Exploiting preprocessing for quantum search to break parameters for $\mathcal{MQ}$ cryptosystems

Benjamin Pring[1]

University of Bath, Bath, BA2 7AY, UK
`b.i.pring@bath.ac.uk`

Abstract. In this paper we re-examine quantum search applied to the Multivariate Quadratic ($\mathcal{MQ}$) hardness problem over the finite field GF(2). This problem is key to the security of a number of proposed *post-quantum* public-key cryptosystems designed to be resistant against attacks from quantum computers and in this paper we give a warning of the dangers of extrapolating parameters based upon the efficiency of quantum search algorithms. Our methods demonstrate that by applying preprocessing to the $\mathcal{MQ}$ problem, we can reduce the computational load on the quantum computer and, in a generalisation of multi-target search for single-targets, improve the efficiency of the basic quantum search oracle for the $\mathcal{MQ}$ problem over GF(2). Our work builds upon the $\mathcal{MQ}$ oracle introduced by Westerbaan and Schwabe [19] and improves it to the extent that it breaks all quantum-resistant security parameters for the Gui cryptosystem [16] proposed by the original authors [15]. Our results hold both in the logical gate model and when the algorithm is fully costed in terms of the Clifford+T universal gate set.

Keywords: quantum search, multivariate quadratic, cryptography

1 Introduction

The Multivariate Quadratic ($\mathcal{MQ}$) problem, informally that of finding a solution to a system of m degree two equations over a finite field in n variables, is a significant problem in cryptography, given that asymmetric encryption systems can be broken by solving large $\mathcal{MQ}$ systems of equations [4] and that a family of public-key cryptosystems rely upon the hardness of solving this problem [16, 18].

These systems of equations are vulnerable to classical algorithms when the system is either underdetermined ($m \ll n$), overdetermined ($m \gg n$) or sparse (they contain equations using far fewer than n variables). The hardest instances of the $\mathcal{MQ}$ problem are therefore thought to be dense systems when $m \approx n$. These algorithms can generally be placed on a spectrum with one extreme being pure Gröbner bases techniques and the other being exhaustive brute force search, which possesses a complexity of $\mathcal{O}\left(2^{n+2}\log_2 n\right)$ [7]. Whilst the best known classical Las Vegas methods for solving systems over GF(2) possess an asymptotic complexity of $\mathcal{O}\left(2^{0.792n}\right)$ [5], it is thought that these methods only have an advantage over classical exhaustive search when $n > 200$. Whilst classical search is obviously still an effective tool, quantum search (sometimes referred to as Grover's algorithm [12]) offers a further advantage. For a search space of size $N = 2^n$ with M satisfying assignments, the basic *query complexity* of classical search is $\mathcal{O}\left(\frac{N}{M}\right)$ whilst the quantum query complexity is $\mathcal{O}\left(\sqrt{\frac{N}{M}}\right)$. This measure of complexity refers *only* to the number of queries the algorithm must make to a *black box oracle*, which upon input of a bitstring of length n, reveals only whether the bitstring is a solution to our system of equations. This black-box oracle can be realised in classical or quantum circuitry via interpreting each bit of the length n bitstring as an assignment to a separate variable and evaluating the system of equations upon this assignment. Once all equations are evaluated, it is easily checked whether the system of equations is satisfied. Quantum search additionally includes a *diffusion step*, the cost of which is often negligible compared to the implementation of the quantum black-box, or *quantum oracle*. The concrete gate count and gate depth for the entire quantum search algorithm to terminate resulting in one of the M items with probability ≈ 1 is given by

$$\frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \cdot \Big(\text{Cost(Quantum oracle)} + \text{Cost(Diffusion step)}\Big), \qquad (1)$$

where implementing the diffusion step costs $\mathcal{O}\big(\log_2 N\big)$. Unfortunately, or fortunately for the security of cryptosystems, this query complexity has been proven to be a lower-bound [8] and so techniques to reduce the total circuit complexity or gate depth cost consist of reducing the cost of the quantum oracle, lowering the query-complexity or accepting a smaller probability of success. We focus on altering the stated problem so that we only obtain partial information concerning the solution. This allows for a reduction of the query complexity at the expensive of increasing the cost of the quantum oracle, resulting in concrete gains.

1.1 Concrete gains

In order to determine the threat that quantum computing poses to cryptographic standards, a recent trend has been to produce quantum resource estimates for various problems, such as quantum search applied to AES [11], preimage attacks on SHA-2 and SHA-3 [1] and that of Shor's algorithm [17]. Ultimately logical gates must be implemented in a particular architecture and one potential choice is to use the Clifford Gate set and the additional T gate, which together form the universal *Clifford+T* gate set for quantum computation. Unfortunately quantum computers are expected to require a vast amount of error correction and the cost of error correction for T gates is predicted to overwhelm that required for the Clifford gate set. This has led to recent quantum resource estimation papers providing resource counts by counting the T gates and the Clifford gates separately, allowing for an approximation of the difficulty of realising these algorithms in potential near-future architecture. Schwabe and Westerbaan introduced the cryptanalysis of the binary $\mathcal{MQ}$ problem [19] and provided low-qubit algorithms to solve it, but only examined the problem in terms of logical gates. Their work demonstrated that the binary $\mathcal{MQ}$ problem becomes vulnerable to quantum computers in the near future, as important instances can be solved with only hundreds of logical qubits, compared to the thousands required to break other cryptographic problems [11, 17, 1]. This motivates our paper and our results provide further evidence that this problem may be even easier to solve.

