Quantum Algorithms for Boolean Equation Solving and Quantum Algebraic Attack on Cryptosystems

Yu-Ao Chen[1,2] and Xiao-Shan Gao[1,2]
[1]KLMM, Academy of Mathematics and Systems Science
Chinese Academy of Sciences, Beijing 100190, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
Email: xgao@mmrc.iss.ac.cn

December 19, 2017

Abstract

Decision of whether a Boolean equation system has a solution is an NPC problem and finding a solution is NP hard. In this paper, we present a quantum algorithm to decide whether a Boolean equation system $\mathcal{F}$ has a solution and compute one if $\mathcal{F}$ does have solutions with any given success probability. The complexity of the algorithm is polynomial in the size of $\mathcal{F}$ and the condition number of $\mathcal{F}$. As a consequence, we have achieved exponential speedup for solving sparse Boolean equation systems if their condition numbers are small. We apply the quantum algorithm to the cryptanalysis of the stream cipher Trivum, the block cipher AES, the hash function SHA-3/Keccak, and the multivariate public key cryptosystems, and show that they are secure under quantum algebraic attack only if the condition numbers of the corresponding equation systems are large.

Keywords. Quantum algorithm, Boolean equation solving, polynomial system solving, HHL algorithm, condition number, stream cipher Trivum, block cipher AES, hash function SHA-3/Keccak, MPKC, 3-SAT, graph isomorphism.

1 Introduction

Solving Boolean equations is a fundamental problem in theoretical computer science. Decision of whether a Boolean equation system has a solution is an NPC problem and finding a solution is NP hard. On the other hand, finding a polynomial-time quantum algorithm for an NPC problem is a basic issue in quantum computing. In this paper, a quantum algorithm for Boolean equation solving will be given, which can be as much as exponentially faster than traditional algorithms for the same task under certain conditions.

1.1 Main results

Let $\mathcal{F} = \{f_1, \dots, f_r\}$ be a set of Boolean polynomials in variables $\mathbb{X} = \{x_1, \dots, x_n\}$ and with total sparseness $T = \sum_{i=1}^{r} \#f_i$, where $\#f_i$ is the number of terms in f_i. Then, we have

Theorem 1.1. *For $\epsilon \in (0,1)$, there is a quantum algorithm which decides whether $\mathcal{F} = 0$ has a solution and computes one if $\mathcal{F} = 0$ does have solutions, with probability at least $1 - \epsilon$ and complexity $\widetilde{O}((n^{3.5} + T^{3.5})\kappa^2 \log 1/\epsilon)$, where κ is the condition number of the Boolean polynomial system $\mathcal{F}$ (refer to Theorem 5.9 for definition).*

As a consequence, we can solve Boolean equation systems using quantum computers with any given success probability and in polynomial-time if the condition number κ of $\mathcal{F}$ and the sparseness T of $\mathcal{F}$ are small, say when κ and T are poly(n). Since T is the size of the input to the algorithm, it should be small for practical problems. For instance, all the equation systems from cryptanalysis in Section 6 are very sparse. Therefore, the key factor is the condition number. As a consequence, we have achieved exponential speedup for sparse Boolean equation solving if its condition number is small.

Let $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ be a set of polynomials in $\mathbb{X}$ with total sparseness $T = \sum_{i=1}^{r} \#f_i$ and $\epsilon \in (0,1)$. A solution $\mathbf{a}$ for $\mathcal{F} = 0$ is called *Boolean*, if each coordinate of $\mathbf{a}$ is 0 or 1. Clearly, deciding whether $\mathcal{F}$ has a Boolean solution is NPC. We also give a quantum algorithm to compute Boolean solutions of $\mathbb{F}$.

Theorem 1.2. *There is a quantum algorithm which decides whether $\mathcal{F} = 0$ has a Boolean solution and computes one if $\mathcal{F} = 0$ does have Boolean solutions, with probability at least $1 - \epsilon$ and complexity $\widetilde{O}(n^{2.5}(n + T)\kappa^2 \log 1/\epsilon)$, where κ is the condition number of the polynomial system $\mathcal{F}$ (refer to Theorem 4.3 for definition).*

We apply Theorem 1.1 to cryptanalysis. As early as in 1946, Shannon [24] pointed out insightfully that "Construct our cipher in such a way that breaking it is equivalent to solving a certain system of simultaneous equations in a large number of unknowns." We know that the analysis of many cryptosystems, such as the stream cipher Trivum, the block cipher AES, the hash function SHA-3/Keccak, and the multivariate public key cryptosystems (MPKC), can be reduced to solving Boolean equations.

Cryptosystems	N_k	N_r	#Vars	#Eqs	T	Complexity
AES-128	4	10	4288	10616	252288	$2^{69.26}c\kappa^2$
AES-192	6	12	7488	18096	421248	$2^{71.83}c\kappa^2$
AES-256	8	14	11904	29520	696384	$2^{74.38}c\kappa^2$
Trivium		1152	3543	4407	24339	$2^{55.50}c\kappa^2$
Trivium		2304	6999	9015	49683	$2^{59.06}c\kappa^2$
	N_h	N_r	#Vars	#Eqs	T	Complexity
Keccak	384	24	76800	77160	611023	$2^{73.12}c\kappa^2$
Keccak	512	24	76800	77288	611540	$2^{73.12}c\kappa^2$

Table 1: Complexities of the quantum algebraic attack

In Table 1, we give the complexities of using Theorem 1.1 to perform quantum algebraic attack to these cryptosystems, where κ is the condition number of the corresponding Boolean equation systems, T is the total sparseness of the Boolean equations, and c is the complexity constant of the HHL algorithm (see Remark 2.4 for definition). For AES-m, $m = 32N_k$ is the key bit-length and N_r is the number of rounds. For Trivium, N_r is the number of rounds. For Keccak, N_h is the output size, N_r is the number of rounds, and the state bit-size b is 1600. From Table 1, we can see that these cryptosystems are secure under quantum algebraic attack only if the condition numbers of their corresponding equation systems are large. This leads to a new criterion for designing cryptosystems that can against the attack of quantum computers: *their corresponding equation systems must have large condition numbers.* Condition numbers for equation systems are generally difficult to estimate, and estimating the conditions for these cryptosystems is an interesting future work.

Many famous problems from computational theory can be reduced to finding a Boolean solution for certain polynomial systems. In this paper, we use Theorem 1.2 to three such problems. The 3-SAT problem is clearly equivalent to Boolean equation solving. For a 3-SAT of r clauses and n variables, the quantum complexity to decide its satisfiability is $\widetilde{O}((n^{2.5}(n+r)\kappa^2 \log 1/\epsilon)$. The *subset sum problem* is: given a set of n integers a_i, is there a non-empty subset whose sum equals to a given number b, which is to find a Boolean solution of the linear equation $\sum_i a_i x_i = b$. We show that there is a quantum algorithm to solve the subset sum problem with complexity $\widetilde{O}(n^{3.5}\kappa^2 \log 1/\epsilon)$. The *graph isomorphism problem* is to determine whether two finite graphs are isomorphic. We do not know whether this problem is NPC. The problem can be described as finding the Boolean solutions for a linear system and the quantum computational complexity is $\widetilde{O}(n^{6.5}\kappa^2 \log 1/\epsilon)$.

1.2 Technical contribution and relation with existing work

The main idea of the quantum algorithm proposed in this paper is that the Boolean solutions of a polynomial system $\mathcal{F}$ can be obtained exactly by solving the Macaulay linear system of $\mathcal{F}$ with the HHL quantum algorithm [16, 2]. For a linear system $Ax = |b\rangle$, the HHL algorithm can obtain an approximation to the solution state $|x\rangle$ exponentially faster than classic algorithms under certain conditions.

Our algorithm is based on three technical contributions: (1) The problem of solving Boolean equations is reduced to the computation of the Boolean solutions for a sparse polynomial system in $\mathbb{C}[\mathbb{X}]$ (see section 5). (2) It is shown that Boolean solutions for a sparse polynomial system in $\mathbb{C}[\mathbb{X}]$ can be computed with the HHL algorithm (see section 4). (3) The computation of Boolean solutions is possible, because we give a formula for the solution of the Macaulay linear system of a polynomial system (see section 3). We will introduce each of these contributions briefly.

Let $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ with $d_i = \deg(f_i)$, $T = \sum_{i=1}^r \#f_i$, and D a positive integer greater than $\max_i d_i$. Consider all the polynomials $m_j f_i$, where m_j are monomials with degree $\le D - d_i$. These equations $m_j f_i = 0$ can be written as a linear system $\mathscr{M}_{\mathcal{F},D}\mathfrak{m}_D = \mathbf{b}_{\mathcal{F},D}$, where $\mathfrak{m}_D$ is the set of all the monomials with degree $\le D$ and $\mathbf{b}_{\mathcal{F},D}$ is the set of the constant terms in $m_j f_i$. The linear system $\mathscr{M}_{\mathcal{F},D}\mathfrak{m}_D = b_{\mathcal{F},D}$ is called the *Macaulay linear system* of $\mathcal{F}$. The F4 algorithm [14] and the XL [11] algorithm to compute the Gröbner basis of $\mathcal{F}$ is to use Gaussian elimination to $\mathscr{M}_{\mathcal{F},D}$ for certain D.

Our contribution here are two folds. First, we show that $\mathscr{M}_{\mathcal{F},D}$ is a T-sparse matrix, which allows us to use the HHL algorithm. Second, we give a formula for the solution of $\mathscr{M}_{\mathcal{F},D}\mathfrak{m}_D = \mathbf{b}_{\mathcal{F},D}$ when using the HHL algorithm [16] to it, which is called the *pseudo solutions* of $\mathcal{F} = 0$.

Secondly, we show how to compute the Boolean solutions for a polynomial system $\mathcal{F}$, which is the solutions of $\mathcal{F}_1 = \mathcal{F} \cup \{x_1^2 - x_1, \ldots, x_n^2 - x_n\}$ over $\mathbb{C}$. Our contribution here is to show that the Boolean solutions of $\mathcal{F} = 0$ can be obtained from the pseudo-solutions of $\mathcal{F}_1 = 0$ with high probability by combining the property of quantum states and that of Boolean solutions.

