Exploration of Vulnerabilities of Fault-Tolerant Quantum Computing

Theodoros Trochatos
Yale University
theodoros.trochatos@yale.edu

Christopher Kang
University of Chicago
ctkang@uchicago.edu

Frederic T. Chong
University of Chicago
chong@cs.uchicago.edu

Jakub Szefer
Northwestern Unviersity
jakub.szefer@northwestern.edu

***Abstract*—Emergence of fault-tolerant quantum computers (FTQC) brings about promise of harnessing the power of quantum computing at larger scale. At the same time, as quantum computers are expected to process more sensitive information, there is a need to understand the security issues in fault-tolerant quantum computers, and develop defenses for attacks that may compromise confidentiality or integrity of the data processed by FTQC. While noisy intermediate-scale quantum (NISQ) computers have already been studied from the security perspective, understanding security issues with FTQC is still an open research question. To address the missing research gap, this work presents the first exploration of possible security vulnerabilities of FTQC. The work presents analysis of possible threat models and outlines potential vulnerabilities of FTQC. Understanding the landscape of the threats can help lead to development of safer FTQC design at both software and hardware levels.**

***Index Terms*—Fault-Tolerant Quantum Computing, FTQC, Security Vulnerabilities**

I. Introduction

Fault-tolerant quantum computing (FTQC) is an approach to quantum computation that ensures reliable operation even in the presence of errors caused by noise, decoherence, and imperfections in quantum hardware. While some quantum algorithms may show benefits on the current Noisy Intermediate Scale Quantum (NISQ) computers, many quantum computing algorithms, such as Shor's [19] or Grover's [11] require use of error-corrected quantum computers to produce efficient solutions to problems at large scale. With the ability of FTQC to execute powerful quantum algorithms such as Shor's, Grover's, or others, there is a need to understand the security issues in fault-tolerant quantum computers, and develop defenses for attacks that may compromise confidentiality or integrity of the data processed by FTQC.

In the realm of NISQ computers, researchers have already demonstrated a variety of potential security attacks. For example, by abusing crosstalk, researchers have shown it may be possible to maliciously flip qubits [16] when two quantum circuits are sharing the same quantum computer. Or, researchers have also shown that basic quantum computing gates, such as the reset gate, leak information when the gate operation is not perfect [12]. These and other prior work mainly abuse NISQ computers where there is no error correction and the qubits are vulnerable to various forms of disturbance.

With the advent of fault-tolerant quantum computing, many of the security issues in NISQ may be avoided. For example, basic attacks that introduce noise or errors into computation may be protected through the error correction. Yet, as this work proposes, there are new, unexplored vulnerabilities in fault-tolerant quantum computing. In general, this work presents exploration of possible vulnerabilities in fault-tolerant quantum computing to highlight the fact that although error correction provides numerous benefits, it does not automatically protect against all security threats, and new security defenses will need to be developed and deployed on top of error correction in FTQC.

The potential vulnerabilities are compounded by the fact that FTQC (and NISQ computers) are expensive and only practically available as remote quantum servers though various cloud-based services. With remote access, users have no physical control of the machines or the cloud operators who manage the remote quantum computers. This presents the challenge of possible security attacks due to untrusted or malicious cloud providers or other untrusted or malicious users who are running their quantum programs on the shared cloud infrastructure (through spatial or temporal sharing).

Already today, quantum computers are easily accessible through cloud-based services such as Amazon Braket [1], IBM Quantum [4], or Microsoft Azure [8]. These services are open to anybody through pay-as-you-go cloud services. Many companies and startups that do not have their own quantum computers already can use these cloud-based quantum computers to run their, often proprietary, quantum circuits. As a recent example, JPMorgan Chase does not own a quantum computer, but leverages cloud-based Quantinuum H-series to develop algorithms for solving linear systems on quantum hardware [20]. With more powerful, FTQC the utilization of quantum computers will continue to raise, and the importance of securing quantum circuits and programs that execute on the quantum computers will continue to increase.