Recently, the classical algorithms of BooleanSolve and XL have been adapted to act as oracles in this manner in conjunction with quantum search and their proposed gate count is lower than that required for quantum search with the oracle of Schwabe of Westerbaan [19]. Whilst these are asymptotically superior algorithms, our methods are based purely search-based, require no such assumptions and may require fewer resources for smaller instances.

1.2 Contributions

We provide a method that can be applied to any binary $\mathcal{MQ}$ oracle that evaluates m equations in n variables and stores their results on m registers to create a *partial search* oracle. This technique defines the quantum search oracle as one which checks whether $n - b$ variables lead to a satisfying solution, where $0 \leq b < n$. Using this strategy increases the cost of the quantum oracle, but allows us to benefit from the drop in query complexity. In order to implement this, we must design a circuit based upon a classical preprocessing stage and after the quantum search terminates, we must locate the remaining b variables. These classical costs will be negligible as b is small compared to n.

Parameters for post-quantum cryptosystems are often chosen based upon the efficiency of the best-known attack, as was the case with the Gui cryptosystem [16, 15] and Schwabe and Westerbaan's $\mathcal{MQ}$ oracle [19] and we highlight that our methods should be viewed as a warning that basic structure can be exploited in quantum algorithms in unexpected ways, so that parameters should be chosen conservatively with regards to the best-known quantum attack.

1.3 Organisation of paper

Section 2 recaps the basics of quantum circuitry required to cost our algorithm. Section 3 reviews quantum search. In Section 4 we review existing binary $\mathcal{MQ}$ oracles and describe our quantum algorithm for partial search. In Section 5 we review our results in the Clifford+T gate model. We give our conclusions in Section 6.

2 Quantum algorithms and circuitry

2.1 Quantum computing

Both classical computers and quantum computers can be thought of as performing operations upon a register which stores memory. In the case of classical computers, the register contains *bits*, each of which can take on the value 0 or 1 and an n-bit state may be in one of the 2^n possible states $x \in \{0, 1\}^n$. In contrast, the register of a quantum computer contains *qubits* and may be in a *superposition* of the 2^n states which a classical register might store. We refer to these 2^n states as the *computational basis states*. By superposition, we mean that the quantum state is

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle \,, \tag{2}$$

where $\alpha_x \in \mathbb{C}$ represents the *amplitude* of any particular state $|x\rangle$. Measuring the quantum register results in the collapse of the quantum state into a single n-bit string, where the probability of measuring state $|x\rangle$ is $|\alpha_x|^2$, so that

$$\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1. \tag{3}$$

Quantum algorithms work via manipulating the α_x amplitudes into a state whereby measurement will collapse the superposition into a single bit-string which contains useful information. The manipulation of these amplitudes is performed by means of *quantum gates* which are the quantum analog of classical logic gates whose fundamental property is that they are reversible, in that no information is lost through computing their output. This reversibility allows computations to be uncomputed, but means that arbitrary boolean functions must be implemented with an additional overhead.

Quantum states themselves may be represented as a vector of complex coefficients and quantum gates as unitary matrices — quantum gates therefore act *linearly* upon the quantum states, so that a quantum circuit implementing the boolean function $h : \{0,1\} \longrightarrow \{0,1\}$ acts by

$$U_h \sum_{x\in\{0,1\}^n} \alpha_x \cdot |x\rangle = \sum_{x\in\{0,1\}^n} \alpha_x \cdot U_h |x\rangle \,. \tag{4}$$

Owing to the property of linearity and the nature of Grover's algorithm, it will be sufficient for our purposes to consider the operation of quantum gates upon the individual computational basis states. At times we will represent algorithms via quantum circuits, which should be read from left (input) to right (output).

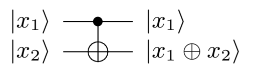
Quantum circuits must be reversible and so to implement an arbitrary boolean function we must adapt the function. This can be achieved by defining the quantum circuit to include the input (n bits), output (m bits) and working memory (w bits) so that the boolean function $h : \{0,1\}^n \longrightarrow \{0,1\}^m$ is interpreted as the quantum circuit $U_h : \{0,1\}^{n+m+w} \longrightarrow \{0,1\}^{n+m+w}$

$$U_h |x\rangle |y\rangle |0\rangle^w \mapsto |x\rangle |y \oplus h(x)\rangle |g(x)\rangle \,. \tag{5}$$

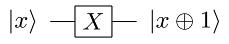
This computation results in a number of so-called "garbage bits", $|g(x)\rangle$, which can be thought of as the end state of the working memory and ancillae bits. If these garbage bits are not dealt with, they will interfere with the diffusion step of quantum search, which relies upon the register being identical on all places apart from the computational basis states representing the search space $|x_1 \dots x_n\rangle$. There is a simple procedure to compute $h(x)$ without garbage bits, which requires an extra space of m-qubits. The procedure consists of computing $|x\rangle |y\rangle |g(x)\rangle |h(x)\rangle$ as in (5) and copying the result to the output register before uncomputing U_h.

2.2 Quantum gates

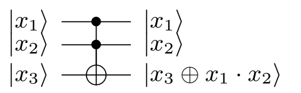
The logical quantum gates we describe below may be decomposed into their Clifford+T gate components as in Figure 2 and we provide further details in section 5. Owing to the linear action of quantum gates, as in (4), it suffices to study the action of these logical gates acting upon an arbitrary computational basis state $|x_1 \dots x_k\rangle$, where $x_i \in \{0,1\}$. We will wish to implement the boolean functions $\wedge_k$ for $k \geq 2$, $\oplus$ and NOT in quantum circuitry. We briefly review these primitives, provided by the designers of the original binary $\mathcal{MQ}$ oracle [19].



