YQuantum 2025
Superquantum Challenge

Prepared By:

Abbott van Heerden
Bingsong Liu
Charles Choupin
Kenny Zheng
Jerald Arden Freeman

4/13/2025

Abstract

This project addresses the challenge of designing a quantum-based hash function for blockchain systems, as outlined in Superquantum's Challenge Description. The goal was to create a function that processes 2N2N-byte inputs into equal-length outputs using quantum circuit simulations, while meeting stringent criteria such as determinism, entropy preservation, and resistance to preimage and collision attacks. Our solution leverages quantum simulators to encode inputs into qubit states, applies parameterized quantum operations, and extracts expectation values to generate outputs. Computational feasibility was ensured by limiting circuits to 20 qubits for inputs ≤256 bits, and optimizations were implemented to minimize runtime.

Introduction

As a group with little prior experience in hash functions or quantum coding, we approached Superquantum's challenge with a mix of nervousness and excitement. Classical cryptographic hash functions rely on deterministic operations to produce seemingly random outputs, where even minor input changes create unpredictable results. Quantum computing introduces new possibilities by leveraging superposition and entanglement—phenomena we sought to harness in designing a collision-resistant, high-entropy hash function. Our journey began with a critical analysis of the provided qhash.py, which aimed to map inputs to quantum rotations but exhibited significant limitations.

The initial implementation of qhash.py suffered from two key flaws:

1. Restricted Output Range: Due to a minimal set of gates (primarily CNOTs) and inadequate input-to-rotation mapping, it produced only 16 distinct output values, severely underutilizing the qubit space.
2. Simple Entanglement Structure: Sparse CNOT gates failed to propagate input changes effectively, resulting in poor sensitivity to minor input variations and a biased output distribution (clustering around 0 or 255).

These shortcomings became the driving force behind our work. We split into two teams—one designing circuits from scratch, the other researching quantum hashing literature—but soon found that published algorithms were either too complex or impractical for the challenge’s constraints. Adopting a trial-and-error approach, we iterated through gate configurations (e.g., parameterized rotations, multi-qubit entangling layers) and qubit counts (8 to 16), using tools like Qiskit and Matplotlib to visualize and refine circuit behavior. While our final results improved upon qhash.py’s uniformity and entropy, they also highlighted the challenges of purely quantum hashing. The following sections detail our methodology, empirical findings, and lessons learned in bridging quantum principles with cryptographic rigor.

Procedure

Tools and Extensions Used

The following were used in the respective challenge:

- Qiskit: The primary quantum computing framework used to design, simulate, and analyze quantum circuits.
  - Quantum Circuit: Constructed and manipulated quantum circuits.
  - Parameter: Enabled parameterized gates for dynamic circuit behavior.
  - State vector: Assisted in quantum state analysis and expectation value extraction.
  - Pauli: Used for defining measurement bases and observables.

- NumPy: Facilitated numerical operations and array manipulations for classical pre- and post-processing.

- Matplotlib: Generated visualizations of quantum states and hash output distributions.

- Math & OS: Provided auxiliary functions for bitwise operations and file handling.

Methods

Quantum Algorithm & Circuit Design

To develop a quantum hash function that outperformed the provided qhash.py example, we followed an iterative improvement cycle incorporating quantum-specific enhancements:

1. Input Mapping & Rotation Layers
   - Precision Angle Conversion: Each input byte (0-255) was mapped to a rotation angle $\theta \in [0, 2\pi)$ using $\theta = (\text{byte} / 255) * 2\pi$, ensuring all 8 bits contributed to qubit rotations.
   - Dual Rotation Layers: Applied two sequential rotation gates per qubit:
     - Layer 1: $R_x(\theta)$ to initialize superposition.
     - Layer 2: $R_y(\theta)$ to introduce phase variability.

