A framework for reducing the overhead of the quantum oracle for use with Grover's algorithm with applications to cryptanalysis of SIKE

Jean-François Biasse[1], **Benjamin Pring**[1]

18th August 2019

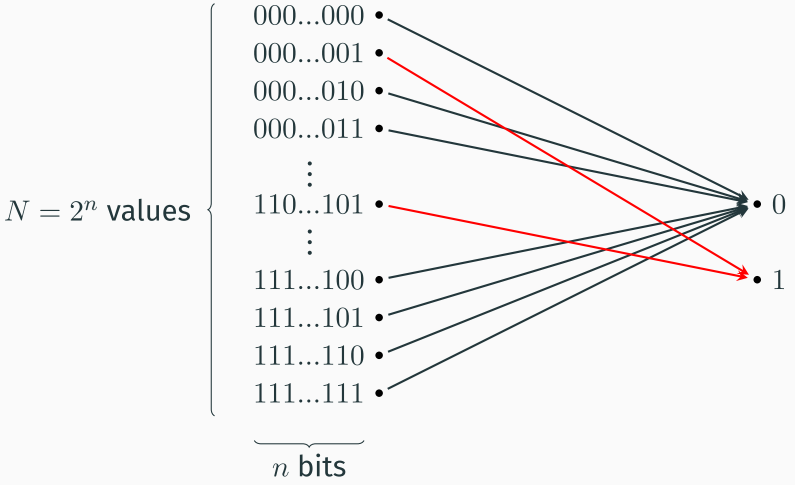
[1]University of South Florida

Overview of the talk

1. Motivation
2. Grover's algorithm — what you need to know
3. **A framework for preprocessing quantum oracles**
4. **Applications**
5. Conclusions

The search problem

$N = 2^n$ items and there exist M items that satisfies a property



The search problem

Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ be such that

$$\chi(x) \mapsto \begin{cases} 1 & \text{if } x \text{ is one of the } M \text{ items we are looking for} \\ 0 & \text{otherwise.} \end{cases}$$

and say we have a circuit that implements χ.

$$x \;-\!\!\boxed{\mathcal{O}_\chi}\!\!-\; \chi(x)$$

Classical queries required: $O(\frac{N}{M})$ (exhaustive search)

Quantum queries required: $O(\sqrt{\frac{N}{M}})$ (Grover's algorithm)

The search problem

Let $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ be such that M elements satisfy $\chi(x) = 1$

$$x \;—\boxed{\mathcal{O}_\chi}—\; \chi(x)$$

Classical queries required: $O(\frac{N}{M})$ (exhaustive search)

Quantum queries required: $O(\sqrt{\frac{N}{M}})$ (Grover's algorithm)

$$\left.\begin{array}{ll}\text{Cost of classical search:} & O\left(\frac{N}{M} \cdot poly(n)\right) \\ \text{Cost of quantum search:} & O\left(\sqrt{\frac{N}{M}} \cdot poly(n)\right)\end{array}\right\} \mathcal{O}_\chi \text{ costs } poly(n) \text{ gates}$$

Quantum states and the computational basis

An n-qubit quantum state can be written in the *computational basis* as

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \qquad \text{where } \alpha_x \in \mathbb{C} \quad \text{and} \quad \sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1.$$

Quantum states and the computational basis

An n-qubit quantum state can be written in the *computational basis* as

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle \qquad \text{where } \alpha_x \in \mathbb{C} \quad \text{and} \quad \sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1.$$

Natural intepretation = superposition of bitstrings.

Quantum states and the computational basis

An n-qubit quantum state can be written in the *computational basis* as

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle \qquad \text{where } \alpha_x \in \mathbb{C} \quad \text{and} \quad \sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1.$$

Natural intepretation = superposition of bitstrings.

Measurement in the computational basis results in $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$. Crucially, measurement collapses the state to $1 \cdot |x\rangle$.

Quantum states and the computational basis

An n-qubit quantum state can be written in the *computational basis* as

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \qquad \text{where } \alpha_x \in \mathbb{C} \quad \text{and} \quad \sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1.$$

Natural intepretation = superposition of bitstrings.

Measurement in the computational basis results in $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$. Crucially, measurement collapses the state to $1 \cdot |x\rangle$.

Quantum algorithm design: evolving an initial quantum state to one whose amplitudes which encode useful information are amplified.

Quantum states and quantum gates I

Quantum states can be viewed as vectors of coefficients.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \tag{1}$$

Quantum states and quantum gates I

Quantum states can be viewed as vectors of coefficients.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \tag{1}$$

Processes which evolve one quantum state to another in a period of continuous time are *unitary operators* acting upon these vectors.

$$U|\psi_{t_0}\rangle = |\psi_{t_1}\rangle \qquad \text{where } U^\dagger U = UU^\dagger = I \tag{2}$$

Note: $U^\dagger = (U^T)^*$ — the conjugate transpose.

Quantum states and quantum gates I

Quantum states can be viewed as vectors of coefficients.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \tag{1}$$

Processes which evolve one quantum state to another in a period of continuous time are *unitary operators* acting upon these vectors.

$$U|\psi_{t_0}\rangle = |\psi_{t_1}\rangle \qquad \text{where } U^\dagger U = UU^\dagger = I \tag{2}$$

Note: $U^\dagger = (U^T)^*$ — the conjugate transpose.

This connection to linear algebra both simplifies and complicates the implementation of Grover's algorithm.