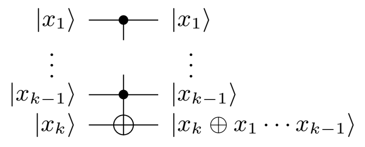
(a) A CNOT gate acting as $x \oplus y$.



(b) An X gate acting as the NOT gate.



(c) A Toffoli gate acting as $x \wedge y$.



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

Fig. 1: Self-inverse gates for building boolean circuits in quantum circuitry.

As the set $\{\wedge, \oplus, \text{NOT}\}$ is a universal boolean gate set, this collection of quantum gates is sufficient to implement arbitrary boolean circuits. We will allow the X gate to act as a primitive gate for the Clifford gate set, though will treat the Toffoli and k-bit Toffoli gate as logical gates for now, whose construction may be optimised dependent upon the availability of ancillae bits.

The Hadamard gate is used in the *diffusion stage* of quantum search and for the construction of Toffoli and k-bit Toffoli gates — facts we will use only for the gate count. Importantly, the Hadamard gate acts upon the state $|1\rangle$ to create

$$H|1\rangle \mapsto \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle\,, \tag{6}$$

which is used to realise the action of the quantum oracle, covered in section 3.2.

2.3 Universal gate sets and error correction

Quantum circuits, much like boolean circuits, may be built out of a finite universal gate set. In line with other quantum resource estimation papers [11, 1, 17], we consider the Clifford+T universal gate set for quantum circuits.

Whilst other universal quantum gate sets are naturally possible, this choice represents a potential, well studied gate set and allows a direct comparison with other literature on quantum resource estimation. The Clifford gate set is generated by the gate set $\{$*Controlled-NOT* (CNOT), *Hadamard* (H) and *Phase* (S)$\}$

and, with the addition of the T gate and its inverse $T^\dagger$, allows approximation of any reversible quantum gate. We will work directly with logical CNOT, X, Hadamard, Toffoli and k-bit Toffoli gates, which may be easily converted into Clifford+T gate representations.

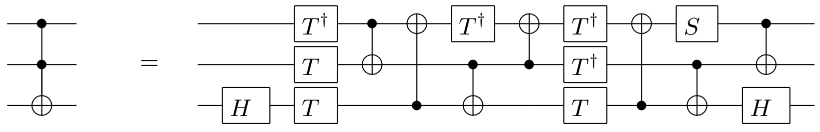


Fig. 2: The logical Toffoli gate decomposed into Clifford+T gates [3, 20].

Owing to the physical scales involved, quantum circuits are inherently vulnerable to noise from their environment. This can be corrected for with an error correction scheme, such as that of surface codes [10], though requires many physical qubits to realise the logical qubits which we use to describe quantum algorithms. The cost of error correction for T gates dominates that required for the Clifford gate set and so it has become a common paradigm to separate quantum resource estimations into the Clifford gate complexity and the T gate complexity. We provide gate count and depth in terms of logical gates and T gates, in line with recent quantum resource estimates [11, 1, 17] in Section 6.

3 Quantum search

3.1 Grover's algorithm

Search problems may be defined by a boolean function $h : \{0,1\}^n \longrightarrow \{0,1\}$ such that $h(x) = 1$ if and only if x is one of the items that fits our criteria. If h is a black-box (that is, when evaluated upon x it reveals no information other than its output), then the number of classical queries we must make to find one of M items in a search space of size N is clearly $\mathcal{O}\left(\frac{N}{M}\right)$.

Grover's quantum search algorithm [12] outputs one of M *marked* items we are searching for in a space of size N and, after an initial setup phase to generate the uniform superposition of n-bit strings, consists of iterating a single procedure, the *Grover iteration*, an optimal number of times before measuring the quantum register. The Grover iteration is composed of two steps — the call to the *quantum oracle* and the *diffusion step* on n-qubits. In essence, the quantum oracle inverts the phase of the items we are searching for, thereby "marking" them and the diffusion step inverts every amplitude around the *mean* of all amplitudes. There is an optimal lower bound to the number of queries to the quantum oracle for measurement to result in a marked item with probability ≈ 1, which is $\approx \frac{\pi}{4} \cdot \sqrt{\frac{N}{M}}$ [12, 8]. Before this bound is reached, the probability of measurement resulting in a marked item will monotonically increase.

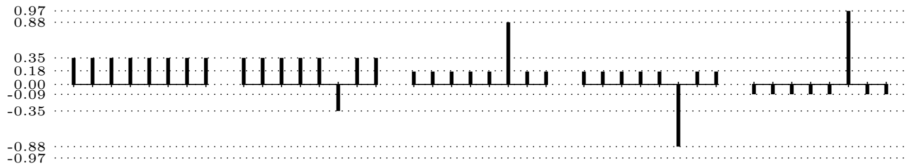


Fig. 3: Evolution of state magnitudes for $N = 8$, $M = 1$ and two iterations.

The total cost of executing Grover's algorithm will be the product of the query complexity with the circuit complexity/depth complexity of the quantum oracle and the diffusion step, therefore being

$$\frac{\pi}{4} \cdot \sqrt{\frac{N}{M}} \cdot \Big(\text{Cost(oracle)} + \text{Cost(diffusion step)}\Big). \tag{7}$$

As we will see, using an efficient exhaustive search method within the quantum oracle simultaneously increases the cost of the oracle whilst both (under mild assumptions and for $M \ll N$) maintaining correctness of the algorithm and reducing the query complexity. As a side benefit, the cost of calling the diffusion step is also mildly reduced. These factors will all lead to real-world gains.

