A New Quantum Solution to the Discrete Logarithm Problem

Matthew Gregoire [1] Richard Howe [2] Guan-Wen Chou [2] Anton Nikulsin [2]

[1]UNC Chapel Hill

[2]North Carolina State University

July 1, 2020

The Discrete Logarithm Problem

Take a modulus N, an integer a, and a power b of a, such that $b = a^m \mod N$.

The Discrete Logarithm Problem

Given the values a and b mod N as above, find the value of m.[1]

- It's easy to compute b when given a and m.
- It's hard to find m when given a and b.
- This fact is the basis of many modern cryptographic protocols.

[1] This can actually be generalized to any group operation! For example, the discrete logarithm has applications in elliptic curve cryptography.

Half Bits

Let a be an integer modulo N, and let b be a power of a, so $b = a^m \mod N$. Also let r be the *order* of a, so $a^r = 1 \mod N$. (We'll continue to use this convention.)

The Half-Bit of b

The *half-bit* of b, denoted $HB_a(b)$, is defined:

$$HB_a(b) = \begin{cases} 0 & 0 \leq m < r/2 \\ 1 & r/2 \leq m < r \end{cases}$$

Essentially, this is the most significant bit of m's binary representation.

Our Project

- In his 1988 thesis, Kaliski presents an algorithm to calculate the discrete logarithm of a value b in polynomial time. [Kaliski, 1988]
- This algorithm relies on an *oracle function* which correctly predicts the half-bit of b with probability at least $1/2 + \epsilon$.
- In a 2017 paper, Kaliski presents a quantum implementation of such an oracle. [Kaliski, 2017]

Main Goal

- Our project implements Kaliski's quantum oracle function in Qiskit.
- We also implement his function to solve the discrete logarithm in Python.

Oracle Construction

The oracle operates in two phases. Given an n-bit input:

Phase One

1. Start in $|0\rangle^{\otimes n} |1\rangle^{\otimes n}$.
2. Apply three specified transformations.
3. Measure the first register to get the value y and collapse the second register to the superposition $|\Psi_y\rangle$.

Phase Two

1. Start in the state $|\Psi_y\rangle |0\rangle$.
2. Apply four specified transformations, one of which depends on the value of y.
3. Measure and output the contents of the second register.

Importantly, *the circuit in phase two depends on the measurement from phase one*.

Example Oracle

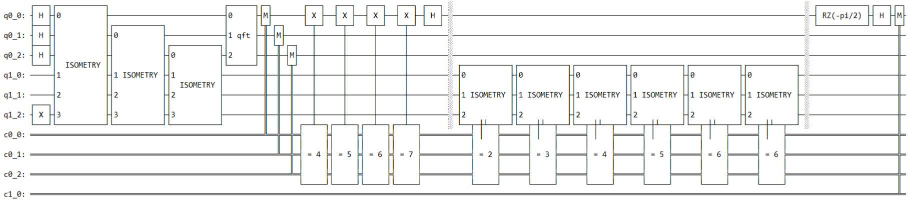

Figure: Oracle generated from `oracle(3, 2, 5)`. This circuit estimates $HB_3(2)$ modulo 5 and puts the value in register `c1_0`.

Limitations

Our oracle works, but is extremely inefficient. Sources of inefficiency:

- Unitary matrices used within the oracle are constructed on-the-fly. Their size is exponential in the input size.
- Qiskit does not allow the result of a classical measurement to dictate how the rest of the circuit is constructed.
- Due to current hardware limitations, this means our oracle can only run in simulators.

[4]:
```
import time

start = time.time()
print(Logarithm(7, 13, 15))
end = time.time()

print("Execution took", int(round(end - start)), "seconds.")
```

```
3
Execution took 206 seconds.
```

Figure: Our algorithm took 206 seconds to determine that $m = 3$ is the value that satisfies $7^m = 13 \mod 15$.

Future Work

- Improve efficiency of the oracle.
  - Look into the `QuantumCircuit.snapshot()` method to improve efficiency on simulators.
  - Increase efficiency of unitary generation.
- Build this into an accessible tutorial explaining the quantum algorithm.
- Create a toy demonstration using this algorithm to break RSA encryption.

References

Burton S. Kaliski Jr. (1988)
Elliptic Curves and Cryptography: A Pseudorandom Bit Generator and Other Tools
Doctoral dissertation, MIT, Cambridge, USA.
https://dspace.mit.edu/bitstream/handle/1721.1/14709/18494044-MIT.pdf

Burton S. Kaliski Jr. (2017)
A Quantum "Magic Box" for the Discrete Logarithm Problem
Cryptology EPrint Archive, Report 2017/745.
https://eprint.iacr.org/2017/745

The End

A huge thank you to IBM Quantum, the Qiskit community, and the organizers of the North Carolina Qiskit Community Summer Jam 2020 for making this project possible!