Thirdly, let $\mathcal{F}$ be a Boolean polynomial system in variables $\mathbb{X}$. Since the HHL algorithm works over $\mathbb{C}$ and does not work for finite fields, we cannot use the HHL algorithm to the Macaulay linear system of $\mathcal{F}$. We prove that the solutions to $\mathcal{F}$ are the same as the Boolean solutions of a 6-sparse polynomial system $\mathcal{F}_2 \subset \mathbb{C}[\mathbb{X}, \mathbb{U}]$ for some extra indeterminates $\mathbb{U}$. Furthermore, the numbers of variables in $\mathbb{U}$ and the numbers of equations in $\mathcal{F}_2$ are linear in the size of $\mathcal{F}$. By computing the Boolean solutions of $\mathcal{F}_2$, we find the solutions of $\mathcal{F} = 0$.

Finally, we compare our algorithm with the HHL algorithm for solving the linear system $Ax = |b\rangle$, where $A \in \mathbb{C}^{N\times N}$, $x, b \in \mathbb{C}^N$. First, the speedup achieved in our algorithm is based on the exponential speed up of the HHL algorithm for solving sparse linear systems. On the other hand, the HHL algorithm has the following subtle properties.

1. The algorithm does not give a solution to $Ax = |b\rangle$, but a state $|x\rangle = (x_1, \ldots, x_N)$.

Measuring of $|x\rangle$ gives $|x_1| : |x_2| : \cdots : |x_N|$ and the complexity will increase to $O(N)$.

2. The algorithm gives an answer $|x\rangle$ even if $A|x\rangle = |b\rangle$ has no solutions.
3. The algorithm works over $\mathbb{C}$, but not over finite fields.
4. The algorithm gives an approximation to the state $|\widehat{x}\rangle$ with any error bound $\nu \in (0, 1)$.

Our algorithm does not have these limitations and gives an exact solution to the Boolean system. It is interesting to see that the second "drawbacks" of the HHL algorithm mentioned above is used to generate the quantum state $|b\rangle$ efficiently (see Lemma 2.5).

The HHL algorithm assumes that $|b\rangle$ is given. There exist no efficient algorithms to generate $|b\rangle$ from b [1] and efficient generation for $|b\rangle$ can be achieved only in some special cases [8]. Fortunately, in our case, $b = \mathbf{b}_{\mathcal{F},D}$ is very sparse: only the first r entries of b are nonzero, which leads to an efficient generation for $|b\rangle$ and the complexity is negligible comparing to that of the HHL algorithm.

The complexity of our algorithm contains the condition number of the Boolean polynomial system, which is inherited from the HHL algorithm. But, the condition number in our case is much more complicated. It is proved in [16] that the dependence on condition number cannot be substantially improved. Also note that, for a symmetric and positive-definite $A \in \mathbb{C}^{N\times N}$, the best classic numerical method for solving the linear equation $Ax = b$ has complexity $\widetilde{O}(N\sqrt{\kappa})$ which also depends on the condition number κ of A [23].

Finally, regarding to the HHL algorithm, it is pointed out in [9] that "Will the quantum solution of linear equations turn out to be a widely used tool, or are its limitations too great for the technique to be of practical significance? Unfortunately, no concrete task has yet been proposed for which the quantum algorithm provides a clear advantage." It seems that the result of this paper does provide a significant application of the HHL algorithm.

2 A modified HHL algorithm

In this section, we give a modified HHL algorithm which will be used in our algorithm for solving Boolean equations.

For a matrix M, the arithmetic square root of each nonzero eigenvalue of the matrix $M^\dagger M$ is called a *singular value* of M, and the quotient of the maximal and minimal singular values is called the *condition number* of M. A matrix M is called *s-sparse* if each row and column of M has at most s nonzero entries.

The following HHL quantum algorithm [16] is proposed to solve a linear equation system $A|x\rangle = |b\rangle$ over $\mathbb{C}$. With the best known algorithm to do the Hamiltonian simulation e^{-iAt} [4, 16, 1, 10], we have

Theorem 2.1 ([16, 4])**.** *Given an s-sparse matrix $A \in \mathbb{C}^{M\times N}$ with the condition number κ, singular values $\lambda_1, \ldots, \lambda_n$, and a unitary quantum state $|b\rangle \in \mathbb{C}^M$. Let $|v_j\rangle$ ($|u_j\rangle$) be the eigenvectors of $A^\dagger A$ ($AA^\dagger$) with respect to the nonzero eigenvalues λ_j^2 of $A^\dagger A$. Then the singular value decomposition of A is $A = \sum_{j=1}^{n} \lambda_j |u_j\rangle\langle v_j|$. For the linear equation system $A|x\rangle = |b\rangle$, HHL algorithm will give an approximation to the solution state $|x\rangle$ for the following vector*

$$\widetilde{x} = \sum_{j=1}^{n} \lambda_j^{-1} |v_j\rangle\langle v_j|b\rangle, \tag{1}$$

in time $\tilde{O}(\log(N + M)s\kappa^2/\epsilon)$ with error bounded by a given $\epsilon \in (0, 1)$.

From [16], we can obtain the following result easily.

Corollary 2.2. *The solution in (1) given by the HHL algorithm has the minimal* $\|\widetilde{x}\| = \sqrt{\langle \widetilde{x}, \widetilde{x} \rangle}$ *among all solutions* $\widetilde{x}$ *of the linear system.*

Ambainis gave a new version of the HHL algorithm with different complexities [2].

Theorem 2.3. *The HHL algorithm can be modified to have complexity* $\tilde{O}(\log(N+M)s\kappa/\epsilon^3)$.

Remark 2.4. *In the cryptanalysis to be given later in this paper, we make the following approximation to the complexities of the HHL algorithm* $\tilde{O}(\log(N+M)s\kappa^2/\epsilon) \simeq c\log(N+M)s\kappa^2/\epsilon$, *where* c *is called the* complexity constant *of the HHL algorithm.*

In Theorem 2.1, $|b\rangle \in \mathbb{C}^M$ is given as a quantum state and there exist no efficient algorithms to generate $|b\rangle$ from b [1]. In the rest of this section, we will modify the HHL algorithm such that the input is A and b instead of A and $|b\rangle$. We first prove a lemma.

Lemma 2.5. *Let* $Bx = c$ *be obtained by adding more "equations"* $0x = 1$ *to* $A|x\rangle = |b\rangle$. *Then using HHL to* $B|x\rangle = |c\rangle$, *we obtain the same solution state as that of* $A|x\rangle = |b\rangle$.

Proof. Let $B = \begin{pmatrix} A \\ 0 \end{pmatrix}$ and $c = \begin{pmatrix} b \\ 1 \end{pmatrix}$, where we use 0 (1) to represent certain maxtix of zeros (ones) with the proper dimension. We have $B^\dagger B = A^\dagger A$. Then, adding some 0 rows to A will not change the nonzero eigenvalues of $A^\dagger A$ and the eigenvectors of $B^\dagger B$ are the same as that of $A^\dagger A$. Now, the lemma follows from Theorem 2.1. □

We have the following modified version of HHL algorithm.

Theorem 2.6. *Assume that only the first* ρ *entries of* $b \in \mathbb{C}^M$ *are nonzero and* $A \in \mathbb{C}^{M\times N}$ *is* s*-sparse. Then, the HHL algorithm can give an* ϵ*-approximation to the solution state (1) of the linear system* $Ax = b$ *in time* $\tilde{O}(\log(N+M)s\kappa^2/\epsilon + T_\rho)$, *where* T_ρ *is the number of nonzero elements in the first* ρ *rows of* A.

Proof. We assume $b = (b_1, \ldots, b_\rho, 0, \ldots, 0)$. By dividing b_i to the i-row of $Ax = b$, we may assume $b_i = 1$ for $i = 1, \ldots, \rho$. This step costs $O(T_\rho)$.

Let $\sigma = 2^{\lceil \log_2 \rho \rceil}$. Then by adding $\sigma - \rho$ one to b, we obtain $c = (1, \ldots, 1, 0, \ldots, 0)$ whose first σ entries are one. Correspondingly, by adding $\sigma - \rho$ zero rows to A after the ρ-th row, we obtain a matrix $B \in \mathbb{C}^{(M+\sigma-\rho)\times N}$. Assuming that the matrix A is represented sparsely, adding some zero rows costs $O(1)$. Then equation system $Ax = b$ becomes

$$Bx = c \tag{2}$$

Without loss of generality, we may add more zero rows to B and c such that $B \in \mathbb{C}^{2^\eta \times N}$ and $c \in \mathbb{C}^{2^\eta}$, where $\eta = \lceil \log_2(M + \sigma - \rho) \rceil$. With these assumptions, we can easily generate the state $|c\rangle$:

$$|c\rangle = \otimes_{i=1}^{\eta - \lceil \log_2 \rho \rceil} |0\rangle \otimes_{i=1}^{\lceil \log_2 \rho \rceil} (H|0\rangle),$$

where H is the Hadamard operator. The complexity of generating $|c\rangle$ is $O(\eta) = O(\log(M))$, since $\sigma = 2^{\lceil \log_2 \rho \rceil} \le 2M$. The equation system (2) becomes

$$\frac{B}{\sqrt{\sigma}}|x\rangle = |c\rangle \tag{3}$$

In (3), we need only divide $\sqrt{\sigma}$ to the first ρ rows of B and the complexity is $O(T_\rho)$. By Lemma 2.5, equation system (3) has the same solution state as that of $A|x\rangle = |b\rangle$, when using the HHL algorithm to them. Since $\eta = \lceil \log_2(M+\sigma-\rho)\rceil$ and $\sigma = 2^{\lceil \log_2 \rho \rceil} \le 2M$, the total complexity is $\tilde{O}(\log(N+M)s\kappa^2/\epsilon + T_\rho + \log M) = \tilde{O}(\log(N+M)s\kappa^2/\epsilon + T_\rho)$. $\square$

Remark 2.7. *If ρ is small, say $\rho = O(\log(N+M))$, then $T_\rho = O(\log(N+M)s)$, which is negligible comparing to the complexity of the HHL algorithm. In this case, the HHL algorithm could be used to the equation system $Ax = b$. Fortunately, the Macaulay linear system to be solved in this paper has this property.*

3 Quantum pseudo-solving of polynomial systems over $\mathbb{C}$

By pseudo-solving of a polynomial system $\mathcal{F}$, we mean to compute the values of the monomials at the solutions of $\mathcal{F} = 0$, which satisfy a set of linear equations.