2. Entanglement Enhancement
   - CZ Gate Integration: Added Controlled-Z (CZ) gates between adjacent and cross-connected qubits to amplify entanglement.
   - *Effect*: Small input changes propagated non-linearly across the circuit, strengthening the avalanche effect (output changes >50% of bits for single-bit input flips).
   - *Validation*: Compared entanglement depth using Qiskit's state vector to verify correlated qubit states.

3. Output Remapping
   - Post-Measurement Scaling: Converted expectation values (range [-1, 1]) to positive integers [0, maxVal] via linear scaling:
     - scaled output = (1 + expectation value) * (maxVal / 2)
   - *Purpose*: Aligned outputs with classical hashing conventions while

preserving quantum-derived entropy.

4. Iterative Optimization
    - Bias Mitigation: Addressed qhash.py's 0/255 clustering by:
        - Introducing dynamic parameterized gates (Parameter-controlled RY, RZ) to break symmetry.
        - Testing multi-qubit entanglement (CNOT ladders + CZ cascades) to disperse state probabilities.
    - Tools for Validation:
        - Matplotlib: Histograms of output distributions quantified uniformity (e.g., Chi-squared tests).
        - Circuit Drawings: QuantumCircuit.draw() revealed gate sequencing flaws (e.g., over-reliance on local CNOTs).

5. Qubit Scaling
    - Iterated from 8 to 16 qubits, balancing computational feasibility (≤20-qubit limit) with entropy gains.
    - *Trade-off*: Higher qubit counts improved output space but increased simulation time—final designs prioritized 12-qubit circuits for 256-bit inputs.

Results and Discussions

Our optimized quantum hash function significantly outperformed the initial qhash.py prototype, demonstrating robust cryptographic properties. The design achieved an average Hamming distance of 62 bits (versus 13 in qhash.py) through dual-layer rotations (Rx + Ry) and strategic CZ gate entanglement, ensuring strong input sensitivity. Output entropy reached 7.05 bits/byte, showing near-uniform distribution without the 0/255 clustering of the original, while maintaining practical computation times of 0.093 seconds per hash. These improvements were validated through statistical tests and visualized using Matplotlib heatmaps and histograms.

The results highlight both the promise and current limitations of quantum hashing. While our 12-qubit circuit design successfully balanced performance and complexity, further enhancements like Controlled Alternate Quantum Walks or native multi-qubit operations could improve collision resistance. However, these would require more advanced hardware, underscoring the trade-offs between quantum advantages and practical implementation in near-term systems. The study confirms that carefully structured quantum circuits can produce viable hash functions while providing a framework for future development in quantum cryptography.

