Exploiting preprocessing to break $\mathcal{MQ}$ system parameters

Benjamin Pring

June 2018

University of Bath

Quantum computers and cryptography

The well-known dangers of quantum computers fit into the two standard camps of algorithmic cryptanalysis.

- Quantum computers provide an exponential speedup for currently used public-key cryptosystems via Shor's algorithms for the factoring and discrete-log problems.

- Quantum computers provide a square-root speedup in query-complexity for brute force search methods.

Quantum computers and cryptography

The well-known dangers of quantum computers fit into the two standard camps of algorithmic cryptanalysis.

- Quantum computers provide an exponential speedup for currently used public-key cryptosystems via Shor's algorithms for the factoring and discrete-log problems.
- Impact: start again with new hardness assumptions.
- Quantum computers provide a square-root speedup in query-complexity for brute force search methods.

Quantum computers and cryptography

The well-known dangers of quantum computers fit into the two standard camps of algorithmic cryptanalysis.

- Quantum computers provide an exponential speedup for currently used public-key cryptosystems via Shor's algorithms for the factoring and discrete-log problems.
- Impact: start again with new hardness assumptions.
- Quantum computers provide a square-root speedup in query-complexity for brute force search methods.
- Impact: when search is thought to be the best approach to solving a problem, such as for breaking AES, we double the size of the key. 128-bit security becomes 64-bit security, and so forth...

Quantum cryptanalysis

Definition (Post quantum cryptanalysis)

The design of algorithms which will never be used for computers which may never exist, to ensure that certain design choices for protocols which may never be executed are never chosen.

Quantum cryptanalysis

Definition (Post quantum cryptanalysis)

The design of algorithms which will never be used for computers which may never exist, to ensure that certain design choices for protocols which may never be executed are never chosen.

Design and estimation of resources for algorithms to achieve

- Complete breaks — the design of new hybrid algorithms which lead to exponential speedups and make a cryptosystem unusable.

Quantum cryptanalysis

Definition (Post quantum cryptanalysis)

The design of algorithms which will never be used for computers which may never exist, to ensure that certain design choices for protocols which may never be executed are never chosen.

Design and estimation of resources for algorithms to achieve

- Complete breaks — the design of new hybrid algorithms which lead to exponential speedups and make a cryptosystem unusable.
- Parameter breaks — design and optimisation of cryptanalysis techniques requiring exponential time to attack cryptosystems.

Quantum cryptanalysis

Definition (Post quantum cryptanalysis)

The design of algorithms which will never be used for computers which may never exist, to ensure that certain design choices for protocols which may never be executed are never chosen.

Design and estimation of resources for algorithms to achieve

- ~~Complete breaks — the design of new hybrid algorithms which lead to exponential speedups and make a cryptosystem unusable.~~
- Parameter breaks — design and optimisation of cryptanalysis techniques requiring exponential time to attack cryptosystems.

↑

This paper

Overview

The MQ-hardness assumption

Quantum computing 101

Quantum search and cost

Quantum circuits and the $\mathcal{MQ}$ evaluation oracle

Our adaptation

The MQ-hardness assumption

One of many candidates

Many post-quantum hardness assumptions we could choose instead of factoring or the discrete-log problem.

- Lattice problems
- Hashing problems
- Isogenies on supersingular curves
- Code-based cryptography
- Hardness of solving systems of equations over finite fields.

One of many candidates

Many post-quantum hardness assumptions we could choose instead of factoring or the discrete-log problem.

- ~~Lattice problems~~
- ~~Hashing problems~~
- ~~Isogenies on supersingular curves~~
- ~~Code-based cryptography~~
- Hardness of solving systems of equations over finite fields.

The hard problem

Definition (The Multivariate Quadratic ($\mathcal{MQ}$) problem)

Let $p^{(1)}, \ldots, p^{(m)} \in \mathbb{F}_q[x_1, \ldots, x_n]$, where $\mathbb{F}_q$ is a finite field of size q and each $p^{(i)}$ is of degree two.

The Multivariate Quadratic ($\mathcal{MQ}$) problem is to find an $\bar{x} = (x_1, \ldots, x_n)$ with $x_i \in \mathbb{F}_q$ such that $p^{(i)}(\bar{x}) = 0$ for $i = 1, \ldots, m$.

The hard problem

Definition (The Multivariate Quadratic ($\mathcal{MQ}$) problem)

Let $p^{(1)}, \ldots, p^{(m)} \in \mathbb{F}_q[x_1, \ldots, x_n]$, where $\mathbb{F}_q$ is a finite field of size q and each $p^{(i)}$ is of degree two.

The Multivariate Quadratic ($\mathcal{MQ}$) problem is to find an $\bar{x} = (x_1, \ldots, x_n)$ with $x_i \in \mathbb{F}_q$ such that $p^{(i)}(\bar{x}) = 0$ for $i = 1, \ldots, m$.

Can be thought of as simply a case of black-box preimage search for

$$\mathcal{P} : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m, \tag{1}$$

but such an approach ignores structure in the problem.

The hard problem

Definition (The Multivariate Quadratic ($\mathcal{MQ}$) problem)

Let $p^{(1)}, \ldots, p^{(m)} \in \mathbb{F}_q[x_1, \ldots, x_n]$, where $\mathbb{F}_q$ is a finite field of size q and each $p^{(i)}$ is of degree two.

The Multivariate Quadratic ($\mathcal{MQ}$) problem is to find an $\bar{x} = (x_1, \ldots, x_n)$ with $x_i \in \mathbb{F}_q$ such that $p^{(i)}(\bar{x}) = 0$ for $i = 1, \ldots, m$.

Can be thought of as simply a case of black-box preimage search for

$$\mathcal{P} : \mathbb{F}_q^n \longrightarrow \mathbb{F}_q^m, \tag{1}$$

but such an approach ignores structure in the problem.

We'll be looking only at the $q = 2$ (binary $\mathcal{MQ}$) case from here on.

Why is it important?

For our purposes: Hidden Field Equations (HFE) cryptosystems.

Interesting topic — but too little time in this talk.

Target for cryptanalysis: The Gui $\mathcal{MQ}$ signature scheme.

Public-key of a Gui(n, D, a, v, k) $\mathcal{MQ}$ signature scheme is the $\mathcal{MQ}$ map

$$\mathcal{P} : \mathbb{F}_2^{n+v} \longrightarrow \mathbb{F}_2^{n-a} \tag{2}$$

Why is it important?

For our purposes: Hidden Field Equations (HFE) cryptosystems.

Interesting topic — but too little time in this talk.

Target for cryptanalysis: The Gui $\mathcal{MQ}$ signature scheme.

Public-key of a Gui(n, D, a, v, k) $\mathcal{MQ}$ signature scheme is the $\mathcal{MQ}$ map

$$\mathcal{P} : \mathbb{F}_2^{n+v} \longrightarrow \mathbb{F}_2^{n-a} \tag{2}$$

Too little time to detail the workings of Gui, but:

- Signing a message $\equiv$ inverting $\mathcal{P}$ a total of k times.
- Easy to invert if we know structure of $\mathcal{P} = L_2 \circ \Phi^{-1} \circ \mathcal{F} \circ \Phi \circ L_1$.
- $\mathcal{F} \in \mathbb{E}[X]$ where $[\mathbb{E} : \mathbb{F}_2] = n$ and degree($\mathcal{F}$) = D.
- Easier to solve using Gröbner/XL techniques than random $\mathcal{MQ}$.
- Search based methods do not exploit this structure.

Classical / quantum methods for solving the $\mathcal{MQ}$ problem.

Leading classical methods

- Exhaustive search methods.
- BooleanSolve.
- Gröbner bases techniques: $XL/F_4/F_5$.

Leading quantum methods

- Quantum search for small parameter sizes.
- QuantumBooleanSolve (BooleanSolve/Grover hybrid).
- GroverXL (XL/Grover hybrid).

Quantum computing 101

Multiple layers of abstraction for quantum computing

Quantum overload

- Physical laws of the universe
- Abstract unitary operators on a Hilbert space
- Logical quantum circuit level
- Fault-tolerant level
- Mapping to eventual quantum hardware architecture

Multiple layers of abstraction for quantum computing

~~Quantum overload~~

- ~~Physical laws of the universe~~
- Abstract unitary operators ~~on a Hilbert space~~
- Logical quantum circuit level
- ~~Fault-tolerant level~~
- ~~Mapping to eventual quantum hardware architecture~~

We'll reduce Grover's quantum search algorithm to the problem of considering boolean reversible circuits, then convert these costs later.

Multiple layers of abstraction for quantum computing

~~Quantum overload~~

- ~~Physical laws of the universe~~
- Abstract unitary operators ~~on a Hilbert space~~
- Logical quantum circuit level
- ~~Fault-tolerant level~~
- ~~Mapping to eventual quantum hardware architecture~~

We'll reduce Grover's quantum search algorithm to the problem of considering boolean reversible circuits, then convert these costs later.

Unitary operators — how we describe algorithms.

Quantum circuits — how we implement unitary operators.

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.

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \tag{3}$$

Quantum states and quantum gates I

Quantum states can be viewed as vectors of coefficients.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \tag{3}$$

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

$$U|\psi_{t_0}\rangle = |\psi_{t_1}\rangle \qquad \text{where } U^\dagger U = UU^\dagger = I \tag{4}$$

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

Quantum states and quantum gates I

Quantum states can be viewed as vectors of coefficients.

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle = \begin{bmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \tag{3}$$

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

$$U\,|\psi_{t_0}\rangle = |\psi_{t_1}\rangle \qquad \text{where } \; U^\dagger U = UU^\dagger = I \tag{4}$$

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 $F : \{0,1\}^n \longrightarrow \{0,1\}$.

$$F(x) = \begin{cases} 1 & \text{if } x \text{ is a target} \\ 0 & \text{otherwise} \end{cases} \qquad S_F \left| x \right\rangle = \begin{cases} -\left| x \right\rangle & \text{if } F(x) = 1 \\ \left| x \right\rangle & \text{if } F(x) = 0 \end{cases} \tag{5}$$

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 $F : \{0,1\}^n \longrightarrow \{0,1\}$.

$$F(x) = \begin{cases} 1 & \text{if } x \text{ is a target} \\ 0 & \text{otherwise} \end{cases} \qquad S_F \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } F(x) = 1 \\ \left|x\right\rangle & \text{if } F(x) = 0 \end{cases} \tag{5}$$

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

$$S_F \sum_{x \in \{0,1\}^n} \alpha_x \left|x\right\rangle = \sum_{x \in \{0,1\}^n} \alpha_x S_F \left|x\right\rangle \tag{6}$$

Quantum states and quantum gates IV

A unitary operator implementing $F : \{0,1\}^n \longrightarrow \{0,1\}$ on bitstrings is enough to realise S_F.