3.1 Sparseness of the modified Macaulay matrices

Let $\mathbb{C}$ be the field of complex numbers and $\mathbb{C}[\mathbb{X}]$ the polynomial ring in the indeterminates $\mathbb{X} = \{x_1, \ldots, x_n\}$. For a polynomial $f \in \mathbb{C}[\mathbb{X}]$, denote $\deg(f)$, $\#f$, and $\mathfrak{m}(f)$ to be the degree of f, the sparseness (the number of terms) of f, and the set of monomials of f. For $S \subset \mathbb{C}[\mathbb{X}]$, we use $\mathbb{V}_{\mathbb{C}}(S) \subset \mathbb{C}^n$ to denote the common zeros of the polynomials in S.

Let $\mathfrak{m}$ denote the set of all the monomials in variables $\mathbb{X}$. In this paper, we will always use the degree reverse lexicographic (DRL) monomial ordering for $x_1 < \cdots < x_n$. We assume that the monomials in $\mathfrak{m} = \{m_0, m_1, \ldots\}$ are arranged in the ascending order w.r.t DRL, that is, $m_0 < m_1 < \cdots$. It is easy to see that $m_0 = 1, m_1 = x_1, \ldots, m_n = x_n, m_{n+1} = x_1^2, \ldots$. Let $\mathfrak{m}_{\le d} = \{m_0, m_1, \ldots, m_{Q_d - 1}\}$ be the set of all monomials of degree $\le d$, where

$$Q_d \quad = \quad \binom{d+n}{n} = O((d+n)^{\min\{n,d\}}). \tag{4}$$

For any $j \in \mathbb{N}$, $Q_d \le j < Q_{d+1}$ if and only if $\deg(m_j) = d+1$.

Let $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ with $d_i = \deg(f_i)$ and $t_i = \#f_i$ for $i = 1, \ldots, r$. Let $D \in \mathbb{N}$ such that $D \ge \max_{i=1}^r d_i$. We will construct a modified Macaulay matrix for $\mathcal{F}$. For each $m_j \in \mathfrak{m}_{\le D - d_i}$, $m_j f_i$ could be considered as a linear function in the monomials in $\mathfrak{m}_{\le D}$. We rewrite these linear functions in matrix form:

$$\begin{array}{cc} & m_1 < m_2 < \cdots < m_{Q_D-1} \\ \begin{array}{c} m_0 f_1 \\ \vdots \\ m_0 f_r \\ m_1 f_1 \\ \vdots \\ m_{Q_{D-d_r}-1} f_r \end{array} & \left(\begin{array}{c} \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \end{array}\right) \end{array} \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_{Q_D-1} \end{pmatrix} = \begin{array}{c} m_0 \\ \begin{pmatrix} -f_1(\mathbf{0}) \\ \vdots \\ -f_r(\mathbf{0}) \\ 0 \\ \vdots \\ 0 \end{pmatrix} \end{array}, \tag{5}$$

denoted as

$$\mathscr{M}_{\mathcal{F},D} \mathbf{m}_D = \mathbf{b}_{\mathcal{F},D}, \tag{6}$$

where $\mathbf{m}_D = (m_1, \ldots, m_{Q_D-1})^T$ and the i-th column of $\mathscr{M}_{\mathcal{F},D}$ consists of the coefficients of m_i in $m_0 f_1, \ldots, m_0 f_r, m_1 f_0, \ldots, m_{Q_{D-d_r}-1} f_r$. $\mathscr{M}_{\mathcal{F},D}$ is called the *(modified) Macaulay matrix* of the polynomial system $\mathcal{F}$ and (6) is called the *Macaulay linear system* of $\mathcal{F}$. $\mathscr{M}_{\mathcal{F},D}$ is a matrix over $\mathbb{C}$ of dimension $(\sum_{i=1}^{r} Q_{D-d_i}) \times (Q_D - 1)$.

Denote $\mathscr{N}_{\mathcal{F},D}$ to be the set of monomials m_k in $\mathbf{m}_D$ such that the k-th column of $\mathscr{M}_{\mathcal{F},D}$ is $\mathbf{0}$. In the other words, $\mathscr{N}_{\mathcal{F},D}$ is the set of monomials not occurring in $m_j f_i$ in (5). We introduce the notation $\widetilde{\mathbf{m}}_{\mathcal{F},D} \in \mathfrak{m}^{Q_D-1}$: for $i = 1, \ldots, Q_D - 1$,

$$\widetilde{\mathbf{m}}_{\mathcal{F},D}(i) = \begin{cases} m_i, & \text{if } m_i \notin \mathscr{N}_{\mathcal{F},D}; \\ 0, & \text{if } m_i \in \mathscr{N}_{\mathcal{F},D}. \end{cases} \tag{7}$$

Using the notations just introduced, we have

Lemma 3.1. *$\mathscr{M}_{\mathcal{F},D}$ is T-sparse and $\max_i t_i$ row sparse, where $T = \sum_{i=1}^{r} t_i$ is called the* total sparseness *of $\mathcal{F}$.*

Proof. Since $m_j f_i$ has t_i terms, each row of $\mathscr{M}_{\mathcal{F},D}$ has at most $\max_i t_i$ nonzero entries. Consider the k-th column corresponding to the monomial m_k. For a fixed $f_i = \sum_j c_{i,j} m_{n_{i,j}}$, we have $m_u m_{n_{i,j}} > m_v m_{n_{i,j}}$ for $m_u > m_v$. Since each coefficient of $m_u f_i$ is strictly shifted to the righthand side with respect to the corresponding coefficients of $m_v f_i$ for $m_u > m_v$, at most t_i monomials of $m_j f_i$ for $m_j \in \mathfrak{m}_{D-d_i}$ are m_k. Thus there exist at most $\sum_i t_i$ nonzero entries per column. The lemma is proved. □

Corollary 3.2. *Suppose that all equations in $\mathcal{F}$ are nonlinear. Let $r = n+1$ and $D = \sum_{i=1}^{n+1} d_i - n$ the Macaulay degree [20], the classic Macaulay matrix is of dimension $(\sum_{i=1}^{n+1} Q_{D-d_i}) \times (Q_D - 1) = \widetilde{O}(nD^n \times D^n)$ and $(\sum_{i=1}^{n+1} t_i)$-sparse.*

Proof. Since all equations in $\mathcal{F}$ are nonlinear, we have $D < D - d_i$. By (4), $Q_{D-d_i} = O(D^n)$ and the corollary is proved. □

$\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ is called a *multivariate quadratic polynomial system (MQ)* if $\deg(f_i) = 2$.

Corollary 3.3. *Let $\mathcal{F} = \{f_1, \ldots, f_r\}$ be a set of MQ. Then $\#f_i \le Q_2 = \frac{(n+1)(n+2)}{2}$, $\mathscr{M}_{\mathcal{F},D}$ is of dimension $\widetilde{O}(rD^n)$ and $O(n^2 r)$-sparse.*

Example 3.4. *Let $f_1 = x_1^2 - x_2$, $f_2 = x_1 - 2$, $D = 2$. Then the Macaulay linear system is*

$$\begin{array}{c} f_1 \\ f_2 \\ x_1 f_2 \\ x_2 f_2 \end{array} \begin{pmatrix} 0 & -1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ -2 & 0 & 1 & 0 & 0 \\ 0 & -2 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \end{pmatrix}.$$

3.2 Solution of the Macaulay linear system

In this section, we give an explicit formula for the solution of the Macaulay linear system. We first introduce the concept of solving degree [19, 6].

Definition 3.5. *Let $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ and $(\mathcal{F})$ the ideal generated by $\mathcal{F}$. D is called the* solving degree *of $\mathcal{F}$, if we can use Buchberger's algorithm to compute the Gröbner basis of $(\mathcal{F})$ such that all the polynomials in the procedure have degrees less than or equal to D. Denote the solving degree of $\mathcal{F}$ by* $\mathrm{Sdeg}(\mathcal{F})$.

In terms of the F4 algorithm [14] or the XL algorithm [11], D is the solving degree of $\mathcal{F}$, if the Gröbner basis of $(\mathcal{F})$ can be obtained from $\mathscr{M}_{\mathcal{F},D}\mathbf{m}_D = \mathbf{b}_{\mathcal{F},D}$ by using Gaussian elimination over $\mathbb{C}$.

For a polynomial $f \in \mathbb{C}[\mathbb{X}]$, denote f^h to be the homogenization of f in $\mathbb{C}[x_0, \mathbb{X}]$. $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ is said to satisfy *Lazard's condition* if $\mathcal{F}^h = \{f_1^h, \ldots, f_r^h\} \subset \mathbb{C}[x_0, \mathbb{X}]$ has a finite number of solutions in the projective space $P^n_{\mathbb{C}}$. Lazard [19] and more recently Caminata-Gorla [6] proved the following upper bound for the solving degree.

Theorem 3.6 ([19])**.** *Let I be an ideal in $\mathbb{C}[\mathbb{X}]$ generated by $\mathcal{F} = \{f_1, \ldots, f_r\}$ of degrees $d_1, \ldots, d_r$, such that $d_1 \geq d_2 \geq \cdots \geq d_r$. Choose any graded monomial order. If $\mathcal{F}$ satisfies Lazard's condition, then* $\mathrm{Sdeg}(\mathcal{F}) \leq d_1 + \cdots + d_{n+1} - n + 1$ *with $d_{n+1} = 1$ if $r = n$.*

Corollary 3.7 ([6])**.** *Denote $d = \max_i d_i$,* $\mathrm{Sdeg}(\mathcal{F}) \leq (n+1)(d-1)+2$ *under Lazard's condition.*

The following lemma gives the solutions to the Macaulay linear system (5).

