A Quantum Algorithm To Locate Unknown Hashes For Known Ngrams Within A Large Malware Corpus

By: Nicholas R. Allgood

Agenda

Part $|000\rangle$ Introduction to Quantum Computing

Part $|001\rangle$ Ngrams

Part $|010\rangle$ Quantum Searching

Part $|011\rangle$ Contribution

Part $|100\rangle$ Evaluation

Part $|101\rangle$ Future Work and Conclusion

Thesis Statement

Quantum computing while gaining a lot of traction in cryptography can also be applied to malware analysis. We will show that quantum algorithms are very applicable to nearly any domain with a little bit of thought and creativity. While an entirely quantum solution for a problem may not always be feasible or possible, exploring quantum algorithms for parts of a solution can still make a significant difference.

Part $|000\rangle$: Introduction to Quantum Computing

Why Quantum?

True Parallelism: Through the use of quantum mechanical properties, we can take advantage of true parallelism.

Short Introduction to Quantum Computing

Quantum computing is the use of quantum mechanical phenomena to provide a model of computation. These phenomena are superposition, entanglement, quantum measurement, and interference.

Quite a bit different than classical computing, though not entirely…

For example, in classical computing you have a bit which can be a 1 or 0, but never 1 and 0 at the same time.

Quantum computing a bit can be 1, 0, or 1 and 0 at the same time.

Types of Quantum Computers

Adiabatic (Quantum Annealer): Used to find the minimum global state and is the more common of quantum computers seen today. Typically useful for optimization problems (NP-Hard) [5].

Analog (Quantum Simulator): These are used to simulate complex quantum interactions and able to detect untraceable quantum details. Most likely will start to see dramatic improvements over classical computing [5].

Universal: The “holy grail” of quantum computing that combines the full power of classical and quantum computation and will be the most general purpose quantum computer [5].

Hybrid: Combines quantum annealing and classical computing together in one machine, but only uses quantum CPU (QPU) for certain operations [6].

Measurement Based: Starts with a fixed entangled state and then a sequence of measurements [16].

Quantum Computing Today

Quantum computers do exist today, but quantum computing is still in its infancy.

Often compared to the 1950’s early days of computing when everything new territory.

We are only just now discovering problems a quantum computer can solve, but a classical can’t (Google’s Quantum Supremacy).

No formalized assembly / machine language, though IBM and Rigetti have an proprietary assembly-like language.

Quantum Computing in Industry

Roswell Park Cancer Institute is using the D-Wave annealer to optimize IMRT radiation treatments [10].

Booz Allen Hamilton is using the D-Wave annealer to optimize satellite convergence [11].

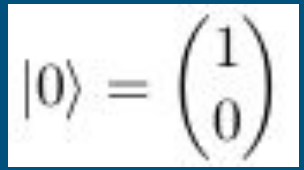
Rigetti has a gate model machine with experiments toward financial portfolio rebalancing*.

Accenture is utilizing 1Qbit for molecule comparisons to accelerate drug discovery for neurological and neurodegenerative conditions [13].

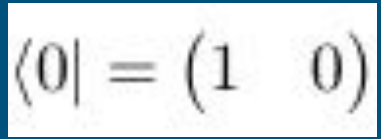
*https://arxiv.org/abs/1911.05296

Notations

A ket is used to represent a vector space as a column.



A bra is used to represent a vector space as a row.



While it may seem like a bra and ket are the same, they can have different uses individually -- hence why they are separate.

Qubits

A quantum bit or qubit is the quantum equivalent of a classical Shannon bit.

A classical bit is a boolean value represented by a 1 or 0 (True or False)

A qubit is a superposition of 0 and 1 at the same time where one number tells you the probability of being 0 and one tells you the probability of being a 1*.

100% chance of being 0, 0% chance of 1

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

0% chance of being 0, 100% chance of 1

In quantum mechanics, you will often see "0" being -1 and "1" being 1

Qubits and Brakets

Ket's and bra's are used to represent a single qubit (though ket's are the most common).

A single qubit is nothing but a vector space, as such we can start to do some math on them. We do an tensor (Kronecker) product on two qubits and we get a larger vector space**.

$$|0\rangle \otimes \langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

The result expands the size of our quantum system exponentially, in addition provides us with the resulting quantum state of the system.

***Literature often implies and omits the tensor symbol.*

$$|0\rangle \langle 0| = |0\rangle \otimes \langle 0|$$

Quantum Registers

Like classical computing, quantum computers have registers known as quantum registers.

These registers are made up of multiple qubits and calculations are done by manipulating the qubits within a quantum register.

