Resource Estimation for Breaking Elliptic Curve Cryptography on Quantum Hardware

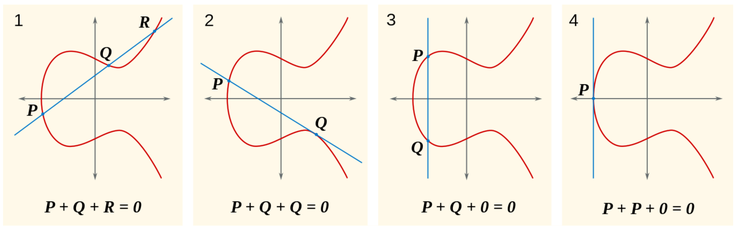
STUDENTS: Zhiyao Li, Lingnan Shen, Mark D'Souza

Abstract

- We implement a large-scale elliptic curve cryptography (ECC) quantum algorithm that computes a private key *k* from a public key *Q* (the *k*-wise elliptic addition of a base point *P*) via a top-down design.
- We follow the subroutines provided in the paper "How to compute a 256-bit elliptic curve private key with only 50 million Toffoli gates" by Daniel Litinski. [1]
- After implementing the algorithm, we perform comprehensive resource analysis using Microsoft's Azure Quantum Resource Estimator and compare our resource estimates to those theoretically anticipated in Litinski's paper.
- This analysis will be useful in determining the efficacy of the ECC algorithm in the age of fault-tolerant quantum computers.

Project Background

- We studied a 256-bit ECC algorithm in Q#, a quantum computing friendly programming language developed by Microsoft. This system has the NIST-recommended minimum key size and so it is reasonable to anticipate that it will be the first widely used cryptosystem to be compromised by quantum computing.
- In our project, we wrote a quantum algorithm using Shor's algorithm for discrete logarithms which would break the ECC systems on a hypothetical quantum computer.
- As of now, the industry accepted resource estimates for breaking a 256-bit elliptic curve are 2330 qubits and 126 billion Toffoli gates. In 2015, the NSA announced a long-term plan in 2015 to transition to a new cipher suite that is resilient to quantum attacks.
- The ECC algorithm is a cryptographic scheme based on elliptic curves with the main goal of determining a private key using a public key as an input.
- An elliptic curve is defined as follows: $y^2 = x^3 + c1\ x + c2$, a typically prime modulus *p* and a base point $P = (Px, Py)$. Points on the curve are integers modulo *p*. A key pair can be created by generating a random integer $0 \le k \le p - 1$ as the private key and computing $Q = [k]P = P + \ldots + P$ as the public key via elliptic curve point addition.
- Given two elliptic curve points $P = (a, b)$ and $Q = (c, d)$, their elliptic sum $R = P + Q$ is given by the diagram below together with some other special cases. A multiple of the base point P can be computed efficiently via repeated addition but there is no known efficient classical algorithm for the reverse operation.

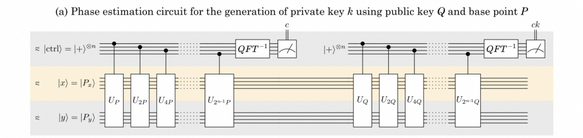


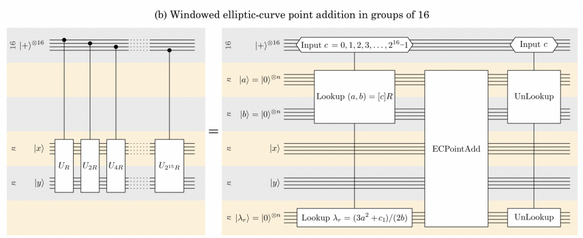
Methods

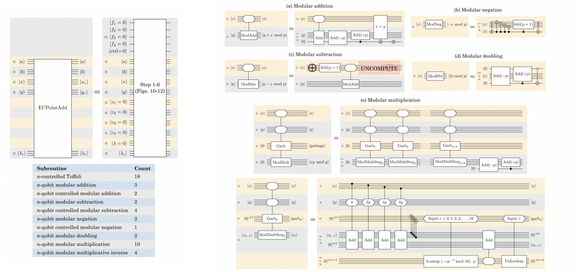
- During the project we have made use of the following resources: the **Azure Quantum Development Kit**, the quantum programming language **Q#** and the **Azure Quantum Resource Estimator**.
- The Modern Quantum Development Kit (Modern QDK) is the software development kit that interfaces Microsoft's cloud computing Azure Quantum service.
- Q# is a high-level programming language part of the QDK which is conducive to running and developing quantum algorithms.
- It draws from elements in Python and C# and lends itself to a model for program writing with loops, if/then statements and common data types. Importantly, it introduces new quantum-specific data structures and operations.
- The diagram on the right gives a flavor of how the language works.

```
operation ModAdd(x: Qubit[], y: Qubit[]): Unit is Adj + Ctl {
    // x, y are the two numbers to be added
    // the result is stored in y
    // |y> -> |y + x mod p>

    use ancilla = Qubit[1];
    IncByLE(x, y + ancilla);

    // Add(−p) to y and ancilla
    Adjoint IncByL(get_p(), y + ancilla);
    // controlled by ancilla and add(p) to y
    Controlled IncByL(ancilla, (get_p(), y));
    ApplyIfLessLE(X, x, y, ancilla[0]);

    X(ancilla[0]);
}
```