Say we have the unitary U_F

$$U_F \left|x\right\rangle \left|y\right\rangle \mapsto \left|x\right\rangle \left|y \oplus F(x)\right\rangle \tag{7}$$

Quantum states and quantum gates IV

A unitary operator implementing $F : \{0,1\}^n \longrightarrow \{0,1\}$ on bitstrings is enough to realise S_F.

Say we have the unitary U_F

$$U_F \left|x\right\rangle \left|y\right\rangle \mapsto \left|x\right\rangle \left|y \oplus F(x)\right\rangle \tag{7}$$

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

$$H \left|1\right\rangle = \frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}} \tag{8}$$

gives us

$$U_F \left|x\right\rangle \left(\frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}\right) \mapsto \left|x\right\rangle \left(\frac{\left|0 \oplus F(x)\right\rangle - \left|1 \oplus F(x)\right\rangle}{\sqrt{2}}\right) = (-1)^{F(x)} \left|x\right\rangle \left(\frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}\right).$$

Unitary operators for boolean circuits

Recall the universal gate set for boolean circuits: $\{\wedge, \neg, \oplus\}$.

Unitary operators for boolean circuits

Recall the universal gate set for boolean circuits: $\{\wedge, \neg, \oplus\}$.

$$X\left|x\right\rangle \mapsto \left|x \oplus 1\right\rangle \tag{9}$$

$$CNOT\left|x\right\rangle\left|y\right\rangle \mapsto \left|x\right\rangle\left|x \oplus y\right\rangle \tag{10}$$

$$\text{Toffoli}\left|x\right\rangle\left|y\right\rangle\left|z\right\rangle \mapsto \left|x\right\rangle\left|y\right\rangle\left|z \oplus (x \wedge y)\right\rangle \tag{11}$$

Unitary operators for boolean circuits

Recall the universal gate set for boolean circuits: $\{\wedge, \neg, \oplus\}$.

$$X\,|x\rangle \mapsto |x \oplus 1\rangle \tag{9}$$

$$CNOT\,|x\rangle\,|y\rangle \mapsto |x\rangle\,|x \oplus y\rangle \tag{10}$$

$$\text{Toffoli}\,|x\rangle\,|y\rangle\,|z\rangle \mapsto |x\rangle\,|y\rangle\,|z \oplus (x \wedge y)\rangle \tag{11}$$

$$X\,|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

Unitary operators for boolean circuits

Recall the universal gate set for boolean circuits: $\{\wedge, \neg, \oplus\}$.

$$X\,|x\rangle \mapsto |x \oplus 1\rangle \tag{9}$$

$$CNOT\,|x\rangle\,|y\rangle \mapsto |x\rangle\,|x \oplus y\rangle \tag{10}$$

$$\text{Toffoli}\,|x\rangle\,|y\rangle\,|z\rangle \mapsto |x\rangle\,|y\rangle\,|z \oplus (x \wedge y)\rangle \tag{11}$$

$$CNOT\,|1\rangle\,|0\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |1\rangle\,|1\rangle$$

Unitary operators for boolean circuits

Recall the universal gate set for boolean circuits: $\{\wedge, \neg, \oplus\}$.

$$X|x\rangle \mapsto |x \oplus 1\rangle \tag{9}$$

$$CNOT|x\rangle|y\rangle \mapsto |x\rangle|x \oplus y\rangle \tag{10}$$

$$\text{Toffoli}|x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z \oplus (x \wedge y)\rangle \tag{11}$$

$$\text{Toffoli}|1\rangle|1\rangle|1\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |1\rangle|1\rangle|0\rangle$$

Reversibility

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

$$U^\dagger U = UU^\dagger = I. \tag{12}$$

Impact: all boolean circuits must implement permutations.

Use of ancillae qubits is crucial for efficient realisation.

Quantum search and cost

Grover's quantum search algorithm and query complexity I

Consider the boolean function F with a domain of size $N = 2^n$

$$F : \{0,1\}^n \longrightarrow \{0,1\} \tag{13}$$

with the promise that $M = \left|F^{-1}(1)\right|$.

Grover's quantum search algorithm and query complexity I

Consider the boolean function F with a domain of size $N = 2^n$

$$F : \{0,1\}^n \longrightarrow \{0,1\} \tag{13}$$

with the promise that $M = \left|F^{-1}(1)\right|$.

If we are provided with a circuit for F and call each execution of this circuit on an input a *query*. How many queries are required on average before we locate an $x \in \{0,1\}^n$ such that $F(x) = 1$?

Grover's quantum search algorithm and query complexity I

Consider the boolean function F with a domain of size $N = 2^n$

$$F : \{0,1\}^n \longrightarrow \{0,1\} \tag{13}$$

with the promise that $M = \left|F^{-1}(1)\right|$.

If we are provided with a circuit for F and call each execution of this circuit on an input a *query*. How many queries are required on average before we locate an $x \in \{0,1\}^n$ such that $F(x) = 1$?

- Given a classical circuit implementing F: $\mathcal{O}\left(\frac{N}{M}\right)$ queries by exhaustive search.

Grover's quantum search algorithm and query complexity I

Consider the boolean function F with a domain of size $N = 2^n$

$$F : \{0,1\}^n \longrightarrow \{0,1\} \tag{13}$$

with the promise that $M = \left|F^{-1}(1)\right|$.

If we are provided with a circuit for F and call each execution of this circuit on an input a *query*. How many queries are required on average before we locate an $x \in \{0,1\}^n$ such that $F(x) = 1$?

- Given a classical circuit implementing F: $\mathcal{O}\left(\frac{N}{M}\right)$ queries by exhaustive search.
- Given a quantum circuit implementing F : $\mathcal{O}\left(\sqrt{\frac{N}{M}}\right)$ queries by Grover's algorithm.

Grover's quantum search algorithm and query complexity II

Let $F : \{0,1\}^n \longrightarrow \{0,1\}$, $N = 2^n$ and $M = \left|F^{-1}(1)\right|$.

Given the unitary operators acting on on n-qubits

$$S_0 \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } \left|x\right\rangle = \left|0\right\rangle \\ \left|x\right\rangle & \text{if } \left|x\right\rangle \neq \left|0\right\rangle \end{cases} \qquad S_F \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } F(x) = 1 \\ \left|x\right\rangle & \text{if } F(x) = 0 \end{cases}, \tag{14}$$

and defining the *Grover iterator* to be

$$G = -H^{\otimes n} S_0 H^{\otimes n} S_F \tag{15}$$

Grover's algorithm consists of the following steps.

Grover's quantum search algorithm and query complexity II

Let $F : \{0,1\}^n \longrightarrow \{0,1\}$, $N = 2^n$ and $M = \left|F^{-1}(1)\right|$.

Given the unitary operators acting on on n-qubits

$$S_0 \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } \left|x\right\rangle = \left|0\right\rangle \\ \left|x\right\rangle & \text{if } \left|x\right\rangle \neq \left|0\right\rangle \end{cases} \qquad S_F \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } F(x) = 1 \\ \left|x\right\rangle & \text{if } F(x) = 0 \end{cases}, \tag{14}$$

and defining the *Grover iterator* to be

$$G = -H^{\otimes n} S_0 H^{\otimes n} S_F \tag{15}$$

Grover's algorithm consists of the following steps.

1. Initialise the quantum register to $\left|0^n\right\rangle$

Grover's quantum search algorithm and query complexity II

Let $F : \{0,1\}^n \longrightarrow \{0,1\}$, $N = 2^n$ and $M = \left|F^{-1}(1)\right|$.

Given the unitary operators acting on on n-qubits

$$S_0 \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } \left|x\right\rangle = \left|0\right\rangle \\ \left|x\right\rangle & \text{if } \left|x\right\rangle \neq \left|0\right\rangle \end{cases} \qquad S_F \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } F(x) = 1 \\ \left|x\right\rangle & \text{if } F(x) = 0 \end{cases}, \tag{14}$$

and defining the *Grover iterator* to be

$$G = -H^{\otimes n} S_0 H^{\otimes n} S_F \tag{15}$$

Grover's algorithm consists of the following steps.

1. Initialise the quantum register to $\left|0^n\right\rangle$
2. Apply the Hadamard transform to compute $\left|\psi_0\right\rangle = H^{\otimes n} \left|0^n\right\rangle$

Grover's quantum search algorithm and query complexity II

Let $F : \{0,1\}^n \longrightarrow \{0,1\}$, $N = 2^n$ and $M = \left|F^{-1}(1)\right|$.

Given the unitary operators acting on on n-qubits

$$S_0 \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } \left|x\right\rangle = \left|0\right\rangle \\ \left|x\right\rangle & \text{if } \left|x\right\rangle \neq \left|0\right\rangle \end{cases} \qquad S_F \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } F(x) = 1 \\ \left|x\right\rangle & \text{if } F(x) = 0 \end{cases}, \tag{14}$$

and defining the *Grover iterator* to be

$$G = -H^{\otimes n} S_0 H^{\otimes n} S_F \tag{15}$$

Grover's algorithm consists of the following steps.

1. Initialise the quantum register to $\left|0^n\right\rangle$
2. Apply the Hadamard transform to compute $\left|\psi_0\right\rangle = H^{\otimes n}\left|0^n\right\rangle$
3. Compute $\left|\psi_k\right\rangle = G^k \left|\psi_0\right\rangle$, via successive applications of G.

Grover's quantum search algorithm and query complexity II

Let $F : \{0,1\}^n \longrightarrow \{0,1\}$, $N = 2^n$ and $M = \left|F^{-1}(1)\right|$.