Remember that qubits are represented by a vector space, so that means multiple qubits put together (tensored) make a quantum register. The quantum register is then represented by the resulting sized vector.

Quantum Logic Gates

Similar to classical computing, quantum computing also has logic gates known as quantum logic gates or quantum gates.

Quantum gates are a foundational part of a universal quantum computer.

Represented as a unitary matrix, which is a special type of square matrix.

Many gates are single qubit gates, but there are some that operate on more than one qubit (which then causes quantum entanglement).

Common Quantum Gates

Pauli Gates

X (NOT)

$$\mathrm{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Y (Rotate Y-Axis)

$$\mathrm{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

Z (Rotate Z-Axis / Phase Flip)

$$\mathrm{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Hadamard (Superposition)

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

SWAP

$$\mathrm{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

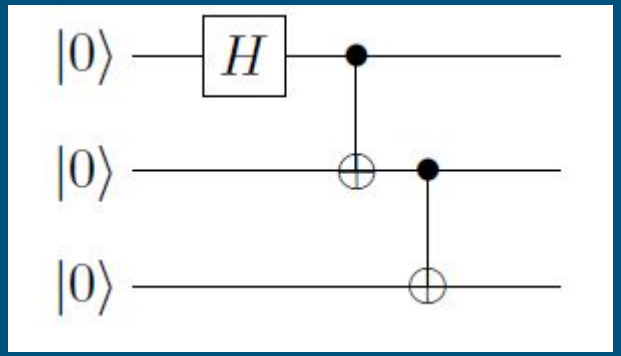
CNOT

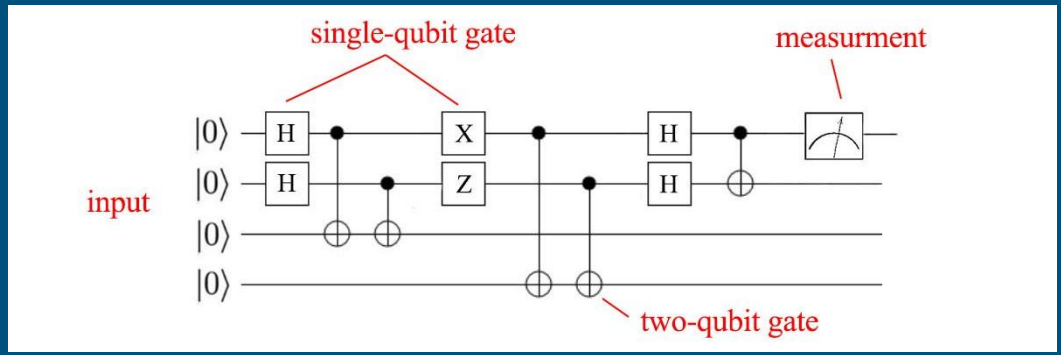
$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Quantum Circuit

Quantum gates are used to build a quantum circuit, much in the same fashion that classical logic gates are used to build a logic circuit.

Examples:





*Original image
https://medium.com/@jonathan_hui/qc-programming-with-quantum-gates-8996b667d256

Quantum Systems and Quantum States

A quantum system “is a portion of the whole universe taken to analyze the wave-particle duality in that system [3].” A quantum state is a particular state of everything going on in that system at that present time (similar to a snapshot).

A quantum computer is a quantum system and its state is a ‘snapshot’ of the all the qubit data and linear combinations being performed on that data.

You will see a lot of references to energy and amplitudes when reading about quantum computing...this is the quantum mechanics coming through...with quantum computing almost being a very thin layer on top.

Superposition

Created using the Hadamard gate.

Creates all permutation of values for the input.

Each of these permutations can be added together “superposed” to create a valid quantum state*.

Simply put, this is what allows us to have a qubit be a 1, 0, or 1 and 0 simultaneously.

*Griffiths, David J. (2004), *Introduction to Quantum Mechanics (2nd ed.)*, Prentice Hall, ISBN 978-0-13-111892-8

Quantum Entanglement

Quantum entanglement is where a particle interacts with another particle and one can’t tell their quantum states apart.

In quantum computing, this is done when we have a multi-qubit gate applied to at least two qubits.

Once applied, the original state of each of the original qubits can no longer be determined.

I.e. The CNOT gate is a gate that causes entanglement on two qubits.

Quantum Measurement

Quantum measurement is when we measure the state of a quantum system and is measured using an observable (which is represented as a matrix)

Once the measured state is returned, we also use that information to determine the probability of getting that state.

Noise in Quantum Systems

Two types: Classical and Quantum.