Quantum states and quantum gates III

A key component of Grover is the concept of the *quantum oracle*, a unitary operator defined by a boolean function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$.

$$\chi(x) = \begin{cases} 1 & \text{if } x \text{ is a target} \\ 0 & \text{otherwise} \end{cases} \qquad \mathcal{O}_\chi \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } \chi(x) = 1 \\ \left|x\right\rangle & \text{if } \chi(x) = 0 \end{cases}$$

Quantum states and quantum gates III

A key component of Grover is the concept of the *quantum oracle*, a unitary operator defined by a boolean function $\chi : \{0,1\}^n \longrightarrow \{0,1\}$.

$$\chi(x) = \begin{cases} 1 & \text{if } x \text{ is a target} \\ 0 & \text{otherwise} \end{cases} \qquad \mathcal{O}_\chi |x\rangle = \begin{cases} -|x\rangle & \text{if } \chi(x) = 1 \\ |x\rangle & \text{if } \chi(x) = 0 \end{cases}$$

Linearity of unitary operators reduces the design of quantum oracles to the problem of implementing a circuit that is correct on bitstrings

$$\mathcal{O}_\chi \left(\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \right) = \sum_{x \in \{0,1\}^n} \alpha_x \mathcal{O}_\chi |x\rangle$$

Quantum states and quantum gates IV

A unitary operator implementing $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ on bitstrings is enough to realise $\mathcal{O}_\chi$.

Say we have the unitary $\mathcal{O}_\chi^{(b)}$

$$\mathcal{O}_\chi^{(b)} |x\rangle |y\rangle \mapsto |x\rangle |y \oplus \chi(x)\rangle$$

Quantum states and quantum gates IV

A unitary operator implementing $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ on bitstrings is enough to realise $\mathcal{O}_\chi$.

Say we have the unitary $\mathcal{O}_\chi^{(b)}$

$$\mathcal{O}_\chi^{(b)} |x\rangle |y\rangle \mapsto |x\rangle |y \oplus \chi(x)\rangle$$

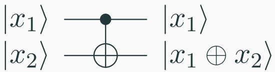
then replacing y with the state (where H is the Hadamard gate)

$$H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

gives us

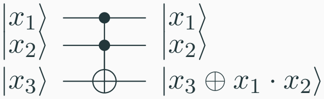
$$\mathcal{O}_\chi^{(b)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \mapsto |x\rangle \left(\frac{|0 \oplus \chi(x)\rangle - |1 \oplus \chi(x)\rangle}{\sqrt{2}}\right) = (-1)^{\chi(x)} |x\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).$$

Building reversible boolean circuits from logical quantum gates



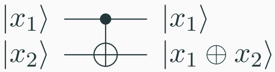
A CNOT gate acting as $x \oplus y$.

An X gate acting as negation.



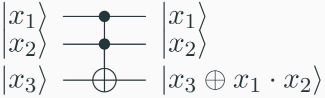
A Toffoli gate acting as $x \cdot y$.

Building reversible boolean circuits from logical quantum gates



A CNOT gate acting as $x \oplus y$.

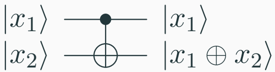
An X gate acting as negation.



A Toffoli gate acting as $x \cdot y$.

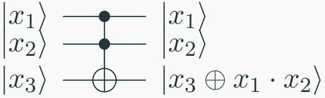
Classicial universal gate set: $\{\oplus, \neg, \wedge\} \leftrightarrow \{$CNOT$, X,$ Toffoli$\}$

Building reversible boolean circuits from logical quantum gates

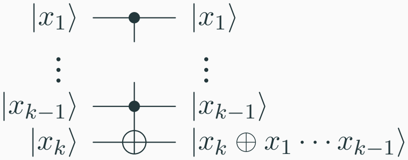


A CNOT gate acting as $x \oplus y$.

An X gate acting as negation.



A Toffoli gate acting as $x \cdot y$.



A k-bit Toffoli gate acting as $\bigwedge_{i=1}^{k-1} x_i$.

Classicial universal gate set: $\{\oplus, \neg, \wedge\} \leftrightarrow \{$CNOT$, X,$ Toffoli$\}$

Reversibility

Implementations of boolean circuits are required to be *reversible* because of the unitary condition

$$U^\dagger U = UU^\dagger = I.$$

Impact: all boolean circuits must implement permutations.

Use of ancillae qubits is crucial for efficient realisation.

Grover basics

Grover's algorithm consists of the following steps.

1. Initialise the quantum register to $|0^n\rangle$
2. Apply the Hadamard transform to compute $|\psi_0\rangle = H^{\otimes n} |0^n\rangle$
3. Compute $|\psi_k\rangle = G^k |\psi_0\rangle$, via successive applications of $G = \mathcal{R}_\psi \mathcal{O}_\chi$.
4. Perform a measurement in the computational basis.

If $k = \left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rfloor$, then with high probability measurement will collapse the state to an element $x \in \{0,1\}^n$ that we are searching for.

Grover's quantum search algorithm and query complexity VI

Let $E_{\mathcal{A}}$ be the cost of implementing the unitary/circuit $\mathcal{A}$.

The cost (circuit-depth or circuit-size) of Grover's algorithm is

$$\left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rfloor \cdot \left(E_{\mathcal{O}_\chi} + E_{\mathcal{R}_\psi} \right)$$

and usually $E_{\mathcal{O}_\chi} \gg E_{\mathcal{R}_\psi}$.

The number of qubits required is dependent upon the circuit-width of the quantum oracle and is at least n.

Grover's quantum search algorithm and query complexity VII

Let $N = 2^{100}$ and $M = 1$.

