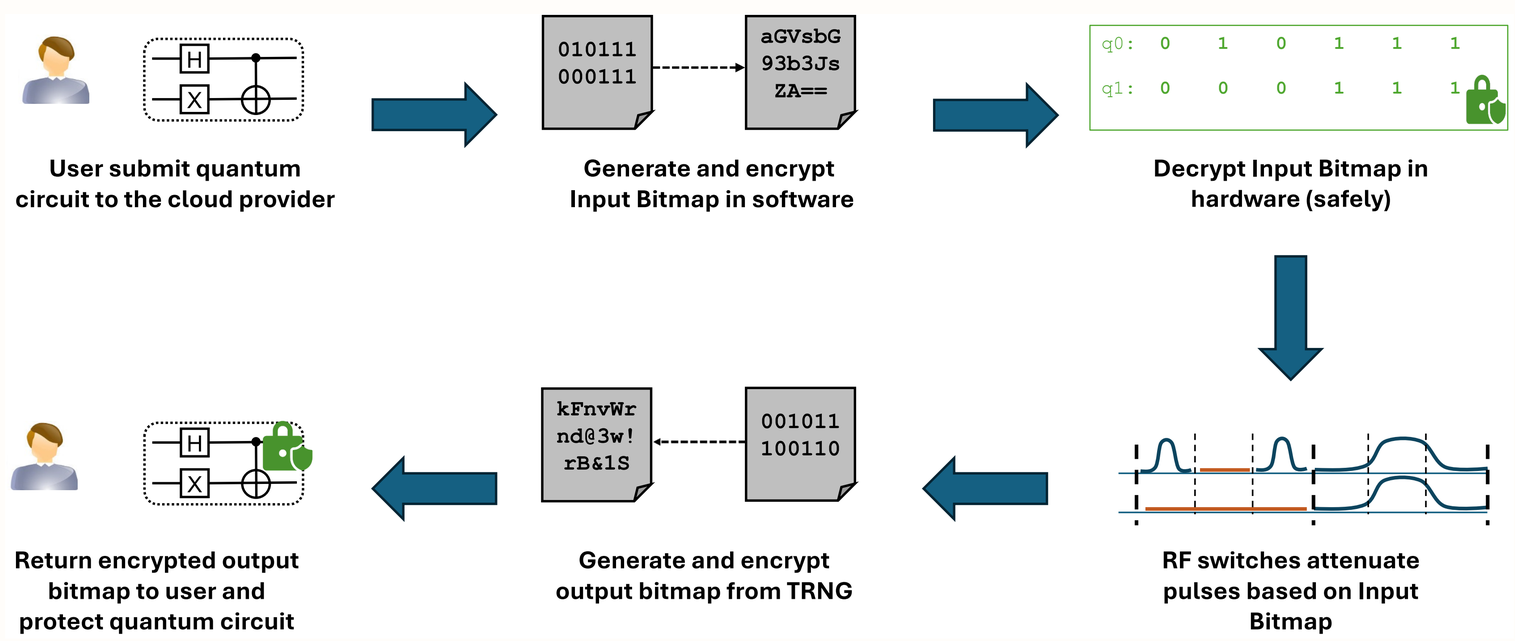

Figure 3: Workflow of SoteriaQ steps followed during software and hardware.