Lemma 3.8. *Let $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ such that $I = (\mathcal{F})$ is a radical zero-dimensional ideal and $\mathbb{V}_{\mathbb{C}}(I) = \{\mathbf{a}_1, \ldots, \mathbf{a}_w\}$. Let D be a solving degree of $\mathcal{F}$. Then any solution $\mathbf{m}_D$ of the linear system $\mathscr{M}_{\mathcal{F},D}\mathbf{m}_D = \mathbf{b}_{\mathcal{F},D}$ is of the form*

$$\widehat{\mathbf{m}}_D = \sum_{i=1}^{w} \eta_i \mathbf{m}(\mathbf{a}_i) + \sum_{m_k \in \mathscr{N}_{\mathcal{F},D}} \mu_k \mathbf{e}_k,$$

where η_i are complex numbers such that $\sum_{i=1}^{w} \eta_i = 1$, μ_k are arbitrary complex numbers, and $\mathbf{e}_k$ is the k-th unit vector in $\mathbb{C}^{Q_D - 1}$.

Proof. For each $m_k \in \mathscr{N}_{\mathcal{F},D}$, the k-th row in $\mathscr{M}_{\mathcal{F},D}$ is a zero column, so m_k can take arbitrary value in the solution of the Macaulay linear system and hence $\mu_k \mathbf{e}_k$ is part of the solution. Delete the k-th column in $\mathscr{M}_{\mathcal{F},D}$ and the k-th row in $\mathbf{m}_D$ and $\mathbf{b}_{\mathcal{F},D}$ to obtain a new system $\widetilde{\mathscr{M}}_{\mathcal{F},D}\widetilde{\mathbf{m}}_{\mathcal{F},D} = \widetilde{\mathbf{b}}_{\mathcal{F},D}$. Since D is a solving degree, we can obtain a Gröbner basis of the ideal I by doing Gaussian elimination on the linear system. Denote $\overline{\mathscr{M}}_{\mathcal{F},D}\widetilde{\mathbf{m}}_{\mathcal{F},D} = \overline{\mathbf{b}}_{\mathcal{F},D}$ to be such a linear system containing a Gröbner basis G of I. Denote $LT(I)$ to be set of the leading monomials of the polynomials in I. Then, the largest monomial in each row $\overline{\mathscr{M}}_{\mathcal{F},D}$ is in $LT(I)$ and each monomial in $LT(G)$ occurs as one of the leading monomials for some row. Thus, the dimension of the solution space of $\widetilde{\mathscr{M}}_{\mathcal{F},D}\widetilde{\mathbf{m}}_{\mathcal{F},D} = \overline{\mathbf{b}}_{\mathcal{F},D}$ is at most $\#(\mathfrak{m} \setminus LT(I)) - 1 = \#V(I) - 1 = w - 1$, where the first equality is true because I is radical. Since each $\widetilde{\mathbf{m}}_{\mathcal{F},D}(\mathbf{a}_i)$ is a solution of $\widetilde{\mathscr{M}}_{\mathcal{F},D}\widetilde{\mathbf{m}}_{\mathcal{F},D} = \overline{\mathbf{b}}_{\mathcal{F},D}$, $\{\sum \eta_i \widetilde{\mathbf{m}}_{\mathcal{F},D}(\mathbf{a}_i) | \sum \eta_i = 1\}$ is a subspace of the solution space of $\widetilde{\mathscr{M}}_{\mathcal{F},D}\widetilde{\mathbf{m}}_{\mathcal{F},D} = \overline{\mathbf{b}}_{\mathcal{F},D}$, that is, the solution space is of dimension at least $w - 1$. Then, $\{\sum \eta_i \widetilde{\mathbf{m}}_{\mathcal{F},D}(\mathbf{a}_i) | \sum \eta_i = 1\}$ is exactly the solution space of $\widetilde{\mathscr{M}}_{\mathcal{F},D}\widetilde{\mathbf{m}}_{\mathcal{F},D} = \overline{\mathbf{b}}_{\mathcal{F},D}$. The lemma is proved. □

Corollary 3.9. *In Lemma 3.8, if $\mathcal{F}$ has a unique solution $\mathbf{a}$, then we have*

$$\widehat{\mathbf{m}}_D = \mathbf{m}_D(\mathbf{a}) + \sum_{m_k \in \mathscr{N}_{\mathcal{F},D}} \mu_k \mathbf{e}_k.$$

Example 3.10. *The equation system in Example 3.4 has a unique solution $x_1 = 2, x_2 = 4$, and $\mathscr{N}_{\mathcal{F},D} = \{x_2^2\}$. By Corollary 3.9, the solution of the Macaulay linear system is* $(2, 4, 4, 8, \mu)$*, where μ is an arbitrary complex number.*

3.3 A quantum algorithm for pseudo-solving of polynomial systems

In this section, we give an explicit formula for the solution of the Macaulay linear system when using the HHL quantum algorithm to it. Using the modified HHL algorithm (Theorem 2.6) to the Macaulay linear system $\mathscr{M}_{\mathcal{F},D}\mathbf{m}_D = \mathbf{b}_{\mathcal{F},D}$, we have

Theorem 3.11. *Let $\mathcal{F} = \{f_1, \dots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ such that $I = (\mathcal{F})$ is a radical zero-dimensional ideal, $\mathbb{V}(I) = \{\mathbf{a}_1, \dots, \mathbf{a}_w\}$, and $\epsilon \in (0, 1)$. Let D be a solving degree of $\mathcal{F}$. Using the modified HHL algorithm to the Macaulay linear system $\mathscr{M}_{\mathcal{F},D}\mathbf{m}_D = \mathbf{b}_{\mathcal{F},D}$, the answer is an approximation to the following solution state*

$$|\widehat{\mathbf{m}}_D\rangle = \sum_{i=1}^{w} \eta_i |\widetilde{\mathbf{m}}(\mathbf{a}_i)\rangle,$$

with a given error bounded by ϵ, where η_i are complex numbers such that $\sum_{i=1}^{w} \eta_i = 1$ and $\|\sum_{i=1}^{w} \eta_i \widetilde{\mathbf{m}}(\mathbf{a}_i)\|$ is minimal. The complexity is $\widetilde{O}(\log(D)nT\kappa^2/\epsilon)$, where $T = \sum_{i=1}^{r} \#f_i$ and κ is the condition number of $\mathscr{M}_{\mathcal{F},D}$.

Proof. By Lemma 3.8,

$$|\widehat{\mathbf{m}}_D\rangle = \sum_{\sum \eta_i = 1} \eta_i |\mathbf{m}_D(\mathbf{a}_i)\rangle + \sum_{m_k \in \mathscr{N}_{\mathcal{F},D}} \mu_k |\mathbf{e}_k\rangle = \sum_{\sum \eta_i = 1} \eta_i |\widetilde{\mathbf{m}}_D(\mathbf{a}_i)\rangle + \sum_{m_k \in \mathscr{N}_{\mathcal{F},D}} \widetilde{\mu}_k |\mathbf{e}_k\rangle.$$

By Corollary 2.2, $\|\widehat{\mathbf{m}}_D\|$ is minimal. Since $\langle \mathbf{e}_k | \widetilde{\mathbf{m}}_D(\mathbf{a}_i)\rangle = 0$, in order for $\|\widehat{\mathbf{m}}_D\|$ to be minimized, each $\widetilde{\mu}_k = 0$. We may change the order of f_i such that, the first ρ polynomials f_i have nonzero constant terms. This step costs $O(r)$. Let $T_\rho = \sum_{1=1}^{\rho} \#f_i$. By Theorem 2.6 and (4), the complexity is $\widetilde{O}(\log(\sum_{i=1}^{r} Q_{D-d_i} + (Q_D - 1))(\sum_{i=1}^{r} t_i)\kappa^2/\epsilon + T_\rho) = \widetilde{O}(\log(rQ_{D-1} + Q_D)T\kappa^2/\epsilon + T_\rho) = \widetilde{O}(\log((n+D)^{\min\{n,D\}})T\kappa^2/\epsilon + T_\rho) = \widetilde{O}(\log(n+D)\min\{n, D\}T\kappa^2/\epsilon + T_\rho)$. Since $n \le D$ for nonlinear systems and $T_\rho \le T$, we prove the theorem. □

In Theorem 3.11, the answer is a linear combination of the monomials of the solutions of $\mathcal{F} = 0$, which is called the *pseudo-solution* for $\mathcal{F} = 0$.

If $\mathcal{F}$ satisfies Lazard's condition, by Corollary 3.7, we have the following three corollaries.

Corollary 3.12. *Set $D = (n+1)(d-1)+2$ in Theorem 3.11, the complexity is $\widetilde{O}(\log(d)nT\kappa^2/\epsilon)$, where $d = \max_i \deg(f_i)$.*

Corollary 3.13. *For MQ, we have $d \le 2$, $T = O(rn^2)$, $D \le n + 3$ and the complexity is $\widetilde{O}(n^3 r\kappa^2/\epsilon)$.*

Corollary 3.14. *If $\mathcal{F} = 0$ has a unique solution $\mathbf{a}$, then the solution state is $|\mathbf{m}\rangle = |\widetilde{\mathbf{m}}(\mathbf{a})\rangle$. The complexity is $\widetilde{O}(\log(d)nT\kappa^2/\epsilon)$.*

By Remark 2.4, the exact complexity for Theorem 3.11 is

Corollary 3.15. *The exact complexity to compute $|\widehat{\mathbf{m}}_D\rangle$ is $c\log(N + M)T\kappa^2/\epsilon$, where $N = \sum_{i=1}^{r} Q_{D-d_i}$, $M = Q_D - 1$, and c is the complexity constant of the HHL algorithm.*

Example 3.16. *If using the modified HHL algorithm to solve the linear system in Example 3.4, by Theorem 3.11, the solution is $(2, 4, 4, 8, 0)$. In order to find the unique solution $x_1 = 2, x_2 = 4$, we need to know how to project $(x_1, x_2, x_1^2, x_1x_2, x_2^2)$ to (x_1, x_2) efficiently.*

Motivated by the above example, we propose the following problem.

Problem 3.17. *Let $|u\rangle$ be an N-dimensional quantum state and $n \ll N$. How can we measure the first n coordinates of $|u\rangle$ efficiently.*

4 Find Boolean solutions for polynomial systems in $\mathbb{C}[\mathbb{X}]$

A solution $\mathbf{a}$ for $\mathcal{F} \subset \mathbb{C}[\mathbb{X}]$ is called *Boolean*, if each coordinate of $\mathbf{a}$ is 0 or 1. In this subsection, we will give a quantum algorithm to compute the Boolean solutions of $\mathcal{F} = 0$.