Given the emergence of proof-of-concept attacks and need to secure quantum computing, especially cloud-based quantum computing, a number of researchers have proposed some defenses. The defenses have been exclusively in the realm of NISQ computers. Researchers have developed a quantum compute antivirus [7], [6], while others have proposed use of obfuscation to protect quantum circuits executing on honest-but-curious cloud-based quantum computers [18]. Whether these defenses can be adapted to FTQC, or what FTQC specific defenses are needed, is an open research question. This

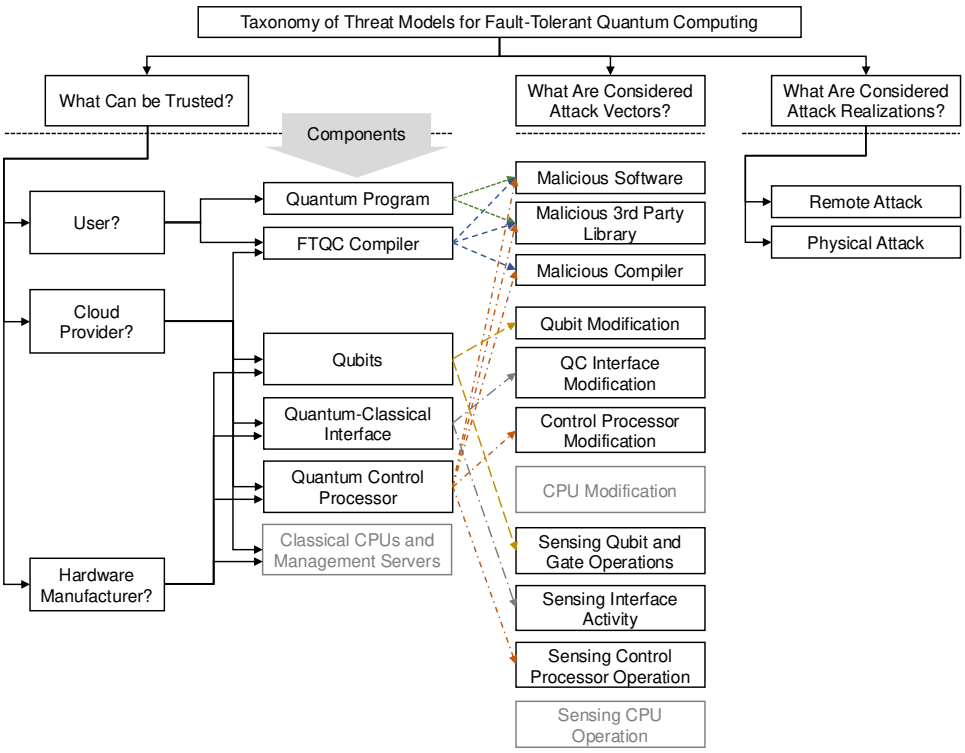


Fig. 1. Taxonomy of threat models for fault-tolerant quantum computing.

work focuses on exploration of vulnerabilities of fault-tolerant quantum computing. The resulting insights can directly drive future work on securing and improving FTQC.

II. Background

This section provides very brief background on FTQC and classical control infrastructure in FTQC.

A. Fault-Tolerant Quantum Computing

FTQC devices rely upon quantum error correction (QEC) to improve the noise resilience of computation. There exist many QEC codes; the surface code is among the best known because it can be implemented with 2D planar connectivity. However, these codes also require specific schemes to effect quantum logic. For example, the surface code can only natively implement the Clifford gates, which are not universal. To perform the 'T' gate needed for universality, specific magic state infrastructure is needed. Furthermore, two-qubit interactions on surface codes require 'lattice surgery,' deforming the code in a predictable pattern. The Clifford+T logical instruction set has encouraged the creation of new compilers and design of both QEC codes and compilers is an active research area. Compiling to the underlying platform requires mapping and routing logical qubits, scheduling their interactions, and planning magic state production.

B. Classical Control Infrastructure

The classical control infrastructure for an FTQC system must accomplish a range of tasks, from controlling physical qubits to decoding error syndromes and orchestrating higher level instructions [21]. These tasks require significant classical infrastructure that can be many times the size of the quantum chip and its refrigerating envelope when considering superconducting qubits. The control infrastructure can be abstracted as a 'quantum control processor' plus associated software and algorithms.

In particular, maintaining a logical qubit via syndrome decoding is a computationally demanding task. Decoding is known to be hard theoretically, with complexity class either NP complete or #P depending on the precise decoding problem [9]. Furthermore, decoding must be performed at the cycle rate of physical hardware; for fast platforms (e.g. superconducting qubits), this means that decoders must operate within microsecond [13] latencies.