Say $E_{\mathcal{O}_\chi} + E_{\mathcal{R}_\psi} \approx E_{\mathcal{O}_\chi} = n^3$

- Query complexity advantage: 2^{50} versus 2^{100}.
- Actual advantage: $2^{69.93}$ versus $2^{119.93}$.
- Each quantum operation will be slower and more expensive.
- Advantageous and easy to run classical search in parallel.
- Disadvantageous to run quantum search in parallel.

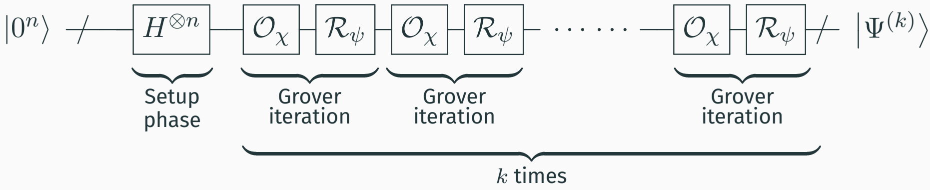
$$\left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rfloor \cdot \left(E_{\mathcal{O}_\chi} + E_{\mathcal{R}_\psi} \right). \tag{3}$$

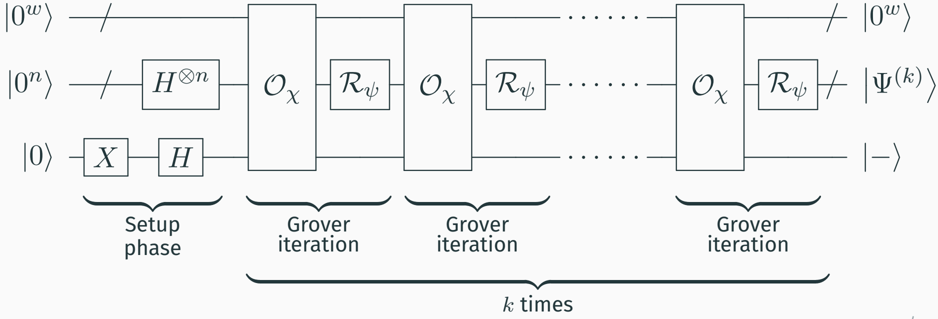
How to optimise?

- Optimise the circuit for $\mathcal{O}_\chi$ (sometimes involves a tradeoff).
- Use fewer Grover iterations (lower success probability).
- Tradeoff between information obtained and complexity.

Grover 101: Circuits for unitaries

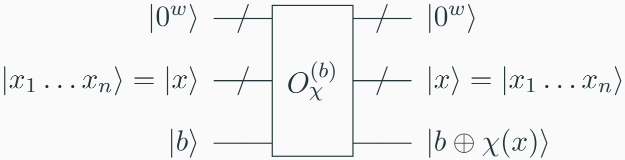
$$\left|\Psi^{(k)}\right\rangle = \left(\mathcal{R}_\psi \mathcal{O}_\chi\right)^k H^{\otimes n} \left|0^n\right\rangle$$





Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$

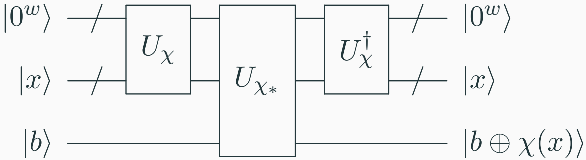
$$\mathcal{O}_\chi^{(b)} |0^w\rangle |x\rangle |b\rangle \mapsto |0^w\rangle |x\rangle |b \oplus \chi(x)\rangle$$



- $\chi : \{0,1\}^n \longrightarrow \{0,1\}$

Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$

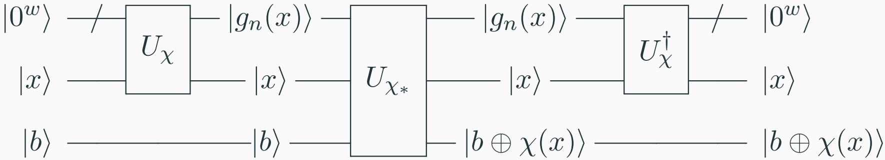
$$U_\chi^\dagger U_{\chi*} U_\chi |0^w\rangle |x\rangle |b\rangle \mapsto |0^w\rangle |x\rangle |b \oplus \chi(x)\rangle$$



- $\chi : \{0,1\}^n \longrightarrow \{0,1\}$

Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$

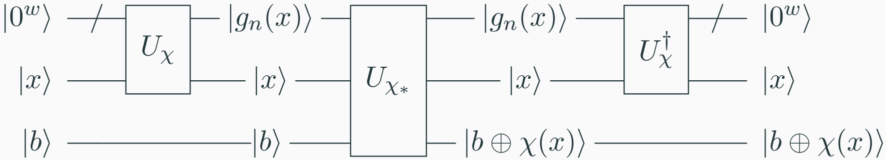
$$U_\chi^\dagger U_{\chi*} U_\chi \left|0^w\right\rangle \left|x\right\rangle \left|b\right\rangle \mapsto \left|0^w\right\rangle \left|x\right\rangle \left|b \oplus \chi(x)\right\rangle$$



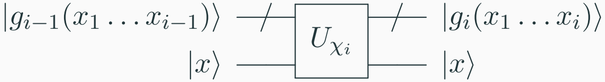
- $\chi : \{0,1\}^n \longrightarrow \{0,1\}$
- $g_n : \{0,1\}^n \longrightarrow \{0,1\}^w$ depends upon $x_1 \dots x_n \in \{0,1\}^n$.

Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$

$$U_\chi^\dagger U_{\chi*} U_\chi \left|0^w\right\rangle \left|x\right\rangle \left|b\right\rangle \mapsto \left|0^w\right\rangle \left|x\right\rangle \left|b \oplus \chi(x)\right\rangle$$

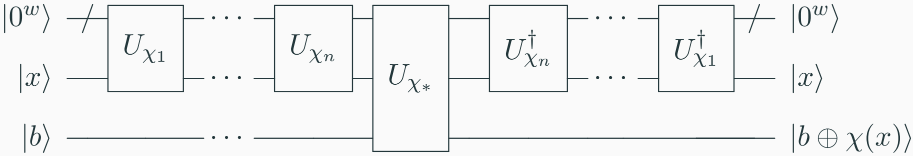


- $\chi : \{0,1\}^n \longrightarrow \{0,1\}$
- $g_i : \{0,1\}^i \longrightarrow \{0,1\}^w$ depends only upon $x_1 \dots x_i \in \{0,1\}^i$.

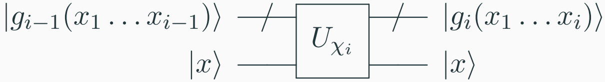


Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$

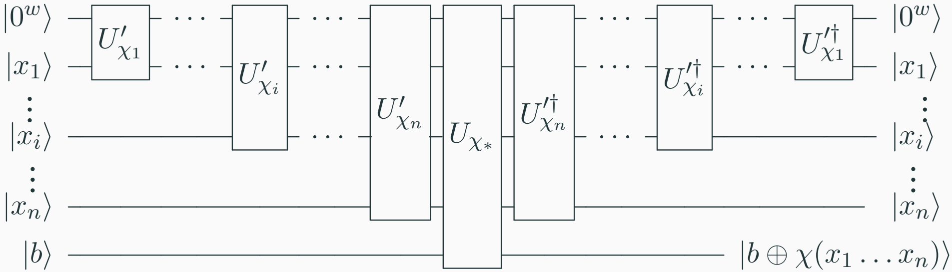
$$U_{\chi_1}^\dagger \cdots U_{\chi_n}^\dagger U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1} |0^w\rangle |x\rangle |b\rangle \mapsto |0^w\rangle |x\rangle |b \oplus \chi(x)\rangle$$



- $\chi : \{0,1\}^n \longrightarrow \{0,1\}$
- $g_i : \{0,1\}^i \longrightarrow \{0,1\}^w$ depends only upon $x_1 \ldots x_i \in \{0,1\}^i$.



Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$



Definition (Bitwise decomposition of $\mathcal{O}_\chi^{(b)}$)

The sequence of $n+1$ unitaries $U_{\chi_1}, \dots, U_{\chi_n}, U_{\chi_*}$ is a *bitwise decomposition of* $\mathcal{O}_\chi^{(b)}$ if $\left(\mathcal{I}^{\otimes w} \otimes O_\chi^{(b)}\right) = U^\dagger_{\chi_n} \cdots U^\dagger_{\chi_1} U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1}$ where $U_{\chi_i} = U'_\chi \otimes \mathcal{I}^{\otimes n-i+1}$ and

$$U'_{\chi_i} \, |g_{i-1}(x_1 \dots x_{i-1})\rangle \, |x_1 \dots x_i\rangle \mapsto |g_i(x_1 \dots x_i)\rangle \, |x_1 \dots x_i\rangle$$
$$U_{\chi_*} \, |g_n(x_1 \dots x_n)\rangle \, |x_1 \dots x_n\rangle \, |b\rangle \mapsto |g_n(x_1 \dots x_n)\rangle \, |x_1 \dots x_n\rangle \, |b \oplus \chi(x_1 \dots x_n)\rangle$$

Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$: sanity check

Definition (Bitwise decomposition of $\mathcal{O}_\chi^{(b)}$)

$U_{\chi_1}, \ldots, U_{\chi_n}, U_{\chi_*}$ is a *bitwise decomposition of* $\mathcal{O}_\chi^{(b)}$ if

$$\left(\mathcal{I}^{\otimes w} \otimes O_\chi^{(b)}\right) = U_{\chi_n}^\dagger \cdots U_{\chi_1}^\dagger U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1}$$

where $U_{\chi_i} = U'_\chi \otimes \mathcal{I}^{\otimes n-i+1}$ and

$U'_{\chi_i} \left|g_{i-1}(x_1 \ldots x_{i-1})\right\rangle \left|x_1 \ldots x_i\right\rangle \mapsto \left|g_i(x_1 \ldots x_i)\right\rangle \left|x_1 \ldots x_i\right\rangle$
$U_{\chi_*} \left|g_n(x_1 \ldots x_n)\right\rangle \left|x_1 \ldots x_n\right\rangle \left|b\right\rangle \mapsto \left|g_n(x_1 \ldots x_n)\right\rangle \left|x_1 \ldots x_n\right\rangle \left|b \oplus \chi(x_1 \ldots x_n)\right\rangle$

Trivial decomposition: if we have circuit for $\mathcal{O}_\chi^{(b)}$ using $w' \leq w$ ancilla

$$U'_{\chi_i} = \mathcal{I}^{\otimes w+n-i} \qquad \text{and} \qquad U_{\chi_*} = \mathcal{I}^{\otimes w-w'} \otimes \mathcal{O}_\chi^{(b)}$$

Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$: sanity check

Definition (Bitwise decomposition of $\mathcal{O}_\chi^{(b)}$)