Classical noise is typically due to known factors such as vibrations, variations thermal energy in a laser emitter, but could be also due to unknown reasons.

Quantum noise is known as decoherence which simply put is the loss of quantum behavior.

While quantum physicists are interested in studying noise, computer scientists wish to reduce the noise as much as possible (to get the most accurate result).

Part $|001\rangle$: Ngrams

Ngrams And Why They Are Useful

Ngrams are a sequence of characters that represent something useful.

Used in a variety of domains such as NLP, machine learning, and DNA sequencing.

Used as building blocks to a larger structure used for classification.

Ngrams For Malware Analysis

Static analysis is still a valuable tool when analyzing malware.

Antivirus signature updates are not scalable and able to keep up.

We can use ngrams to represent known malicious byte sequences.

Each byte sequence could represent things such as malicious instructions to known malicious strings, even their location in the executable.

Using ngrams to create features from a PE32 executable has been proven to be very effective in malware detection [Shalaginov 2018].

Sample Ngrams

ngram	Literal
0x6261642e7275	“bad.ru”
0x41646a757374546f6b656e50	“AdjustTokenP”
0x646f776e6c6f61642e657865	“download.exe”
0x80	x86 INT

Hash Grams

While useful, n-grams for malware detection have known issues, especially for larger n-grams.

Utilizes a rolling hash for a set of n-grams in a document, and the top-k number of recurring n-grams are selected.

Has significantly less overhead than previous research.

Frequency of n-grams follows a Zipfan power-law distribution where frequency of an n-gram is inversely proportional to its rank in the table.

KiloGram

KiloGram [1] takes the concept of n-grams for malware analysis and scales them to even larger n-grams (larger than a 8gram).

While using a similar approach to hash grams, KiloGram incorporates a space saving data structure for storing the top-k n-grams.

This algorithm makes an additional pass over the data after initially hashed and inserts into the space saving data structure.

Also follows Zipfan distribution.

Creates ngrams from common byte sequences between malicious and benign software.

Uses a rolling hash to create a “hash gram” for an ngram. Also used to create a “whitelist” of what to process for the space saving data structure.

KiloGram can then be used to create a libsvm compatible dataset to be used for machine learning.

Part $|010\rangle$: Quantum Searching

Grover's Algorithm

One of the first quantum searching algorithms created by Lov K. Grover

Inspiration for Shor's factoring algorithm.

Has since been improved vastly over the years and widely researched.

Only more efficient on unsorted and unstructured data.

Makes heavy "use" and reference of the Bloch Sphere.

Bloch Sphere

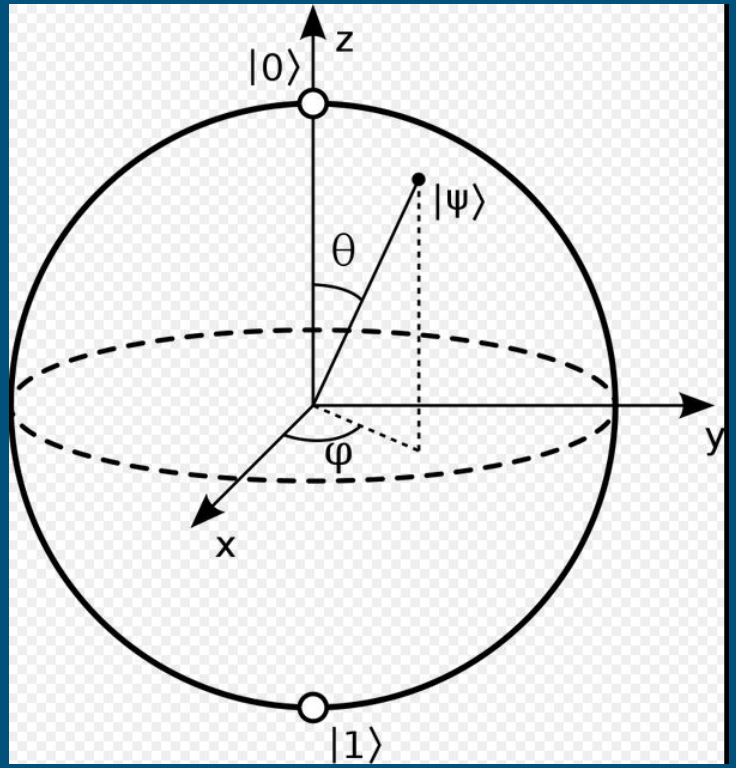
The Bloch Sphere is a geometrical representation of a two-level quantum mechanical system (qubits).