4.1 A quantum algorithm to find Boolean solutions

For $\mathcal{F} \subset \mathbb{C}[\mathbb{X}]$, the Boolean solutions of $\mathcal{F}$ are $\mathbb{V}_{\mathbb{C}}(\mathcal{F}, \mathbb{H}_{\mathbb{X}})$, where $\mathbb{H}_{\mathbb{X}} = \{x_1^2 - x_1, \ldots, x_n^2 - x_n\}$. We first prove a lemma.

Lemma 4.1. *For $\mathcal{F} \subset \mathbb{C}[\mathbb{X}]$, $I = (\mathcal{F}, \mathbb{H}_{\mathbb{X}})$ is radical and satisfies Lazard's condition.*

Proof. Since $(\mathcal{F}) + (x_1 - a_1, \ldots, x_n - a_n)$ is a maximal ideal in the ring $\mathbb{C}[\mathbb{X}]$,

$$(\mathcal{F}, \mathbb{H}_{\mathbb{X}}) = \bigcap_{a_1,\ldots,a_n \in \{0,1\}} ((\mathcal{F}) + (x_1 - a_1, \ldots, x_n - a_n)),$$

is an intersection of maximal ideals and I is a radical ideal. Considering the zero of I^h at infinity, $x_0 = 0$ and each $x_i^2 - x_i x_0 = 0$ implies $x_i = 0$, so I satisfies Lazard's condition. □

By Lemma 4.1, we can use Theorem 3.11 to compute $|\widehat{\mathbf{m}}_D\rangle$ where the solving degree D is given in Theorem 3.6. Denote $\mathbf{0}$ as $(0, \ldots, 0)$ and $\mathbf{1}$ as $(1, \ldots, 1)$. Our quantum algorithm to compute Boolean solutions is given below.

Algorithm 4.2.

Input: $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ with $T = \sum_{i=1}^r \# f_i$ and $d_i = \deg(f_i)$. Also $\epsilon \in (0, 1)$.
Output: A Boolean solution $\mathbf{a} \in \mathbb{V}_{\mathbb{C}}(\mathcal{F}, \mathbb{H}_{\mathbb{X}})$ or $\emptyset$ meaning that $\mathbb{V}_{\mathbb{C}}(\mathcal{F}, \mathbb{H}_{\mathbb{X}}) = \emptyset$, with success probability at least $1 - \epsilon$.

Step 1: If $\mathcal{F}(\mathbf{0}) = \mathbf{0}$, return $\mathbf{0}$. If $\mathcal{F}(\mathbf{1}) = \mathbf{0}$, return $\mathbf{1}$. Set $k = 1$.

Step 2: Let $\mathcal{F}_1$ be obtained from $\mathcal{F}$ by replacing x_i^m in $\mathcal{F}$ with x_i for all i and $m \in \mathbb{N}$. Let $\mathbb{Y} = \mathbb{X}$.

Step 3: Let $\mathcal{F}_2 = \mathcal{F}_1 \cup \mathbb{H}_{\mathbb{Y}}$, and $D = \mathrm{Sdeg}(\mathcal{F}_2)$ as in Theorem 3.6.

Step 4: Use the modified HHL algorithm (Theorem 2.6) to the Macaulay linear system $\mathscr{M}_{\mathcal{F}_2,D}\,\mathbf{m}_D = \mathbf{b}_{\mathcal{F}_2,D}$ to obtain a state $|\widehat{\mathbf{m}}\rangle$ with the error bound $\sqrt{\epsilon_1/n}$, where ϵ_1 can be chosen arbitrarily in $(0, 1)$ such as $\epsilon_1 = 1/2$.

Step 5: Measuring $|\widehat{\mathbf{m}}\rangle$, we obtain a state $|\mathbf{e}_k\rangle$ or equivalently, the number k.

Step 6: Let $m_k = \prod_{i=1}^{u_k} x_{n_i}$. Set $x_{n_i} = 1$ in $\mathcal{F}_1 \subset \mathbb{C}[\mathbb{Y}]$ for $i = 1, \ldots, u_k$.

Step 7: Remove 0 from $\mathcal{F}_1$. Set $\mathbb{Y} = \mathbb{Y} \setminus \{x_{n_i} \,|\, i = 1, \ldots, u_k\}$.

Step 8: If $1 \in \mathcal{F}_1$ or $\mathbb{Y} = \emptyset$ then goto **Step 11**

Step 9: If $\mathcal{F}_1 \neq \emptyset$ and $\mathcal{F}_1(\mathbf{0}) \neq \mathbf{0}$, then goto **Step 3**.

Step 10: Return $(a_1, \ldots, a_n)$ where $a_i = 0$ if $x_i \in \mathbb{Y}$ else $a_i = 1$.

Step 11: If $\lceil \log_{\epsilon_1} \epsilon \rceil < k$ then return $\emptyset$, else $k = k + 1$ and goto **Step 2**.

We first show what exactly the algorithm does in the following theorem and then prove the claim in the rest of this section.

Theorem 4.3. *Algorithm 4.2 has the following properties.*

- *If the algorithm returns a solution, then it is a Boolean solution of $\mathcal{F} = 0$. Equivalently, if $\mathcal{F}$ has no Boolean solutions, the algorithm returns $\emptyset$.*
- *If $\mathcal{F}$ has Boolean solutions, the algorithm computes one with probability at least $1 - \epsilon$.*
- *The complexity of the algorithm is $\widetilde{O}(n^{2.5}(n+T)\kappa^2 \log 1/\epsilon)$ if using the HHL algorithm [16] and $\widetilde{O}(n^{3.5}(n+T)\kappa \log 1/\epsilon)$ if using Ambainis' algorithm [2], where κ is the maximal condition number for all matrixes $\mathscr{M}_{\mathcal{F}_2,D}$ in Step 4 of the algorithm, called the* condition number *for the polynomial system $\mathcal{F}$.*

In the rest of this section, we will prove the correctness of Theorem 4.3.

First, we briefly explain each step. In Step 1, we first check two easy solutions in time $O(T)$. In Step 2, we replace x_i^m by x_i in time $\widetilde{O}(nT)$. As a consequence, $\deg(\mathcal{F}) \leq n$. In Step 3, the solving degree from Theorem 3.6 can be used due to Lemma 4.1. We have $D = O(nd) < O(n^2)$. In Step 4, we use the modified HHL to solve the Macaulay linear system.

In Step 5, we measure the quantum state $|\widehat{\mathbf{m}}\rangle$ and obtain an interger k. In Step 6, we will show later that with high provability $m_k = \prod_{i=1}^{u_k} x_{n_i} \neq 0$ at a solution $\mathbf{a} = (\mathbf{a_1}, \ldots, \mathbf{a_n})$ of $\mathcal{F} = 0$. Since a_i is either 0 or 1, $\prod_{i=1}^{u_k} a_{n_i} \neq 0$ implies $a_{n_i} = 1$ for all n_i. We thus set $x_{n_i} = 1$ in Step 7 and try to find the other coordinates of $\mathbf{a}$.

In Step 8, $1 \in \mathcal{F}_1$ implies $\mathbb{V}_{\mathbb{C}}(\mathcal{F}_1, \mathbb{H}_{\mathbb{X}}) = \emptyset$. From step 1, $\mathcal{F}(\mathbf{1}) \neq \mathbf{0}$. If $\mathbb{Y} = \emptyset$, then $\mathbb{V}_{\mathbb{C}}(\mathcal{F}_1, \mathbb{H}_{\mathbb{X}}) = \emptyset$.

In Step 9, if $\mathcal{F}_1 = \emptyset$ or $\mathcal{F}_1(\mathbf{0}) = \mathbf{0}$, then we find a solution of $\mathcal{F}$: $x_i = 0$ for any $x_i \in \mathbb{Y}$ and $x_j = 1$ for any $x_j \notin \mathbb{Y}$, which will be returned in Step 10. Otherwise, we repeat the procedure for $\mathcal{F}_1$ in Step 11.

Secondly, we prove the correctness of Theorem 4.3. It is easy to see that if the algorithm returns a solution, then it is a Boolean solution of $\mathcal{F} = 0$. The second property of Theorem 4.3 follows from Lemma 4.6.

The following lemma gives the successful probability for Step 5.

Lemma 4.4. *In Steps 5 and 6, with a probability $> 1 - \epsilon_1/n$, $\mathbb{V}_{\mathbb{C}}(\mathcal{F}_1, \mathbb{H}_{\mathbb{X}}) \neq \emptyset$ implies that there exists an $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{V}_{\mathbb{C}}(\mathcal{F}_1, \mathbb{H}_{\mathbb{X}})$ with $a_{n_i} = 1$ for $i = 1, \ldots, u_k$.*

Proof. Let $|\mathbf{m}_D\rangle = \sum_{j=1}^{Q_D-1} \alpha_j |\mathbf{e}_j\rangle$ be the solution state and $|\widehat{\mathbf{m}}_D\rangle = \sum_{j=1}^{Q_D-1} \beta_j |\mathbf{e}_j\rangle$ be the approximate state obtained with the HHL algorithm. By the definition of quantum measurement, we have a probability $\| \sum_{k,\alpha_k=0} \beta_k |\mathbf{e}_k\rangle\|^2 < \| \sum_{k=1}^{Q_D-1} (\beta_k - \alpha_k)|\mathbf{e}_k\rangle\|^2 = \|\widehat{\mathbf{m}}_D - \mathbf{m}_D\|^2 < \epsilon_1/n$ to obtain some state $|\mathbf{e}_k\rangle$ under the condition $m_k(\mathbf{a}) = 0$ for all $\mathbf{a} \in \mathbb{V}(\mathcal{F}_1)$. Conversely, we have a probability $> 1 - \epsilon_1/n$ to obtain some state $|\mathbf{e}_k\rangle$ such that $m_k(\mathbf{a}) = 1$ for some $\mathbf{a} \in \mathbb{V}(\mathcal{F}_1)$. □

Lemma 4.5. *The loop from Step 3 to Step 9 will run at most n times, and returns $\emptyset$ with probability $< \epsilon_1$ when $\mathcal{F} = 0$ has Boolean solutions.*

Proof. Since at each loop, the values of at least one x_i will be determined in Step 6, we will repeat this loop for at most n times. By Lemma 4.4, when $\mathcal{F} = 0$ has Boolean solutions, the algorithm returns $\emptyset$ with probability $< 1 - (1 - \epsilon_1/n)^n < \epsilon_1$. □

Lemma 4.6. *The loop from Step 2 to Step 11 will run at most $\lceil \log_{\epsilon_1} \epsilon \rceil$ times and with probability $\geq 1 - \epsilon$, returns a Boolean solution of $\mathcal{F} = 0$ when $\mathcal{F} = 0$ has Boolean solutions.*

Proof. By Lemma 4.5, if $\mathcal{F}$ has Boolean solutions, then the probability that we reach step 11 is $< \epsilon_1$. The number of loops from Step 2 to Step 11 is at most $\lceil \log_{\epsilon_1} \epsilon \rceil$. Then, if $\mathcal{F}$ has Boolean solutions, then the probability that the algorithm returns $\emptyset$ is $\epsilon_1^{\lceil \log_{\epsilon_1} \epsilon \rceil} < \epsilon$. $\square$

Finally, we will estimate the complexity of Algorithm 4.2. Step 4 is the dominate step in terms of complexities. The complexities for other steps are very low comparing to that of Step 4. So, we just omit them from the complexity analysis.

