Fast Quantum Algorithm for Solving Multivariate Quadratic Equations

Jean-Charles Faugère[2,1], Kelsey Horan[3], Delaram Kahrobaei[3,4], Marc Kaplan[1,5], Elham Kashefi[1,5], and Ludovic Perret[1,2]

[1] UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France
CNRS, UMR 7606, LIP6, F-75005, Paris, France
[2] INRIA, Paris Center,
Jean-Charles.Faugere@inria.fr, ludovic.perret@lip6.fr
[3] PhD Program in Computer Science, The Graduate Center, The City University of New York
365 5th Ave, New York, NY 10016, USA
[4] New York University, Tandon School of Engineering, Brooklyn, NY 11201, USA
khoran@gradcenter.cuny.edu, dkahrobaei@gc.cuny.edu
[5] School of Informatics
University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, UK
kapmarc@gmail.com, Elham.Kashefi@lip6.fr

Abstract. In August 2015 the cryptographic world was shaken by a sudden and surprising announcement by the US National Security Agency (NSA) concerning plans to transition to post-quantum algorithms. Since this announcement post-quantum cryptography has become a topic of primary interest for several standardization bodies. The transition from the currently deployed public-key algorithms to post-quantum algorithms has been found to be challenging in many aspects. In particular the problem of evaluating the quantum-bit security of such post-quantum cryptosystems remains vastly open. Of course this question is of primarily concern in the process of standardizing the post-quantum cryptosystems. In this paper we consider the quantum security of the problem of solving a system of m *Boolean multivariate quadratic equations in* n *variables* ($\mathtt{MQ}_2$); a central problem in post-quantum cryptography. When $n = m$, under a natural algebraic assumption, we present a Las-Vegas quantum algorithm solving $\mathtt{MQ}_2$ that requires the evaluation of, on average, $O(2^{0.462n})$ quantum gates. To our knowledge this is the fastest algorithm for solving $\mathtt{MQ}_2$.

Keywords: Multivariate Quadratic Equations, Quantum Computation, Quantum Complexity

1 Introduction

The goal of this paper is to study the complexity of solving *systems of Boolean multivariate quadratic equations* ($\mathtt{MQ}_2$) in the quantum setting. This classical NP-hard problem [22] is stated as follows:

$\mathtt{MQ}_2$
Input. $f_1(x_1,\ldots,x_n),\ldots,f_m(x_1,\ldots,x_n) \in \mathbb{F}_2[x_1,\ldots,x_n]$.
Goal. Find – if any – a vector $(z_1,\ldots,z_n) \in \mathbb{F}_2^n$ such that:

$$f_1(z_1,\ldots,z_n) = 0,\ldots,f_m(z_1,\ldots,z_n) = 0.$$

$\mathtt{MQ}_2$ is a fundamental problem with many applications in cryptography, coding theory and beyond. Typically, the security of multivariate schemes is directly related to the hardness of $\mathtt{MQ}_2$, e.g. [20,27,9,5,16,17]. $\mathtt{MQ}_2$ is then central to evaluating the security of such multivariate cryptosystems. Besides multivariate cryptography, the security of a wide variety of cryptosystems is related to $\mathtt{MQ}_2$, via algebraic cryptanalysis [30]. This includes *post-quantum cryptosystems* [6] such as code-based cryptography [19,18], lattice-based cryptography [2,1], ...

The status of post-quantum cryptography is currently completely evolving. It is quickly moving from a purely academic theme to a topic of major industrial interest. This is mainly driven by the fact that post-quantum cryptography has recently received much attention from the standardization and policy sectors. The triggering event appears to be the announcement in August 2015 by the National Security Agency (NSA) of preliminary plans to transition the existing systems to quantum resistant algorithms[6]:

> "*Currently, Suite B cryptographic algorithms are specified by the National Institute of Standards and Technology (*NIST*) and are used by* NSA*'s Information Assurance Directorate in solutions approved for protecting classified and unclassified National Security Systems (*NSS*). Below, we announce preliminary plans for transitioning to* **quantum resistant** *algorithms.*"

This was quickly followed by an announcement by NIST, detailing the transition process [15]. NIST then released in January 2016 a call to select standards for post-quantum public-key cryptosystems: public-key exchange, signature and public-key encryption [29]. The threat to see a large computer in a medium term was considered to be sufficient by NIST to organize a renewal of the public-key cryptosystems deployed in practice.

A key issue for the wide adoption of quantum-safe standards in the future is our confidence in their security. There is, therefore, a great need to develop *quantum cryptanalysis* against post-quantum cryptosystems. It is clear that a challenge in the next years will be to precisely evaluate the *quantum-bit security* of post-quantum cryptosystems submitted to the NIST standardization process.

We study here how quantum techniques can be used to improve the complexity of solving $\mathtt{MQ}_2$; an important problem in post-quantum cryptography. In [4], the authors provide a theoretical upper limit on the speed-up that can be obtained in the quantum setting. They demonstrated that – relative to an oracle chosen uniformly at random – a problem in NP can not be decided by any quantum algorithm in $o(2^{n/2})$. On the other hand, Grover's algorithm [25] is a quantum algorithm than can decide any problem of NP in $O(2^{n/2})$; including $\mathtt{MQ}_2$. Thus, Grover's algorithm is essentially optimal in the setting of [4]. We emphasize that this does not rule out the possibility of a greater than quadratic speed-up in the quantum setting. However, it is mandatory to take advantage of the problem structure to achieve this.

In this paper, we present an algorithm that beats the $O(2^{n/2})$ bound for solving $\mathtt{MQ}_2$. To do so, we combine Grover's technique with a Gröbner basis-based algorithm.

1.1 State of the Art

Classical Setting. The question of solving $\mathtt{MQ}_2$ has been investigated with various algorithmic techniques in the literature. We list below those techniques with the best asymptotic complexity.

Exhaustive search. The first, most obvious, technique for solving $\mathtt{PoSSo}_q$ is exhaustive search. For $q = 2$, the authors of [10] describe a fast exhaustive search for $\mathtt{MQ}_2$ and provide the exact cost of this approach :

$$4 \log_2(n)\, 2^n \text{ binary operations.}$$

A classical (and challenging) theme for $\mathtt{MQ}_2$ is to design algorithms that are asymptotically faster than exhaustive search, i.e. that beat the $O(2^n)$ barrier.

Approximation algorithm. Recently, the authors of [28] proposed new techniques which solve $\mathtt{MQ}_2$ faster than a direct exhaustive search. The techniques from [28] allows for the approximation of a system $F = \big(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\big) \in \mathbb{F}_2[x_1, \ldots, x_n]$ by a single, high-degree, multivariate polynomial P over $n' < n$ variables. The polynomial P is constructed such that it vanishes on the same zeroes as the original system F with high probability. We then must

[6] https://www.nsa.gov/ia/programs/suiteb_cryptography/

perform an exhaustive search on P to recover, with high probability, the zeroes F. This leads to an algorithm for solving $\mathtt{MQ}_2$ with complexity

$$O^*\big(2^{0.8765\,n}\big).$$

The notation O^* omits polynomial factors.

Hybrid approaches. To date, the best methods for solving $\mathtt{MQ}_2$ are based on Gröbner bases [13,12]. More precisely, the fastest methods are hybrid techniques which combine exhaustive search and Gröbner bases algorithms [8,7,3]. `BooleanSolve`, an algorithm originally presented in [3], falls into this category and is the asymptotically fastest approach to solving $\mathtt{MQ}_2$(Section 2.1). When $m = n$, the deterministic variant of `BooleanSolve` has complexity bounded by $O(2^{0.841n})$, while a Las-Vegas variant has expected complexity

$$O(2^{0.792n}).$$

We emphasize that all stated complexities for `BooleanSolve` are obtained under the assumption of a natural algebraic hypothesis on the input system. In contrast, the complexities of [10,28] do not rely on any such assumption.