When thinking about quantum computing, it’s very helpful to utilize the Bloch Sphere as a model.

The Bloch sphere is also an intuitive way to look at particles and their respective spins.

We have several states which correspond to the Paul gates mentioned earlier. (X, Y, Z). We have “spin-up” , “spin-down”, “diagonal spin”, “rotations” etc

Bloch Sphere (Diagram)



* https://en.wikipedia.org/wiki/Bloch_sphere#/media/File:Bloch_sphere.svg

Amplitude Amplification

Grover’s utilizes amplitude amplification initially created by Gilles Brassard in 1997 and later that year by Grover.

In fact, amplitude amplification is more or less a generalization of Grover’s Algorithm.

The fundamental idea is to increase (amplify) the probabilities of the desired results, and this is accomplished by using a sequence of reflections.

Amplitude amplification causes the reflections to be rotated closer to the desired quantum state along the Bloch Sphere.

The target state is marked as $\sin^2(\Theta)$ and the algorithm is applied m-times. The target state is marked by an oracle.

The probability of obtaining the correct state is $\sin^2((2m+1)\Theta)$

Simply put, we have a target state on a sphere, we keep rotating it until we find the correct result. Each rotation brings us slightly closer.

Phase Flipping

Amplitude amplification incorporates what is known as phase flipping.

Performed by Pauli-Z and Control-Z gates.

Flips the sign of the amplitudes within the state vector.

For example, say we look at the following matrix and we wish to locate the value at row 1, column 3:

$$\begin{bmatrix} AB & CD & \mathbf{EF} \\ 12 & 97 & 85 \\ 2D & 3F & 9C \end{bmatrix}$$

Phase Flip

$$\begin{bmatrix} AB & CD & \mathbf{-EF} \\ 12 & 97 & 85 \\ 2D & 3F & 9C \end{bmatrix}$$

Problem Formation

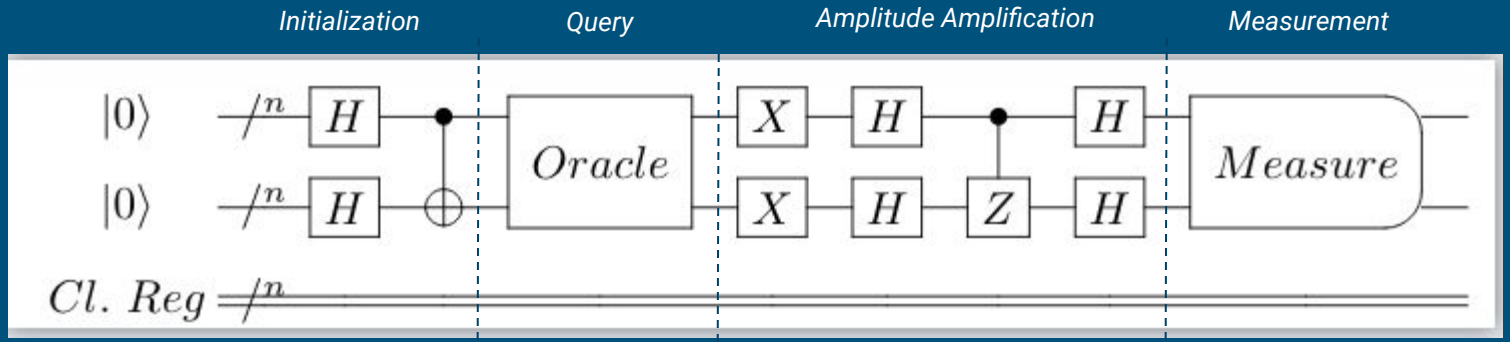
Before we can search for our target value, we have to change our thinking when it comes to searching an unstructured database.

Typically we want to get a known key and then we can look up its corresponding value. With Grover's, we flip this idea around.

We know the value we are looking for , but we don't actually know the key.

All key and value pairs are entangled.

Example Grover Quantum Circuit



**Example of quantum circuit representing Grover's algorithm on gate model machines.

Grover's Quantum Search

Initialization: All states "zeroed" out and entangled.