We have $D \leq (n+1)(d-1)+2$ and $\mathscr{M}_{\mathcal{F}_2,D}$ is of dimension $(\sum_{i=1}^{r} Q_{D-d_i}) \times (Q_D - 1)$ and $(2n+T)$-sparseness. By Corollary 3.15, the complexity of Step 4 is $c\log(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1)(2n+T)\kappa^2\sqrt{n/\epsilon_1}$.

By Lemma 4.5, the loop from Step 3 to Step 9 will run at most n times. Then the complexity for the loop from Step 3 to Step 9 is $\sum_{j=0}^{n-1}(c\log(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1)(2(n-j)+T)\kappa^2\sqrt{n/\epsilon_1}) = c\log(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1)(n(n+1)+nT)\kappa^2\sqrt{n/\epsilon_1}$.

By Lemma 4.6, the loop from Step 2 to Step 11 will run at most $\lceil \log_{\epsilon_1} \epsilon \rceil$ times. Then the total complexity of the algorithm is $c\log(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1)(n(n+1)+nT)\kappa^2\sqrt{n/\epsilon_1}\lceil \log_{\epsilon_1} \epsilon \rceil = c\log(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1)n^{1.5}(n+1+T)\kappa^2\sqrt{2}\lceil \log_2 1/\epsilon \rceil$, by choosing ϵ_1 to be $1/2$.

Since $\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1 \leq rQ_{D-1} + Q_D$, we have $\log(\sum_{i=1}^{r} Q_{D-d_i} + Q_D - 1)) < \log(rQ_{D-1} + Q_D) = \log(r\binom{n+D-1}{n} + \binom{n+D}{n}) = \log((rn/(n+D)+1)\binom{n+D}{n}) = \log(rn/(n+D)+1) + \log\binom{n+D}{n} \leq \log(nr/(dn+d+1)+1) + \log\binom{n+D}{n} \leq \log(r+1) + \log\binom{n+D}{n}$.

To estimate the number $\binom{n+D}{n}$, we need to use Stirling's approximation:

$$\sqrt{2\pi}n^{n+1/2}e^{-n} \leq n! \leq en^{n+1/2}e^{-n}, \tag{8}$$

where e is Euler's number.

Lemma 4.7. *For $n, \gamma \in \mathbb{R}_{\geq 1}$,*

$$\binom{n+\gamma n}{n} \leq \frac{e(\gamma+1)^{1/2}}{2\pi(\gamma n)^{1/2}}e^n \leq \frac{e^{n+1}}{\sqrt{2n\pi}}. \tag{9}$$

Proof. $\binom{n+\gamma n}{n} = \frac{(n+\gamma n)!}{(\gamma n)!n!} \leq \frac{e(n+\gamma n)^{n+\gamma n+1/2}e^{-n-\gamma n}}{\sqrt{2\pi}n^{n+1/2}e^{-n}\sqrt{2\pi}(\gamma n)^{\gamma n+1/2}e^{-\gamma n}} = \frac{en^{n+\gamma n+1/2}(\gamma+1)^{n+\gamma n+1/2}}{2\pi n^{n+\gamma n+1}\gamma^{\gamma n+1/2}} = \frac{e(\gamma+1)^{1/2}}{2\pi(\gamma n)^{1/2}}(1+1/\gamma)^{\gamma n} \leq \frac{e(\gamma+1)^{1/2}}{2\pi(\gamma n)^{1/2}}e^n$. Considering $\gamma \geq 1$, we have $\binom{n+\gamma n}{n} \leq \frac{e2^{1/2}}{2\pi n^{1/2}}e^n$. $\square$

Let $\gamma = D/n = ((n+1)(d-1)+2)/n = d+(d+1)/n$, and we have $\log_2\binom{n+D}{n} \leq \log_2(\frac{e^{n+1}}{\sqrt{2n\pi}}) = (n+1)\log_2 e - \log_2(\sqrt{2n\pi}) < n\log_2 e$. Finally, we have the exact complexity for Algorithm 4.2:

Lemma 4.8. *The exact time complexity for Algorithm 4.2 is $\sqrt{2}c(n\log_2 e + \log_2 r)n^{1.5}(n+1+T)\kappa^2\lceil \log_2 1/\epsilon \rceil$. Moreover, it equals to $\widetilde{O}(n^{2.5}(n+T)\kappa^2\log 1/\epsilon)$, because $r \leq T$.*

We have completed the proof of Theorem 4.3.

Remark 4.9. *The purpose of the loop started in Step 11 is to use less qubits. If we do not use this loop, we need to set $\epsilon_1 = \epsilon$ in step 4 and the precision needed is $\sqrt{\frac{\epsilon}{n}}$. With the loops started in Step 11, we can use a large value for $\epsilon_1 \in (0,1)$, say $\epsilon_1 = 1/2$, then the precision needed in Step 4 is $\sqrt{\epsilon_1/n} = \frac{1}{\sqrt{2n}}$ which is generally larger than $\sqrt{\frac{\epsilon}{n}}$.*

Remark 4.10. *Note that the degrees $d_i = \deg(f_i)$ does not appear in the complexity of the algorithm. Also note that due to Step 2, we have $\deg(f_i) \leq n$.*

We can easily improve our algorithm to the following form.

Remark 4.11. *Given $(a_1, \ldots, a_n) \in \mathbb{C}^n$, $\mathcal{F} \subset \mathbb{C}[\mathbb{X}]$, we can obtain an element in $\mathbb{V}_{\mathbb{C}}(\mathcal{F}, x_1^2 - a_1x_1, \ldots, x_n^2 - a_nx_n)$ by Algorithm 4.2, where we need to replace $\mathbb{H}_{\mathbb{X}}$ with $(x_1^2 - a_1x_1, \ldots, x_n^2 - a_nx_n)$ and $x_{n_i} = 1$ with $x_{n_i} = a_{n_i}$ in Step 6.*

4.2 Obtain all the Boolean solutions

We will show how to find all Booelan solutions of $\mathcal{F}$. For a Boolean solution $\mathbf{a}$ of $\mathcal{F}$, the following lemma shows how to construct a polynomial system $\mathcal{F}_1$ satisfying $\mathbb{V}_{\mathbb{C}}(\mathcal{F}_1, \mathbb{H}_{\mathbb{X}}) = \mathbb{V}_{\mathbb{C}}(\mathcal{F}, \mathbb{H}_{\mathbb{X}}) \backslash \{\mathbf{a}\}$.

Lemma 4.12. *For $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{V}_{\mathbb{C}}(\mathcal{F}, \mathbb{H}_{\mathbb{X}})$, we have*

$$\mathrm{Proj}_{\mathbb{X}} \mathbb{V}_{\mathbb{C}}(\mathcal{F}, \mathbb{H}_{\mathbb{X}}, \mathbb{S}, f_{\mathbf{a}}) = \mathbb{V}_{\mathbb{C}}(\mathcal{F}, \mathbb{H}_{\mathbb{X}}) \setminus \{\mathbf{a}\}$$

where $\mathbb{S} = \{\bar{x}_i + x_i - 1 \,|\, i = 1, \ldots, n\}$, $f_{\mathbf{a}} = \prod_{a_i=0} \bar{x}_i \prod_{a_i=1} x_i$, and $\bar{x}_i$ are new variables.

Then we can use the Algorithm 4.2 to find all Boolean Solutions for $\mathcal{F} = 0$.

Algorithm 4.13.

Input: $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{C}[\mathbb{X}]$ with $T = \sum_{i=1}^r \# f_i$ and $d_i = \deg(f_i)$. Also $\epsilon \in (0, 1)$.
Output: $\mathbb{V}_{\mathbb{C}}(\mathcal{F}, \mathbb{H}_{\mathbb{X}})$.

Step 1: Set $S = \emptyset$. $\mathcal{F}_1 = \mathcal{F} \cup \{x_1 + \bar{x}_1 - 1, \ldots, x_n + \bar{x}_n - 1\} \subset \mathbb{C}[\mathbb{X}, \overline{\mathbb{X}}]$ with $\overline{\mathbb{X}} = \{\bar{x}_1, \ldots, \bar{x}_n\}$.

Step 2: Use Algorithm 4.2 to compute Boolean solutions of $\mathcal{F}_1 = 0$. If we obtain $\emptyset$, return S. Else we obtain a Boolean solution $\mathbf{a} = (a_1, \ldots, a_n)$.

Step 3: $S = S \cup \{\mathbf{a}\}$, $\mathcal{F}_1 = \mathcal{F}_1 \cup \{\prod_{a_i=0} \bar{x}_i \prod_{a_i=1} x_i\}$. Goto Step 2.