Proposed algorithms and architectures for QEC decoders have explored a variety of ideas to improve performance. To improve algorithms complexity, approximate decoding algorithms can be used; some, for example, solve easy cases and offload hard cases [5], [15]. To reduce latency, dedicated decoding chips are placed either near or in the fridge. For example, CryoCMOS chips can be placed within the fridge [3] and FPGAs can be placed outside the fridge. However, these solutions all have drawbacks: approximate algorithms are still demanding computationally and specialized chips are expensive to fabricate, especially on novel platforms like CryoCMOS.

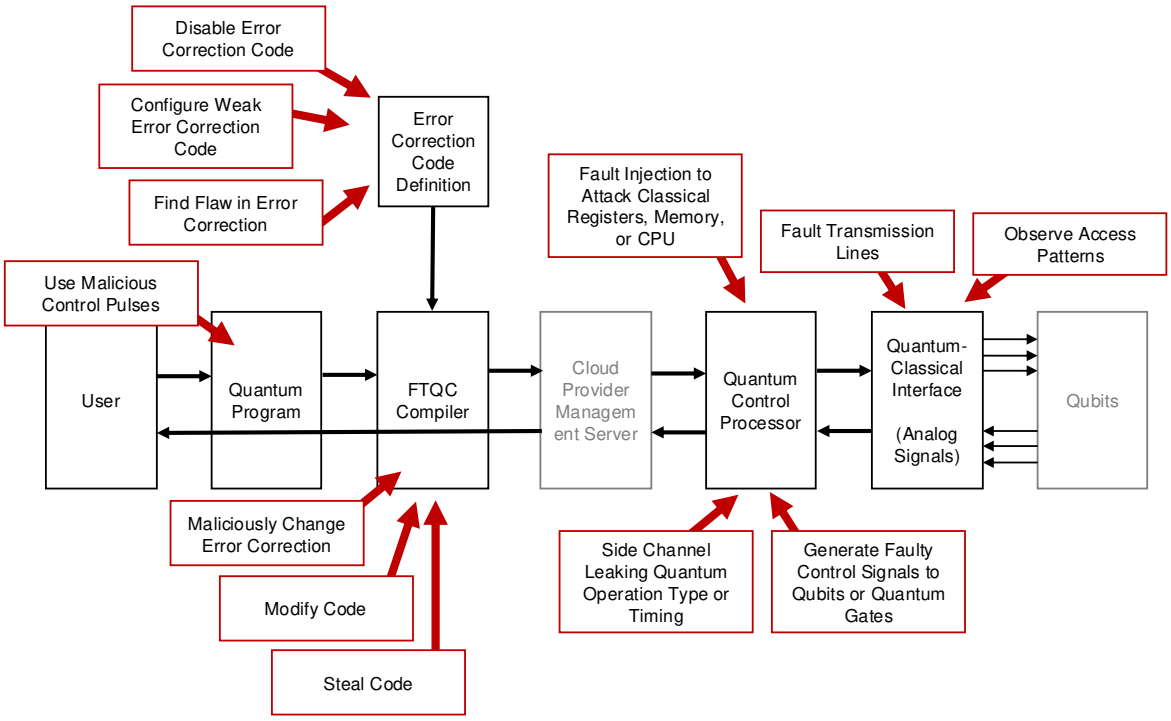


Fig. 2. Classification of vulnerabilities of fault-tolerant quantum computing.

C. Compiling Circuits to FTQC Devices

Compilation of quantum algorithms is a nascent and active area of research. Compilation involves multiple levels of abstraction, requiring program transformation from the algorithmic, logical, and physical levels. For example, programs may first be described mathematically, then translated into logical instructions, then optimized for physical qubit instructions.

There are many potential compiler optimizations which may be employed. For example, compiling multi-controlled `NOT` gates can be done with clean ancillas, dirty ancillas [22], or in-place [2]. Each of these compilation schemes would affect the physical operations actually performed. Furthermore, these implementations must be mapped and routed on real devices, incurring additional overheads and room for optimization.

D. Security of FTQC, Classical Control, and FTQC Compilers

This far, security aspects of FTQC, classical control, and compilers have not been considered. As many aspects of FTQC are being actively developed, now is the time to incorporate security into the design. To help give guidance, this paper presents taxonomy of threat models in the following section, and classification of vulnerabilities of FTQC in the subsequent section.