Quantum Setting. The hardness of $\mathtt{MQ}_q$ has been directly considered in [31], and somewhat indirectly in [16].

Quantum exhaustive search. In [31], the authors proposed simple quantum algorithms for solving $\mathtt{MQ}_2$. The principle is to perform a fast exhaustive search by using Grover's algorithm. The authors derive precise resource estimates for their algorithms, demonstrating that we can solve $m-1$ binary quadratic equations in $n-1$ binary variables using $O(m+n)$ qubits and requiring the evaluation of $O\big(mn^2 2^{n/2}\big)$ quantum gates. The authors also describe a variant using $O\big(n+\log_2(m)\big)$ qubits but with twice as many quantum gates required, when compared to the first approach. In essence, this work constructs a quantum oracle to be used along with amplitude amplification performed by Grover's algorithm. The oracle is fairly simple and takes advantage of the structure of the $\mathtt{MQ}_2$ problem, developing a straightforward way to evaluate a system of equations on a superposition of all possible boolean variable assignments. Then, Grover's algorithm is utilized to amplify those inputs which satisfy all provided equations.

Quantum hybrid approach. The main goal of [16] is to construct a multivariate signature scheme based on random instances of $\mathtt{MQ}_2$ and $\mathtt{MQ}_q$ (for field bigger than $q > 2$). However, in order to derive secure parameters, the authors considered a quantum variant of the hybrid approach from [8,7] using Grover's algorithm. They used this approach to explicitly compute the quantum-bit security of random instances of $\mathtt{MQ}_q$ for given parameters. However, the authors of [16] do not provide the asymptotic complexity of their approach. In this paper, we provide such an asymptotic analysis and build our quantum algorithm on top of `BooleanSolve`. It should be mentioned that `BooleanSolve` is inspired, but different, from [8,7]. So, the quantum algorithm presented here is different from the one sketched in [16].

1.2 Organization of the Paper and Main Results

Overview of the results. The main result of this paper is the fastest known quantum algorithm algorithm for solving $\mathtt{MQ}_2$ (Section 3.1). More precisely:

Theorem 1 (summarized from Section 4). *There is a quantum algorithm that solves* $\mathtt{MQ}_2$ *and requires to*

- *evaluate* $O(2^{0.47n})$ *quantum gates for the deterministic variant,*
- *evaluate an expected number of* $O(2^{0.462n})$ *quantum gates for the probabilistic variant.*

Overview of the results. A natural step towards developing a quantum algorithm for the $\texttt{MQ}_2$ problem which outperforms quantum exhaustive search via Grover's algorithm [31] would be the quantization of a classical algorithm for $\texttt{MQ}_2$ which outperforms classical exhaustive search. A first candidate for such quantization is the approximation algorithm [28] described above. The quantization of such algorithm for use in Grover's algorithm requires building a quantum circuit. Unfortunately, a basic approach to quantize the approximation algorithm mentioned does not seem to be possible, even for $\texttt{MQ}_2$.
Fortunately, we have been able to quantize `BooleanSolve` using amplitude amplification techniques [25,11]. Under a natural algebraic assumption the new algorithm beats quantum exhaustive search, i.e. $O(2^{n/2})$. This is arguably a significant complexity result for a central problem in post-quantum cryptography, but more generally in computer science. The originality of our algorithm is to instantiate Grover's algorithm with a non-trivial oracle that implements the quantum circuit corresponding essentially to a simplified Gröbner basis computation (Section 3.2). We construct the quantum circuit required to implement the simplified Gröbner basis computation.

Cryptographic implications. The complexity analysis is especially important for selecting parameters in multivariate cryptography. It shows that in order to reach a quantum security level of 2^s, one has to consider an instance of $\texttt{MQ}_2$ with at least $s/0.462 = 2.16 \cdot s$ variables. In the table below, we provide the minimal number of variables n (second column) required to reach a precise security level (first column) The public-key in a multivariate cryptosystem is usually given by set of boolean equations. We report in the last column the minimum size required for a given security level.

quantum sec. level	n	$O(n^3)$
64	139	167.36 KB
80	173	326,4 KB
128	277	1.33 MB
256	555	10.65 MB

Finally, we mention that in the signature scheme from [16], the authors proposed to use an instance of $\texttt{MQ}_2$ with $n = m = 256$ variables to achieve a quantum security level of 128 bits. According to our new result, the quantum security is slightly less, i.e. 118 bits.

Organisation. After this introduction, the paper is organized as follows. In Section 2, we first review the two main components of our quantum algorithm : `BooleanSolve` (Section 2.1) and Grover's algorithm (Section 2.2). We describe the new quantum algorithm, `QuantumBooleanSolve`, in Section 3.1. We construct the quantum circuit for a simplified Gröbner basis computation, used as Grover's oracle, in Section 3.2. Finally, we derive in Section 4 the complexity of our algorithm.

2 Preliminaries

In the following we assume familiarity with standard classical and quantum computational notation, such as the standard bra-ket notation for specifying a quantum state. We use the following subsections to overview the classical and quantum algorithms which will be of use in this paper.

2.1 Classical `BooleanSolve`

As explained in the introduction, `BooleanSolve` [3] is the fastest asymptotic algorithm for $\texttt{MQ}_2$. From now on, we will refer to this algorithm as `ClassicalBooleanSolve`. We will indeed present a quantum version of this algorithm, `QuantumlBooleanSolve`, in Section 3.

Essentially, `ClassicalBooleanSolve` first specializes a subset of the variables $x_1, \ldots, x_k$ and then checks the consistency of the specialized system using *Macaulay matrices* (Definition 1). If the specialized system is found to be consistent, the original algebraic system is determined to have

no solution. If the specialized system is inconsistent then the algorithm conducts an exhaustive search on the remaining $n-k$ variables and recovers the solutions for the $\mathtt{MQ}_2$ instance.

We cover the more relevant aspects of the theory behind the algorithm in an effort to keep this paper self contained, and refer the reader to additional preliminary and theoretical information which can be found in the original work [3].

Definition 1. *Let $f \in \mathbb{F}_2[x_1,\ldots,x_n]$, and $\phi(f)$ be the square-free part of f, i.e. the reduction of f modulo $\langle x_i^2 - x_i\rangle_{1\le i\le n}$. The* **Boolean Macaulay matrix** *of degree d for a set of polynomials $F=(f_1,\ldots,f_m)\in\mathbb{F}_2[x_1,\ldots,x_n]^m$, denoted by $\mathcal{M}_d^{\text{acaulay}}(F)$, has the following structure: the rows are the coefficients of polynomials $\{\phi(tf_i)\}$ where $1\le i\le m$, $deg(tf_i)=d,t$ is a square-free monomial, and the columns are the square free monomials in the polynomial ring of degree at most d ordered descendingly with respect to Degree Reverse Lexicographic (*DRL*) ordering.*

We recall below some bounds on boolean Macaulay matrices that will be useful in the complexity analysis.

Proposition 1. *([3]) Let $F=(f_1,\ldots,f_m)\in\mathbb{F}_2[x_1,\ldots,x_n]^m$. Denote by r_{Mac} (resp. $c_{\text{Mac}}, s_{\text{Mac}}$) the number of rows (resp. columns, number of nonzero entries) of the associated boolean Macaulay matrix $\mathcal{M}_d^{\text{acaulay}}(F)$. If $1\le d<\frac{n}{2}$, then*

$$c_{\text{Mac}}<\frac{1-x}{1-2x}\binom{n}{d},\quad r_{\text{Mac}}<m\frac{x^2}{(1-2x)(1-x)}\binom{n}{d},\quad s_{\text{Mac}}<mn^2\frac{x^2}{(1-2x)(1-x)}\binom{n}{d}$$

where $x=\frac{d}{n}$.