Amplitude Amplification (Repeated $\sqrt{N}$ times):

Oracle:

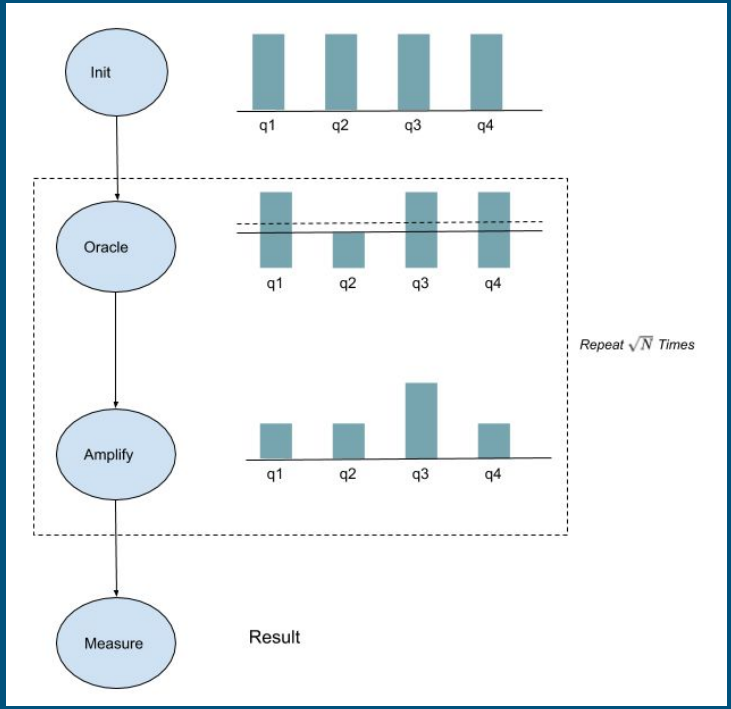
"Uncomputes" state based on target value.

Phase flips state "tags" that value.

"Re-computes" state to input state.

Amplify: Amplifies the amplitudes of the entangled quantum state. This separates the data we want from the data we don't.

Measurement: "Collapses" state of entanglement, have key and value pair with high probability.



**https://www.nature.com/articles/s41467-017-01904-7/figures/1

Part $|011\rangle$: Contribution

Using KiloGram Results

KiloGram being a scalable solution for large ngrams, was chosen to provide the analysis of benign vs. malicious software.

KiloGram has many options, including an option to get both the hash and ngram’s found.

Used KiloGram to output a hash table containing hashes as keys and ngrams as values.

Why rehash ?

The hashes generated are used internally by KiloGram, but discarded if ngrams are desired.

Common practice to simply run hashing algorithm again over list of ngrams if desired.

Keeping hashes can be beneficial if needed later, especially for larger ngrams.

Asymptotics

Grover's search only takes $O(\sqrt{N})$ number of lookups on real quantum hardware.

Classically, a linear value search would take $O(N)$ number of lookups

One could also potentially reverse the lookup table, but provided it's a single-linked list, this would take $O(N) + O(1) = O(N)$

Re-hashing everything would take $O(MN)$

Quantum Solution

Used the KiloGram resulting hash-table and use on a quantum computer.

Grover’s algorithm allows us to search an embedded hash-table on a quantum computer.

We have a known malicious list of ngrams and want to avoid rehashing.

For any quantum performance increase, data must be unstructured.

Quantum searching will provide a quadratic speedup over classical searching.

Part $|100\rangle$: Evaluation

Caveats

Limited availability of real quantum hardware (5 qubit and 14 qubit for IBMQ).

Grover’s by default not designed for quantum annealing (not a stochastic Hamiltonian).

Simulation much slower due to having to visit each of the permutations with every iteration. (Due to Grover’s not being able to take advantage of Schmidt decomposition) [Pendault 2017].

Hardware limitations for simulators allow for simulation of ~30 qubits.

Qrack Quantum Simulator

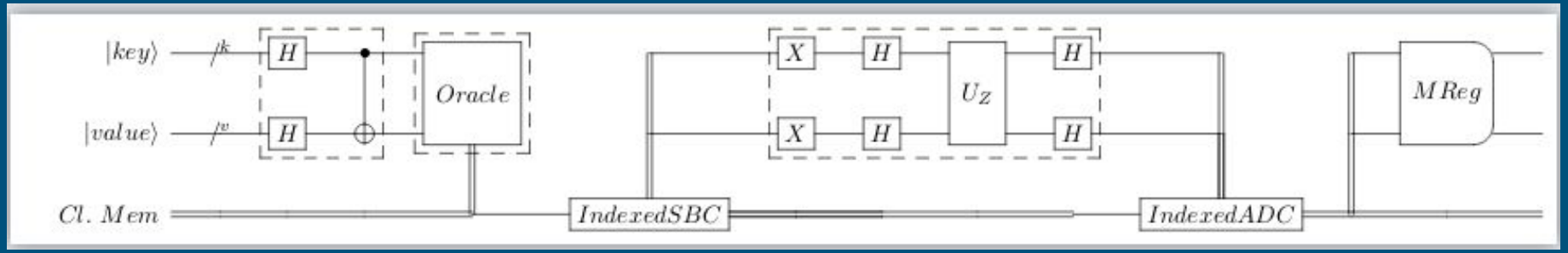
Open source, written in C++ by Dan Strano.