$U_{\chi_1}, \ldots, U_{\chi_n}, U_{\chi_*}$ is a *bitwise decomposition of* $\mathcal{O}_\chi^{(b)}$ if

$$\left(\mathcal{I}^{\otimes w} \otimes O_\chi^{(b)}\right) = U_{\chi_n}^\dagger \cdots U_{\chi_1}^\dagger U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1}$$

where $U_{\chi_i} = U'_\chi \otimes \mathcal{I}^{\otimes n-i+1}$ and

$U'_{\chi_i} \left|g_{i-1}(x_1 \ldots x_{i-1})\right\rangle \left|x_1 \ldots x_i\right\rangle \mapsto \left|g_i(x_1 \ldots x_i)\right\rangle \left|x_1 \ldots x_i\right\rangle$
$U_{\chi_*} \left|g_n(x_1 \ldots x_n)\right\rangle \left|x_1 \ldots x_n\right\rangle \left|b\right\rangle \mapsto \left|g_n(x_1 \ldots x_n)\right\rangle \left|x_1 \ldots x_n\right\rangle \left|b \oplus \chi(x_1 \ldots x_n)\right\rangle$

$$f(x_1, x_2, x_3, x_4, x_5) = x_1x_2 + x_1x_5 + x_3x_5 + x_3x_4 + x_2 + x_4 + x_5 + 1$$

Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$: sanity check

Definition (Bitwise decomposition of $\mathcal{O}_\chi^{(b)}$)

$U_{\chi_1}, \ldots, U_{\chi_n}, U_{\chi_*}$ is a *bitwise decomposition of* $\mathcal{O}_\chi^{(b)}$ if

$$\left(\mathcal{I}^{\otimes w} \otimes O_\chi^{(b)}\right) = U_{\chi_n}^\dagger \cdots U_{\chi_1}^\dagger U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1}$$

where $U_{\chi_i} = U'_\chi \otimes \mathcal{I}^{\otimes n-i+1}$ and

$$U'_{\chi_i} \left|g_{i-1}(x_1 \ldots x_{i-1})\right\rangle \left|x_1 \ldots x_i\right\rangle \mapsto \left|g_i(x_1 \ldots x_i)\right\rangle \left|x_1 \ldots x_i\right\rangle$$
$$U_{\chi_*} \left|g_n(x_1 \ldots x_n)\right\rangle \left|x_1 \ldots x_n\right\rangle \left|b\right\rangle \mapsto \left|g_n(x_1 \ldots x_n)\right\rangle \left|x_1 \ldots x_n\right\rangle \left|b \oplus \chi(x_1 \ldots x_n)\right\rangle$$

$$f(x_1, x_2, x_3, x_4, x_5) = 1 + x_2 \cdot \underbrace{(x_1 + 1)}_{y_1} + x_4 \cdot \underbrace{(x_3 + 1)}_{y_4} + x_5 \cdot \underbrace{(1 + x_1 + x_3)}_{y_5}$$

Structure in the quantum bit oracle $\mathcal{O}_\chi^{(b)}$: sanity check

Definition (Bitwise decomposition of $\mathcal{O}_\chi^{(b)}$)

$U_{\chi_1}, \ldots, U_{\chi_n}, U_{\chi_*}$ is a *bitwise decomposition of* $\mathcal{O}_\chi^{(b)}$ if

$$\left(\mathcal{I}^{\otimes w} \otimes O_\chi^{(b)}\right) = U_{\chi_n}^\dagger \cdots U_{\chi_1}^\dagger U_{\chi_*} U_{\chi_n} \cdots U_{\chi_1}$$

where $U_{\chi_i} = U'_\chi \otimes \mathcal{I}^{\otimes n-i+1}$ and

$$U'_{\chi_i} \left|g_{i-1}(x_1 \ldots x_{i-1})\right\rangle \left|x_1 \ldots x_i\right\rangle \mapsto \left|g_i(x_1 \ldots x_i)\right\rangle \left|x_1 \ldots x_i\right\rangle$$
$$U_{\chi_*} \left|g_n(x_1 \ldots x_n)\right\rangle \left|x_1 \ldots x_n\right\rangle \left|b\right\rangle \mapsto \left|g_n(x_1 \ldots x_n)\right\rangle \left|x_1 \ldots x_n\right\rangle \left|b \oplus \chi(x_1 \ldots x_n)\right\rangle$$

$$f(x_1, x_2, x_3, x_4, x_5) = 1 + x_2 \cdot \underbrace{(x_1 + 1)}_{y_1} + x_4 \cdot \underbrace{(x_3 + 1)}_{y_4} + x_5 \cdot \underbrace{(1 + x_1 + x_3)}_{y_5}$$

$$U_{\chi_i} \left|\sum_{j=1}^{i-1} x_i y_i\right\rangle \left|x_1 \ldots x_i\right\rangle \mapsto \left|\sum_{j=1}^{i} x_i y_i\right\rangle \left|x_1 \ldots x_i\right\rangle$$

$$U_{\chi_*} \left|\sum_{j=1}^{5} x_i y_i\right\rangle \left|x_1 \ldots x_i\right\rangle \left|b\right\rangle \mapsto \left|f(x_1, \ldots, x_n)\right\rangle \left|b \oplus (f(x_1, \ldots, x_5) \stackrel{?}{=} 0)\right\rangle$$

Computational gains

Assume $E_{\mathcal{O}_{\chi}{}^{(b)}} \gg E_{\mathcal{R}_n} \in O(n)$

Cost of Grover for $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $M = |\chi^{-1}(1)|$

$$\approx \underbrace{\frac{\pi}{4} \cdot \frac{2^{n/2}}{\sqrt{M}}}_{\text{Query complexity}} \cdot \underbrace{E_{O_{\chi}^{(b)}}}_{\text{Cost of oracle}}$$