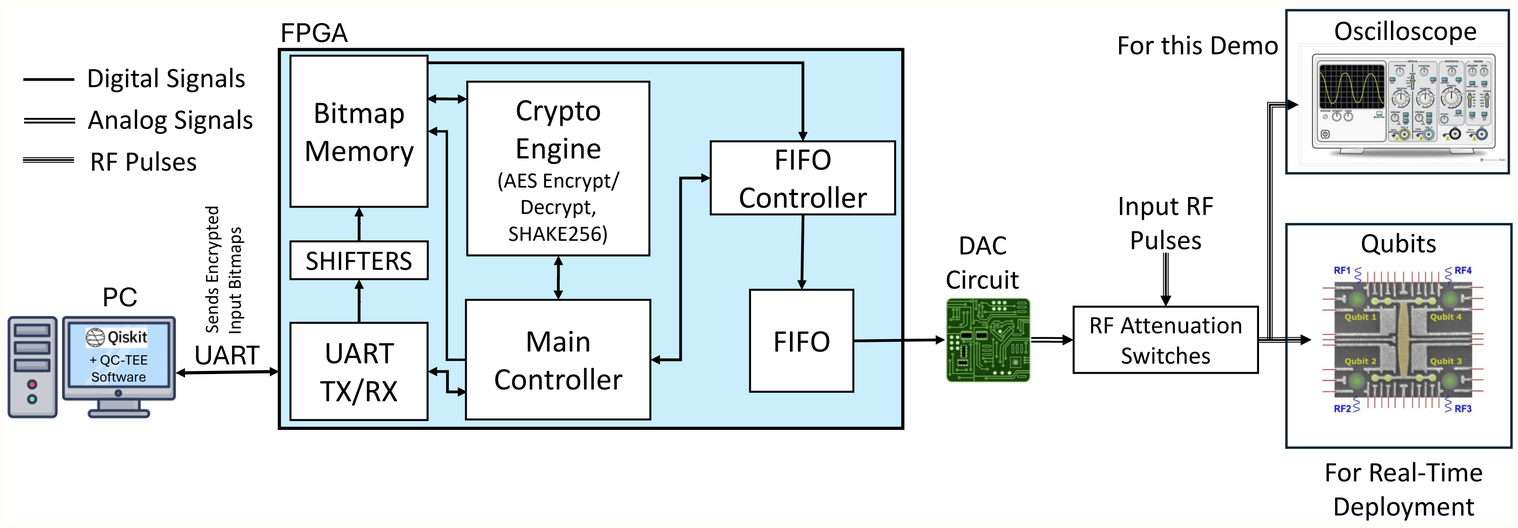

Figure 4: Block diagram of the complete SoteriaQ design.

Architecture of SoteriaQ

To realize SoteriaQ, a number of hardware components need to be added to the internals of the dilution refrigerator. All components are available today and use very low power and area compared to existing quantum computer components. A block diagram of the major components and their connections is shown in Figure 4. The memory is used to store the decrypted input bitmap, which is later used to control the attenuation of the pulses. First, a Decryption Engine is used to decrypt the encrypted input bitmap and then store the decrypted input bitmap in the Input Bitmap Memory. The memory is used to store the decrypted input bitmap, which is later used to control the attenuation of the pulses. The hardware security engine is a hardware implementation of a state machine that controls the attenuation switches. Attenuation Switches are used to attenuate the decoy control pulses, which were added to confuse the potential attackers, and are not actually used for computation. In most of the cases, we are able to achieve more than 2^{256} combinations required by the attacker to guess the circuit. We present the security analysis of our design in Figure 5.

Security Analysis

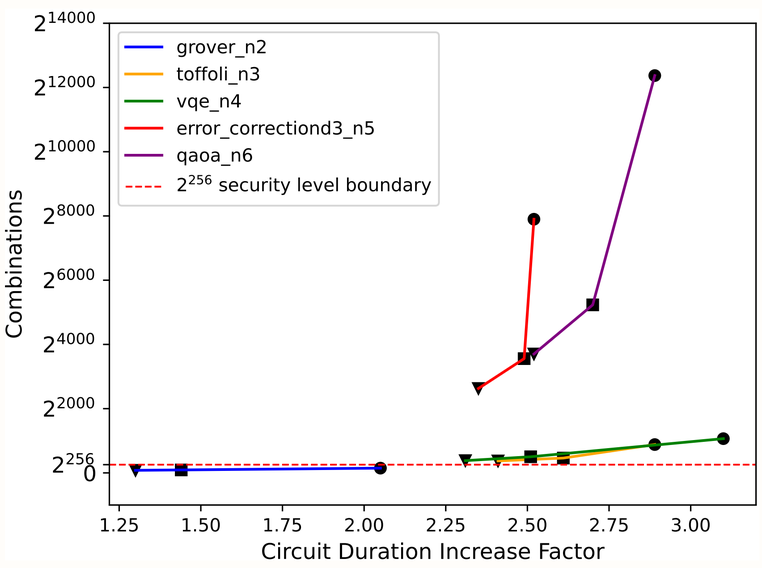

Figure 5: Combinations required by the attacker to guess circuit versus circuit duration increase factor, for each of the three obfuscation levels.

References

[1] T. Trochatos, C. Xu, S. Deshpande, Y. Lu, Y. Ding, and J. Szefer, "A quantum computer trusted execution environment," *IEEE Computer Architecture Letters*, vol. 22, no. 2, pp. 177–180, 2023.