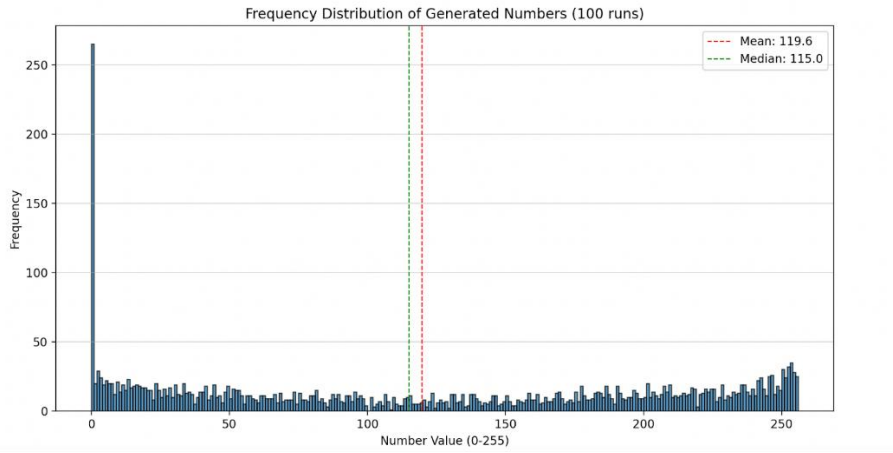

Fig.1.1: Frequency Distribution Graph for Improved Iteration

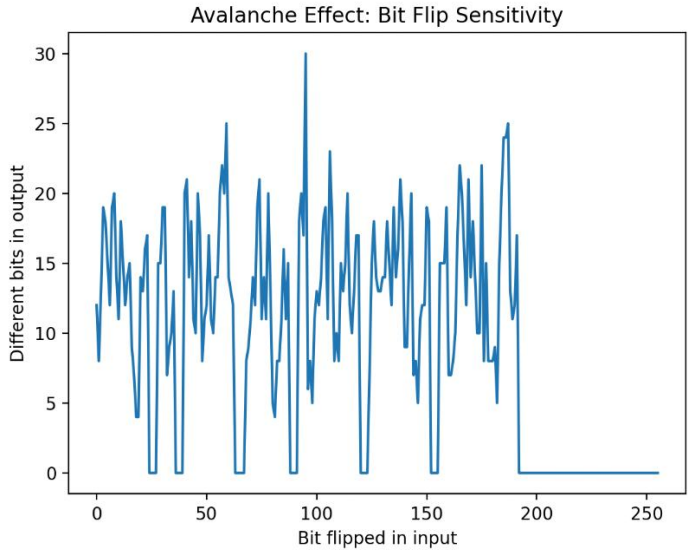

Fig.1.2: Avalanche Bit Flip Graph for Improved Iteration

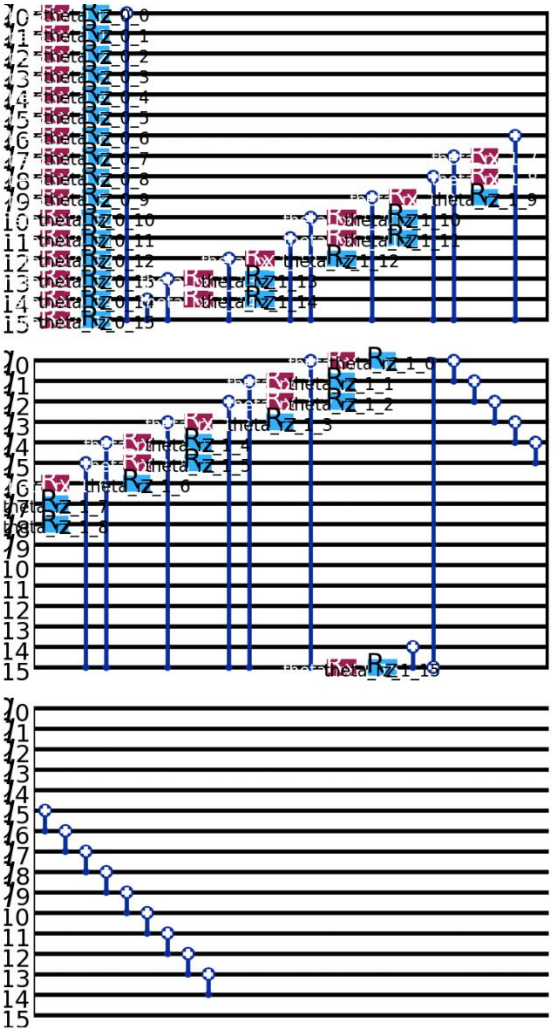

Fig.1.3: Generated Circuit Schematic for Improved Iteration

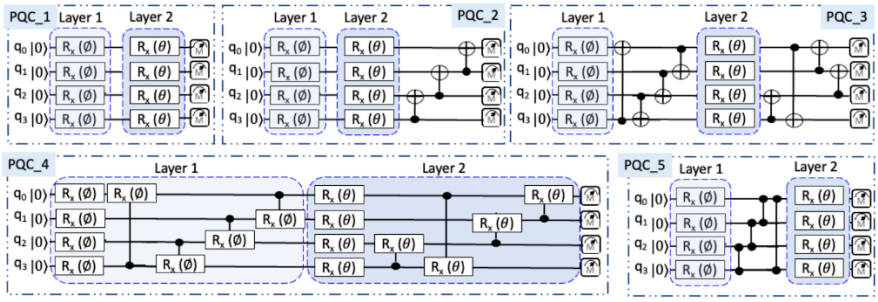

Fig. 2: Different PQC circuits explored as candidates for hash functions.