Structure of the E.C.C. Quantum Algorithm





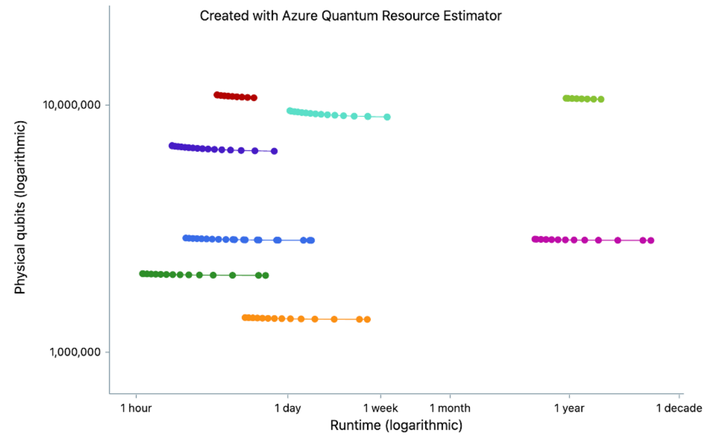


Subroutine	Count
n-controlled Toffoli	18
n-qubit modular addition	3
n-qubit controlled modular addition	2
n-qubit modular subtraction	2
n-qubit controlled modular subtraction	4
n-qubit modular negation	2
n-qubit controlled modular negation	1
n-qubit modular doubling	2
n-qubit modular multiplication	10
n-qubit modular multiplicative inverse	4

- In the first quantum circuit, the algorithm consists of two phase estimation steps with two different sets of unitaries performing elliptic curve point addition with base point *P* and public key *Q* acting on two quantum registers.
- The 'windowing technique' outlined in the second quantum circuit allows us to reduce the number of lookup additions.
- The next step is to code the ECPointAdd operation, a lengthy 6-step process.
- The left hand figure shows a schematic of the input registers of this operation as well as the total count of the n-controlled Toffoli and modular arithmetic subroutines.
- The right hand figure shows a similar decomposition of the modular arithmetic subroutines.

Resource Estimation Analysis

- We obtain the resource estimation for executing the algorithm on different hardware (Majorana, superconducting, and ion trap) with a error budget of 0.333.
- The fastest architecture is Majorana based Floquet code with a runtime of **1 hour**. The minimum number of qubits required is also Majorana based architecture with surface code requiring **1.3 million physical qubits**.
- The low T factory fraction in the algorithm results in a low tradeoff between time and space (number of physical qubits)



Run name	Runtime	Physical qubits	T factory fraction	rQOPS
Majorana with 1e-6, floquet	1 hours	2,079,728	1.78 %	3,986,666,667
Majorana with 1e-4 error, floquet	2 hours	6,856,304	5.73 %	2,146,666,667
Superconducting with 1e-4 error, surface	3 hours	2,897,496	2.34 %	1,610,000,000
Superconducting with 1e-3 error, surface	5 hours	10,998,120	4.85 %	837,200,000
Majorana with 1e-6, surface	10 hours	1,381,744	1.84 %	465,111,112
Majorana with 1e-4 error, surface	1 days	9,484,776	6.61 %	182,000,000
Ion trap with 1e-4 error, surface	177 days	2,868,616	1.36 %	1,073,334
Ion trap with 1e-3 error, surface	340 days	10,652,720	1.76 %	558,134

Future Work and References

Potential future work includes:

- More economic implementation of the modular inversion operation by parallelism, method outlined in ref [1].
- Resource estimation on generating multiple private keys of the same elliptic curve by repeating the second phase estimation with different public keys.

[1] Litinski, Daniel. "How to compute a 256-bit elliptic curve private key with only 50 million Toffoli gates." arXiv preprint arXiv:2306.08585 (2023).

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

ADVISERS: Mathias Soeken (Microsoft), Mariia Mykhailova (Microsoft) , Arka Majumdar (UW)

SPONSORS: Microsoft

Accelerating Quantum-Enabled Technologies Program (AQET)