`ClassicalBooleanSolve` [3] is based on a fundamental property of Macaulay matrices. Let $F=(f_1,\ldots,f_m)\in\mathbb{F}_2[x_1,\ldots,x_n]^m$ and $\mathbf{M}=\mathcal{M}_d^{\text{acaulay}}(F)$ be the corresponding boolean Macaulay matrix in degree d. It holds that if the linear system

$$u\cdot\mathbf{M}=(0,0,\ldots,0,1)$$

has a solution then F does not have a solution in $\mathbb{F}_2^n$. This reduces the problem of deciding the consistency of non-linear equations to the problem of solving a linear system.

We now need to determine which degree of the Macaulay matrix should be considered. This degree is the so-called *witness degree* defined below:

Definition 2. *([3]) Let $F=(f_1,\ldots,f_m)\in\mathbb{F}_2[x_1,\ldots,x_n]^m$ and $I\subset\mathbb{F}_2[x_1,\ldots,x_n]$ be the ideal defined by F. We set:*

$$\begin{aligned} I_{\le d} &= \big\{p\in\mathbb{F}_2[x_1,\ldots,x_n]\,|\,p\in I,\deg(p)\le d\big\},\\ J_{\le d} &= \big\{p\in\mathbb{F}_2[x_1,\ldots,x_n]\,|\,\exists h_1,\ldots,h_{m+n},\forall i\in\{1,\ldots,m+n\},\deg(h_i)\le d-2,\\ &\qquad p=\sum_{i=1}^{m}h_if_i+\sum_{j=1}^{n}h_{m+j}(x_j^2-x_j)\big\}. \end{aligned}$$

The **witness degree** *for F, denoted $d_{\text{wit}}(F)$, is the smallest integer d_0 such that*

$$I_{\le d_0}=J_{\le d_0}\ \text{ and }\ \langle\{\text{LM}(f)|f\in I_{\le d_0}\}\rangle=\text{LM}(I),$$

where $\text{LM}(f)$ is the leading monomial of the polynomial f with respect to DRL *ordering.*

Alternatively, the witness degree for F can be defined as the degree where any polynomial in a (minimal) Gröbner basis of the system is obtained as a linear combination of the rows of the Macaulay matrix in this degree. Therefore, given $F=(f_1,\ldots,f_m)\in\mathbb{F}_2[x_1,\ldots,x_n]^m$, the witness degree provides an upper bound on the degree d_0 of $\mathcal{M}_{d_0}^{\text{acaulay}}(F)$ required to adequately determine the consistency of F.

Under some algebraic assumptions, the witness degree can be computed explicitly from the Hilbert series:

$$\mathrm{HS}(m,n,k) = \frac{(1+t)^{n-k}}{(1-t)(1+t^2)^m}. \tag{1}$$

The witness degree, denoted by $d_{\mathrm{wit}}(m,n,k)$, is given by the index of the first nonzero coefficient of (1).

Now that we have reviewed all necessary background information, we can present the algorithm from [3] for solving $\mathtt{MQ}_2$.

`ClassicalBooleanSolve`

Input: $f_1,\ldots,f_m \in \mathbb{F}_2[x_1,\ldots,x_n]^m$ with $\deg(f_i)=2$ for all $i \in \{1,\ldots,m\}$.
Output: All boolean solutions to $f_1 = \ldots = f_m = 0$

procedure `ClassicalBooleanSolve(m,n,k)`
  $S = \{\}$
  $d_0 \leftarrow d_{\mathrm{wit}}(m,n,k)$
  for $(a_{n-k+1},\ldots,a_n) \in \mathbb{F}_2^k$ **do**
    for $i = 1 \ldots m$ **do**
      $\tilde{f}_i(x_1,\ldots,x_{n-k}) \leftarrow f_i(x_1,\ldots,x_{n-k},a_{n-k+1},\ldots,a_n) \in \mathbb{F}_2[x_1,\ldots,x_{n-k}]$
    end for
    $\mathbf{M} \leftarrow \mathcal{M}_{d_0}^{\mathrm{acaulay}}(\tilde{f_1},\ldots,\tilde{f_m})$
    if $u \cdot \mathbf{M} = r = (0,\ldots,0,1)$ is inconsistent, determined by the `SparseLinearSystemSolver` **then**
      $T =$ solutions of the system $\tilde{f_1} = \ldots = \tilde{f_m} = 0$ found by exhaustive search
      $S \leftarrow S \cup T$
    end if
  end for
  Return S
end procedure

There are two variants of `ClassicalBooleanSolve` : deterministic and Las-Vegas. The only difference is on the algorithm used in `SparseLinearSystemSolver`, presented in Section 3.2, which can be deterministic or probabilistic. The computational complexity of `ClassicalBooleanSolve` is lower bounded by the complexity of the consistency check of the Macaulay matrices in degree d_0. Therefore, a complete complexity analysis will merely determine the time required to complete the consistency check in term of the input parameters. This yields:

Theorem 2. *([3]) Let $\theta, 2 \le \theta \le 3$ be such that any two $n \times n$ matrices [23] can be multiplied in $O(n^\theta)$ operations in the underlying field. For any $\epsilon > 0$, and $\alpha \ge 1$ and sufficiently large $m = \lceil \alpha n \rceil$, the complexity of all tests of consistency of Macaulay matrices in* `ClassicalBooleanSolve`$(\mathtt{m},\mathtt{n},\mathtt{k})$ *is:*

- $O(2^{(1-\gamma+\theta F_\alpha(\gamma)+\epsilon)n})$ *in the deterministic variant,*
- $O(2^{(1-\gamma+2F_\alpha(\gamma)+\epsilon)n})$ *in the probabilistic variant,*

where $\gamma = 1 - \frac{k}{n}$, $F_\alpha(\gamma) = -\gamma \log_2(D^D(1-D)^{(1-D)})$ *with* $D = M(\frac{\alpha}{\gamma})$, *and*

$$M(x) = -x + \frac{1}{2} + \frac{1}{2}\sqrt{2x^2 - 10x - 1 + 2(x+2)\sqrt{x(x+2)}}.$$

This complexity is obtained by evaluating the cost of checking the consistency of $2^k = 2^{(1-\gamma)n}$ Macaulay matrices.
To derive the asymptotic complexity, we need to assume a certain algebraic condition on the systems considered during the algorithm.

Definition 3. *Let $F = (f_1, \ldots, f_m)$ be quadratic polynomials in $\mathbb{F}_2[x_1, \ldots, x_n]$ and $(1-\gamma)n \leq n$. The system F is called* γ**-strong semi-regular** *if both the set of its solutions and the set*

$$\left\{(a_{n-k+1}, \ldots, a_n) \in \mathbb{F}_2^k \mid d_{\text{wit}}\big(F(x_1, \ldots, x_{n-k}, a_{n-k+1}, \ldots, a_n)\big) > d_{\text{wit}}(m, n, k)\right\}$$

have cardinality at most $2^{(1-\gamma+2F_\alpha(\gamma)+\epsilon)}$, with $\epsilon > 0$ and F_α as in Theorem 2.

Under this assumption, we can now minimize, in term of k, the complexities of Theorem 2. The results are provided for various values of θ: 3 which is the upper bound, 2.376 which is the current best theoretical bound [21], and 2 which requires careful consideration of the linear algebra problem.