3.2 The quantum oracle

Defining the boolean function $h : \{0,1\}^n \longrightarrow \{0,1\}$ to be the function for which $h(x) = 1$ if and only if x is one of the M items we are searching for, the *quantum oracle* is a quantum circuit which acts upon each computational basis by

$$|x\rangle \mapsto \begin{cases} -|x\rangle & \text{if } h(x) = 1 \\ \ \ |x\rangle & \text{otherwise.} \end{cases} \tag{8}$$

Whilst this can be performed in several ways, our method requires the use of the $|-\rangle$ state described by (6), so that the quantum oracle is realised by the action,

$$|x\rangle \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big) \mapsto |x\rangle \Big(\frac{|0 \oplus h(x)\rangle - |1 \oplus h(x)\rangle}{\sqrt{2}}\Big) = (-1)^{h(x)}\,|x\rangle \Big(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\Big). \tag{9}$$

If we ignore the parts of the register which are identical for all computational basis states, then the action of the quantum oracle is as described by (8). Manipulating the phase of the amplitude according to an implemented boolean function is sometimes referred to as *phase kickback*. The resource requirements for the quantum oracle are naturally dependent upon the search problem.

3.3 The diffusion step

The *diffusion step* affects the amplitudes via the operation of *inversion around the mean*. Details may be found in the reference section [14], but in essence the diffusion step acts upon the amplitudes of the computational basis states so that

$$\alpha_x \mapsto 2 \cdot \langle\alpha\rangle - \alpha_x \qquad \text{where} \qquad \langle\alpha\rangle = \frac{1}{N} \sum_{x\in\{0,1\}^n} \alpha_x. \tag{10}$$

As applying the quantum oracle with phase kickback means that the amplitude of all marked items is negative, we have that the amplitude of marked items after the diffusion step is increased and those of unmarked items are decreased. The diffusion step upon n-qubits is commonly implemented by a circuit involving $2n$ Hadamard gates, $2n$ X gates and a single $(n+1)$-bit Toffoli gate.

4 An $\mathcal{MQ}$ partial search oracle

Definition 1 (The Multivariate Quadratic ($\mathcal{MQ}$) problem).
Given $f^{(1)}, \dots, f^{(m)} \in \mathbb{F}_q[x_1, \dots, x_n]$, where $\mathbb{F}_q$ is the finite field of size q and each equation is of degree two, the Multivariate Quadratic ($\mathcal{MQ}$) problem is to find an $\bar{x} = (\bar{x}_1, \dots, \bar{x}_n)$ with $\bar{x}_i \in \mathbb{F}_q$ such that $f^{(i)}(\bar{x}) = 0$ for $i = 1, \dots, m$.

The *binary* $\mathcal{MQ}$ problem is simply the specialised case when $q = 2$. As a minor modification, we consider the equivalent problem of searching for an $\bar{x} \in \{0,1\}^n$ such that all equations are satisfied when $f^{(i)}(\bar{x}) = 1$ for $i = 1, \dots, m$. This is trivially obtained via addition of 1 to all equations and allows us to easily output whether all equations are satisfied via one $(m+1)$-bit Toffoli gate. In all cases we will assume that $n \leq m < 2n$, as efficient algorithms exist when $m \geq 2n$ [13] and when $m < n$, the system can be reduced to this problem [21].

4.1 Previous work of Schwabe and Westerbaan

Design We review the original binary $\mathcal{MQ}$ oracle [19], which we will use in our partial search oracle and for comparison. The single equation

$$f^{(k)}(x_1, \dots, x_n) = \sum_{1\leq i<j\leq n} a_{i,j}^{(k)} x_i x_j + \sum_{1\leq i\leq n} b_i^{(k)} x_i + c^{(k)} \tag{11}$$

with $a_{i,j}^{(k)}, b_i^{(k)}, c^{(k)} \in \{0,1\}$ may be placed into the equivalent representation

$$f^{(k)}(x_1, \dots, x_n) = c^{(k)} + \sum_{i=1}^{n} y_i^{(k)} x_i \qquad \text{with} \qquad y_i^{(k)} = b_i^{(k)} + \sum_{j=i+1}^{n} a_{i,j}^{(k)} x_j. \tag{12}$$

This representation is then exploited to compute in a manner that is space efficient and reversible. To load the equation $f^{(k)}$ into the register $\left|E^{(k)}\right\rangle$ with the temporary storage qubit $|t\rangle$, the following method may be used. We note $\left|E^{(k)}\right\rangle$ and $|t\rangle$ are initialised to $|0\rangle$.

For $i = 1, \ldots, n$

(a) Compute $y_i^{(k)}$ in the $|t\rangle$ register. The linear component involving the variables is computed using CNOTs with $|x_j\rangle$ as the control and $|t\rangle$ as the target when $a_{i,j} = 1$. The addition of $b_i^{(k)}$ is handled via an X gate.

(b) Add $x_i y_i^{(k)}$ to the register $\left|E^{(k)}\right\rangle$. This is accomplished via a Toffoli gate with the first control set to $|x_i\rangle$ and the second on the register $|t\rangle$, which holds the value $y_i^{(k)}$ by our previous step.