Given the unitary operators acting on on n-qubits

$$S_0 \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } \left|x\right\rangle = \left|0\right\rangle \\ \left|x\right\rangle & \text{if } \left|x\right\rangle \neq \left|0\right\rangle \end{cases} \qquad S_F \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } F(x) = 1 \\ \left|x\right\rangle & \text{if } F(x) = 0 \end{cases}, \tag{14}$$

and defining the *Grover iterator* to be

$$G = -H^{\otimes n} S_0 H^{\otimes n} S_F \tag{15}$$

Grover's algorithm consists of the following steps.

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

Grover's quantum search algorithm and query complexity III

If we define the normalised superpositions

$$|Good\rangle = \frac{1}{\sqrt{M}} \sum_{\substack{x\in\{0,1\}^n \\ x:F(x)=1}} |x\rangle \quad \text{and} \quad |Bad\rangle = \frac{1}{\sqrt{N-M}} \sum_{\substack{x\in\{0,1\}^n \\ x:F(x)=0}} |x\rangle \tag{16}$$

and the angle $\theta = \arcsin\left(\sqrt{\frac{M}{N}}\right)$, then choosing $k \in \mathbb{N}_0$ and performing steps $1-3$ of Grover's algorithm leave us with the quantum state

$$|\psi_k\rangle = \sin\Big((2k+1)\theta\Big)\,|Good\rangle + \cos\Big((2k+1)\theta\Big)\,|Bad\rangle\,. \tag{17}$$

Grover's quantum search algorithm and query complexity III

If we define the normalised superpositions

$$|Good\rangle = \frac{1}{\sqrt{M}} \sum_{\substack{x \in \{0,1\}^n \\ x: F(x)=1}} |x\rangle \quad \text{and} \quad |Bad\rangle = \frac{1}{\sqrt{N-M}} \sum_{\substack{x \in \{0,1\}^n \\ x: F(x)=0}} |x\rangle \tag{16}$$

and the angle $\theta = \arcsin\left(\sqrt{\frac{M}{N}}\right)$, then choosing $k \in \mathbb{N}_0$ and performing steps $1-3$ of Grover's algorithm leave us with the quantum state

$$|\psi_k\rangle = \sin\Big((2k+1)\theta\Big)|Good\rangle + \cos\Big((2k+1)\theta\Big)|Bad\rangle . \tag{17}$$

Measurement will therefore result in a solution with probability

$$\sin^2\Big((2k+1)\theta\Big) \tag{18}$$

and the optimal $k \in \mathbb{N}$ is easily computed to be $\left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rfloor$.

Grover's quantum search algorithm and query complexity IV

Why does this work?

Grover's quantum search algorithm and query complexity IV

Why does this work? Step one creates the *uniform superposition*

$$H^{\otimes n} |0^n\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x \in \{0,1\}^n} |x\rangle \tag{19}$$

Grover's quantum search algorithm and query complexity IV

Why does this work? Step one creates the *uniform superposition*

$$H^{\otimes n}\left|0^n\right\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x\in\{0,1\}^n} |x\rangle \tag{19}$$

whilst the *Grover iterator* can be rewritten

$$G = D_n S_F, \qquad \text{where} \quad D_n = -H^{\otimes n} S_0 H^{\otimes n} \tag{20}$$

where D_n is the *diffusion operator on* n*-qubits*

Grover's quantum search algorithm and query complexity IV

Why does this work? Step one creates the *uniform superposition*

$$H^{\otimes n} |0^n\rangle = \frac{1}{2^{\frac{n}{2}}} \sum_{x \in \{0,1\}^n} |x\rangle \tag{19}$$

whilst the *Grover iterator* can be rewritten

$$G = D_n S_F, \qquad \text{where} \quad D_n = -H^{\otimes n} S_0 H^{\otimes n} \tag{20}$$

where D_n is the *diffusion operator on n-qubits* whose action is

$$\sum_{x \in \{0,1\}^n} \alpha_x |x\rangle \mapsto \sum_{x \in \{0,1\}^n} \Big(2 \cdot \langle \alpha \rangle - \alpha_x\Big) |x\rangle \tag{21}$$

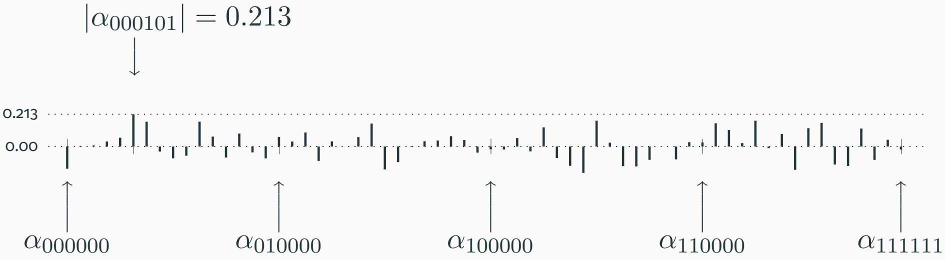
where

$$\langle \alpha \rangle = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} \alpha_x. \tag{22}$$

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$. Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step zero: We have a random quantum state $\sum_{x \in \{0,1\}^n} \alpha_x \left|x\right\rangle$.



Probability of measuring $000101 \approx 100 \cdot | \quad 0.213|^2 = 4.547\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step one : Initialise the quantum state to $|000000\rangle$.

Probability of measuring $000101 \approx 100 \cdot | \quad 0.000|^2 = 0.000\%$.

Grover’s quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$. Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step two : Initialise the uniform superposition by computing $H^{\otimes n} |0^n\rangle$.

Probability of measuring $000101 \approx 100 \cdot | \quad 0.125|^2 = 1.563\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(1): Apply S_F

Probability of measuring $000101 \approx 100 \cdot |-0.125|^2 = 1.563\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$. Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(1): Apply S_F and then D_n.

Probability of measuring $000101 \approx 100 \cdot | \quad 0.367|^2 = 13.483\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(2): Apply S_F

Probability of measuring $000101 \approx 100 \cdot |-0.367|^2 = 13.483\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(2): Apply S_F and then D_n.

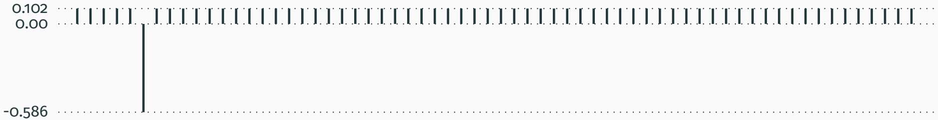
Probability of measuring $000101 \approx 100 \cdot | \quad 0.586|^2 = 34.390\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(3): Apply S_F

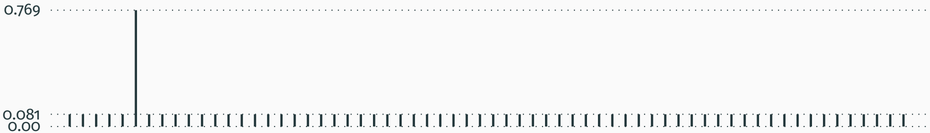


Probability of measuring $000101 \approx 100 \cdot |-0.586|^2 = 34.390\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$. Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(3): Apply S_F and then D_n.



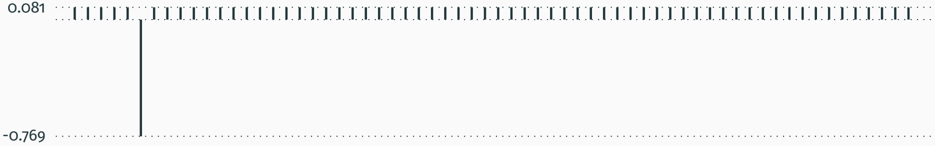
Probability of measuring $000101 \approx 100 \cdot | \quad 0.769|^2 = 59.138\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(4): Apply S_F



Probability of measuring $000101 \approx 100 \cdot |-0.769|^2 = 59.138\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(4): Apply S_F and then D_n.

Probability of measuring $000101 \approx 100 \cdot | \quad 0.904|^2 = 81.638\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(5): Apply S_F

Probability of measuring $000101 \approx 100 \cdot |-0.904|^2 = 81.638\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(5): Apply S_F and then D_n.

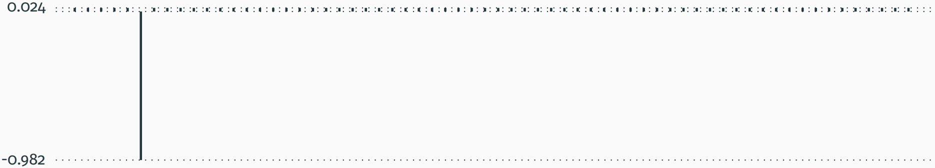
Probability of measuring $000101 \approx 100 \cdot | \quad 0.982|^2 = 96.352\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(6): Apply S_F



Probability of measuring $000101 \approx 100 \cdot |-0.982|^2 = 96.352\%$.

Grover's quantum search algorithm and query complexity V

Let $N = 2^6$ and $M = 1$. Let $\chi(5) = 1$.

Fact: $\frac{\pi}{4} \cdot \sqrt{\frac{2^6}{1}} \approx 6.283$

Step three(6): Apply S_F and then D_n.

Probability of measuring $000101 \approx 100 \cdot |\quad 0.998|^2 = 99.659\%$.

Grover's quantum search algorithm and query complexity VI

The cost (circuit-depth or circuit-size) of Grover's algorithm, excluding the negligible setup phase of computing $|\psi_0\rangle = H^{\otimes} |0^n\rangle$ is therefore

$$\left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rfloor \cdot \Big(\mathrm{Cost}(S_F) + \mathrm{Cost}(D_n) \Big). \tag{23}$$

The diffusion step is relatively cheap, costing $\mathcal{O}(n)$ gates, so the polynomial overhead of Grover's algorithm will usually be dominated by the cost of the quantum oracle.

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 $\mathrm{Cost}(S_F) + \mathrm{Cost}(D_n) \approx \mathrm{Cost}(S_F) = n^3$

- 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.

$$\left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rfloor \cdot \Big(\mathrm{Cost}(S_F) + \mathrm{Cost}(D_n) \Big). \tag{24}$$

How to optimise?

- Optimise the circuit for S_F (often involves a tradeoff).