Lemma 1. *Let the notations be as in Theorem 2. The function $1 - \gamma + \theta F_\alpha(\gamma)$ is bounded by:*

- $1 - 0.112\alpha$*, when* $\theta = 3$ *and* $\gamma = 0.27\alpha$*,*
- $1 - 0.159\alpha$*, when* $\theta = 2.376$ *and* $\gamma = 0.40\alpha$*,*
- $1 - 0.208\alpha$*, when* $\theta = 2$ *and* $\gamma = 0.55\alpha$*.*

Finally:

Theorem 3. `ClassicalBooleanSolve` *is correct and solves* $\mathtt{MQ}_2$*. If $m = n$, then the algorithm has complexity $O(2^{0.841n})$, if the system is 0.40-strong semi-regular, for the deterministic variant, and of expectation $O(2^{0.792n})$, if the the system is 0.55-strong semi-regular, for the Las-Vegas probabilistic variant.*

This is essentially the cost of the first step, i.e. testing the Macaulay matrices, since the second step, i.e. exhaustive search, has negligible cost when compared to the consistency check.

2.2 Grover's Algorithm

Grover's algorithm [25], often called *database search*, is a quantum algorithm that can be implemented to reduce computation time for the exhaustive search of a function over the entire function domain. The problem solved by Grover's algorithm is as follows: given a function $f : \{0,1\}^n \to \mathbb{F}_2$, determine the unique $x^* \in \{0,1\}^n$ such that $f(x^*) = 1$.
Determining such a x^* with a classical computer requires exhaustive search on the entire function domain of f. Classical computation techniques cannot do better than evaluating f over every possible input, resulting in time complexity of $O^*(2^n)$. Grover's quantum algorithm can determine x^* with merely $2^{\frac{n}{2}}$ evaluations of $\mathcal{F}$, the quantum circuit which evaluates the function f.
Grover's algorithm can be extended to perform exhaustive search over a function where $|f^{-1}(1)| = M$ with $M \geq 1$, as well as searching over a function where the preimage of 1 has arbitrary size. Here we present a simple version of the algorithm.
In the quantum oracle model, when presented with a quantum oracle for the evaluation of f, the problem is to locate an x^* such that $f(x^*) = 1$. The algorithm utilizes two unitary operations. First, a rotation $O_f^{\pm} : \alpha_x|x\rangle \to (-1)^{f(x)}\alpha_x|x\rangle$ which flips the sign of the phase of the desired x^*.

$$O_f^{\pm} : \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} x \to \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n, x \neq x^*} |x\rangle - \frac{1}{\sqrt{2^n}}|x^*\rangle$$

Second, a diffusion operator D which rotates the state around the average amplitude, $\mu = \frac{1}{2^n}\sum_{x \in 2^n} a_x$ of $x \in 2^n$,

$$D : \sum_{x \in \{0,1\}^n} \alpha_x|x\rangle \to \sum_{x \in \{0,1\}^n} (2\mu - \alpha_x)|x\rangle$$

Successive application of these two oracles performs amplitude amplification on the quantum computer, essentially taking the state of the computer from a uniform superposition over all inputs to a state that, when measured, with high probability will return x^*. To converge to such a final state the oracles $O_f^{\pm}$ and D must be applied $\lceil \frac{\pi}{4}\sqrt{\frac{2^n}{|f^{-1}(1)|}} \rceil$ times.

The algorithm proceeds as follows: begin by using a Hadamard gate, $H^{\oplus n}$, to prepare the quantum computer in a uniform superposition over all possible inputs, $\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle$. Following this, apply $O_f^{\pm} D$ to the quantum state $\lceil \frac{\pi}{4} \sqrt{\frac{2^n}{|f^{-1}(1)|}} \rceil$ times. Finally, measure to obtain x^* with high probability. The computational complexity of Grover's algorithm is $O(2^n \cdot \mathcal{F})$ where $\mathcal{F}$ is the complexity of the quantum oracle for f.

Theorem 4 (Amplitude Amplification ([11])). *Let $\mathcal{A}$ be a quantum algorithm that, with no measurement, produces a superposition $\sum_{x \in G} a_x |x\rangle + \sum_{y \in B} a_y |y\rangle$. Let $a = \sum_{x \in G} |a_x|^2$ be the probability of obtaining, after measurement, a state in the good subspace G. Then, there exists a quantum algorithm that calls $\mathcal{A}$ and $\mathcal{A}^{-1}$ as subroutines $O(\frac{1}{\sqrt{a}})$ times and produces an outcome $x \in G$ with a probability at least $max(a, 1-a)$.*

The key to successfully performing Grover's algorithm for the function f is to determine the quantum circuit for the function, in order to construct $O_f^{\pm}$. It is sufficient to provide an oracle that computes the function f, i.e. provide a unitary operator U_f in the form of a quantum circuit which calculates $|x\rangle|y\rangle \to |x\rangle|y \oplus f(x)\rangle$, evaluating the function at a superposition of all possible inputs. Then, Grover's algorithm can be used to amplify the desired output for measurement. What remains is to show that the quantum analog of `ClassicalBooleanSolve` is reversible and computable on a quantum computer. In the following section, we will construct the quantum circuit for the algorithm `ClassicalBooleanSolve` and analyze the complexity of the circuit.

2.3 Quantum Gates

The following gates are quantum gates of interest which operate on qubits, each directly corresponding to reversible classical gates. For qubits $|x\rangle, |y\rangle, |z\rangle$ the gates perform the following operations:

- *CNOT* (XOR, Feynman)
$$\mathrm{CNOT}|x\rangle|y\rangle = |x\rangle|x + y\rangle$$
- *Toffoli* (AND)
$$\mathrm{T}|x\rangle|y\rangle|z\rangle = |x\rangle|y\rangle|z + xy\rangle$$
- *X* (NOT)
$$\mathrm{X}|x\rangle = |\bar{x}\rangle = |1 + x\rangle$$
- *n-qubit Toffoli* (AND)
$$\mathrm{T}_n|x_1\rangle \dots |x_n\rangle = |x_1\rangle \dots |x_{n-1}\rangle|x_n + (x_1 \dots x_{n-1})\rangle$$
- *Swap*
$$\mathrm{S}|x\rangle|y\rangle = |y\rangle|x\rangle$$

It is important to note that $T_1 = X$, $T_2 = \mathrm{CNOT}$ and $T_3 = T$. In terms of computational complexity, X, SWAP and CNOT gates are relatively cheap to compute, while the n-qubit Toffoli gates are more expensive; T_n is equivalent to computing $2n$ CNOT gates. Accounting for these equivalences can change the reported computational complexity of a given circuit. Additionally, it is important to note that one can emulate a Toffoli gate over $\mathbb{F}_q$ using $\log(q)$ basic T gates, in order to determine the additional resources required for any general extension of quantum computations over $\mathbb{F}_2$ to quantum computations over $\mathbb{F}_q$.

3 A Quantum Version of `BooleanSolve`

3.1 `QuantumBooleanSolve`

We explain here how to combine the `ClassicalBooleanSolve` algorithm (Section 2.1) with Grover's algorithm (Section 2.2). `ClassicalBooleanSolve` conducts two exhaustive searches over the variables. The first exhaustive search is over the last k variables, specializing $x_{n-k+1}, \ldots, x_n$ and projecting to the k last components of a solution. The second exhaustive search, when necessary, is on the first $n-k$ variables, and allows the algorithm to determine the entire solution. It is clear that one can utilize Grover's algorithm, as in [14], to quantize the second exhaustive search and obtain a speed up over the classical complexity. In what follows, `QuantumSearch` will refer to the quantum algorithm [14] that solves $\texttt{MQ}_2$. We will see that Grover's algorithm can be used to speed-up the first exhaustive search as well. Essentially, we will quantize the consistency check on Macaulay matrices by providing a quantum circuit which can be used as the function oracle in Grover's algorithm.