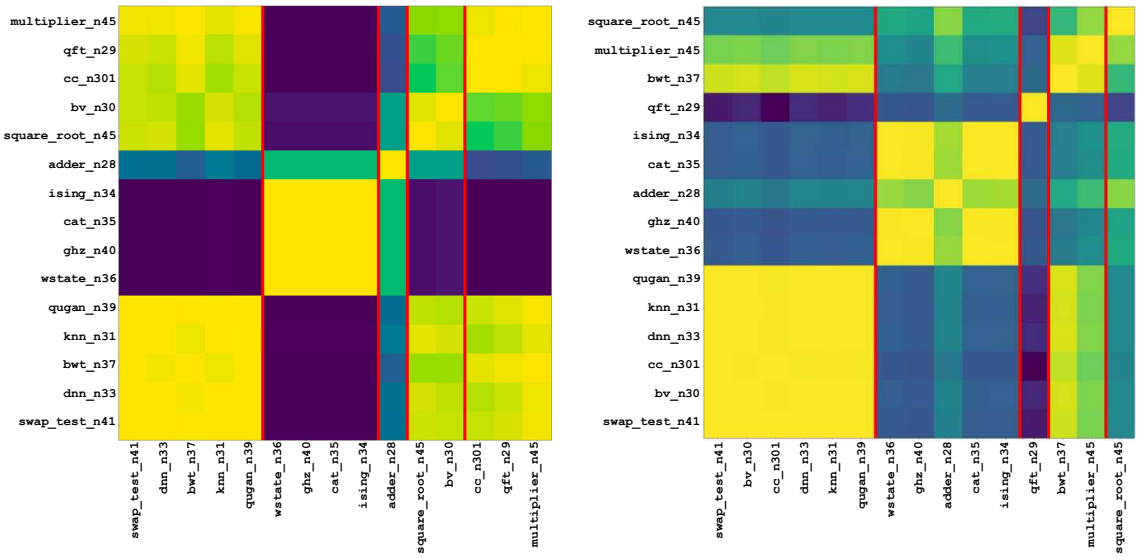
III. Taxonomy of Threat Models

We first provide a taxonomy of threat models. Different users may have different threats they are worried about, what one user considers a vulnerability, another may not be concerned about. Thus, it is important to have a taxonomy of threat models, which then is used to evaluate what vulnerabilities are a concern under each particular threat model.

Figure 1 shows a taxonomy of threat models for fault-tolerant quantum computing. When deciding on a threat model, a user (or cloud provider or manufacturer) should decide who and what can be trusted, what are considered attack vectors, and what are the considered attack realization. A specific threat model is effectively a combination of answers to each of these questions.

For example, a cloud provider may be worried about untrusted users and using malicious software to perform remote attacks. Or, a cloud user may be worried about an untrusted cloud provider performing sensing on operation of the quantum control processor through a physical attack. Or a cloud provider may be worried about an untrusted hardware manufacturer who has inserted a hardware Trojan through modification of qubit design.

In the Figure 1 colored and dashed arrows highlight various attack vectors when a particular component is assumed untrusted. For example, a quantum program could be directly malicious (label “Malicious Software” in the taxonomy figure) or an otherwise honest user could download a third-party library that contains malicious code (label “Malicious 3^{rd} Party Library” in the taxonomy figure). Further, a quantum compiler may be directly malicious (label “Malicious Software” in the taxonomy figure), or the compiler developer can download a third-party library that contains malicious code (label “Malicious 3^{rd} Party Library” in the taxonomy figure). On the hardware side, an untrusted cloud provider could try to modify the qubits and hardware they control (label “Qubit



(a) RandomWalk (b) ShortestPath

Fig. 3. Clustered heatmaps of large size transpiled QASMBench [10] benchmarks with RandomWalk and ShortestPath graph kernels.

Modification" in the taxonomy figure) or given their physical access to the devices they could try to sense operation of the qubits ((label "Sensing Qubit and Gate Operations" in the taxonomy figure).

A. Choice of Threat Model(s)

The choice of threat model is a subjective choice. Some threats may be more plausible, e.g. user downloading a malicious third-party library. Other threats may be less plausible, e.g. a cloud provider opening up a dilution refrigerator and physically modifying the qubit hardware. Yet, on the other hand, a malicious hardware manufacturer could easily modify the hardware. The threat model taxonomy cannot answer the question about which threat models a user (or cloud provider or manufacturer) should consider, but it is a means of thinking about what could be trusted (or untrusted) and what are possible attack vectors and attack realizations.