(c) Perform step (a) again to uncompute $y_i^{(i)}$ from the $|t\rangle$ register.

Once the nonconstant part of the equation is loaded, an X gate may be used to perform addition of the constant $c^{(k)}$ if required. To illustrate this, we convert the equation

$$f^{(k)} = x_1x_4 + x_2x_4 + x_1x_5 + x_3x_4 + x_3x_5 + x_4x_5 + x_1 + +x_3 + x_5 + 1 \qquad (13)$$

into

$$\begin{aligned} y_1^{(k)} &= x_4 + x_5 + 1 & y_2^{(k)} &= x_4 & y_3^{(k)} &= x_4 + x_5 + 1 \\ y_4^{(k)} &= x_5 & y_5^{(k)} &= 1. \end{aligned} \qquad (14)$$

which for loading $x_1 y_1^{(k)}$ into the 7th register using the 6th register for temporary storage corresponds to the following circuit

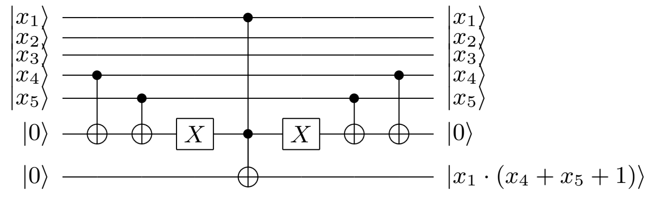


Fig. 4: Addition of $x_1 y_1^{(k)}$ to an equation register via a temporary storage register.

Including the cost of uncomputing the equation registers, this oracle therefore requires $2mn$ Toffoli gates for multiplication, $4mn + 2m$ X gates, $2m(n^2 - n)$ CNOT gates and $n + m + 1$ qubits.

4.2 A partial search oracle

We propose a quantum oracle that can be constructed from *any* circuit that evaluates m binary $\mathcal{MQ}$ equations and stores their result in m registers. For a concrete example we will use the oracle described in section 4.1. Our circuit solves the *k-partial search problem* for the binary $\mathcal{MQ}$ problem, which can simplify solving the initial binary $\mathcal{MQ}$ problem. In essence our strategy is to relax the problem and use preprocessing to reduce the computational load. This can either be optimised towards total gate count or the total number of T gates.

Definition 2 (The k-Partial Search problem). *Given $h:\{0,1\}^n \longrightarrow \{0,1\}$ and a promise that there exist M bit-strings for which $h(x)=1$, the k-partial search problem is to locate at least k bits of a bitstring for which $h(x)=1$.*

If we can obtain the first $n-b$ bits of a valid bitstring, then substitution of these values into a system of m equations in n variables will result in a system of m equations in b variables, a far easier problem to solve either classically via either Gröbner bases/XL algorithms [9, 5, 13] or a classical/quantum search.

4.3 Concept

Our starting point is to view each quadratic equation as a sum of three components, so that the equation $f^{(k)}$ with original representation

$$f^{(k)}(x_1,\ldots,x_n)=\sum_{i=1}^{n}\sum_{j=i+1}^{n}a_{i,j}^{(k)}x_ix_j+\sum_{i=1}^{n}b_i^{(k)}x_i+c^{(k)} \tag{15}$$

is viewed as the equation

$$f^{(k)}(x_1,\ldots,x_n)=g_1^{(k)}(x_1,\ldots,x_{n-b})+g_2^{(k)}(x_1\ldots,x_n)+g_3^{(k)}(x_{n-b+1},\ldots,x_n), \tag{16}$$

where

$$g_1^{(k)}(x_1,\ldots,x_{n-b})=\sum_{i=1}^{n-b}\sum_{j=i+1}^{n-b}a_{i,j}^{(k)}x_ix_j+\sum_{i=1}^{n-b}b_i^{(k)}x_i \tag{17}$$

$$g_2^{(k)}(x_1,\ldots,x_n)=\sum_{i=1}^{n-b}\sum_{j=n-b+1}^{n}a_{i,j}^{(k)}x_ix_j \tag{18}$$

$$g_3^{(k)}(x_{n-b+1},\ldots,x_n)=\sum_{i=n-b+1}^{n}\sum_{j=i+1}^{n}a_{i,j}^{(k)}x_ix_j+\sum_{i=n-b+1}^{n}b_i^{(k)}x_i+c^{(k)}. \tag{19}$$

In this way g_1 consists of the quadratic and linear terms involving only the first $n-b$ variables, g_2 consists of the quadratic terms for which one variable is from the first $n-b$ variables and the second variable is from the last b variables, whilst g_3 consists of the quadratic and linear terms involving the last b variables and the constant term.

Our first observation is that substitution of the last b variables will result in $g_1^{(k)}$ remaining as in (17), whilst $g_2^{(k)}$ will become a linear equation in the first $n-b$ variables and $g_3^{(k)}$ will evaluate to a constant bit. Our circuit design will exploit this, so that our search space for Grover's algorithm will be defined upon the first $n-b$ variables and the oracle itself will evaluate the $g_1^{(k)}$ equations and then implement a method similar to exhaustive search of the final b variables. This exhaustive search is based upon a classical preprocessing step which generates $m\cdot 2^b$ linear equations obtained by substituting $g_2^{(k)}+g_3^{(k)}$ with the last b variables.