Let $F = (f_1, \ldots, f_m) \in \mathbb{F}_2[x_1, \ldots, x_n]^m$. We consider the function $F^{\text{cons}}_{F,k} : \mathbb{F}_2^k \mapsto \{0,1\}$ which evaluates F on $(x_1, \ldots, x_{n-k}, y_1, \ldots, y_k)$ with $(y_1, \ldots, y_k) \in \mathbb{F}_2^k$ and returns 1 only if the non-linear system defined below is consistent :

$$\begin{aligned}\tilde{F} &= (\tilde{f}_1, \ldots, \tilde{f}_m) \\ &= \big(f_1(x_1, \ldots, x_{n-k}, y_1, \ldots, y_k), \ldots, f_m(x_1, \ldots, x_{n-k}, y_1, \ldots, y_k)\big) \in \mathbb{F}_2^m[x_1, \ldots, x_{n-k}]\end{aligned}$$

This is reduced to check whether the linear system below has a solution:

$$u \cdot \mathcal{M}_d^{\text{acaulay}}(\tilde{F}) = (0, 0, \ldots, 0, 1), \text{ for a well chosen } d. \tag{2}$$

In order to quantize `ClassicalBooleanSolve`, we then proceed in two steps. We first use Grover's algorithm, along with the quantum circuit which evaluates $F^{\text{cons}}_{F,k}$ to determine $\mathbf{a}_2 := (a_{n-k+1}, \ldots, a_n) \in \mathbb{F}_2^k$ such that $F^{\text{cons}}_{F,k}(\mathbf{a}_2) = 1$, i.e. $\mathbf{a}_2$ is such that the non-linear system below is consistent:

$$\tilde{F} := (\tilde{f}_1, \ldots, \tilde{f}_m) = \big(f_1(x_1, \ldots, x_{n-k}, \mathbf{a}_2), \ldots, f_m(x_1, \ldots, x_{n-k+1}, \mathbf{a}_2)\big).$$

The $\mathbf{a}_2$ then corresponds to the variable assignments of the last components in a solution for the system F. We can then use Grover's algorithm, via `QuantumSearch` on $\tilde{F}$ to find the remainder of a complete solution corresponding to $\mathbf{a}_2$.

1: **procedure** `QuantumBooleanSolve(m,n,k)`
2: $\quad \mathbf{a}_2 := (a_{n-k+1}, \ldots, a_n) := \texttt{GroverSearch}(F^{\text{cons}}_{F,k})$
3: $\quad \tilde{F} \leftarrow (\tilde{f}_1, \ldots, \tilde{f}_m) = \big(f_1(x_1, \ldots, x_{n-k}, \mathbf{a}_2), \ldots, f_m(x_1, \ldots, x_{n-k+1}, \mathbf{a}_2)\big)$
4: $\quad \mathbf{a}_1 := (a_1, \ldots, a_{n-k}) := \texttt{QuantumSearch}(\tilde{F})$
5: $\quad$ Return $(\mathbf{a}_1, \mathbf{a}_2)$
6: **end procedure**

The most essential part of determining the advantage of using Grover's algorithm to improve the computational complexity of `ClassicalBooleanSolve` is to construct the quantum circuit for $F^{\text{cons}}_{F,k}$). Below, we construct the quantum circuit that solves (2).

3.2 Quantum Oracle

`QuantumBooleanSolve` consists of constructing a quantum oracle for the consistency check of Macaulay matrices, i.e. for $F^{\text{cons}}_{F,k}$. We provide the classical complete sparse linear system solver for classical consistency checks below, as presented by Giesbrecht et al. [24], and then provide an outline of the quantum circuit. Classical algorithm complexity is provided in a black-box model, where we assume access to a black-box for computing matrix-matrix and matrix-vector products. Then, the complexity is given as the number of calls to such black boxes, as well as the number of additional field operations required.

Classical case. `SparseLinearSystemSolver` is the classical algorithm employed to determine the consistency of the Macaulay matrices in degree d_0. The algorithm takes as input a matrix A, a vector b and a subset of the field or a field extension $\mathcal{L}$, and outputs either a solution x to $Ax = b$ or a certificate of inconsistency, u. This classical algorithm requires $O(n)$ evaluations of black box algorithms for matrix-vector multiplications, as well as an additional $O(n^2 \log n \log\log n)$ additional field operations. Subroutines of the `SparseLinearSystemSolver` can be found below.

`SparseLinearSystemSolver`

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}, \mathcal{L} \subset \mathbb{F}$ with $|\mathcal{L}| > 2n(n-1)$
Output: Any of the following 3 return values, as an evaluation of the matrix A: *(nonsingular, x)* where $x = A^{-1}b$, *(singular-consistent, x)* with x a random element of the solution space, *(singular-inconsistent, u)* with $u^tA = 0$ and $u^tb \neq 0$, certifying the inconsistency of the system.

1: **procedure** `SparseLinearSystemSolver`
2: $(\hat{f}(z), x) \leftarrow$ `Wiedemann`$(A, b, \mathcal{L})$
3: **if** $x \in \mathbb{F}^{n\times 1}$ and $Ax = b$ **then**
4: Return *(nonsingular,x)*
5: **end if**
6: $\alpha_2, \alpha_3, \ldots, \alpha_n, \beta_2, \ldots, \beta_n, \gamma_1, \ldots, \gamma_n \xleftarrow{\$} \mathcal{L}$
7:

$$U = \begin{bmatrix} 1 & \alpha_2 & \alpha_3 & \alpha_4 & \ldots & \alpha_{n-1} & \alpha_n \\ 0 & 1 & \alpha_2 & \alpha_3 & \ldots & \alpha_{n-2} & \alpha_{n-1} \\ 0 & 0 & 1 & \alpha_2 & \ldots & \alpha_{n-3} & \alpha_{n-2} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}$$

8:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 & 0 \\ \beta_2 & 1 & 0 & 0 & \ldots & 0 & 0 \\ \beta_3 & \beta_2 & 1 & 0 & \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \beta_n & \beta_{n-1} & \beta_{n-2} & \beta_{n-3} & \ldots & \beta_2 & 1 \end{bmatrix}$$

9: $B \leftarrow UAL$
10: $\hat{f}(z) \leftarrow$ `Wiedemann`$(B, 0, \mathcal{L})$ the minpoly of B
11: $f(z) \leftarrow \frac{\hat{f}(z)}{z}$
12: **if** $z | f(z)$ **then**
13: Repeat all steps following the random generation of U, L
14: **end if**
15: $r \leftarrow \deg(f)$
16: $c \leftarrow Ub$
17: **if** *(True,u)* $\leftarrow$ `RandomSol`$(B^t, 0, f, \mathcal{L})$ and $u^tc \neq 0$ **then**
18: Return *(singular-inconsistent, u)*
19: **end if**
20: **if** *(True, x)* $\leftarrow$ `RandomSol`$(B, c, f, \mathcal{L})$ **then**
21: Return *(singular-consistent, x)*
22: **end if**
23: Return to Step 6
24: **end procedure**

`RandomSol`: a subroutine of `SparseLinearSystemSolver`, intended to return a random element of the solution space to the system $Ax = b$. The algorithm is stated to require $O(r)$ evaluations of the black-box for matrix-vector product, and $O(nr)$ additional field operations.

`RandomSol`

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^{n\times 1}, f(z) \in \mathbb{F}[z]$ with $f(0) \neq 0$, and $\mathcal{L} \subset \mathbb{F}$ with $|\mathcal{L}| > 2n(n-1)$

Output: One of the following two return values: *(False)* indicating no solution, *(True, $\hat{x}$)* with $\hat{x}$ a random solution to the system $Ax = b$.

procedure `RandomSol`
  $w \leftarrow (w_1, \ldots, w_n)^t$ with $w_i \in_\$ \mathcal{L}$
  $r \leftarrow \deg(f(z))$
  $b' = (b'_1, \ldots, b'_r) \leftarrow b + Aw$, the first r entries of the calculated vector b'
  A_r, the leading $r \times r$ submatrix of A
  $x \leftarrow -\sum_{i=1}^{n} \frac{f_i}{f_0} A_r^i b'$
  if $Ax = b$ **then**
    Return *(True, x)*
  end if
  Return *(False)*
end procedure

`Wiedemann`: a subroutine of `SparseLinearSystemSolver`. The deterministic version of the `Wiedemann` algorithm is presented below, as seen in [32].