Theorem 4.14. *Let $w = \#D$. Then Algorithm 4.13 finds w Boolean solutions of $\mathcal{F} = 0$ with complexity $\widetilde{O}(n^{2.5}(n + T + w)w\kappa^2 \log 1/\epsilon)$, and probability at least $(1 - \epsilon)^w$.*

Proof. By Theorem 4.3, the complexity of the algorithm is $\sum_{i=0}^{w-1} \widetilde{O}((2n)^{2.5}(2n + 3n + T + i)\kappa^2 \log 1/\epsilon) = \widetilde{O}(n^{2.5}(n + T + w)w\kappa^2 \log 1/\epsilon)$. □

4.3 Computing Boolean solutions to linear systems and applications

Many well-known problems in computation theory can be described as finding the Boolean solutions for linear systems. In this section, we consider two such problems and their computational complexities using our quantum algorithm.

The *subset sum problem* is an important problem in complexity theory and cryptography. The problem is: given a set of integers, is there a non-empty subset whose sum is a given number? The problem can be described as finding the Boolean solutions for a linear system. We have the following result.

Proposition 4.15. *Let $A \in \mathbb{Z}^{r \times n}$ for $r < n$ and $b \in \mathbb{Z}^r$. There is a quantum algorithm to find the Boolean solutions to the linear system $Ax = b$ with probability $\geq 1 - \epsilon$ and complexity $\widetilde{O}(n^{3.5} r\kappa^2 \log 1/\epsilon)$*

Proof. We have r linear equation of sparseness $(n+1)$ and n quadratic binomials. Thus $T = 2n + nr + r$, by Theorem 4.3, we can use Algorithm 4.2 to find a Boolean solution for $Ax = b$ in time $\widetilde{O}(n^{2.5}(n + 2n + nr + r)\kappa^2 \log 1/\epsilon) = \widetilde{O}(n^{3.5} r\kappa^2 \log 1/\epsilon)$. □

The *graph isomorphism problem* is another well-known problem in computational theory, which is to determine whether two finite graphs are isomorphic. We do not know whether it is NPC or P. The problem can be described as solving the Boolean solutions for a linear system. Let A and B in $\mathbb{F}_2^{n\times n}$ be the adjacent matrices for two graphs, the graph isomorphism problem is to decide whether there exists a permutation matrix P such that $AP = PB$.

Proposition 4.16. *There is a quantum algorithm to decide whether two graphs with n vertices are isomorphism with probability* $\geq 1-\epsilon$ *and complexity* $\widetilde{O}(n^{6.5}\kappa^2 \log 1/\epsilon)$.

Proof. Let $A = (a_{ij})$, $B = (b_{ij})$, $P = (x_{ij})$ with $\sum_i x_{ij} = 1$ for each j, $\sum_j x_{ij} = 1$ for each i, and $x_{ij}^2 - x_{ij} = 0$ for each i, j. Thus in the equation system, the number of $2n$-sparse linear equations is n^2, the number of $(n+1)$-sparse linear equations is $2n$, and the number of quadratic binomials is n^2. Thus $T = 2n^3 + 4n^2 + 2n$, by Theorem 4.3, we can use Algorithm 4.2 to find a Boolean solution for $AP = PB$ in time $\widetilde{O}((n^2)^{2.5}(n^2 + 2n^3 + 4n^2 + 2n)\kappa^2 \log 1/\epsilon) = \widetilde{O}(n^8\kappa^2 \log 1/\epsilon)$.

Due to the special property of the problem, the complexity could be reduced as follows. Considering the loop from Step 3 to Step 9 in Algorithm 4.2, since exactly n of x_{ij} equal to 1 in the permutation matrix P, the number of loops will be n instead of n^2. Thus the error bound in step 4 will be $\sqrt{\epsilon_1/n}$ instead of $\sqrt{\epsilon_1/n^2}$. Finally, we can use Algorithm 4.2 to find a Boolean solution for $AP = PB$ in time $\widetilde{O}(n^{8-1.5}\kappa^2 \log 1/\epsilon) = \widetilde{O}(n^{6.5}\kappa^2 \log 1/\epsilon)$. □

By Propositions 4.15 and 4.16, in order to determine the quantum complexity of these two problems, we need only to study the condition numbers of the corresponding equation systems.

5 Solving Boolean equation systems

In this section, we will give a quantum algorithm to solve Boolean equations by converting the problem into computing the Boolean solutions of a sparse polynomial system over $\mathbb{C}$.

5.1 Reduce Boolean systems to polynomial systems over $\mathbb{C}$

Let $\mathbb{F}_2$ be the field consisting of 0 and 1. We will consider the problem of equation solving over $\mathbb{F}_2$, or equivalently, solving Boolean equations. Let $\mathbb{X} = \{x_1, \ldots, x_n\}$ be a set of indeterminants and

$$\mathcal{R}_2[\mathbb{X}] = \mathbb{F}_2[\mathbb{X}]/(\mathbb{H}_{\mathbb{X}}),$$

where $\mathbb{H}_{\mathbb{X}} = \{x_1^2 - x_1, \ldots, x_n^2 - x_n\}$. Then $\mathcal{R}_2[\mathbb{X}]$ is a Boolean ring and every ideal in $\mathcal{R}_2$ is radical. Elements in $\mathcal{R}_2$ are called *Boolean polynomials*, which have the form $\sum_i m_i$ and m_i are Boolean monomials with degree at most one for each x_i. Similar to Section 3, we use $\mathbb{V}_{\mathbb{F}_2}(\mathcal{F})$ to denote the zeros of $\mathcal{F} \subset \mathcal{R}_2[\mathbb{X}]$ in $\mathbb{F}_2$.

We first show how to reduce a given Boolean polynomial into several s-sparse Boolean polynomials for a given $s \in \mathbb{N}_{\geq 3}$. Let $f = \sum_{i=1}^t m_i$ be a Boolean polynomial. Set $S_t = \lceil \frac{t-s}{s-2} \rceil$ and $\mathbb{U}_f = \{u_1, \ldots, u_{S_t}\}$ be a set of new variables depending on f. We define a Boolean polynomial set $S(f, s)$ as follows. If $t \leq s$, then $S(f, s) = \{f\}$. Otherwise, let

$$S(f, s) = \{\check{f}_1, \ldots, \check{f}_{S_t+1}\} \subset \mathcal{R}_2[\mathbb{X}, \mathbb{U}_f] \tag{10}$$

where $\check{f}_1 = \sum_{k=1}^{s-1} m_k + u_1$, $\check{f}_j = \sum_{k=(j-1)(s-2)+2}^{j(s-2)+1} m_k + u_{j-1} + u_j$ for $j = 2, \ldots, S_t$, and $\check{f}_{S_t+1} = \sum_{k=S_t(s-2)+2}^{t} m_k + u_{S_t}$. $S(f, s)$ is called the *splitting set* for f.

For the convenience of presentation, in this paper, we give new meaning to the notation: $\lceil e \rceil = 0$ if $e \leq 0$. With this assumption, $\#\mathbb{U}_f = \lceil \frac{t-s}{s-2} \rceil$ and $\#S(f, s) = \lceil \frac{t-s}{s-2} \rceil + 1$.

For a set $\mathcal{F}$ of Boolean polynomials, denote $\mathbb{U}(\mathcal{F}, s) = \bigcup_{f \in \mathcal{F}} \mathbb{U}(f, s)$, and

$$S(\mathcal{F}, s) = \bigcup_{f \in \mathcal{F}} S(f, s) \subset \mathcal{R}_2[\mathbb{X}, \mathbb{U}(\mathcal{F}, s)]. \tag{11}$$

The following results are easy to check.

Lemma 5.1. *Let $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathcal{R}_2[\mathbb{X}]$ and $t_i = \#f_i$. Then $S(\mathcal{F}, s)$ is s-sparse, $\deg(S(\mathcal{F}, s)) = \deg(\mathcal{F})$, $\#S(\mathcal{F}, s) = r + \sum_i \lceil \frac{t_i - s}{s-2} \rceil$, $\#(\mathbb{X} \cup \mathbb{U}(\mathcal{F}, s)) = n + \sum_i \lceil \frac{t_i - s}{s-2} \rceil$. In particular, $\#S(\mathcal{F}, 3) = T - 2r$ and $\#(\mathbb{X} \cup \mathbb{U}(\mathcal{F}, 3)) = n + T - 3r$, where $T = \sum_i t_i$.*

Lemma 5.2. *Let $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathcal{R}_2[\mathbb{X}]$. For a given $s \in \mathbb{N}_{\geq 3}$, we have*

$$(S(\mathcal{F}, s)) \bigcap \mathcal{R}_2[\mathbb{X}] = (\mathcal{F}), \ \mathrm{Proj}_{\mathbb{X}} \mathbb{V}_{\mathbb{F}_2}(S(\mathcal{F}, s)) = \mathbb{V}_{\mathbb{F}_2}(\mathcal{F}),$$

where $(S(\mathcal{F}, s))$ is the ideal generated by $S(\mathcal{F}, s)$ in $\mathcal{R}_2[\mathbb{X}, \mathbb{U}(\mathcal{F}, s)]$.

The following example shows that we cannot use a method of equation solving over $\mathbb{C}$ to solve Boolean equations directly.

Example 5.3. *Let $f = x_1 + x_2 + 1$. Then $\mathbb{V}_{\mathbb{F}_2}(f) = \{(0, 1), (1, 0)\}$. But $\mathbb{V}_{\mathbb{C}}(f, x_1^2 - x_1, x_2^2 - x_2) = \emptyset$.*

The following lemma shows how to transfer Boolean equation solving to equation solving over $\mathbb{C}$.

Lemma 5.4. *Let $\mathcal{F} = \{f_1, \ldots, f_r\}$ be a set of Boolean polynomials with $t_i = \#f_i$. In $\mathbb{C}[\mathbb{X}]$, let $F_i = \prod_{k=f_i(\mathbf{0})}^{\lfloor t_i/2 \rfloor} (f_i - 2k)$ and let*

$$C(\mathcal{F}) = \{F_1, \ldots, F_r\} \cup \mathbb{H}_{\mathbb{X}}. \tag{12}$$

Then $\mathbb{V}_{\mathbb{F}_2}(\mathcal{F}) = \mathbb{V}_{\mathbb{C}}(C(\mathcal{F}))$. Furthermore, $(C(\mathcal{F}))$ is a radical ideal in $\mathbb{C}[\mathbb{X}]$ and satisfies Lazard's condition.