The key idea behind our method will be that if b is small, then after the m $g_1^{(k)}$ equations have been evaluated, the addition of the $m \cdot 2^b$ linear equations will be a minor cost compared to that of computing the $g_1^{(k)}$. The use of classical preprocessing reduces this exhaustive search on the final b variables to a circuit using only CNOT gates and $(m+1)$-bit Toffoli gates. This allows us to define the search space on $n-b$ variables and benefit from the drop in query complexity.

4.4 Our hybrid classical/quantum algorithm

Classical preprocessing We first choose an optimal $0 \leq b < n$, which can be easily derived for a fixed n and m via the equations in section 5.3. We note that b will be small ($b = 6$ for $n = m = 80$). We then create the m tuples $(g_1^{(k)}, g_2^{(k)}, g_3^{(k)})$ as in (17), (18) and (19). After this is done, we evaluate the $g_2^{(k)}$ and $g_3^{(k)}$ equations upon all 2^b assignments of the last b variables and store the results. We then have m $g_1^{(k)}$ equations, $m \cdot 2^b$ linear $g_2^{(k)}$ equations and $m \cdot 2^b$ $g_3^{(k)}$ constants. So long as b is small, this will be a negligible cost.

The quantum oracle The search space for our oracle is defined upon the first $n-b$ variables. The oracle first loads m registers with the evaluation of the m $g_1^{(k)}(\bar{x}_1, \ldots, \bar{x}_{n-b})$ equations via any circuit design for evaluation of polynomials and then implements 2^b sub-circuits. Each subcircuit performs the addition of the linear equation from the preprocessed $g_2^{(k)}$ and the constant bit from the preprocessed $g_3^{(k)}$ to the respective equation registers and checks whether the equations have been satisfied via an $(m+1)$-bit Toffoli gate wired to the phase flip bit $|-\rangle$. This has the effect of inverting the phase whenever the first $n-b$ variables and the last b variables captured in the m linear equations that the subcircuit implements lead to a satisfying solution together.

The final subcircuit additionally performs uncomputation to leave each equation register in the state $g_1^{(k)}(\bar{x}_1, \ldots, \bar{x}_{n-b})$. The equation registers may then all be uncomputed via the method used to load them with the values $g_1^{(k)}$.

The 2^b subcircuits are equivalent to performing classical exhaustive search upon the 2^b values that the variables $x_{n-b+1}, \ldots, x_n$ may be assigned. Excluding the first and last of these circuits, each circuit must only add the difference between each pre-processed sum, meaning that we use 2^b $(m+1)$-bit Toffoli gates and at most $(2^b+1) \cdot m(n-b)$ CNOT gates, which can be executed in $(2^b+1) \cdot m$ layers with our assumption that $m \geq n$. There will additionally be at most $(2^b+1) \cdot m$ X gates, which can be performed in (2^b+1) layers. As the search space is only defined on $n-b$ variables, the entire circuit requires b fewer qubits for the oracle than would otherwise be required.

Classical postprocessing It then remains for us to substitute the $n-b$ values obtained from the quantum search bitstring into our system of m equations. We will then have to solve a system of m equations in b variables, but as b will be small, this cost will be negligible compared to the quantum search stage.

Benefits The major benefit of this method is that for small b we can exploit the smaller query complexity, $\mathcal{O}\left(2^{\frac{n-b}{2}}\right)$ as opposed to $\mathcal{O}\left(2^{\frac{n}{2}}\right)$ and the data gathered from the preprocessing stage in an efficient manner. The preprocessing stage helps us via shifting the computational load from the Toffoli gates to CNOT gates, which require less error correction, and $(m+1)$-bit Toffoli gates, whose T count can be more readily optimised. We will obtain gains so long as the combined benefits of the lower query complexity applied to the entire modified oracle result in a a lower gate count to that of the original oracle.

The technique is also embarassingly parallel, in that if we are lucky to have an excess of qubits, then after computing the m values of $g_1^{(k)}$, we can copy these values to m empty registers and execute the 2^b subcircuits in parallel with multiple phase flip bits. Hence any qubits used for parallelism in the circuit for computing $g_1^{(k)}$ need not be idle during the exhaustive search step.

4.5 Cost analysis

We note that the case $b = 0$ is that of the original evaluation circuit being used as a quantum oracle without modification. As with the case of the original oracle, circuit depth is dependent upon the number of additional qubits available — for reasons of space, we leave discussion concerning circuit depth to an expanded version of this paper, which details the circuit depth of Schwabe and Westerbaan's oracle and provides new $\mathcal{MQ}$ evaluation oracles optimised for depth and T gate count.

Circuit-size/depth complexity Fixing a cost metric of a function to be Cost$(\cdot)$, a procedure to evaluate m equations in n variables as Eval(n,m), the $(m+1)$-bit Toffoli gate as Toffoli$(m+1)$ and the diffusion step on n variables as Diff(n), we obtain our maximum advantage when

$$2^{\frac{n-b}{2}} \cdot \Big(2 \cdot \mathrm{Cost}\big(\mathrm{Eval(n\text{-}b,m)}\big) + 2^b \cdot \mathrm{Cost}\big(\mathrm{Toffoli}(m+1)\big) + \mathrm{Cost}\big(\mathrm{Diff(n\text{-}b)}\big)\Big) \tag{20}$$

is minimised. We examine the exact benefits in terms of Clifford+T gates in section 5 after detailing the cost of the Toffoli and $(k+1)$-bit Toffoli gates.