Computational gains

Assume $E_{\mathcal{O}_\chi{}^{(b)}} \gg E_{\mathcal{R}_n} \in O(n)$

Cost of Grover for $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $M = |\chi^{-1}(1)|$

$$\approx \underbrace{\frac{\pi}{4} \cdot \frac{2^{n/2}}{\sqrt{M}}}_{\substack{\text{Query}\\\text{complexity}}} \cdot \underbrace{E_{O_\chi^{(b)}}}_{\substack{\text{Cost of}\\\text{oracle}}} \quad = \quad \underbrace{\frac{\pi}{4} \cdot \frac{2^{n/2}}{\sqrt{M}}}_{\substack{\text{Query}\\\text{complexity}}} \cdot \underbrace{\left(2\sum_{i=1}^{n} E_{U_i} + U_{\chi_*}\right)}_{\substack{\text{Cost of}\\\text{decomposed}\\\text{oracle}}}$$

Computational gains

Assume $E_{\mathcal{O}_\chi{}^{(b)}} \gg E_{\mathcal{R}_n} \in O(n)$

Cost of Grover for $\chi : \{0,1\}^n \longrightarrow \{0,1\}$ and $M = |\chi^{-1}(1)|$

$$\approx \underbrace{\frac{\pi}{4} \cdot \frac{2^{n/2}}{\sqrt{M}}}_{\text{Query complexity}} \cdot \underbrace{E_{O_\chi^{(b)}}}_{\text{Cost of oracle}} \quad = \quad \underbrace{\frac{\pi}{4} \cdot \frac{2^{n/2}}{\sqrt{M}}}_{\text{Query complexity}} \cdot \underbrace{\left(2\sum_{i=1}^{n} E_{U_i} + U_{\chi_*}\right)}_{\text{Cost of decomposed oracle}}$$

Basic idea: modify the oracle to work on a smaller search-space

- Increases the cost contribution of the quantum oracle
- Decreases the number of queries
- Can balance costs for certain problems
- Can apply preprocessing to decrease cost of additional queries

Adding additional targets to the search-space

Goal: modify the oracle to use Grover on a search-space of size 2^{n-k}

- Choose a $0 \leq k < n$ and compute

$$U_{\chi_{n-k}} \cdots U_{\chi_1} |0^w\rangle |x_1 \ldots x_{n-k}\rangle \; |0^k\rangle \mapsto |g_{n-k}(x_1 \ldots x_{n-k})\rangle \; |0^k\rangle$$

- For $z_1 \ldots z_k \in \{0,1\}^k$:
  - Change the last register to $|z_1 \ldots z_k\rangle$
  - Execute $U_k = U^\dagger_{\chi_{n-k+1}} \cdots U^\dagger_{\chi_n} U_{\chi_*} U_{\chi_n} \cdots U_{\chi_{n-k+1}}$
- Restore the last register to the state $|0^k\rangle$ and execute $U^\dagger_{\chi_1} \cdots U^\dagger_{\chi_{n-k}}$:

$$|0^w\rangle |x_1 \ldots x_{n-k}\rangle \; |0^k\rangle \; |b \bigoplus_{z_1 \ldots z_k \in \{0,1\}^k} \chi(x_1 \ldots x_{n-k} z_1 \ldots z_k)\rangle$$

Cost $\approx \quad \dfrac{\pi}{4} \cdot \dfrac{2^{(n-k)/2}}{\sqrt{M}} \cdot \left(2 \cdot \sum_{i=1}^{n-k} E_{U_i} + 2 \cdot 2^k \sum_{i=1}^{k} E_{U_i} + 2^k U_{\chi_*} \right)$

Preprocessing

Preprocessing only helps!

- Allows shifting of costs to earlier part
- Allows us to reduce or remove quantum gates

  $$|a_1a_2a_3a_40\rangle\,|b\rangle \mapsto |a_1a_2a_3a_40\rangle\,|b \oplus (a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge 0)\rangle$$

  can be removed, whilst

  $$|a_1a_2a_3a_41\rangle\,|b\rangle \mapsto |a_1a_2a_3a_41\rangle\,|b \oplus (a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge 1)\rangle$$

  becomes

  $$|a_1a_2a_3a_4\rangle\,|b\rangle \mapsto |a_1a_2a_3a_4\rangle\,|b \oplus (a_1 \wedge a_2 \wedge a_3 \wedge a_4)\rangle\,.$$

- In particular, this can drop the cost of $E_{U_{\chi_{n-k+1}}}, \ldots, E_{U_{\chi_n}}, E_{U_{\chi_*}}$

Preprocessing

Preprocessing only helps!

- Allows shifting of costs to earlier part
- Allows us to reduce or remove quantum gates
- In particular, this can drop the cost of $E_{U_{\chi_{n-k+1}}}, \ldots, E_{U_{\chi_n}}, E_{U_{\chi *}}$
- After hardwiring we can remove qubits

$$|0^w\rangle \, |x_1 \ldots x_{n-k}\rangle \; |0^k\rangle \; |b \bigoplus_{z_1 \ldots z_k \in \{0,1\}^k} \chi(x_1 \ldots x_{n-k} z_1 \ldots z_k)\rangle$$

$$\text{Cost} \approx \quad \frac{\pi}{4} \cdot \frac{2^{(n-k)/2}}{\sqrt{M}} \cdot \left(2 \cdot \sum_{i=1}^{n-k} E_{U_i} + 2 \cdot 2^k \sum_{i=1}^{k} E_{U_i} + 2^k U_{\chi *} \right)$$

Preprocessing

Preprocessing only helps!