Grover's quantum search algorithm and query complexity VII

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

Say $\mathrm{Cost}(S_F) + \mathrm{Cost}(D_n) \approx \mathrm{Cost}(S_F) = n^3$

- 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.

$$\left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rfloor \cdot \Big(\mathrm{Cost}(S_F) + \mathrm{Cost}(D_n)\Big). \tag{24}$$

How to optimise?

- Optimise the circuit for S_F (often involves a tradeoff).
- Use fewer Grover iterations (lower success probability).

Grover's quantum search algorithm and query complexity VII

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

Say $\mathrm{Cost}(S_F) + \mathrm{Cost}(D_n) \approx \mathrm{Cost}(S_F) = n^3$

- 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.

$$\left\lfloor \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \right\rfloor \cdot \Big(\mathrm{Cost}(S_F) + \mathrm{Cost}(D_n)\Big). \tag{24}$$

How to optimise?

- Optimise the circuit for S_F (often involves a tradeoff).
- Use fewer Grover iterations (lower success probability).
- Tradeoff between information obtained and complexity.

Quantum circuits and the $\mathcal{MQ}$ evaluation oracle

The $\mathcal{MQ}$ problem again

Minor change in problem:

Assume from now on that we are searching for $x_1, \ldots, x_n$ such that

$$p^{(1)}(x_1, \ldots, x_n) = \cdots = p^{(m)}(x_1, \ldots, x_n) = 1.$$

The $\mathcal{MQ}$ problem again

Minor change in problem:

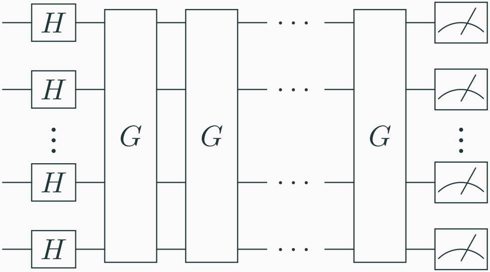
Assume from now on that we are searching for $x_1, \ldots, x_n$ such that

$$p^{(1)}(x_1, \ldots, x_n) = \cdots = p^{(m)}(x_1, \ldots, x_n) = 1.$$

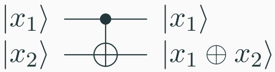
(Equivalent to the $\mathcal{MQ}$ problem — simply add $+1$ to each $p^{(k)}$).

Saves a few gates and makes the complexity slightly more compact.

The Grover circuit

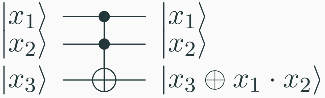


Building reversible boolean circuits from logical quantum gates



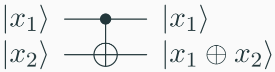
A CNOT gate acting as $x \oplus y$.

An X gate acting as negation.



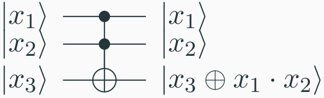
A Toffoli gate acting as $x \cdot y$.

Building reversible boolean circuits from logical quantum gates



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

An X gate acting as negation.

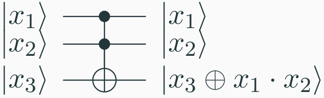


A Toffoli gate acting as $x \cdot y$.

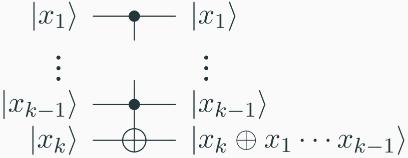
Classicial universal gate set: $\{\oplus, \neg, \wedge\} \leftrightarrow \{\text{CNOT}, X, \text{Toffoli}\}$

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: $\{\oplus, \neg, \wedge\} \leftrightarrow \{\text{CNOT}, X, \text{Toffoli}\}$

Constructing the $\mathcal{MQ}$ evaluation oracle

If we have a unitary operator which implements

$$U_E \left|x\right\rangle \left|0^m\right\rangle \mapsto \left|x\right\rangle \ \left|p^{(1)}(x)\right\rangle \dots \left|p^{(m)}(x)\right\rangle \tag{25}$$

then we simply exploit the $\left|-\right\rangle = \frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}$ state and use a single $m+1$-bit Toffoli gate U_C to obtain our quantum oracle as

$$S_F \left|x\right\rangle \left|0^m\right\rangle \left|-\right\rangle = U_E U_C U_E \left|x\right\rangle \left|0^m\right\rangle \left|-\right\rangle \tag{26}$$

(27)

(28)

(29)

Constructing the $\mathcal{MQ}$ evaluation oracle

If we have a unitary operator which implements

$$U_E \left|x\right\rangle \left|0^m\right\rangle \mapsto \left|x\right\rangle\ \left|p^{(1)}(x)\right\rangle \dots \left|p^{(m)}(x)\right\rangle \tag{25}$$

then we simply exploit the $\left|-\right\rangle = \frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}$ state and use a single $m+1$-bit Toffoli gate U_C to obtain our quantum oracle as

$$\begin{aligned} S_F \left|x\right\rangle \left|0^m\right\rangle \left|-\right\rangle &= U_E U_C U_E \left|x\right\rangle \left|0^m\right\rangle \left|-\right\rangle && (26)\\ &= U_E U_C \left|x\right\rangle\ \left|p^{(1)}(x)\right\rangle \dots \left|p^{(m)}(x)\right\rangle \left|-\right\rangle && (27)\\ & && (28)\\ & && (29) \end{aligned}$$

Constructing the $\mathcal{MQ}$ evaluation oracle

If we have a unitary operator which implements

$$U_E \left|x\right\rangle \left|0^m\right\rangle \mapsto \left|x\right\rangle \, \left|p^{(1)}(x)\right\rangle \ldots \left|p^{(m)}(x)\right\rangle \tag{25}$$

then we simply exploit the $\left|-\right\rangle = \frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}$ state and use a single $m+1$-bit Toffoli gate U_C to obtain our quantum oracle as

$$\begin{aligned} S_F \left|x\right\rangle \left|0^m\right\rangle \left|-\right\rangle &= U_E U_C U_E \left|x\right\rangle \left|0^m\right\rangle \left|-\right\rangle && (26)\\ &= U_E U_C \left|x\right\rangle \, \left|p^{(1)}(x)\right\rangle \ldots \left|p^{(m)}(x)\right\rangle \left|-\right\rangle && (27)\\ &= (-1)^{F(x)} U_E \left|x\right\rangle \, \left|p^{(1)}(x)\right\rangle \ldots \left|p^{(m)}(x)\right\rangle \left|-\right\rangle && (28)\\ & && (29) \end{aligned}$$

Constructing the $\mathcal{MQ}$ evaluation oracle

If we have a unitary operator which implements

$$U_E \left|x\right\rangle \left|0^m\right\rangle \mapsto \left|x\right\rangle\ \left|p^{(1)}(x)\right\rangle \ldots \left|p^{(m)}(x)\right\rangle \tag{25}$$

then we simply exploit the $\left|-\right\rangle = \frac{\left|0\right\rangle - \left|1\right\rangle}{\sqrt{2}}$ state and use a single $m+1$-bit Toffoli gate U_C to obtain our quantum oracle as

$$\begin{aligned}
S_F \left|x\right\rangle \left|0^m\right\rangle \left|-\right\rangle &= U_E U_C U_E \left|x\right\rangle \left|0^m\right\rangle \left|-\right\rangle && (26)\\
&= U_E U_C \left|x\right\rangle\ \left|p^{(1)}(x)\right\rangle \ldots \left|p^{(m)}(x)\right\rangle \left|-\right\rangle && (27)\\
&= (-1)^{F(x)} U_E \left|x\right\rangle\ \left|p^{(1)}(x)\right\rangle \ldots \left|p^{(m)}(x)\right\rangle \left|-\right\rangle && (28)\\
&= (-1)^{F(x)} \left|x\right\rangle\ \left|0^m\right\rangle \left|-\right\rangle && (29)
\end{aligned}$$

Cost of the quantum oracle

$S_F = U_E U_C U_E$, where

- U_E — evaluates $p^{(1)}(x), \ldots, p^{(m)}(x)$
- U_C — outputs 1 iff $p^{(1)}(x) = \cdots = p^{(m)}(x) = 1$.

If $S_F = U_E U_C U_E$, then the total cost of this operation is therefore

$$2 \times U_E + 1 \times U_C.$$

U_C — simply an $m+1$-bit Toffoli gate.

U_E — can be constructed via $X, CNOT$ and Toffoli gate primitives.

The $\mathcal{MQ}$ quantum oracle

Requirements: $n + m + 2$ qubits.

- n-qubits — search-space.
- m-qubits — storage for evaluated equations.
- 1-qubit — working memory.
- 1-qubit — for the $|-\rangle$ state.

$$
\begin{array}{r l}
\text{Search-qubits}\left\{\begin{array}{c} |x_1\rangle \;\text{———————}\; |x_1\rangle \\ \vdots \qquad\qquad\qquad \vdots \\ |x_n\rangle \;\text{———————}\; |x_n\rangle \end{array}\right. \\
\text{Temporary storage}\left\{\begin{array}{c} |0\rangle \;\text{———————}\; |0\rangle \end{array}\right. \\
\text{Evaluated equations}\left\{\begin{array}{c} |0\rangle \;\text{———————}\; |0\rangle \\ \vdots \qquad\qquad\qquad \vdots \\ |0\rangle \;\text{———————}\; |0\rangle \end{array}\right.
\end{array}
$$

An evaluation-based quantum search oracle for the $\mathcal{MQ}$ problem

The $\mathcal{MQ}$ quantum oracle I

(Previous work by Schwabe and Westerbaan). Take a single equation

$$p^{(k)}(x_1, \dots, x_n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n} b_i^{(k)} x_i + c^{(k)}. \tag{30}$$

and rewrite it as

$$p^{(k)}(x_1, \dots, x_n) = \sum_{i=1}^{n} x_i y_i \tag{31}$$

where

$$y_i = b_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_j. \tag{32}$$

The $\mathcal{MQ}$ quantum oracle I

(Previous work by Schwabe and Westerbaan). Take a single equation

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n} b_i^{(k)} x_i + c^{(k)}. \tag{30}$$

and rewrite it as

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i y_i \tag{31}$$

where

$$y_i = b_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_j. \tag{32}$$

Evaluation strategy for each equation:

- Compute y_i onto the workspace via the X and CNOT gates.