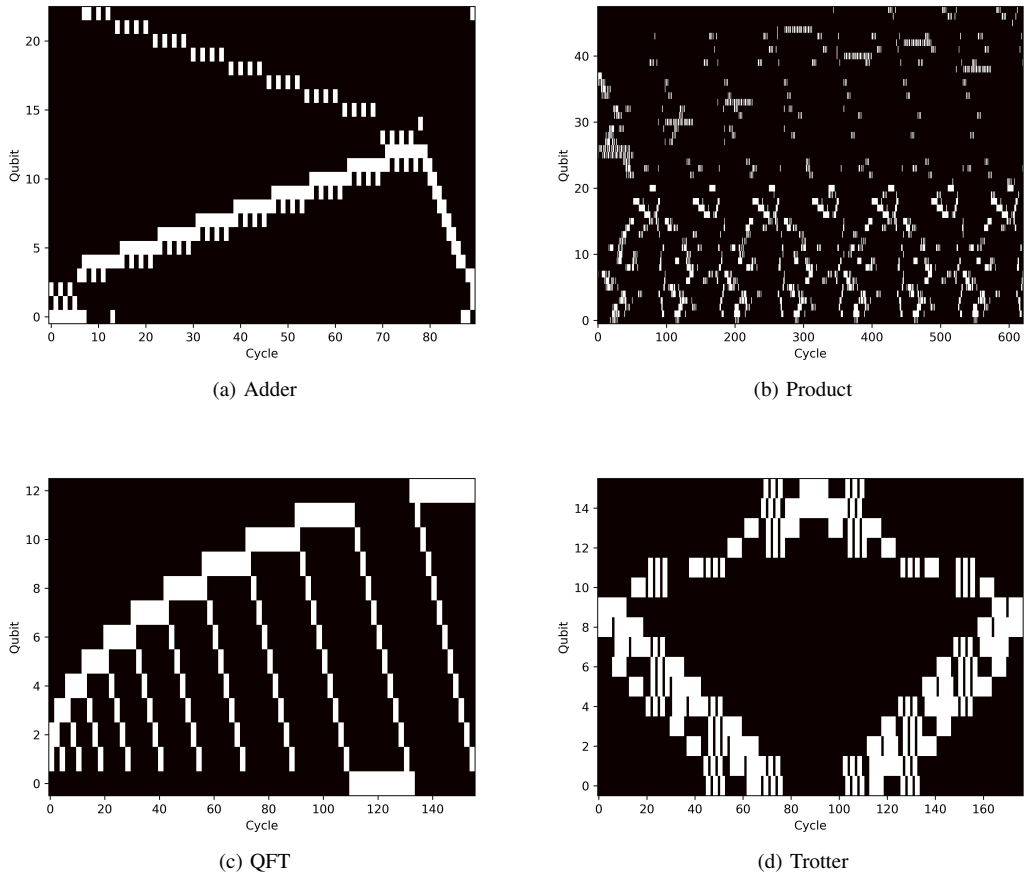
IV. Classification of Vulnerabilities of Different Components

While noisy intermediate-scale quantum (NISQ) computers have already been studied from the security perspective, understanding security issues with FTQC is still an open research question. To address the missing research gap, this section presents the first exploration of possible security vulnerabilities of FTQC by developing a taxonomy of vulnerabilities of FTQC.

Possible vulnerabilities are presented for each component of a FTQC system. Whether the possible vulnerabilities are a real security concern, depends on whether the component is considered untrusted by the user (or cloud provider or manufacturer). This in turn depends on the threat model that is being considered. Section III discussed the possible threat models through use of our taxonomy.

Figure 2 shows various vulnerabilities that could affect FTQC systems. We focus on five components, the quantum programs, FTQC compiler, error correction code definition, quantum control processor, and the quantum-classical interface. We separate the error correction code definition to highlight the fact that otherwise correct and trusted control processor could be configured to use wrong or incorrect or weak error correction.