Takes advantage of GPU for processing using OpenCL library.

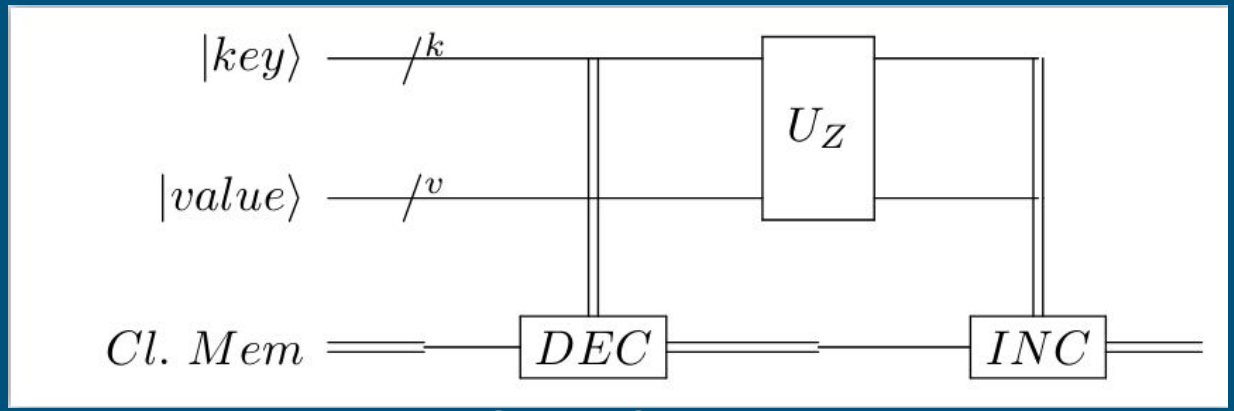
Based on VM6502 MOS 6502 CPU simulator, with quantum enhancements.

Updated frequently with many of the latest research enhancements.

Qrack Grover's Circuits



Grover's Circuit



Grover's Oracle

Sample Data Summary

Benign	Benign Files	Malicious	Malicious Files
Windows 7 System32	4565	Vxheaven 2015	284151
MAML	691	VirusShare 2018	131072

Sample Datasets

Benign	Malicious	N-gram Size	Kept n-grams
Windows 7 System32	Vxheaven 2015	3 bytes	64
Windows 7 System32	Vxheaven 2015	2 bytes	16384
Windows 7 System32	Vxheaven 2015	2 bytes	4096
MAML	VirusShare 2018	3 bytes	64
MAML	VirusShare 2018	2 bytes	2048
MAML	VirusShare 2018	2 bytes	1024

Memory Allocation For Simulation

Assuming double precision floating point

Let n be the number of qubits.

Let r be the number of real bytes in double precision.

Let i be the number of imaginary bytes in double precision.

We use $2^{n+(r+i)}$

Qubits	Real Bytes	Imaginary Bytes	Total Memory
4	2	2	256 bytes
8	2	2	4 KB
16	2	2	1 MB
24	2	2	≈ 268 MB
28	2	2	≈ 4 GB
30	2	2	≈ 16 GB
32	2	2	≈ 64 GB
40	2	2	≈ 17 TB

Number of Lookups

Let N be the number of ngrams

Number of Lookups = $\sqrt{N}$

Number of Ngrams	Number of Lookups
64	8
128	$\approx$ 12
256	12
512	$\approx$ 23
1024	32
2048	$\approx$ 46
4096	64
8192	$\approx$ 91
16384	128

Qrack Code (Oracle)

Algorithm 1 Qrack Implementation of Grover's Oracle

procedure TAGVALUE($tPerms, qReg, valueSt, valLen$)

$qReg \leftarrow DEC(tPerms, valueSt, valLen)$

$qReg \leftarrow ZeroPhaseFlip(tPrems, valueSt, valLen)$;

$qReg \leftarrow INC(tPrems, valueSt, valLen)$;

end procedure

Thesis page 32

Qrack Code (Amplitude Amplification)

Thesis page 33

Algorithm 2 Qrack Implementation of Amplitude Amplification

procedure AMPLITUDEAMPLIFIATION( )