The $\mathcal{MQ}$ quantum oracle I

(Previous work by Schwabe and Westerbaan). Take a single equation

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n} b_i^{(k)} x_i + c^{(k)}. \tag{30}$$

and rewrite it as

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i y_i \tag{31}$$

where

$$y_i = b_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_j. \tag{32}$$

Evaluation strategy for each equation:

- Compute y_i onto the workspace via the X and CNOT gates.
- Use a Toffoli gate to add $x_i y_i$ to the equation register.

The $\mathcal{MQ}$ quantum oracle I

(Previous work by Schwabe and Westerbaan). Take a single equation

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n} b_i^{(k)} x_i + c^{(k)}. \tag{30}$$

and rewrite it as

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i y_i \tag{31}$$

where

$$y_i = b_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_j. \tag{32}$$

Evaluation strategy for each equation:

- Compute y_i onto the workspace via the X and CNOT gates.
- Use a Toffoli gate to add $x_i y_i$ to the equation register.
- Uncompute y_i onto the workspace via the X and CNOT gates.

The $\mathcal{MQ}$ quantum oracle II

$$\begin{aligned} p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\ &= x_1 \underbrace{\left(x_2 + x_4\right)}_{y_1} + x_2 \underbrace{\left(x_3 + 1\right)}_{y_2} + x_3 \underbrace{\left(x_4\right)}_{y_3} + x_4 \underbrace{\left(1\right)}_{y_4} + 1 \end{aligned}$$

The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1,x_2,x_3,x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$

The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$

The $\mathcal{MQ}$ quantum oracle II

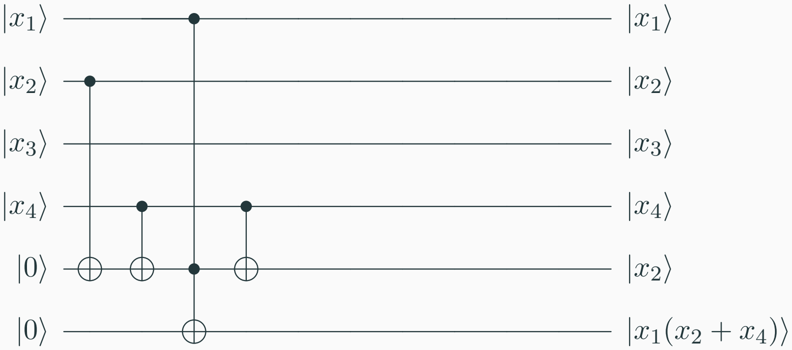
$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$

The $\mathcal{MQ}$ quantum oracle II

$$p^{(k)}(x_1,x_2,x_3,x_4) = x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1$$
$$= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1$$

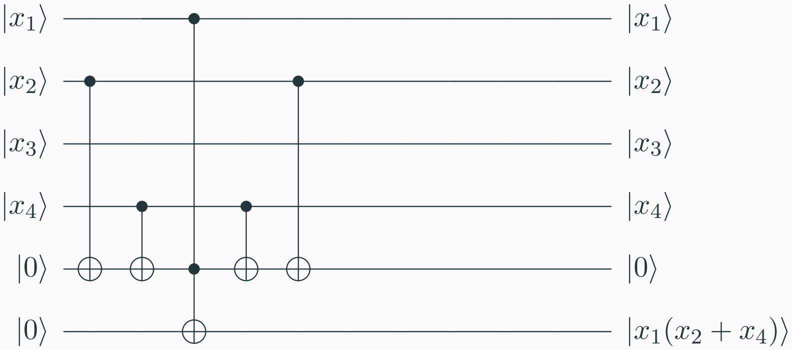
The $\mathcal{MQ}$ quantum oracle II

$$\begin{aligned} p^{(k)}(x_1,x_2,x_3,x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\ &= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1 \end{aligned}$$



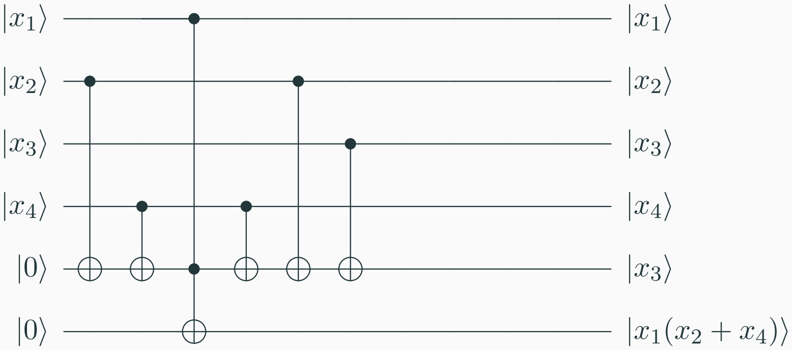
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



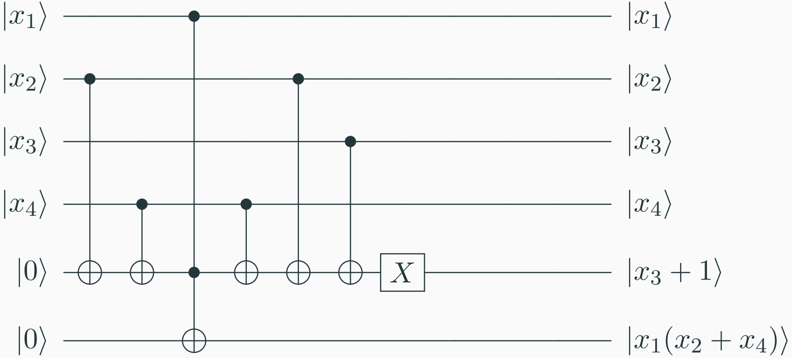
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{\left(x_2 + x_4\right)}_{y_1} + x_2 \underbrace{\left(x_3 + 1\right)}_{y_2} + x_3 \underbrace{\left(x_4\right)}_{y_3} + x_4 \underbrace{\left(1\right)}_{y_4} + 1
\end{aligned}
$$



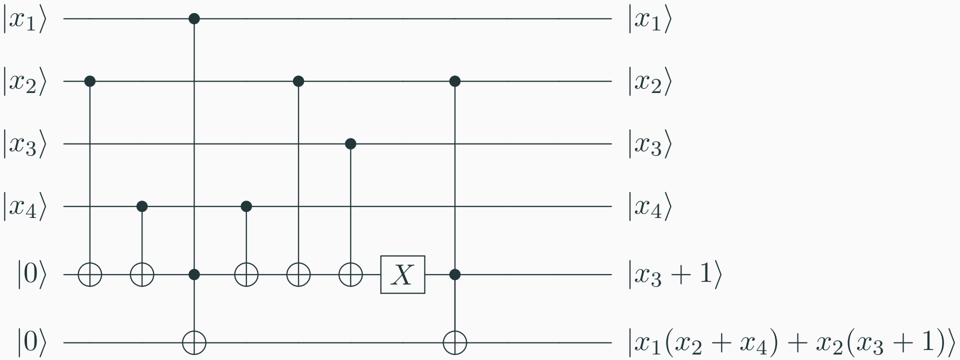
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1,x_2,x_3,x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



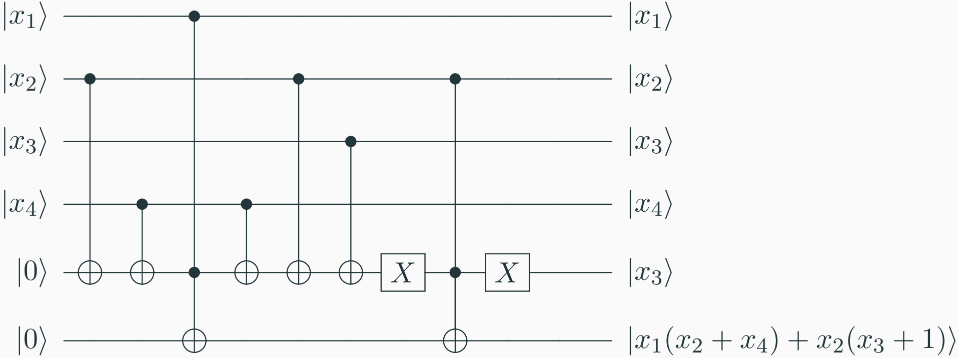
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



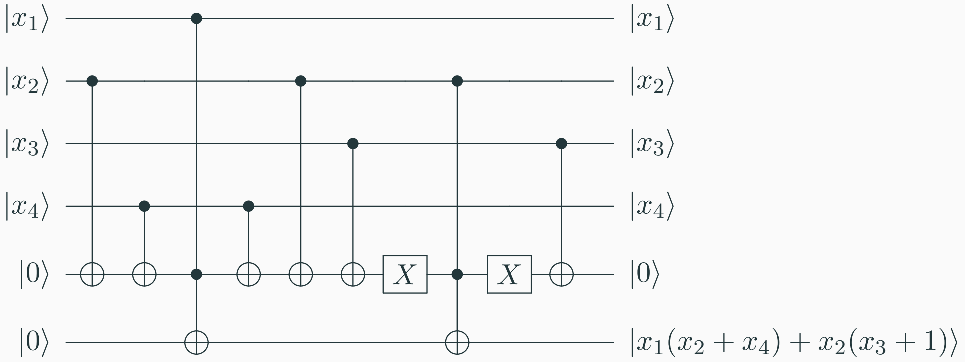
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



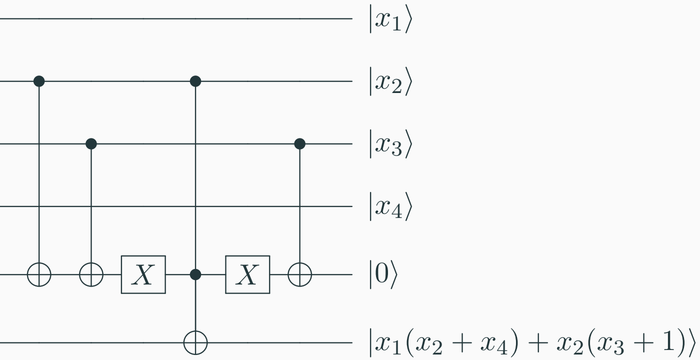
The $\mathcal{MQ}$ quantum oracle II

$$\begin{aligned} p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\ &= x_1 \underbrace{\left(x_2 + x_4\right)}_{y_1} + x_2 \underbrace{\left(x_3 + 1\right)}_{y_2} + x_3 \underbrace{\left(x_4\right)}_{y_3} + x_4 \underbrace{\left(1\right)}_{y_4} + 1 \end{aligned}$$