4.6 Valid parameters

We note that if $M > 1$, then multiple solutions may share the same first $n-b$ values. The quantum search procedure described in Section 4.4 may then invert the phase an even number of times, which will be equivalent to no phase inversion, hence the quantum search may fail. In reality, we are dealing with the case where $M \ll N$ (often $M \approx 1$), can fix variables in order to obtain this scenario, or even permute the variable indices upon failure and try again. Nevertheless, a negligible chance of failure remains and without a promise that there exists no collision on the first $n-b$ bits of any two solutions, we must view our adaption as a heuristic adaption to quantum search. One option to correct for this would be to increment a counter [19] controlled on the output of the $(m+1)$-bit Toffoli gate and invert the phase if and only if the counter has been incremented. This would provide for correctness, but the overhead required may destroy our gains.

To obtain gains we must choose an optimal value for b, which is dependent upon the cost of implementing the $(m+1)$-bit Toffoli gate. The next section reviews the T gate cost metric and the cost of our partial search oracle.

5 Analysis and impact in the T gate model

Up until now we have kept our analysis in the logical gate model. In this section we expand upon Section 2.3, briefly justifying the costing of quantum circuitry in terms of separate counts for the Clifford gates and T gates. We provide Clifford+T costs for the Toffoli and k-bit Toffoli gates and provide analysis for partial search oracle. Full details of error correction methods are beyond the scope of this paper and we refer the reader to [1] for a more detailed treatment.

5.1 Error correction and T gates

The cost of error correction for T gates dominates that of those from the Clifford gate set. In essence this is because fault-tolerant implementation of the T gate requires a process referred to as *magic state distillation*. This process requires creation of the so-called *magic state*

$$|\Theta\rangle = \frac{|0\rangle + e^{\frac{i\pi}{4}} |1\rangle}{\sqrt{2}}, \tag{21}$$

in an ancillae qubit which is consumed upon application of each T gate [1].

In order for the entire algorithm to be implemented correctly, we must have that each $|\Theta\rangle$ is produced with an error rate of less than or equal to $\frac{1}{T_U}$, where T_U is the total number of T gates used in the entire quantum algorithm [1]. Creation of these magic states may be handled by so-called *magic state distillation factories*, which produce magic states by processing multiple noisy $|\Theta\rangle$ states and distilling them into a single $|\Theta\rangle$ state with less noise. This process may be repeated until a $|\Theta\rangle$ state is produced with the required error threshold. Given that this process is required only for the implementation of T gates, papers which estimate the cost of quantum algorithms include a separation of cost metrics in either in terms of Clifford gates and T gates [11, 1], or only Toffoli gates [17].

5.2 Clifford+T gate costs for logical gates

Our primitive Clifford gates will be {CNOT, S, H, X}. It then remains for us to provide costs for implementations of the logical Toffoli and k-bit Toffoli gates.

The Toffoli and k-bit Toffoli gates We will use the construction in Figure 2 for the Toffoli gate, which requires 7 T gates and 10 Clifford gates.

From (20), it is obvious that the cost of the $(m+1)$-bit Toffoli gate plays a large part in the efficiency of our methods. We detail more efficient constructions in an expanded version of this paper, but choose an older and relatively inefficient construction to demonstrate the gains that can be made by using our partial search method. The k-bit Toffoli gate can be constructed by elementary means using only Toffoli gates and one ancilla qubit, which breaks down into $80k-240$ Clifford gates and $52k-168$ T gates [6].Without using at least one ancilla qubit, the cost of implementing the k-bit Toffoli gate will be $\mathcal{O}\left(k^2\right)$ [22, 6] and we note that any improvement in circuit design for the multiple controlled Toffoli gate translates has a favourable impact upon the concrete cost of our partial search algorithm. Optimisation programs such as TPar [2] may provide further gains.

5.3 Clifford+T gate costs for our partial search oracle

We provide the cost of the oracle circuits in terms of Clifford gates and in T gates and parameterised by n, m and b, using the original $\mathcal{MQ}$ oracles[19] as our evaluation circuit. This gives us a total cost in terms of Clifford gates of

$$\frac{\pi}{4}\cdot 2^{\frac{n-b}{2}}\cdot\Big(2m(n-b)^2+23m(n-b)+2^b(m(n-b)+81m-160)+84n-160\Big) \tag{22}$$

and a total cost in terms of T gates of

$$\frac{\pi}{4}\cdot 2^{\frac{n-b}{2}}\cdot\Big(14m(n-b)+2^b(52m-116)+52(n-b)-116\Big). \tag{23}$$

Given a fixed n and m, an optimum value of b can easily be found and from these equations it is clear our gains come from ensuring the drop in the contribution from the query complexity is not outweighed by the term involving 2^b.

6 Impact and conclusions

Parameter choice for quantum security The Gui cryptosystem [16] is a proposed *Multivariate Quadratic* public-key signature scheme that is thought to be resistant to quantum attacks. The Gui signature scheme relies upon k (known as the repetition factor) applications of an HFEv- (Hidden Field Equations with Vinegar variables and Minus equations) derived central map, which is a system of m equations in n variables over GF(2). Forging a signature therefore requires inverting the *central map* k times. Whilst parameters for security against only classical attacks were provided in the original paper [16], the authors provide parameters for security against an attack by a quantum computer in a later paper [15], including the parameters for $\lambda = 80, 128$ and 256 bit quantum security (it should require at least 2^λ quantum gates to break the scheme).