$idxLen \leftarrow 10$

$valLen \leftarrow 16$

$cryIdx = idxLen + valLen$

$ngrams \leftarrow ngramtable[indexLength]$

$ngram \leftarrow 0xf3d7$

$qReg \leftarrow CreateQuantumInterface(*params)$

$qReg \leftarrow SetPermutation(0)$

$qReg \leftarrow H(valLen, idxLen)$

$qReg \leftarrow IndexedLDA(valLen, idxLen, 0, valLen, ngrams)$

for $i \leftarrow 0$ *to* $floor\left[\frac{\pi}{4\arcsin^2(\frac{1}{\sqrt{2^N}})}\right]$ **do**

TagValue(ngram, qReg, 0, valLen)

$qReg \leftarrow X(cryIdx)$

$qReg \leftarrow IndexedSBC(valLen, idxLen, 0, valLen, cryIdx, ngrams)$

$qReg \leftarrow X(cryIdx)$

$qReg \leftarrow H(valLen, idxLen)$

$qReg \leftarrow ZeroPhaseFlip(valLen, idxLen)$

$qReg \leftarrow H(valLen, idxLen)$

$qReg \leftarrow IndexedADC(valLen, idxLen, 0, valLen, cryIdx, ngrams)$

Return $qReg \leftarrow MReg(17, 27)$

end procedure

Short Example

Known ngram: 0xf3d7

Iterations: $floor\left[\frac{\pi}{4\arcsin^2\left(\frac{1}{\sqrt{2^N}}\right)}\right]$

Resulting hash: 0x3a9

```
Device #0, Built JIT.
Device #1, Built JIT.
Default platform: NVIDIA CUDA
Default device: GeForce GTX 1660
OpenCL device #0: GeForce GTX 1660
OpenCL device #1: GeForce GTX 1660
         0> chance of match:0.00876619
         1> chance of match:0.0242241
         2> chance of match:0.0471087
         3> chance of match:0.0770629
         4> chance of match:0.113619
         5> chance of match:0.156206
         6> chance of match:0.20416
         7> chance of match:0.256732
         8> chance of match:0.313102
         9> chance of match:0.372389
        10> chance of match:0.433668
        11> chance of match:0.495982
        12> chance of match:0.558359
        13> chance of match:0.619825
        14> chance of match:0.67942
        15> chance of match:0.736215
        16> chance of match:0.789322
        17> chance of match:0.837913
        18> chance of match:0.881229
        19> chance of match:0.918595
        20> chance of match:0.949426
        21> chance of match:0.973241
        22> chance of match:0.98967
        23> chance of match:0.998456
        24> chance of match:0.99946
After measurement (of value, key, or both):
Chance of match:1
Ngram: f3d7
Hash: 3a9
Total Iterations: 25
```

Part $|101\rangle$: Future Work and Conclusion

Conclusions

Quantum computing is very active in development and are making great strides in multiple domains.

All sub-domains within cyber security could potentially benefit in some way from making use of a full or partial quantum algorithms.

While the future is uncertain, the past has proven that there will always be a threat from malicious software.

Quantum computing will also be used by the malicious software creators and potentially already has begun.

Future Work

These experiments can be applied to any domain where n-grams are used.

Could also be utilized with other hashing methods (such as when looking up a known malicious SHA-256 file hash).

Working on an enhanced Grover's algorithm that can take advantage of Schmidt decomposition for better simulator performance.

Expand KiloGram further to utilize a variant of Grover's to pick top-k hashes instead of quick select.

Questions ?

Thanks!

Acknowledgements

Dr. Charles Nicholas

Dr. Samuel Lomanoco

Dr. Edward Raff (KiloGram)

Mr. Ajinkya Borle (Revisions / Mentor)

Mr. RJ Joyce (Revisions)

Mr. Dan Strano (Qrack Simulator / Mentor)

Master's Research
https://github.com/nallg00d

References

[1] Edward Raff, William Fleming, Richard Zak, Hyrum Anderson, Bill Fin-layson, Charles K. Nicholas, and Mark Mclean. KiloGrams: Very LargeN-Grams for Malware Classification. InProceedings of KDD 2019Workshop on Learning and Mining for Cybersecurity (LEMINCS'19),2019.

[2] D.R. Simon. On the power of quantum computing. InFoundationsof Computer Science, 1994 Proceedings., 35th Annual Symposium on:116–123, 1994.

[3] Lov K. Grover. A fast quantum mechanical algorithm for databasesearch.Proceedings of the twenty-eighth annual ACM symposium onTheory of computing - STOC '96, 1996.

[4] IBM. IBM quantum experience. https://quantum-computing.ibm.com,2020.

[5] D-Wave. D-wave. https://dwavesys.com, 2020.