The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$

$|x_1\rangle$

$|x_2\rangle$

$|x_3\rangle$

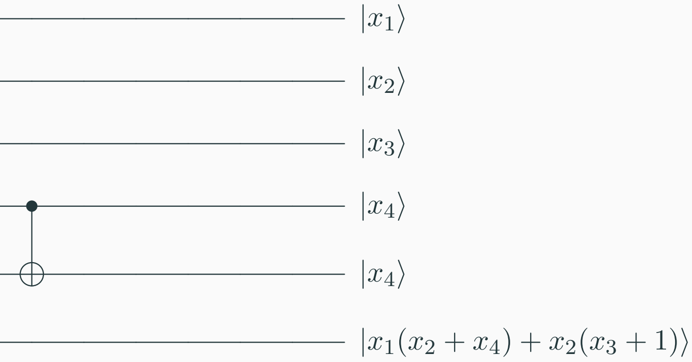
$|x_4\rangle$

$|0\rangle$

$|x_1(x_2 + x_4) + x_2(x_3 + 1)\rangle$

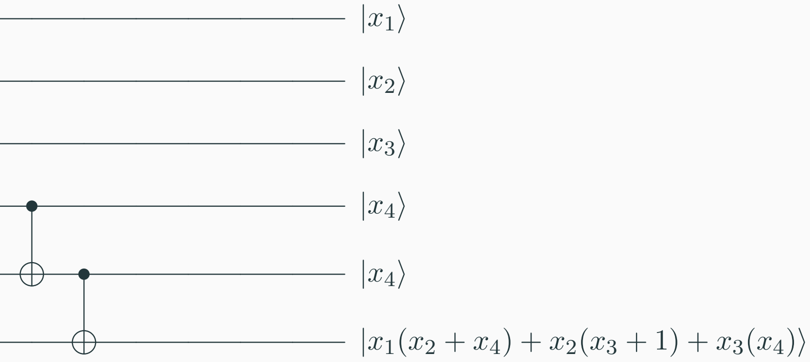
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



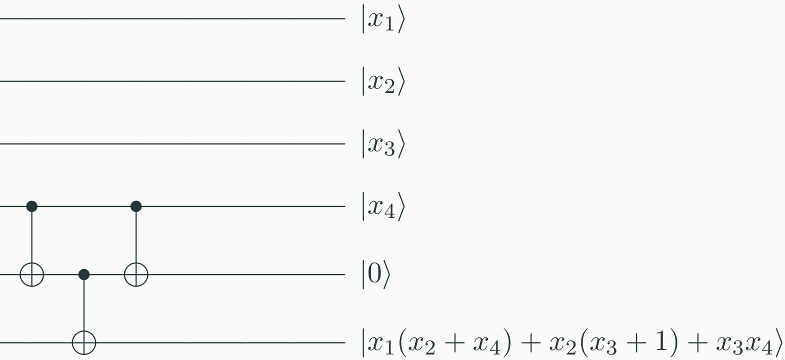
The $\mathcal{MQ}$ quantum oracle II

$$\begin{aligned} p^{(k)}(x_1,x_2,x_3,x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\ &= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1 \end{aligned}$$



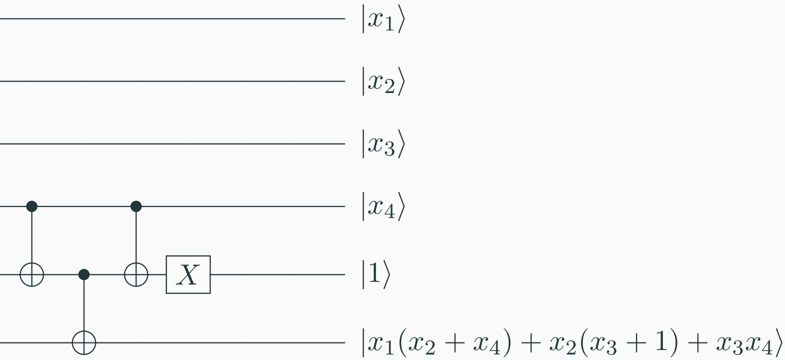
The $\mathcal{MQ}$ quantum oracle II

$$\begin{aligned} p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\ &= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1 \end{aligned}$$



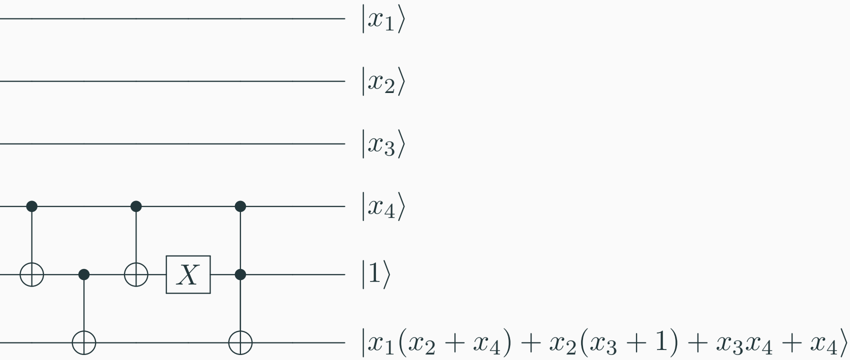
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



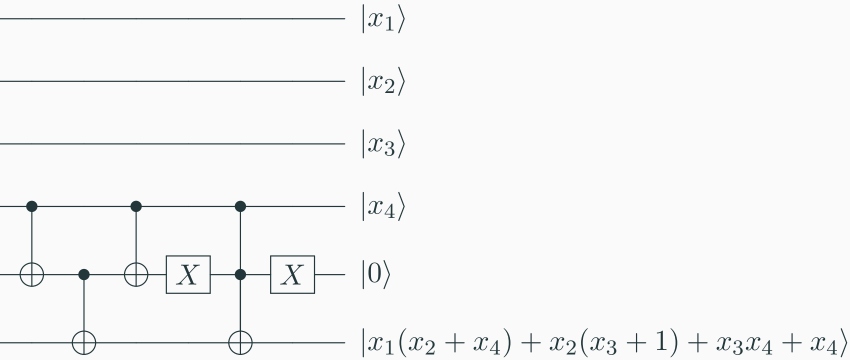
The $\mathcal{MQ}$ quantum oracle II

$$\begin{aligned} p^{(k)}(x_1,x_2,x_3,x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\ &= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1 \end{aligned}$$



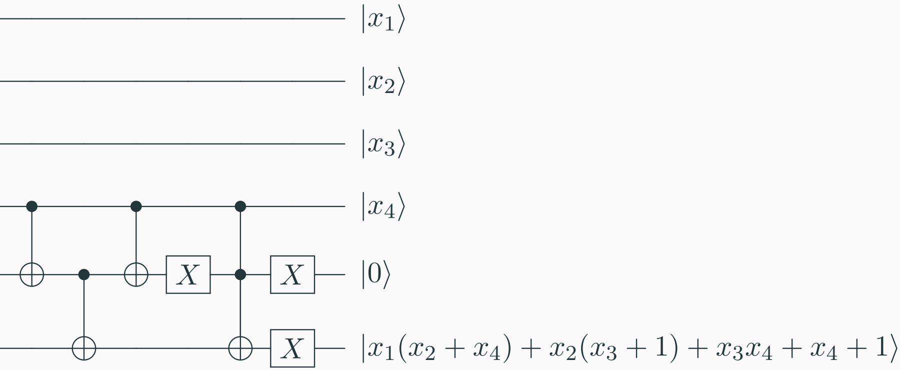
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



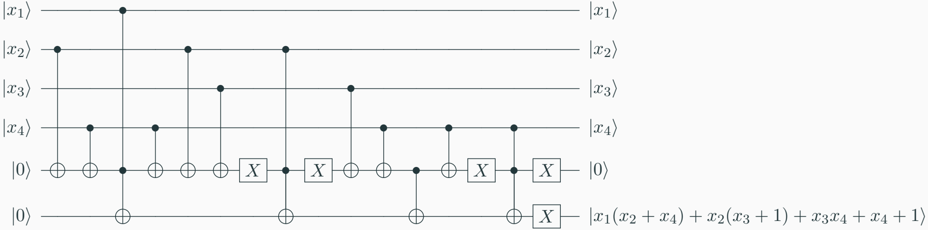
The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



The $\mathcal{MQ}$ quantum oracle II

$$
\begin{aligned}
p^{(k)}(x_1, x_2, x_3, x_4) &= x_1x_2 + x_1x_4 + x_2x_3 + x_3x_4 + x_2 + x_4 + 1 \\
&= x_1 \underbrace{(x_2 + x_4)}_{y_1} + x_2 \underbrace{(x_3 + 1)}_{y_2} + x_3 \underbrace{(x_4)}_{y_3} + x_4 \underbrace{(1)}_{y_4} + 1
\end{aligned}
$$



The $\mathcal{MQ}$ quantum oracle III

(Previous work by Schwabe and Westerbaan):

$$p^{(k)}(x_1, \ldots, x_n) = c^{(k)} + \sum_{i=1}^{n} x_i y_i \quad \text{where} \quad y_i = b_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_j \tag{33}$$

Upper bounding of cost to evaluate a single equation:

- $2n + 1$ X gates for the addition of constants (with uncomputation).
- $n^2 - n$ CNOT gates to compute the y_i (with uncomputation).
- n Toffoli gates adding $x_i y_i$ to the equation register

Cost to evaluate m equations and store them in m registers:

$$m \cdot (n^2 + 2n + 1). \tag{34}$$

We'll ignore specific gate count for now.

The $\mathcal{MQ}$ quantum oracle IV

(Previous work by Schwabe and Westerbaan)

Cost for G: $2\times$ evaluation cost + $1\times$ comparison cost + diffusion cost.