- Allows shifting of costs to earlier part
- Allows us to reduce or remove quantum gates
- In particular, this can drop the cost of $E_{U_{\chi_{n-k+1}}}, \ldots, E_{U_{\chi_n}}, E_{U_{\chi*}}$
- After hardwiring we can remove qubits

$$|0^w\rangle \, |x_1 \ldots x_{n-k}\rangle \, |b \bigoplus_{z_1 \ldots z_k \in \{0,1\}^k} \chi(x_1 \ldots x_{n-k} z_1 \ldots z_k)\rangle$$

$$\text{Cost} \approx \quad \frac{\pi}{4} \cdot \frac{2^{(n-k)/2}}{\sqrt{M}} \cdot \left(2 \cdot \sum_{i=1}^{n-k} E_{U_i} + 2 \cdot 2^k \sum_{i=1}^{k} E_{U_i} + 2^k U_{\chi*} \right)$$

Application: Multivariate Quadratic oracle

Problem: find a zero of $f^{(1)}, \ldots, f^{(m)} \in \mathbb{F}_2[x_1, \ldots, x_n]$ in $\mathbb{F}_2$.

$$f^{(k)}(x_1, \ldots, x_n) = c^{(k)} + \sum_{1 \leq i \leq n} x_i y_i^{(k)} \quad \text{where} \quad y_i^{(k)} = b_i^{(k)} + \sum_{1 \leq j < n}^{i-1} a_{j,i}^{(k)} x_j$$

- Original cost of quantum search if $n = m$: $O(2^{n/2} m n^2)$.
- Using preprocessing reduces this to $O(2^{n/2} m n^{3/2})$
- Shifting computation of $y_i^{(k)}$ reduces this to $O(2^{n/2} m n)$

$$O\left(2^{n/2} 2^{-k/2} \cdot \left(m(n-k)^2 + 2^k m \cdot (n-k)\right)\right) \qquad \text{(optimal } k \approx \log_2 n\text{)}$$

$$O\left(2^{n/2} 2^{-k/2} \cdot \left(mn^2 + 2^k m\right)\right) \qquad \text{(optimal } k \approx 2 \log_2 n\text{)}$$

Application: SIKE

Definition (Claw finding problem)

Given finite sets $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ and functions $f : \mathcal{X} \longrightarrow \mathcal{Z}$ and $g : \mathcal{Y} \longrightarrow \mathcal{Z}$, find $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $f(x) = g(y)$.

Goal: find a degree-2^e isogeny between two elliptic curves $E_0/\mathbb{F}_{p^2}$ and $E_1/\mathbb{F}_{p^2}$, where $e \approx \frac{\log p}{2}$ using

$$\underbrace{f_{e_1} : \{0,1\}^{e_1} \longrightarrow \mathbb{F}_{p^2}}_{\text{Computes an isogeny-path from } E_0} \quad \text{and} \quad \underbrace{g_{e_2} : \{0,1\}^{e_2} \longrightarrow \mathbb{F}_{p^2}}_{\text{Computes an isogeny-path from } E_1} \quad \text{st. } e_1 + e_2 = e$$

- Classical algorithm for f_{e_1}, g_{e_2} is $O(e \log e)$ EC operations [JDF11]

Question[JS19a]: Might Grover competitive with Tani's claw-finding algorithm?

Application: SIKE

Find a degree-2^e isogeny between $E_0/\mathbb{F}_{p^2}$ and $E_1/\mathbb{F}_{p^2}$ where $e \approx \frac{\log p}{2}$

$$\underbrace{f_{e_1} : \{0,1\}^{e_1} \longrightarrow \mathbb{F}_{p^2}}_{\text{Computes an isogeny-path from } E_0} \quad \text{and} \quad \underbrace{g_{e_2} : \{0,1\}^{e_2} \longrightarrow \mathbb{F}_{p^2}}_{\text{Computes an isogeny-path from } E_1} \quad \text{st. } e_1 + e_2 = e$$

- Assume cost for evaluating degree-2^e isogeny is C_e
- Tactic used for comparison with Tani's algorithm:
  - Choose $e_1 \approx e_2$
  - Set $U_{\chi_{e-e_1}} \cdots U_{\chi_1}$ to evaluate and store $f_{e_1}(x_1 \ldots x_{e_1})$
  - Set $U_{\chi_e} \cdots U_{\chi_{e-e_1+1}}$ to evaluate and store $g_{e_2}(x_1 \ldots x_{e_1})$
  - Set U_{χ_*} to compare the two stored values and output if they match

$$O\left(p^{1/4} C_e\right)$$

Application: SIKE

Find a degree-2^e isogeny between $E_0/\mathbb{F}_{p^2}$ and $E_1/\mathbb{F}_{p^2}$ where $e \approx \frac{\log p}{2}$

$$\underbrace{f_{e_1} : \{0,1\}^{e_1} \longrightarrow \mathbb{F}_{p^2}}_{\text{Computes an isogeny-path from } E_0} \quad \text{and} \quad \underbrace{g_{e_2} : \{0,1\}^{e_2} \longrightarrow \mathbb{F}_{p^2}}_{\text{Computes an isogeny-path from } E_1} \quad \text{st. } e_1 + e_2 = e$$

- Assume cost for evaluating degree-2^e isogeny is C_e
- Tactic used for comparison with Tani's algorithm:
  - Require that $e_1 + e_2 = e$ and use a preprocessed secondary-search
  - Set $U_{\chi_{e-e_1}} \cdots U_{\chi_1}$ to evaluate and store $f_{e_1}(x_1 \dots x_{e_1})$
  - $U_{\chi_e} \cdots U_{\chi_{e-e_1+1}}$ evaluates $g_{e_2}(x_1 \dots x_{e_1})$ using only $O(\log p)$ X gates
  - Set U_{χ_*} to compare the two stored values and output if they match

$$O\Big(p^{1/4} 2^{-e_2/2} \cdot \big(2C_{e_1} + 2^{e_2} \log p\big)\Big)$$

Application: SIKE

Find a degree-2^e isogeny between $E_0/\mathbb{F}_{p^2}$ and $E_1/\mathbb{F}_{p^2}$ where $e \approx \frac{\log p}{2}$

$$\underbrace{f_{e_1} : \{0,1\}^{e_1} \longrightarrow \mathbb{F}_{p^2}}_{\text{Computes an isogeny-path from } E_0} \quad \text{and} \quad \underbrace{g_{e_2} : \{0,1\}^{e_2} \longrightarrow \mathbb{F}_{p^2}}_{\text{Computes an isogeny-path from } E_1} \quad \text{st. } e_1 + e_2 = e$$