[6] Daniel Strano and Benn Bollay. Qrack a comprehensive, gpu accel-erated framework for developing universal virtual quantum processors.https://github.com/vm6502q/qrack, 2020.

[7] Andrii Shalaginov, Sergii Banin, Ali Dehghantanha, and Katrin Franke.Machine learning aided static malware analysis: A survey and tutorial.Cyber Threat Intelligence, page 7–45, 2018.

[8] Edward Raff and Charles Nicholas. Hash-grams: Faster n-gram featuresfor classification and malware detection. InProceedings of the ACMSymposium on Document Engineering 2018, DocEng '18, New York,NY, USA, 2018. Association for Computing Machinery.

[9] P.W. Shor. Algorithms for quantum computation: discrete logarithmsand factoring.Proceedings 35th Annual Symposium on Foundations ofComputer Science, Santa Fe, NM, pages 124–134, 1994.

[10] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantumamplitude amplification and estimation.Quantum Computation andInformation, page 53–74, 2002.

[11] D. Coppersmith. An approximate fourier transform useful in quantumfactoring, 2002.

[12] Walter Rudin.Real and Complex Analysis, 3rd Ed.McGraw-Hill, Inc.,USA, 1987.

[13] DanielStranoandBennBollay.Vm6502qandqrack.https://vm6502q.readthedocs.io/en/latest/index.html, 2020.

[14] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph Bardin,Rami Barends, Rupak Biswas, Sergio Boixo, Fernando Brandao, DavidBuell, Brian Burkett, Yu Chen, Jimmy Chen, Ben Chiaro, RobertoCollins, William Courtney, Andrew Dunsworth, Edward Farhi, BrooksFoxen, Austin Fowler, Craig Michael Gidney, Marissa Giustina, RobGraff, Keith Guerin, Steve Habegger, Matthew Harrigan, MichaelHartmann, Alan Ho, Markus Rudolf Hoffmann, Trent Huang, TravisHumble, Sergei Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyan-tyn Kechedzhi, Julian Kelly, Paul Klimov, Sergey Knysh, AlexanderKorotkov, Fedor Kostritsa, Dave Landhuis, Mike Lindmark, Erik Lucero,Dmitry Lyakh, Salvatore Mandr`a, Jarrod Ryan McClean, MatthewMcEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, MasoudMohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill,Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John Platt, ChrisQuintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas Rubin, DanielSank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin Jeffery Sung, MattTrevithick, Amit Vainsencher, Benjamin Villalonga, Ted White, Z. JamieYao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John Martinis.Quantum supremacy using a programmable superconducting processor.Nature, 574:505–510, 2019.

[15] Michael A. Nielsen and Isaac L. Chuang.Quantum Computation and Quantum Information. Cambridge University Press, 2000.

[16] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.Introduction to Algorithms, pages 911–916. MIT Press, seconde dition, 1990.

[17] David Deutsch and Richard Jozsa. Rapid Solution of Problems by Quantum Computation.Proceedings of the Royal Society of London Series A, 439(1907):553–558, December 1992.

[18] Chao-Yang Pang, Ri-Gui Zhou, Cong-Bao Ding, and Ben-Qiong Hu.Quantum search algorithm for set operation.Quantum Information Processing, 12(1):481–492, Mar 2012.

[19] Patrick J. Coles, Stephan Eidenbenz, Scott Pakin, Adetokunbo Ade-doyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Andrey Lokhov, Alexander Ma-lyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Dan O'Malley, Diane Oyen, Lakshman Prasad, Randy Roberts, Phil Romero,Nandakishore Santhi, Nikolai Sinitsyn, Pieter Swart, Marc Vuffray, JimWendelberger, Boram Yoon, Richard Zamora, and Wei Zhu. Quantum algorithm implementations for beginners, 2018.

[20] Nicolas J. Cerf and Chris Adami. Information theory of quantum entanglement and measurement.Physica D: Nonlinear Phenomena,120(1-2):62–81, Sep 1998.

[21] G. Brassard and P. Hoyer. An exact quantum polynomial-time algorithm for simon's problem.Proceedings of the Fifth Israeli Symposium onTheory of Computing and Systems.

[22] Sachin Jain and Yogesh Kumar Meena. Byte level n–gram analysis for malware detection. In K. R. Venugopal and L. M. Patnaik, editors,Computer Networks and Intelligent Computing, pages 51–59, Berlin,Heidelberg, 2011. Springer Berlin Heidelberg

[23] LibSVM. https://www.csie.ntu.edu.tw/~cjlin/libsvm/