Considering quantum programs, with FTQC the possibility to cause crosstalk attacks may be limited, as the users execute atop logical qubits. However, if a user is able to get access to control of physical qubits (through the ability to disable error correction or specify custom error correction algorithm that allows them to control physical qubits), then users could generate attacks by having malicious control pules. Considering the error correction code definition, if the provider allows for custom error correction, or attackers find ways to manipulate the error correction, then they can undermine the system. Further, attackers could find flaws in the error correction algorithms or implementations that then facilitate attacks. Considering the FTQC compiler, it could change the error correction, modify code, or even steal user's code. Users will be worried about code modification or stealing (if compiler is hosted in the cloud), while cloud providers will be worried about compiler properly implementing error correction (if compiler is hosted by users locally before they submit compile circuits to the cloud). Considering quantum control processor, it is vulnerable to various side and fault injection attacks. Targeting classical operations, attackers can steal digital or analog data about qubit or gate operations being executed. Quantum classical interface is the direct connection between the control processor and the qubits. Transmission lines could be monitored to learn qubit access times and



(a) Adder

(b) Product

(c) QFT

(d) Trotter

Fig. 4. Sample qubit access patterns for common quantum subroutines. Within each graph, moving *right* is later in program execution, while each *row* is a specific qubit. A qubit is white if it is interacting with another qubit or not. (a) is an adder circuit, (b) is a multiplier circuit, (c) is a Quantum Fourier Transform (QFT) circuit, (d) is a Trotter circuit.

patterns, or faults could be injected into the transmission lines.

V. Example Attacks

In this section, we present two preliminary results demonstrating potential weaknesses in FTQC concerning program identification. We focus on possible side channel leaks in the quantum control processor. We simulate attacker's access to information about which qubits are interacting with each other as different quantum circuits execute, i.e. we consider the threat model of an untrusted cloud provider collecting information about quantum gates and the qubits they involved. We consider only two-qubit interactions among logical qubits as a quantum program executes.

A. Clustering Connectivity Graphs

As the most basic approach, we generate connectivity graphs, where each node represents a qubit, while the edge weight represents how often each pair of qubits has interacted during the execution of the circuit.

We generate graphs that represent a subset of QASMBench [10] benchmarks. To identify whether the graphs have structure, we take two steps. First, we compute the similarity among the graphs, then, we cluster the provided graphs using graph kernels. Graph kernels [14] are used to compare the similarity of different graphs. A graph kernel g is defined over the space of graphs $\mathcal{G}$ as follows:

$$g : \mathcal{G} \times \mathcal{G} \mapsto \mathbb{R} \tag{1}$$

For example, for graphs G_1, G_2, $g(G_1, G_2)$ represents the similarity of graphs G_1, G_2 under some metric g. We use the Python library Grakel [17] and ShortestPath and RandomWalk kernels.

Figure 3 shows the clustered heatmaps of the large size transpiled QASMBench [10] benchmarks with RandomWalk and ShortestPath graph kernels. More similar graphs are shown in light color, and less similar graphs are shown in dark color. On the diagonal we can see that by definition, each graph is similar to itself and thus the diagonal is light colored. As we can see, different benchmarks are similar to each other, and groups of benchmarks get clustered together. Future understanding of the reasons behind why the benchmarks are similar and how that relates to the clusters can help understand if there are possible information leaks from the connectivity graphs that an attacker may gather. For example, if a new

algorithm is being executed, its similarity to a known circuit could leak information about that new algorithm.

B. Qubit Access Pattern

As a second approach, we consider qubit access pattern. We generate qubit access patterns by analyzing which qubits would be active, i.e. interacting with another qubit, at each time-step of circuit execution.

We analyze four quantum subroutines: quantum adder, quantum product, QFT, and Trotter circuits. We observe that access patterns of qubits can also reveal some insights about program structure. In Figure 4, we show two-qubit (interacting) gates in time. Note that different subroutines have different access patterns; this may serve as a unique signature for different subroutines. In particular, access patterns will also be harder to obfuscate, as they reflect fundamental data interactions that must occur for the program to succeed.

VI. Conclusion

In this work, we analyzed the possible threat models and potential vulnerabilities of the fault-tolerant quantum computers. We presented a taxonomy of different threat models and classification of vulnerabilities in FTQC. In addition, two example attack scenarios were analyzed from among the possible threat models and vulnerabilities. The main focus was to identify the structure of the executed quantum program. Based on the taxonomy of threat models and the classification of vulnerabilities, future work can study each threat model and each vulnerability to build a comprehensive understanding of threats to FTQC, and eventually propose software or hardware defenses.

VII. Acknowledgments

This work is funded in part by NSF grant 2332406. This work is also funded in part by the STAQ project under award NSF Phy-232580; in part by the US Department of Energy Office of Advanced Scientific Computing Research, Accelerated Research for Quantum Computing Program; and in part by the NSF Quantum Leap Challenge Institute for Hybrid Quantum Architectures and Networks (NSF Award 2016136), in part based upon work supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, and in part by the Army Research Office under Grant Number W911NF-23-1-0077. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] Amazon braket. https://aws.amazon.com/braket/.