For reasons of space, we refer the reader to the original paper for details of Gui, but summarise the number of equations, variables and repetition factor along with the number of gates using the original $\mathcal{MQ}$ oracle and the $\mathcal{MQ}$ oracle with our adaptations in Table 1 below. We use the Clifford+T gate count as we believe this allows for a fair comparison of our methods. Our results still stand (and are in fact better) in a purely logical gate model [19], but we feel that this makes for a fair comparison, given that we use far more multiple controlled Toffoli gates and we do not wish to hide any costs by abstracting this primitive.

λ [15]	$n = m$	k	#Gates using [19]	#Gates using our method	b chosen for our method
80	117	2	$2^{80.99}$	$2^{78.38}$	7
128	209	2	$2^{129.40}$	$2^{126.26}$	8
256	457	2	$2^{256.71}$	$2^{252.93}$	10

Table 1: Number of Clifford+T gates required to break Gui [16, 15].

As is plain from the table above, which can easily be computed by adding formulae (22) and (23), our methods break the proposed quantum-resistant parameters for Gui given in [15]. As our results do not imply any structural weakness in Gui, new parameters for the cryptosystem can easily be chosen to ensure it is secure for the relevant security level, though in doing so we risk again leaving the cryptosystem open to any future optimisations. It is our hope that this demonstrates that when deciding on parameters for a given security level, even when taking into account the best-known quantum algorithms we must be especially conservative, as the structure of the problem may be vulnerable to techniques such as pre processing, which may reduce the overhead of the quantum oracle.

Conclusions We have demonstrated that techniques such as reformulation of the $\mathcal{MQ}$ problem, classical preprocessing and adaptation of multi-target search techniques for single targets can provide us with concrete improvements which can break the quantum resistant parameters of $\mathcal{MQ}$ cryptosystems. Furthermore we have proved that in relation to the Clifford+T gate set, we can specifically lower the total number of T gates and therefore drastically reduce the amount of error correction required. The techniques described in this paper demonstrate both how quantum algorithms may be optimised and that parameters should be chosen conservatively with regards to best-known quantum algorithms.

Acknowledgements The author kindly thanks James Davenport and Christophe Petit for their helpful discussions. Benjamin Pring is funded by an EPRSC grant.

References

[1] Amy, M., Di Matteo, O., Gheorghiu, V., Mosca, M., Parent, A., Schanck, J.: Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. In: Selected Areas in Cryptography-SAC 2016. Springer (2016)

[2] Amy, M., Maslov, D., Mosca, M.: Polynomial-time T-depth optimization of Clifford+ T circuits via matroid partitioning. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 33(10), 1476–1489 (2014)
[3] Amy, M., Maslov, D., Mosca, M., Roetteler, M.: A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 32(6), 818–830 (2013)
[4] Bard, G.: Algebraic cryptanalysis. Springer Science & Business Media (2009)
[5] Bardet, M., Faugère, J.C., Salvy, B., Spaenlehauer, P.J.: On the complexity of solving quadratic boolean systems. Journal of Complexity 29(1), 53–75 (2013)
[6] Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Physical review A 52(5), 3457 (1995)
[7] Bouillaguet, C., Chen, H.C., Cheng, C.M., Chou, T., Niederhagen, R., Shamir, A., Yang, B.Y.: Fast exhaustive search for polynomial systems in $\mathbb{F}_2$. In: Int. Workshop on Cryptographic Hardware and Embedded Systems. pp. 203–218. Springer (2010)
[8] Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. arXiv quant-ph/9605034 (1996)
[9] Courtois, N.T., Patarin, J.: About the XL algorithm over GF (2). In: Proc. of the 2003 RSA Cryptographers' track. pp. 141–157. Springer-Verlag (2003)
[10] Fowler, A.G., Mariantoni, M., Martinis, J.M., Cleland, A.N.: Surface codes: Towards practical large-scale quantum computation. Physical Review A 86(3), 032324 (2012)
[11] Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: quantum resource estimates. In: International Workshop on Post-Quantum Cryptography. pp. 29–43. Springer (2016)
[12] Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proc. of the 28^{th} annual ACM symp. on Theory of computing. pp. 212–219. ACM (1996)
[13] Joux, A., Vitse, V.: A crossbred algorithm for solving boolean polynomial systems. IACR Cryptology ePrint Archive 2017, 372 (2017)
[14] Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press (2010)
[15] Petzoldt, A., Chen, M.S., Ding, J., Yang, B.Y.: Hmfev- an efficient multivariate signature scheme. In: International workshop on post-quantum cryptography. pp. 205–223. Springer (2017)
[16] Petzoldt, A., Chen, M.S., Yang, B.Y., Tao, C., Ding, J.: Design principles for HFEv-based multivariate signature schemes. In: Int. Conference on the Theory and Application of Cryptology and Information Security. pp. 311–334. Springer (2015)
[17] Roetteler, M., Naehrig, M., Svore, K.M., Lauter, K.: Quantum resource estimates for computing elliptic curve discrete logarithms. Asiacrypt (2017)
[18] Sakumoto, K., Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In: CRYPTO. vol. 6841, pp. 706–723. Springer (2011)
[19] Schwabe, P., Westerbaan, B.: Solving binary $\mathcal{MQ}$ with Grover's algorithm. In: SPACE 2016. pp. 303–322. Springer (2016)
[20] Selinger, P.: Quantum circuits of T-depth one. Phys. Rev. A 87(4), 042302 (2013)
[21] Thomae, E., Wolf, C.: Solving underdetermined systems of multivariate quadratic equations revisited. In: Public Key Cryptography. vol. 7293, pp. 156–171. Springer (2012)
[22] Toffoli, T.: Reversible computing. Automata, Languages and Programming pp. 632–644 (1980)