Cost for Grover with $M = 1$:

$$\frac{\pi}{4} \cdot 2^{\frac{n}{2}} \cdot \Big(2m(n^2 + 2n + 1) + \text{Toffoli(m+1)} + \big(\text{Toffoli(n)} + 4n\big)\Big)$$

Our adaptation

Our adaptation I

Aim: optimise the $\mathcal{MQ}$ oracle so that it breaks the parameters for Gui.

Trick 1: obtain a subset of the bits of the solution, then solve again.

Trick 2: offload computation to preprocessing.

Our adaptation II

Each $\mathcal{MQ}$ equation

$$p^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n} b_i^{(k)} + c^{(k)}$$

can be written (for a fixed $0 \leq b \leq n$)

$$g_1^{(k)}(x_1, \ldots, x_{n-b}) + g_2^{(k)}(x_1, \ldots, x_n) + g_3^{(k)}(x_{n-b+1}, \ldots, x_n)$$

where

$$g_1^{(k)}(x_1, \ldots, x_{n-b}) = \sum_{i=1}^{n-b} \sum_{j=i+1}^{n-b} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n-b} b_i^{(k)} x_i$$

$$g_2^{(k)}(x_1, \ldots, x_n) = \sum_{i=1}^{n-b} \sum_{j=n-b+1}^{n} a_{i,j}^{(k)} x_i x_j$$

$$g_3^{(k)}(x_{n-b+1}, \ldots, x_n) = \sum_{i=n-b+1}^{n} \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_i x_j + \sum_{i=n-b+1}^{n} b_i^{(k)} x_i + c^{(k)}$$

Our adaptation III

Trivial that

$$p^{(k)}(\bar{x}_1, \ldots, \bar{x}_{n-b}, \bar{x}_{n-b+1}, \ldots, \bar{x}_n) = 1$$

$$\Updownarrow$$

$$g_1^{(k)}(\bar{x}_1, \ldots, \bar{x}_{n-b}) + g_2^{(k)}(\bar{x}_1, \ldots, \bar{x}_{n-b}, \bar{x}_{n-b+1}, \ldots, \bar{x}_n) + g_3^{(k)}(\bar{x}_{n-b+1}, \ldots, \bar{x}_n) = 1$$

Our adaptation IV

For a fixed $0 \leq b \leq n$, and $\bar{x}_{n-b+1} \ldots \bar{x}_n \in \{0,1\}^b$

$$g_1^{(k)}(x_1, \ldots, x_{n-b}) = \sum_{i=1}^{n-b} \sum_{j=i+1}^{n-b} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n-b} b_i^{(k)} x_i$$

$$g_2^{(k)}(x_1, \ldots, x_{n-b}, x_{n-b+1}, \ldots, x_n) = \sum_{i=1}^{n-b} \sum_{j=n-b+1}^{n} a_{i,j}^{(k)} x_i x_j$$

$$g_3^{(k)}(x_{n-b+1}, \ldots, x_n) = \sum_{i=n-b+1}^{n} \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_i x_j + \sum_{i=n-b+1}^{n} b_i^{(k)} x_i + c^{(k)}$$

Our adaptation IV

For a fixed $0 \leq b \leq n$, and $\bar{x}_{n-b+1} \ldots \bar{x}_n \in \{0,1\}^b$

$$g_1^{(k)}(x_1, \ldots, x_{n-b}) = \sum_{i=1}^{n-b} \sum_{j=i+1}^{n-b} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n-b} b_i^{(k)} x_i$$

$$g_2^{(k)}(x_1, \ldots, x_{n-b}, \bar{x}_{n-b+1}, \ldots, \bar{x}_n) = \sum_{i=1}^{n-b} \sum_{j=n-b+1}^{n} a_{i,j}^{(k)} x_i \bar{x}_j$$

$$g_3^{(k)}(\bar{x}_{n-b+1}, \ldots, \bar{x}_n) = \sum_{i=n-b+1}^{n} \sum_{j=i+1}^{n} a_{i,j}^{(k)} \bar{x}_i \bar{x}_j + \sum_{i=n-b+1}^{n} b_i^{(k)} \bar{x}_i + c^{(k)}$$

Our adaptation IV

For a fixed $0 \leq b \leq n$, and $\bar{x}_{n-b+1} \ldots \bar{x}_n \in \{0,1\}^b$

$$g_1^{(k)}(x_1, \ldots, x_{n-b}) = \sum_{i=1}^{n-b} \sum_{j=i+1}^{n-b} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n-b} b_i^{(k)} x_i$$

$$g_2^{(k)}(x_1, \ldots, x_{n-b}, \bar{x}_{n-b+1}, \ldots, \bar{x}_n) = \sum_{i=1}^{n-b} a_{i,j}'^{(k)} x_i$$

$$g_3^{(k)}(\bar{x}_{n-b+1}, \ldots, \bar{x}_n) = c'^{(k)}$$

Our adaptation IV

For a fixed $0 \leq b \leq n$, and $\bar{x}_{n-b+1} \dots \bar{x}_n \in \{0,1\}^b$

$$g_1^{(k)}(x_1, \dots, x_{n-b}) = \sum_{i=1}^{n-b} \sum_{j=i+1}^{n-b} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n-b} b_i^{(k)} x_i$$

$$g_2^{(k)}(x_1, \dots, x_{n-b}, \bar{x}_{n-b+1}, \dots, \bar{x}_n) = \sum_{i=1}^{n-b} a_{i,j}'^{(k)} x_i$$

$$g_3^{(k)}(\bar{x}_{n-b+1}, \dots, \bar{x}_n) = c'^{(k)}$$

Let i be the integer representation of $\bar{x}_{n-b+1} \dots \bar{x}_n \in \{0,1\}^b$ and define

$$h_i^{(k)}(x_1, \dots, x_{n-b}) = g_2^{(k)}(x_1, \dots, x_{n-b}, \bar{x}_{n-b+1}, \dots, \bar{x}_n) + g_3^{(k)}(\bar{x}_{n-b+1}, \dots, \bar{x}_n).$$

Our adaptation IV

For a fixed $0 \leq b \leq n$, and $\bar{x}_{n-b+1} \ldots \bar{x}_n \in \{0,1\}^b$

$$g_1^{(k)}(x_1, \ldots, x_{n-b}) = \sum_{i=1}^{n-b} \sum_{j=i+1}^{n-b} a_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{n-b} b_i^{(k)} x_i$$

$$g_2^{(k)}(x_1, \ldots, x_{n-b}, \bar{x}_{n-b+1}, \ldots, \bar{x}_n) = \sum_{i=1}^{n-b} a'^{(k)}_{i,j} x_i$$

$$g_3^{(k)}(\bar{x}_{n-b+1}, \ldots, \bar{x}_n) = c'^{(k)}$$

Let i be the integer representation of $\bar{x}_{n-b+1} \ldots \bar{x}_n \in \{0,1\}^b$ and define

$$h_i^{(k)}(x_1, \ldots, x_{n-b}) = g_2^{(k)}(x_1, \ldots, x_{n-b}, \bar{x}_{n-b+1}, \ldots, \bar{x}_n) + g_3^{(k)}(\bar{x}_{n-b+1}, \ldots, \bar{x}_n).$$

For some $x_1 \ldots x_{n-b} \in \{0,1\}^{n-b}$ and $0 \leq i < 2^b$ the equation

$$p^{(k)}(x_1, \ldots, x_{n-b}) = g_0^{(k)}(x_1, \ldots, x_{n-b}) + h_i^{(k)}(x_1, \ldots, x_{n-b})$$

is satisfied.

Our adaptation V

For some $0 \leq i < 2^b$ the following equation will be satisfied

$$p^{(k)}(x_1, \ldots, x_{n-b}) = g_0^{(k)}(x_1, \ldots, x_{n-b}) + h_i^{(k)}(x_1, \ldots, x_{n-b}). \tag{35}$$

The equation will be satisfied if and only if we have chosen the correct $x_1, \ldots, x_{n-b}$ and the correct $0 \leq i < 2^b$.

Our adaptation V

For some $0 \leq i < 2^b$ the following equation will be satisfied

$$p^{(k)}(x_1, \ldots, x_{n-b}) = g_0^{(k)}(x_1, \ldots, x_{n-b}) + h_i^{(k)}(x_1, \ldots, x_{n-b}). \tag{35}$$

The equation will be satisfied if and only if we have chosen the correct $x_1, \ldots, x_{n-b}$ and the correct $0 \leq i < 2^b$. This leads to a strategy:

- Redefine the search problem to be on the variables $x_1, \ldots, x_{n-b}$.

Our adaptation V

For some $0 \leq i < 2^b$ the following equation will be satisfied

$$p^{(k)}(x_1, \ldots, x_{n-b}) = g_0^{(k)}(x_1, \ldots, x_{n-b}) + h_i^{(k)}(x_1, \ldots, x_{n-b}). \tag{35}$$

The equation will be satisfied if and only if we have chosen the correct $x_1, \ldots, x_{n-b}$ and the correct $0 \leq i < 2^b$. This leads to a strategy:

- Redefine the search problem to be on the variables $x_1, \ldots, x_{n-b}$.
- Each evaluation of the quantum oracle consists of
    1. Evaluate $g_0^{(k)}(x_1, \ldots, x_{n-b})$ on the equation registers.

Our adaptation V

For some $0 \leq i < 2^b$ the following equation will be satisfied

$$p^{(k)}(x_1, \ldots, x_{n-b}) = g_0^{(k)}(x_1, \ldots, x_{n-b}) + h_i^{(k)}(x_1, \ldots, x_{n-b}). \tag{35}$$

The equation will be satisfied if and only if we have chosen the correct $x_1, \ldots, x_{n-b}$ and the correct $0 \leq i < 2^b$. This leads to a strategy:

- Redefine the search problem to be on the variables $x_1, \ldots, x_{n-b}$.
- Each evaluation of the quantum oracle consists of
    1. Evaluate $g_0^{(k)}(x_1, \ldots, x_{n-b})$ on the equation registers.
    2. Add $h_0^{(k)}(x_1, \ldots, x_{n-b})$ to the equation registers and test if satisfied.