[2] Adriano Barenco, Charles H Bennett, Richard Cleve, David P DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical review A*, 52(5):3457, 1995.

[3] E Charbon, F Sebastiano, A Vladimirescu, H Homulle, S Visser, L Song, and R M Incandela. Cryo-cmos for quantum computing. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 13–5. IEEE, 2016.

[4] Andrew Cross. The ibm q experience and qiskit open-source quantum computing software. In *APS March meeting abstracts*, volume 2018, pages L58–003, 2018.

[5] Poulami Das, Christopher A Pattison, Srilatha Manne, Douglas M Carmean, Krysta M Svore, Moinuddin Qureshi, and Nicolas Delfosse. Afs: Accurate, fast, and scalable error-decoding for fault-tolerant quantum computers. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 259–273. IEEE, 2022.

[6] Sanjay Deshpande, Chuanqi Xu, Theodoros Trochatos, Yongshan Ding, and Jakub Szefer. Towards an antivirus for quantum computers. In *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 37–40, 2022.

[7] Sanjay Deshpande, Chuanqi Xu, Theodoros Trochatos, Hanrui Wang, Ferhat Erata, Song Han, Yongshan Ding, and Jakub Szefer. Design of quantum computer antivirus. In *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 260–270, 2023.

[8] Johnny Hooyberghs and Johnny Hooyberghs. Azure quantum. *Introducing Microsoft Quantum Computing for Developers: Using the Quantum Development Kit and Q#*, pages 307–339, 2022.

[9] Antonio deMarti iOlius, Patricio Fuentes, Román Orús, Pedro M Crespo, and Josu Etxezarreta Martinez. Decoding algorithms for surface codes. *arXiv preprint arXiv:2307.14989*, 2023.

[10] Ang Li, Samuel Stein, Sriram Krishnamoorthy, and James Ang. Qasmbench: A low-level qasm benchmark suite for nisq evaluation and simulation, 2022.

[11] Aamir Mandviwalla, Keita Ohshiro, and Bo Ji. Implementing grover's algorithm on the ibm quantum computers. In *2018 IEEE international conference on big data (big data)*, pages 2531–2537. IEEE, 2018.

[12] Allen Mi, Shuwen Deng, and Jakub Szefer. Securing reset operations in nisq quantum computers. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2279–2293, 2022.

[13] Hartmut Neven. Meet willow, our state-of-the-art quantum chip, Dec 2024.

[14] Giannis Nikolentzos, Giannis Siglidis, and Michalis Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 2021.

[15] Gokul Subramanian Ravi, Jonathan M Baker, Arash Fayyazi, Sophia Fuhui Lin, Ali Javadi-Abhari, Massoud Pedram, and Frederic T Chong. Better than worst-case decoding for quantum error correction. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 88–102, 2023.

[16] Mohan Sarovar, Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout. Detecting crosstalk errors in quantum information processors. *Quantum*, 4:321, 2020.

[17] Giannis Siglidis, Giannis Nikolentzos, Stratis Limnios, Christos Giatsidis, Konstantinos Skianis, and Michalis Vazirgiannis. Grakel: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020.

[18] Theodoros Trochatos, Chuanqi Xu, Sanjay Deshpande, Yao Lu, Yongshan Ding, and Jakub Szefer. A quantum computer trusted execution environment. *IEEE Computer Architecture Letters*, 22(2):177–180, 2023.

[19] CH Ugwuishiwu, UE Orji, CI Ugwu, and CN Asogwa. An overview of quantum cryptography and shor's algorithm. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(5), 2020.

[20] Romina Yalovetzky, Pierre Minssen, Dylan Herman, and Marco Pistoia. Solving linear systems on quantum hardware with hybrid hhl++. *Scientific Reports*, 14(1):20610, 2024.

[21] Fang Zhang, Xing Zhu, Rui Chao, Cupjin Huang, Linghang Kong, Guoyang Chen, Dawei Ding, Haishan Feng, Yihuai Gao, Xiaotong Ni, et al. A classical architecture for digital quantum computers. *ACM Transactions on Quantum Computing*, 5(1):1–24, 2023.

[22] Ben Zindorf and Sougato Bose. Efficient implementation of multi-controlled quantum gates. *arXiv preprint arXiv:2404.02279*, 2024.