- Assume cost for evaluating degree-2^e isogeny is C_e
- Tactic used for comparison with Tani's algorithm:
  - Optimal $e_2 \approx \log_2\left(\frac{C_e}{\log p}\right)$ with a preprocessed secondary-search
  - Set $U_{\chi_{e-e_1}} \cdots U_{\chi_1}$ to evaluate and store $f_{e_1}(x_1 \ldots x_{e_1})$
  - $U_{\chi_e} \cdots U_{\chi_{e-e_1+1}}$ evaluates $g_{e_2}(x_1 \ldots x_{e_1})$ using only $O(\log p)$ X gates
  - Set U_{χ_*} to compare the two stored values and output if they match

$$O\left(p^{1/4} \cdot \sqrt{C_e \log p}\right)$$

Application: SIKE

Improvement:

$$O\left(p^{1/4}C_e\right) \longrightarrow O\left(p^{1/4}\sqrt{C_e \log p}\right)$$

- $C_e \in O(e \log e)$ elliptic curve operations
- Elliptic curve operations $\in O(\log p \log\log p)$ (conservative [JS19b])

$$O\left(p^{1/4}\log^2 p(\log\log p)^2\right) \longrightarrow O\left(p^{1/4}\log^{3/2} p(\log\log p)\right)$$

- Elliptic curve operations $\in O(\log^2 p \log\log p)$ (realistic [RNSL17])

$$O\left(p^{1/4}\log^3 p(\log\log p)^2\right) \longrightarrow O\left(p^{1/4}\log^2 p(\log\log p)\right)$$

Application: SIKE

Improvement:

$$O\left(p^{1/4}C_e\right) \longrightarrow O\left(p^{1/4}\sqrt{C_e \log p}\right)$$

- $C_e \in O(e \log e)$ elliptic curve operations
- Elliptic curve operations $\in O(\log p \log\log p)$ (conservative [JS19b])

$$O\left(p^{1/4}\log^2 p(\log\log p)^2\right) \longrightarrow O\left(p^{1/4}\log^{3/2} p(\log\log p)\right)$$

- Elliptic curve operations $\in O(\log^2 p \log\log p)$ (realistic [RNSL17])

$$O\left(p^{1/4}\log^3 p(\log\log p)^2\right) \longrightarrow O\left(p^{1/4}\log^2 p(\log\log p)\right)$$

Conservative/unoptimised vs more realistic/optimised

$$O\left(p^{1/4}\log^2 p(\log\log p)^2\right) \quad \text{vs} \quad O\left(p^{1/4}\log^2 p(\log\log p)\right)$$

Applications: SIKE

Attack cost	SIKE-434			SIKE-610		
	G	D	W	G	D	W
Grover [JS19b]	132	122	10	177	167	10
Grover (Ours with assumptions from [JS19b])	126	116	10	171	160	10
Grover (Ours with higher costs)	130	120	10	175	165	10
Tani[JS19b] (optimal # gates)	124	114	25	169	159	25
Tani[JS19b] (optimal $D \times W$)	131	122	10	177	166	10
VW [JS19b] (optimal # gates)	132	14	128	177	14	173
VW [JS19b] (optimal $D \times W$)	132	14	128	177	14	173

- Grover may be superior in the Depth $\times$ Width-cost metric.
  For SIKE-434: 2^{126} for Grover's algorithm compared to 2^{132} for Tani.
- Grover may be competitive in the gate based metric.
  For SIKE-434: 2^{126} for Grover's algorithm compared to 2^{124} for Tani.

Conclusions

- Generic framework easily applicable to problems
- Optimisation of older algorithm to solve an instance of $\mathcal{MQ}(\mathbb{F}_2, n, m)$

$$O\left(2^{n/2}mn^2\right) \longrightarrow O\left(2^{n/2}mn\right)$$

- Minor improvements in claw-finding techniques using Grover

$$O\left(p^{1/4}C_e\right) \longrightarrow O\left(p^{1/4}\sqrt{C_e \log p}\right)$$

- Optimisations only increase query-complexity
- Cost of Grover may be slightly lower than thought
- Using Grover as a black-box with overheads may be risky

References i

- David Jao and Luca De Feo, Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies, International Workshop on Post-Quantum Cryptography, Springer, 2011, pp. 19–34.
- Samuel Jaques and John M. Schanck, Quantum cryptanalysis in the ram model: Claw-finding attacks on sike, Advances in Cryptology – CRYPTO 2019 (Cham) (Alexandra Boldyreva and Daniele Micciancio, eds.), Springer International Publishing, 2019, pp. 32–61.
- Samuel Jaques and John M Schanck, Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE, Tech. report, Cryptology ePrint Archive, Report 2019/103, 2019. https://eprint. iacr. org …, 2019.

References ii

Martin Roetteler, Michael Naehrig, Krysta M Svore, and Kristin Lauter, Quantum resource estimates for computing elliptic curve discrete logarithms, International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2017, pp. 241–270.