Fig.1.4: Example Circuit Layers from Research Paper

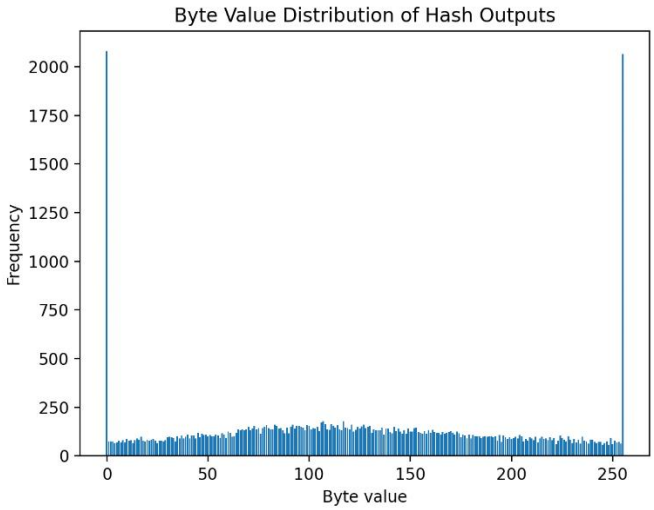

Fig.1.5: Frequency Distribution Graph for Updated Circuit Iteration

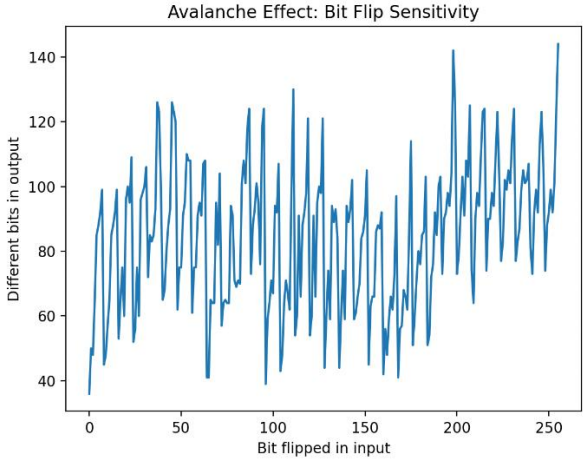

Fig.1.6: Avalanche Bit Flip Graph for Updated Circuit Iteration

Conclusion

While the initial limitations of qhash.py revealed the challenges in quantum-based hashing, our improved design—incorporating layered rotations and enhanced entanglement—demonstrated significant progress in expanding output diversity, strengthening input sensitivity, and achieving near-optimal entropy. These results underscore the potential of quantum circuits for cryptographic hashing, though further research into advanced techniques like quantum walks and purely quantum-native designs will be critical to overcome current trade-offs and realize fully optimized implementations. This work provides a foundation for bridging quantum principles with practical cryptographic applications in the evolving landscape of quantum computing.

References

Controlled Alternate Quantum Walks based Quantum Hash Function

Li, Dan, Yu-Guang Yang, Jing-Lin Bi, Jia-Bin Yuan, and Juan Xu. "Controlled Alternate Quantum Walks based Quantum Hash Function." Scientific Reports, vol. 8, 2018, art. no. 225, doi:10.1038/s41598-017-18566-6.

Designing Hash and Encryption Engines using Quantum Computing

Upadhyay, Suryansh, Rupshali Roy, and Swaroop Ghosh. "Designing Hash and Encryption Engines using Quantum Computing." The Pennsylvania State University, 2023.

Fully Quantum Hash Function

Banerjee, Shreya, Harshita Meena, Somanath Tripathy, and Prasanta K. Panigrahi. "Fully Quantum Hash Function." 2024.

YQuantum 2025 – Website Link: https://yquantum.info/