Cost estimates for quantum preimage attacks

Matt Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, John Schanck

Institute for Quantum Computing
University of Waterloo

February 26, 2016
Given a bijection

$$H : \{0, 1\}^k \rightarrow \{0, 1\}^k,$$

Grover’s algorithm finds the preimage of

$$y \in \{0, 1\}^k$$

using $$\Theta(\sqrt{2^k})$$ queries to an oracle that computes

$$f : \{0, 1\}^k \rightarrow \{0, 1\}$$

$$x \mapsto \delta(H(x), y).$$
To estimate the cost of Grover’s algorithm…

Write down a circuit.

\[ |0\rangle \quad \xrightarrow{H} \quad G \quad G \quad G \quad \ldots \]

\[ |1\rangle \quad \xrightarrow{H} \quad G \quad G \quad \ldots \]
To estimate the cost of Grover’s algorithm...

\[ |0\rangle \xrightarrow{H} \]
\[ |1\rangle \xrightarrow{H} \]

You’ll need the cost of one Grover iteration

\[ |x\rangle \xrightarrow{H} \]
\[ |0\rangle \]
\[ |\rangle \]

So you want to break a hash function...
To estimate the cost of Grover’s algorithm...

\[ |0\rangle \quad \begin{array}{c} H \end{array} \quad G \quad G \quad G \quad \cdots \]
\[ |1\rangle \quad \begin{array}{c} H \end{array} \quad \begin{array}{c} \begin{array}{c} \text{…} \end{array} \end{array} \]

i.e. the diffusion operator

\[ |x\rangle \quad \begin{array}{c} H \end{array} \quad \begin{array}{c} 2|0\rangle\langle 0| - I \end{array} \quad H \quad \begin{array}{c} \text{Of} \end{array} \quad |x\rangle \]
\[ |0\rangle \quad \begin{array}{c} \text{Of} \end{array} \quad |0\rangle \]
\[ |\rangle \quad |\rangle \]
To estimate the cost of Grover’s algorithm…

\[ |0\rangle \xrightarrow{H} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \]

and the oracle.

\[ |x\rangle \xrightarrow{O_f} |x\rangle \]

\[ |0\rangle \xrightarrow{H} 2|0\rangle\langle 0| - I \]

So you want to break a hash function…
To estimate the cost of Grover’s algorithm...

The oracle needs to be instantiated

\[ O_f \rightarrow U_f \]
To estimate the cost of Grover’s algorithm...

The oracle needs to be instantiated

\[ |x\rangle \xrightarrow{U_H} |x\rangle \]
\[ |0\rangle \xrightarrow{X_y} |a\rangle \]
\[ |\text{-}\rangle \xrightarrow{X_y} |\text{-}\rangle \]

with a reversible implementation of H.

\[ |x\rangle \xrightarrow{U_H} |a \oplus H(x)\rangle \]
The logical layer

1. Compile to a universal gate set (Clifford+T)
2. Optimize the circuit (minimize T-count)
Our contribution:

1. T-count optimized reversible implementations of SHA-2 and SHA-3 functions.

<table>
<thead>
<tr>
<th></th>
<th>#gates</th>
<th>depth</th>
<th>#qubits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>Clifford</td>
<td>T</td>
</tr>
<tr>
<td>SHA-256</td>
<td>193,280</td>
<td>4,510,960</td>
<td>88,256</td>
</tr>
<tr>
<td>SHAKE-256</td>
<td>499,200</td>
<td>34,030,165</td>
<td>576</td>
</tr>
</tbody>
</table>
The fault-tolerant layer

*Without significant future effort, the classical processing will almost certainly limit the speed of any quantum computer, particularly one with intrinsically fast quantum gates.*

The fault-tolerant layer

Our contribution:

2. Cost model for comparing Grover against classical brute force search.
The fault-tolerant layer

1. Assume surface code quantum computing.
2. Estimate additional resources required by fault tolerance layer, e.g. magic state distillation factories.
3. Cost only the classical resources.
4. Assume 1 classical core per logical qubit.
5. Assume 1 surface code cycle $\approx$ 1 application of $H$.
6. Compute cost as surface code cycles $\times$ logical qubits.
The oracular layer
The reversible layer
The logical layer
The fault-tolerant layer

The fault-tolerant layer

See our forthcoming paper for details.

Thanks!