Our adaptation V

For some $0 \leq i < 2^b$ the following equation will be satisfied

$$p^{(k)}(x_1, \ldots, x_{n-b}) = g_0^{(k)}(x_1, \ldots, x_{n-b}) + h_i^{(k)}(x_1, \ldots, x_{n-b}). \tag{35}$$

The equation will be satisfied if and only if we have chosen the correct $x_1, \ldots, x_{n-b}$ and the correct $0 \leq i < 2^b$. This leads to a strategy:

- Redefine the search problem to be on the variables $x_1, \ldots, x_{n-b}$.
- Each evaluation of the quantum oracle consists of
  1. Evaluate $g_0^{(k)}(x_1, \ldots, x_{n-b})$ on the equation registers.
  2. Add $h_0^{(k)}(x_1, \ldots, x_{n-b})$ to the equation registers and test if satisfied.
  3. For $i = 1, \ldots, 2^b - 1$:
     3.1 Add the difference $h_i^{(k)}(x_1, \ldots, x_{n-b}) - h_{i-1}^{(k)}(x_1, \ldots, x_{n-b})$ and test.

Our adaptation V

For some $0 \leq i < 2^b$ the following equation will be satisfied

$$p^{(k)}(x_1, \ldots, x_{n-b}) = g_0^{(k)}(x_1, \ldots, x_{n-b}) + h_i^{(k)}(x_1, \ldots, x_{n-b}). \tag{35}$$

The equation will be satisfied if and only if we have chosen the correct $x_1, \ldots, x_{n-b}$ and the correct $0 \leq i < 2^b$. This leads to a strategy:

- Redefine the search problem to be on the variables $x_1, \ldots, x_{n-b}$.
- Each evaluation of the quantum oracle consists of
  1. Evaluate $g_0^{(k)}(x_1, \ldots, x_{n-b})$ on the equation registers.
  2. Add $h_0^{(k)}(x_1, \ldots, x_{n-b})$ to the equation registers and test if satisfied.
  3. For $i = 1, \ldots, 2^b - 1$:
     3.1 Add the difference $h_i^{(k)}(x_1, \ldots, x_{n-b}) - h_{i-1}^{(k)}(x_1, \ldots, x_{n-b})$ and test.
  4. Restore the equation registers to the state $g_0^{(k)}(x_1, \ldots, x_{n-b})$ via adding $h_{2^b-1}^{(k)}(x_1, \ldots, x_{n-b})$ to the equation registers.

Our adaptation V

For some $0 \leq i < 2^b$ the following equation will be satisfied

$$p^{(k)}(x_1, \ldots, x_{n-b}) = g_0^{(k)}(x_1, \ldots, x_{n-b}) + h_i^{(k)}(x_1, \ldots, x_{n-b}). \tag{35}$$

The equation will be satisfied if and only if we have chosen the correct $x_1, \ldots, x_{n-b}$ and the correct $0 \leq i < 2^b$. This leads to a strategy:

- Redefine the search problem to be on the variables $x_1, \ldots, x_{n-b}$.
- Each evaluation of the quantum oracle consists of
  1. Evaluate $g_0^{(k)}(x_1, \ldots, x_{n-b})$ on the equation registers.
  2. Add $h_0^{(k)}(x_1, \ldots, x_{n-b})$ to the equation registers and test if satisfied.
  3. For $i = 1, \ldots, 2^b - 1$:
     3.1 Add the difference $h_i^{(k)}(x_1, \ldots, x_{n-b}) - h_{i-1}^{(k)}(x_1, \ldots, x_{n-b})$ and test.
  4. Restore the equation registers to the state $g_0^{(k)}(x_1, \ldots, x_{n-b})$ via adding $h_{2^b-1}^{(k)}(x_1, \ldots, x_{n-b})$ to the equation registers.
  5. Uncompute $g_0^{(k)}(x_1, \ldots, x_{n-b})$.

Our adaptation VI

Remarks

- $h_i^{(k)}$ can all be precomputed.
- Requires strictly b fewer qubits.
- Embarrassingly parallel if we have more qubits.

Our adaptation VI

Remarks

- $h_i^{(k)}$ can all be precomputed.
- Requires strictly b fewer qubits.
- Embarrassingly parallel if we have more qubits.

What does the new method cost?

Our adaptation VI

Remarks

- $h_i^{(k)}$ can all be precomputed.
- Requires strictly b fewer qubits.
- Embarrassingly parallel if we have more qubits.

What does the new method cost?

Evaluate $g_1^{(k)}(x_1, \ldots, x_{n-b})$: $(n-b)^2 + 2(n-b)$ gates.

Our adaptation VI

Remarks

- $h_i^{(k)}$ can all be precomputed.
- Requires strictly b fewer qubits.
- Embarrassingly parallel if we have more qubits.

What does the new method cost?

Evaluate $g_1^{(k)}(x_1, \ldots, x_{n-b})$: $(n-b)^2 + 2(n-b)$ gates.

Evaluate $h_i^{(k)}(x_1, \ldots, x_{n-b})$: $n - b + 1$ gates

Our adaptation VI

Remarks

- $h_i^{(k)}$ can all be precomputed.
- Requires strictly b fewer qubits.
- Embarrassingly parallel if we have more qubits.

What does the new method cost?

Evaluate $g_1^{(k)}(x_1, \ldots, x_{n-b})$: $(n-b)^2 + 2(n-b)$ gates.

Evaluate $h_i^{(k)}(x_1, \ldots, x_{n-b})$: $n-b+1$ gates

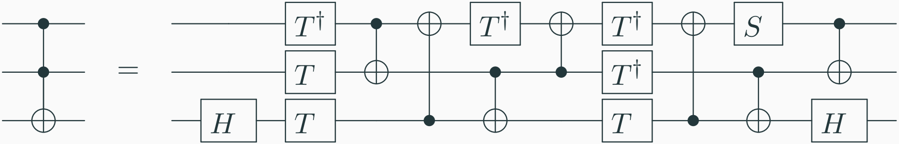
Total cost

$$\frac{\pi}{4} \cdot 2^{\frac{n-b}{2}} \cdot \Big(2m\big((n-b)^2 + 2(n-b)\big) \\ + 2^b\big(m(n-b+1) + \text{Toffoli(m+1)}\big) + m(n-b+1) \\ + \text{Toffoli}(n) + 4n\Big)$$

Clifford+T gate costs

Need to choose a quantum gate set to make a fair comparison!

- Clifford+T gate set — advantageous for error correction.
- Clifford+T = $\{\text{CNOT}, S, S^\dagger, H\} \cup \{T, T^\dagger\}$.
- Toffoli gates and k-bit Toffoli gates expensive, but $\mathcal{O}(k)$ cost.
- We use 2^b $(m+1)$-bit Toffoli gates per Grover iteration vs. only 1.
- Can simply convert using best-known implementations.



The logical Toffoli gate decomposed into Clifford+T gates.

Clifford+T gate costs

Cost in Clifford gates:

$$\frac{\pi}{4} \cdot 2^{\frac{n-b}{2}} \cdot \Big(2m(n-b)^2 + 23m(n-b) + 2^b(m(n-b) + 81m - 160) + 84n - 160\Big)$$

Cost in T gates

$$\frac{\pi}{4} \cdot 2^{\frac{n-b}{2}} \cdot \Big(14m(n-b) + 2^b(52m - 116) + 52(n-b) - 116\Big)$$

Separation of costs is unimportant for cryptanalysis of Gui.

Impact on key-sizes I

Let's rewind (reverse?) back to Gui.

The security of this system relies upon solving a system of underdefined equations. Essentially there are four parameters

- n — the size of the trapdoor extension field.
- v — the number of vinegar variables (# elements we fix).
- a — the number of minus equations (equations).
- k — the repetition factor (# times we must invert the $\mathcal{MQ}$ system).

This essentially forms the $\mathcal{MQ}$ map

$$Gui : \mathbb{F}_2^{n+v} \longrightarrow \mathbb{F}_2^{n-a}, \tag{36}$$

which must be inverted k times in order to break Gui.

By a few tricks, solving an underdefined ($m < n$) systems of equation can be reduced to that of solving m equations in m variables.

Impact on key-sizes II

Authors initially suggest query complexity to derive parameters.

In a subsequent paper, the same authors suggest a new set of parameters is derived from the new binary $\mathcal{MQ}$ oracle, using

$$k \cdot 2^{\frac{n-a}{2}} \cdot 2 \cdot (n-a)^3, \tag{37}$$

the number of quantum gates required to break Gui.

Impact on key-sizes II

Authors initially suggest query complexity to derive parameters.

In a subsequent paper, the same authors suggest a new set of parameters is derived from the new binary $\mathcal{MQ}$ oracle, using

$$k \cdot 2^{\frac{n-a}{2}} \cdot 2 \cdot (n-a)^3, \tag{37}$$

the number of quantum gates required to break Gui.

- Counts Toffoli/multiple-controlled-Toffoli gates as single gates.
- Clifford+T gate implementation of these primitives is costly.
- Estimation and security holds when all costs are accounted for.

Impact on key-sizes III

Gui(n, D, a, v, k)

Gui$(120, 9, 3, 3, 2)$: 80-bit security.

Gui$(212, 9, 3, 4, 2)$: 128-bit security.

Gui$(464, 9, 7, 8, 2)$: 256-bit security.

Applying our method and accounting for costs, we obtain

λ	$n = m$	k	$\mathcal{MQ}$ #Gates	Our #Gates	b	#Qubits used
80	117	2	$2^{80.99}$	$2^{78.38}$	7	236/229
128	209	2	$2^{129.40}$	$2^{126.26}$	8	420/412
256	457	2	$2^{256.71}$	$2^{252.93}$	10	916/906

Number of Clifford+T gates and qubits required to break Gui.

Caveats and conclusions

Caveats:

- Adaptation does NOT break the NIST proposal for Gui, owing to the value MAXDEPTH chosen to be 2^{64}.
- Adaptation cannot be applied to the *low-qubit* $\mathcal{MQ}$ oracle, which uses half the number of qubits but double the gates.

Caveats and conclusions

Caveats:

- Adaptation does NOT break the NIST proposal for Gui, owing to the value MAXDEPTH chosen to be 2^{64}.
- Adaptation cannot be applied to the *low-qubit* $\mathcal{MQ}$ oracle, which uses half the number of qubits but double the gates.

Conclusions

- Exploiting structure of problems obviously helps.
- Offload computation to classical preprocessing where possible.
- Hybrid algorithms may provide far better real-world performance.
- Choose security parameters based upon oracle query complexity.

Thanks!

Questions?