`Wiedemann`

Input: $A \in \mathbb{F}^{n \times n}$, $b \in \mathbb{F}^{n \times 1}$, $\mathcal{L} \subset \mathbb{F}$
Output: One of the following two return values: x such that $x \in \mathbb{F}^{n \times 1}$ and $Ax = b$ or $\hat{f}(z)$, a factor of minpoly$(A) \in \mathbb{F}[z]$.

procedure `Wiedemann`
  $S = \{\}$
  for i = 0 ... 2n-1 **do**
    $S = S \cup \{A^i b\}$
  end for
  $k \leftarrow 0$
  $g_0(z) \leftarrow 1$
  while $\deg(g_k) < n$ and $k < n$ **do**
    $u_{k+1} \leftarrow e_{k+1}$
    $s_k \leftarrow \{(u_{k+1}, A^i b)\}_{i=0}^{2n-1}$
    $gs_k \leftarrow \{(u_{k+1}, A^i g_k(A) b)\}_{i=0}^{2n-1-\deg(g_k)}$
    $f_{k+1} \leftarrow$ minpoly(gs_k) using the Berlekamp-Massey algorithm
    $g_{k+1} \leftarrow f_{k+1} g_k$
    $k \leftarrow k + 1$
  end while
  Return $x = -\sum_{i=1}^{\deg(g_k)} g_k[i] A^{i-1} b$
end procedure

From the presentation of the classical algorithm `SparseLinearSystemSolver`, and the subroutines, it is clear that the significant operations performed classically are matrix multiplication and vector-matrix products. The complexity analysis for such operations is given in a black-box model. Therefore, it is sufficient to show that these operations can be executed (in comparable time) by a quantum circuit on a quantum computer, in order to construct a quantum oracle for matrix consistency checking.

Quantum case. As the basic operations implemented by the classical consistency check are linear algebra on matrices, it is important to verify that this linear algebra can be computed on a superposition of inputs via a quantum circuit. We must check that a reversible unitary operation can compute the required algebraic computations on a quantum computer, with comparable computational complexity to their classical analogs.

Equality Testing. Binary equality testing, i.e. checking if $b = \tilde{b}$, can be computed via one CNOT and one X gate, as follows: $|b\rangle|\tilde{b}\rangle|0\rangle \rightarrow |b\rangle|\tilde{b}\rangle|b \oplus \tilde{b} \oplus 1\rangle$.

Matrix Vector Multiplication. The formula for matrix vector multiplication Ax calculates each element of the product vector $b = (b_1, \ldots, b_n) = (a_{11}x_1 + a_{12}x_2 + a_{1n}x_n, a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n, \ldots, a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n)$. To compute each b_i we require at most n products between the elements of A and of x. Therefore, the matrix-vector product requires at most n^2 products. It remains to show that the computation is reversible.
Figure 1 shows a reversible circuit for computing the inner product between two vectors. This is simply done by replacing the products by Toffoli gates. This quantum circuit is also itself reversible; applying the circuit twice leads to identity. In total, the inner product of two n-bit vectors can be computed, on a superposition of inputs, using n Toffoli gates. Therefore, the computation of such a matrix vector multiplication requires at most n^2 Toffoli gates.

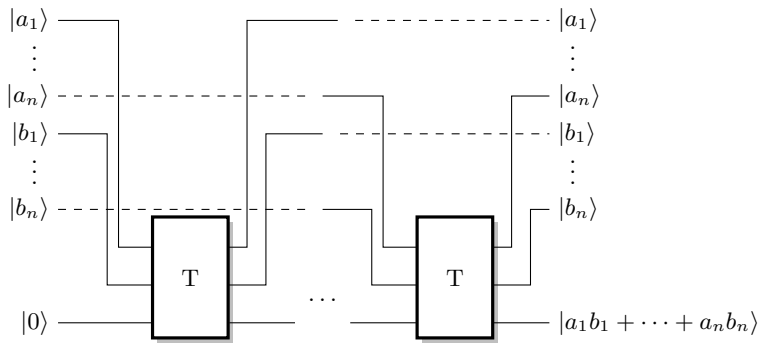

Fig. 1. Computing the inner product on a superposition of inputs.

Matrix Multiplication. In the same fashion it is possible to compute matrix products in the quantum setting. Each column of the matrix can be computed using matrix vector multiplication, which, in turn, can be implemented using n^2 Toffoli gates. In total, a reversible quantum circuit for computing matrix multiplication on a superposition of inputs requires at most n^3 Toffoli gates, for square matrices.

Utilizing this quantum circuit for inner product between two vectors the quantum oracle for consistency checking can be constructed. Despite the fact that this naive quantum matrix multiplication is computed in $O(n^3)$, time greater than $O(n^\omega)$ where $\omega \sim 2.376$ is the current classical complexity of matrix multiplication with the Coppersmith-Winograd algorithm, quantizing this computation will result in a lower quantum computational time for the classical $\mathtt{MQ}_2$ problem. `ClassicalBooleanSolve`, as well as `SparseLinearSystemSolver` and the provided subroutines are analyzed in the black box model, where matrix multiplication such as $x \to Ax$ for a vector x and a matrix A are given by black boxes. We have provided the above construction to assure that such computations can be carried out by a quantum computer, reversibly and without entanglement concerns.

4 Complexity Analysis

We can now study the complexity of `QuantumBooleanSolve` (Section 3.1). This analysis consists of constructing the quantum oracle $\mathcal{QBS}$ implementing $F^{\text{cons}}_{F,k} : \{0,1\}^k \to \{0,1\}$, which on input $a \in \mathbb{F}_2^k$ specializes the polynomial system F and indicates the consistency of the associated Macaulay matrix M_a of appropriate degree. This can be done by analyzing `SparseLinearSystemSolver` and any associated subroutines separately, either illustrating the equivalence between the complexity of the classical function and the quantum circuit or proving that the quantum circuit is more efficient.

For example, the quantization of the subroutine `RandomSol` would consist of constructing a quantum circuit $\mathcal{QRS}$. `RandomSol` takes as input a matrix $A \in \mathbb{F}^{n\times n}$, a vector $b \in \mathbb{F}^{n\times 1}$, a polynomial $f(z) \in \mathbb{F}[z]$ with $f(0) \neq 0$ (and $\mathcal{L} \subset \mathbb{F}$, which is in the case of $\mathtt{MQ}_2$ a field extention of $\mathbb{F}_2$. We would build

$$\mathcal{QRS} : |a_{11}\rangle \dots |a_{nn}\rangle |f[0]^{-1}\rangle |f[1]\rangle \dots |f[n]\rangle |r\rangle |w_1\rangle \dots |w_n\rangle |b_1\rangle \dots |b_n\rangle |0\rangle \dots |0\rangle \rightarrow$$
$$|a_{11}\rangle \dots |a_{nn}\rangle |f[0]^{-1}\rangle |f[1]\rangle \dots |f[n]\rangle |r\rangle |w_1\rangle \dots |w_n\rangle |b_1\rangle \dots |b_n\rangle |0\rangle \dots |0\rangle |b'_1\rangle \dots |b'_r\rangle |x_1\rangle \dots |x_n\rangle |s_A\rangle$$

This quantum circuit takes as input the elements of the matrix A, the coefficients of the function f, the elements of the random vector w, $r = \deg(f)$, and the elements of the vector b, as well as wires for computation space, and returns $b'_r = b + Aw$, $x = -\sum_{i=1}^{n} \frac{f[i]}{f[0]} A^i_r b'_r$, a boolean s_A which takes the value of 1 if $Ax = b$ and 0 otherwise, along with the input for reversibility.

Theorem 5. *The quantum circuit*

$$\mathcal{QRS} : |A\rangle |f\rangle |w\rangle |r\rangle |b\rangle |0\rangle \dots |0\rangle \rightarrow |A\rangle |f\rangle |w\rangle |r\rangle |b\rangle |0\rangle \dots |0\rangle |b'\rangle |x\rangle |s_A\rangle$$

implementing `RandomSol` *requires* $O(n^3 + 2n^2 + 3n + 1)$ *quantum gates to compute. In the black-box model, when provided with an oracle to compute matrix-vector and matrix-matrix products,* $\mathcal{QRS}$ *requires* $O(r)$ *evaluations of the black box, and* $O(nr)$ *operations in the base field* $\mathbb{F}$*, which is equivalent to the classical complexity of* `RandomSol`*.*

A proof of the above theorem is fairly straightforward when directly analyzing a quantum analogue of the classical algorithm provided above for `RandomSol`. It is clear that steps 4, 6, and 7 of `RandomSol` are the only steps computed by the $\mathcal{QRS}$ quantum circuit. Firstly, step 4 consists of the computation of $b'_r = (b'_1, \dots, b'_r) = b + Aw$, the first r entries of the vector b'. This is merely matrix-vector multiplication and vector-vector addition; we compute the r entries of b' with rn T gates for multiplication and r CNOT gates for addition, totaling $O(rn+r) \leq O(n^2+n)$ quantum gates. In the black box model, we have 1 oracle query for matrix-vector multiplication and $O(r)$ field operations for addition of two vectors. Secondly, step 6 consists of computing for $i = 1 \dots n$ the matrix-vector product $(A)\cdot(A^{i-1}b')$ with n^2 T gates, followed by the computation of $\frac{f_i}{f_0}$ via one T-gate, and computing the ith term of the sum with an additional T gate. This is $O(n(n^2+2)+1)$ quantum gates to compute x when we consider the additional NOT gate at the end of the computation. In the black box model, we have $O(n)$ black box matrix-vector product queries and $O(nr)$ field operations for the sum. Finally, the equality test conducted in step 7 consists of computing the matrix-vector product Ax with n^2 T gates, followed by n CNOT gates to compute, element by element, $(Ax)_i \oplus b_i$, and then one T_{n+1} gate to compute the value s_A. In the black box model, this is 1 call to the matrix-vector product oracle. Therefore, we have established the equivalence of the classical complexity of the subroutine `RandomSol` with the quantum oracle implementing the function $\mathcal{QRS}$ in the black-box model.

Similar arguments demonstrate the equivalence of `SparseLinearSystemSolver` as well as the entire quantum circuit $\mathcal{QBS}$. Due to the equivalence of the classical and quantum consistency checks in the black-box model, it is straightforward to adapt Theorem 2 to `QuantumBooleanSolve`, as follows.

Theorem 6. *Let* $\theta, 2 \leq \theta \leq 3$ *is such that any two* $n \times n$ *matrices can be multiplied in* $O(n^\theta)$ *operations in the underlying field. For any* $\epsilon > 0$*, and* $\alpha \geq 1$ *and sufficiently large* $m = \lceil \alpha n \rceil$*, testing the consistency of all Macaulay matrices in* `QuantumBooleanSolve(m,n,k)` *requires the:*

- *evaluation of* $O(2^{(\frac{1-\gamma}{2} + \theta F_\alpha(\gamma) + \epsilon)n})$ *quantum gates in the deterministic variant;*
- *evaluation, on average,* $O(2^{(\frac{1-\gamma}{2} + 2F_\alpha(\gamma) + \epsilon)n})$ *quantum gates in the probabilistic variant,*

where $\gamma = 1 - \frac{k}{n}$, $F_\alpha(\gamma) = -\gamma \log_2(D^D(1-D)^{(1-D)})$ *with* $D = M(\frac{\alpha}{\gamma})$ *and* $M(x) = -x + \frac{1}{2} + \frac{1}{2}\sqrt{2x^2 - 10x - 1 + 2(x+2)\sqrt{x(x+2)}}$ *and*

The above complexity is obtained through the full evaluation of the cost of the consistency check oracle, $\mathcal{QBS}$, which is equivalent to the cost of the classical consistency check in the black box model. The quantum circuit for $\mathcal{QBS}$ can then be run in superposition over all generated Macaulay matrices, $\sum_{a\in\mathbb{F}_2^k} |\mathbf{M_a}\rangle$. Amplitude amplification is then utilized, as in Grover's algorithm, to determine the $\mathbf{a} \in \mathbb{F}_2^k$ such that $\mathbf{M_a}$ is inconsistent.

If we are guaranteed only one input $\mathbf{a} \in \mathbb{F}_2^k$ is such that the generated Macaulay matrix $\mathbf{M_a}$ is inconsistent, the algorithm requires $O(2^{k/2}) = O(2^{(\frac{(1-\gamma)n}{2})})$ evaluations of the quantum circuit $\mathcal{QBS}$ implementing $F_{F,k}^{\text{cons}}$ for $F \in \mathbb{F}_2[x_1, \ldots, x_n]^m$ as well as the diffusion gate D for Grover's algorithm. When we have more than one $\mathbf{a} \in \mathbb{F}_2$ such that $\mathbf{M_a}$ is inconsistent, amplitude amplification must be run

$$O\left(\frac{\pi}{4}\sqrt{\frac{2^k}{|\mathbf{a} \in \mathbb{F}_2^k : \mathbf{M_a} \text{ inconsistent}|}}\right)$$

times to recover such $\mathbf{a} \in \mathbb{F}_2^k$.

As in the classical analysis of `ClassicalBooleanSolve`, in the case that the Macaulay matrices are found to be inconsistent, the full system F may be consistent. We therefore must determine the remainder of the solution, once we have found $\mathbf{a} \in \mathbb{F}_2^k$ such that $M_\mathbf{a}$ is inconsistent. This exhaustive search can be performed using Grover's algorithm with a quantum oracle for the specialized system $\tilde{F}_a$, where if $a = (y_1, \ldots, y_k)$ we have $\tilde{F} = (\tilde{f}_1, \ldots, \tilde{f}_m) = (f_1(x_1, \ldots, x_{n-k}, y_1, \ldots, y_k), \ldots, f_m(x_1, \ldots, x_{n-k}, y_1, \ldots, y_k))$. Similarly to the classical analysis, we find that an overwhelming amount of computational cost is the consistency check performed by $\mathcal{QBS}$; the cost of the second exhaustive search, performed over the remaining $n-k$ variables, is negligible. By definition of strong semi-regularity, the number of such searches is bounded by $O(2^{(1-2\gamma+2F_\alpha(\gamma)_\epsilon)n})$, and therefore the cost of the second exhaustive search is bounded by the cost of the consistency check.

To derive the asymptotic complexity, we now minimize (for example, numerically) the exponents stated in Theorem 6.

Lemma 2. *Let the notations be as in Theorem 6 and $\alpha = 1$. Then, the function $\frac{(1-\gamma)}{2} + \theta F_\alpha(\gamma)$ is bounded by:*

- $0.477 = 1 - 0.523$*, when* $\theta = 3$ *and* $\gamma = 0.1$*,*
- $0.47 = 1 - 0.53$*, when* $\theta = 2.376$ *and* $\gamma = 0.13$*,*
- $0.462 = 1 - 0.538$*, when* $\theta = 2$ *and* $\gamma = 0.17$

It can be remarked that the value of θ has a minimal impact on the bounds provided in the Lemma below; less than in the classical setting (see Section 2.1). Note that these results can be extended to any $\alpha \geq 1$.

To assure the reader that such computations can be performed on a quantum computer, we have provided a naive matrix-vector and matrix-matrix product circuit computed via inner product in the previous section. Finally, in summary, we have:

Theorem 7. `QuantumlBooleanSolve` *is correct and solves* $\mathtt{MQ}_2$*. If $m = n$, then – for any $\epsilon > 0$ – the deterministic variant of the algorithm requires to evaluate $O(2^{(0.47+\epsilon)n})$ quantum gates provided that the system is* 0.13*-strong semi-regular. The Las-Vegas probabilistic variant requires to evaluate an expected number of $O(2^{(0.462+\epsilon)}n)$ quantum gates if the system is* 0.17*-strong semi-regular.*

This theorem follows directly from the equivalence of the classical and quantum complexity of the consistency checks, as well as the above Lemma 2.
This complexity should be directly compared to the ideal of a pure quadratic speed-up over the classical complexity of `ClassicalBooleanSolve`, which is $O(2^{(0.396n)})$, as well as a quadratic speed-up on the classical approximation algorithm from [28] which is $O(2^{(0.438n)})$. Note that none of these complexities have been obtained so far and are thus an open challenge.

5 Acknowledgment

The first and last authors are partially supported by the french Programme d'Investissement d'Avenir under national project RISQ1 P141580[7]. Delaram Kahrobaei is partially supported by an ONR (Office of Naval Research) grant N00014-15-1-2164, as well as a PSC-CUNY grant from the CUNY Research Foundation. We also would like to thanks the referees of PKC'18 for their comments on the first version of this document.

References

1. Albrecht, M.R., Cid, C., Faugère, J., Fitzpatrick, R., Perret, L.: Algebraic algorithms for LWE problems. IACR Cryptology ePrint Archive 2014, 1018 (2014), http://eprint.iacr.org/2014/1018
2. Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I. Lecture Notes in Computer Science, vol. 6755, pp. 403–415. Springer (2011), https://doi.org/10.1007/978-3-642-22006-7_34
3. 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)
4. Bennett, C.H., Bernstein, E., Brassard, G., Vazirani, U.V.: Strengths and weaknesses of quantum computing. SIAM J. Comput. 26(5), 1510–1523 (1997), http://dx.doi.org/10.1137/S0097539796300933
5. Berbain, C., Gilbert, H., Patarin, J.: QUAD: A multivariate stream cipher with provable security. J. Symb. Comput. 44(12), 1703–1723 (2009), http://dx.doi.org/10.1016/j.jsc.2008.10.004
6. Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.): Post-quantum cryptography. Mathematics and Statistics Springer-11649; ZDB-2-SMA, Springer Berlin Heidelberg, Berlin, Heidelberg (2009), http://opac.inria.fr/record=b1128738
7. Bettale, L., Faugère, J.C., Perret, L.: Hybrid Approach for Solving Multivariate Systems over Finite Fields. Journal of Mathematical Cryptology 3(3), 177–197 (2010), http://www-salsa.lip6.fr/~jcf/Papers/JMC2.pdf
8. Bettale, L., Faugère, J.C., Perret, L.: Solving Polynomial Systems over Finite Fields: Improved Analysis of the Hybrid Approach. In: Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation. pp. 67–74. ISSAC '12, ACM, New York, NY, USA (2012), http://www-polsys.lip6.fr/~jcf/Papers/FBP12.pdf
9. Bettale, L., Faugère, J.C., Perret, L.: Cryptanalysis of multivariate and odd-characteristic HFE variants. pp. 441–458
10. Bouillaguet, C., Chen, H.C., Cheng, C.M., Chou, T., Niederhagen, R., Shamir, A., Yang, B.Y.: Fast exhaustive search for polynomial systems in f_2. pp. 203–218
11. Brassard, G., Høyer, P., Mosca, M., Tapp, A.: Quantum amplitude amplification and estimation. In: Quantum computation and information (Washington, DC, 2000), Contemp. Math., vol. 305, pp. 53–74. Amer. Math. Soc., Providence, RI (2002), http://dx.doi.org/10.1090/conm/305/05215
12. Buchberger, B.: Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. Journal of Symbolic Computation 41(3-4), 475–511 (2006)
13. Buchberger, B., Collins, G.E., Loos, R.G.K., Albrecht, R.: Computer algebra symbolic and algebraic computation. SIGSAM Bull. 16(4), 5–5 (1982)
14. Carlet, C., Hasan, M.A., Saraswat, V. (eds.): Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings, Lecture Notes in Computer Science, vol. 10076. Springer (2016), https://doi.org/10.1007/978-3-319-49445-6
15. Chen, L., Jordan, S., Liu, Y.K., Moody, D., Peralta, R., Perlner, R., Smith-Tone, D.: Report on post-quantum cryptography. Reasearch report NISTIR 8105, NIST (2003), http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf
16. Chen, M., Hülsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: From 5-pass MQ -based identification to MQ -based signatures. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10032, pp. 135–165 (2016), https://doi.org/10.1007/978-3-662-53890-6_5

[7] https://risq.fr/?page_id=31&lang=en

17. Chen, M.S., Hlsing, A., Rijneveld, J., Samardjiska, S., Schwabe, P.: Sofia: Mq-based signatures in the qrom. Cryptology ePrint Archive, Report 2017/680 (2017), http://eprint.iacr.org/2017/680
18. Faugère, J.C., Otmani, A., Perret, L., Tillich, J.P.: Algebraic Cryptanalysis of McEliece Variants with Compact Keys. In: Proceedings of Eurocrypt 2010. Lecture Notes in Computer Science, vol. 6110, pp. 279–298. Springer Verlag (2010), http://www-salsa.lip6.fr/~jcf/Papers/Eurocrypt2010.pdf
19. Faugère, J.C., Perret, L., De Portzamparc, F.: Algebraic Attack against Variants of McEliece with Goppa Polynomial of a Special Form. In: Advances in Cryptology Asiacrypt 2014. Kaohsiung, Tawan (Sep 2014), http://hal.inria.fr/hal-01064687
20. Faugère, J.C., Joux, A.: Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using gröbner bases. pp. 44–60
21. Gall, F.L.: Powers of tensors and fast matrix multiplication. In: Nabeshima, K., Nagasaka, K., Winkler, F., Szántó, Á. (eds.) International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014. pp. 296–303. ACM (2014), http://doi.acm.org/10.1145/2608628.2608664
22. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman (1979)
23. von zur Gathen, J., Gerhard, J.: Modern Computer Algebra (3. ed). Cambridge University Press (2013)
24. Giesbrecht, M., Lobo, A., Saunders, B.D.: Certifying inconsistency of sparse linear systems. In: Proceedings of the 1998 international symposium on Symbolic and algebraic computation. pp. 113–119. ACM (1998)
25. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219. ACM (1996)
26. Ivanyos, G., Santha, M.: Solving systems of diagonal polynomial equations over finite fields. Theor. Comput. Sci. 657, 73–85 (2017), https://doi.org/10.1016/j.tcs.2016.04.045
27. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced oil and vinegar signature schemes. pp. 206–222
28. Lokshtanov, D., Paturi, R., Tamaki, S., Williams, R., Yu, H.: Beating brute force for systems of polynomial equations over finite fields, to appear, 27th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)
29. NIST: Proposed submission requirements and evaluation criteria for the post-quantum cryptography standardization process (DRAFT)., http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-draft-aug-2016.pdf
30. Perret, L.: Bases de Gröbner en Cryptographie Post-Quantique. (Gröbner bases techniques in Quantum-Safe Cryptography) (2016), https://tel.archives-ouvertes.fr/tel-01417808
31. Westerbaan, B., Schwabe, P.: Solving binary mq with grover's algorithm. In: Carlet, C.; Hasan, A.; Saraswat, V.(ed.), Security, Privacy, and Advanced Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016. pp. 303–322. Berlin: Springer-Verlag (2016)
32. Wiedemann, D.: Solving sparse linear equations over finite fields. IEEE transactions on information theory 32(1), 54–62 (1986)