Proof. Let $f_i = \sum_{k=1}^{t_i} m_{ik}$ and $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{V}_{\mathbb{F}_2}(\mathcal{F})$. When we regard f_i as a polynomial in $\mathbb{C}[\mathbb{X}]$, $f_i(\mathbf{a}) = \sum_{k=1}^{t_i} m_{ik}(\mathbf{a})$ is an even integer between $f_i(\mathbf{0})$ and t_i, because $f_i(\mathbf{a}) \equiv 0 \mod 2$. Thus $\mathbf{a}$ is a zero of $F_i = \prod_{k=f_i(\mathbf{0})}^{\lfloor t_i/2 \rfloor} (f_i - 2k)$. The other direction is easy: a zero $\mathbf{a}$ of $F_i = \prod_{k=f_i(\mathbf{0})}^{\lfloor t_i/2 \rfloor} (f_i - 2k)$ satisfies $f_i(\mathbf{a}) = 2k$ for some k, and hence $f_i(\mathbf{a}) = 0$ in $\mathcal{R}_2[\mathbb{X}]$. By Lemma 4.1, $(C(\mathcal{F}))$ is radical and satisfies Lazard's condition. □

Corollary 5.5. *Let F_i be defined in Lemma 5.4. Then the solving degree of $C(\mathcal{F})$ satisfies $\leq (n+1)(t/2+1)d - n + 1$, where $t = \max_i \#f_i$ and $d = \max_i \deg(f_i)$.*

Proof. By Corollary 3.7, the solving degree of $(C(\mathcal{F}))$ satisfies $\mathrm{Sdeg}(C(\mathcal{F})) \leq (n+1)(\max_i \deg(F_i) - 1) + 2 \leq (n+1)(t/2+1)d - n + 1$. □

We summarize the results of this subsection as the following theorem.

Theorem 5.6. *Let $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathcal{R}_2[\mathbb{X}]$ with $t_i = \#f_i$ and $d_i = \deg(f_i)$. For a given $s \in \mathbb{N}_{\geq 3}$, let $\mathbb{Y} = \mathbb{X} \cup \mathbb{U}(\mathcal{F}, s)$. Then we have a polynomial set*

$$P(\mathcal{F}, s) = C(S(\mathcal{F}, s)) \subset \mathbb{C}[\mathbb{Y}]$$

such that $\mathbb{V}_{\mathbb{F}_2}(\mathcal{F}) = \text{Proj}_{\mathbb{X}}\mathbb{V}_{\mathbb{C}}(P(\mathcal{F}, s))$ and $(P(\mathcal{F}, s))$ is a 0-dimensional radical ideal in $\mathbb{C}[\mathbb{Y}]$ and satisfies Lazard's condition. Furthermore, $P(\mathcal{F}, s)$ is $s(s+1)^{\lfloor s/2 \rfloor}$-sparse, $\deg(P(\mathcal{F}, s)) \leq (\lfloor s/2 \rfloor + 1)\deg(\mathcal{F})$, $\#P(\mathcal{F}, s) = r + n + 2\sum_i \lceil \frac{t_i - s}{s-2} \rceil$, $\#\mathbb{Y} = n + \sum_i \lceil \frac{t_i - s}{s-2} \rceil$, $\text{Sdeg}(P(\mathcal{F}, s)) < (2n + sn + 5rt)d/2 + 2 = O((sn + rt)d)$.

Proof. For any $\check{F} = \prod_{k=\check{f}(\mathbf{0})}^{\lfloor s/2 \rfloor}(\check{f} - 2k) \in C(S(\mathcal{F}, s))$, $\#\check{f} \leq s$ implies $\#\check{F} \leq s(s+1)^{\lfloor s/2 \rfloor}$. By Corollary 5.5, the solving degree of $P(\mathcal{F}, s)$ is $\text{Sdeg}(P(\mathcal{F}, s)) \leq (\#\mathbb{Y} + 1)(s/2 + 1)\deg(S(\mathcal{F}, s)) - \#\mathbb{Y} + 1 = (n + \sum_i \lceil \frac{t_i - s}{s-2} \rceil + 1)(s/2 + 1)d - (n + \sum_i \lceil \frac{t_i - s}{s-2} \rceil) + 1 \leq (n + r(t-2)/(s-2))((s/2 + 1)d - 1) + 2 < (2n + sn + 5rt)d/2 + 2 = O((sn + rt)d)$. □

In our algorithm, we use $s = 3$ and from Theorem 5.6, we have

Corollary 5.7. *For $s = 3$, $P(\mathcal{F}, 3)$ is 6-sparse, $\deg(P(\mathcal{F}, 3)) \leq 2\deg(\mathcal{F})$, $\#P(\mathcal{F}, 3) = 2T + n - 5r$, $\#\mathbb{Y} = T + n - 3r$, $\text{Sdeg}(P(\mathcal{F}, 3)) < 2.5(n + rt)d + 2 = O((n + rt)d)$.*

Proof. We need only to show that $P(\mathcal{F}, 3)$ is 6-sparse and other results are easy to verify. For $s = 3$, we can replace $F_i = \prod_{k=f_i(\mathbf{0})}^{\lfloor t_i/2 \rfloor}(f_i - 2k)$ with

$$\hat{F}_i = \begin{cases} f_i - 2, & \text{if } f_i(\mathbf{0}) = 1; \\ 2m_{i1}m_{i2} + 2m_{i1}m_{i3} + 2m_{i2}m_3 - m_{i1} - m_{i2} - m_{i3}, & \text{if } f_i(\mathbf{0}) = 0, \end{cases} \tag{13}$$

where $f_i = m_{i1} + m_{i2} + m_{i3}$. In $P(\mathcal{F}, 3)$, we have $m^2 = m$ for any monomial m. Then, $F_i = f_i(f_i - 2) = m_{i1}^2 - 2m_{i1} + m_{i2}^2 - 2m_{i2} + m_{i3}^2 - 2m_{i3} + 2m_{i1}m_{i2} + 2m_{i1}m_{i3} + 2m_{i2}m_{i3} = 2m_{i1}m_{i2} + 2m_{i1}m_{i3} + 2m_{i2}m_3 - m_{i1} - m_{i2} - m_{i3} \pmod{\mathbb{H}_{\mathbb{Y}}}$, which is 6-sparse. □

5.2 Quantum algorithm for Boolean equation solving

In this subsection, we will give a quantum algorithm to solve Boolean equations.

Algorithm 5.8.

Input: $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathcal{R}_2[\mathbb{X}]$ and $\epsilon \in (0, 1)$.

Output: A zero of $\mathcal{F}$ or $\emptyset$ meaning that $\mathbb{V}_{\mathbb{F}_2}(\mathcal{F}) = \emptyset$ with probability $> 1 - \epsilon$.

Step 1: Compute $\mathcal{F}_1 = S(\mathcal{F}, 3) \subset \mathcal{R}_2[\mathbb{Y}]$ as defined in (11), where $\mathbb{Y} = \mathbb{X} \cup \mathbb{U}(\mathcal{F}, 3)$.

Step 2: Compute $\mathcal{F}_2 = C(\mathcal{F}_1) \subset \mathbb{C}[\mathbb{Y}]$ as defined in (12).

Step 3: Use Algorithm 4.2 to find a Boolean solution of $\mathcal{F}_2 = 0$ over $\mathbb{C}[\mathbb{Y}]$, with probability bound ϵ. Return $\emptyset$ if Algorithm 4.2 returns $\emptyset$, else we have a Boolean solution $\mathbf{a}$ for $\mathcal{F}_2$.

Step 4: Return $\text{Proj}_{\mathbb{X}}\mathbf{a}$.

Theorem 5.9. *Algorithm 5.8 has the following properties.*

- *If the algorithm returns a solution, then it is a solution of $\mathcal{F} = 0$. Equivalently, if $\mathbb{V}_{\mathbb{F}_2}(\mathcal{F}) = \emptyset$, the algorithm returns $\emptyset$.*

- *If $\mathbb{V}_{\mathbb{F}_2}(\mathcal{F}) \neq \emptyset$, the algorithm computes a solution of $\mathcal{F} = 0$ with probability $> 1 - \epsilon$.*
- *The complexity is $\widetilde{O}((n^{3.5} + T^{3.5})\kappa^2 \log 1/\epsilon)$ if using HHL and $\widetilde{O}((n^{4.5} + T^{4.5})\kappa \log 1/\epsilon)$ if using Ambainis' algorithm [2], where $T = \sum_i \#f_i$ and κ is the maximal condition number of the polynomial system $\mathcal{F}_2$, called the* condition number *of the Booolean system $\mathcal{F}$.*

Proof. In Step 2, we split $\mathcal{F}$ to 3-sparse polynomials and then turn them into a polynomial system over $\mathbb{C}$ in time $O(T)$. By Theorem 5.6, $\mathbb{V}_{\mathbb{F}_2}(\mathcal{F}) = \mathbb{V}_{\mathbb{C}}(\mathcal{F}_2)$. Thus, we need only to solve $\mathcal{F}_2$ over $\mathbb{C}$ in Step 3.

By Theorems 4.3 and 5.6, the complexity of Step 3 is $\widetilde{O}((n + \sum_i \lceil \frac{t_i - s}{s-2} \rceil)^{2.5}(n + \sum_i \lceil \frac{t_i - s}{s-2} \rceil + (r + \sum_i \lceil \frac{t_i - s}{s-2} \rceil) s(s+1)^{\lfloor s/2 \rfloor})\kappa^2 \log 1/\epsilon) = (n + T/s)^{2.5}(n + Ts^{s/2})\kappa^2 \log 1/\epsilon)$. To minimize the complexity, we choose $s = 3$. By Corollary 5.7, the complexity is $\widetilde{O}((n + T - 3r)^{2.5}(n + T - 3r + 3(T - 2r))\kappa^2 \log 1/\epsilon) = \widetilde{O}((n^{3.5} + T^{3.5})\kappa^2 \log 1/\epsilon)$. □

Remark 5.10. *In Step 8 of Algorithm 4.2, we can replace $\mathbb{Y} = \emptyset$ with $\mathbb{X} \cap \mathbb{Y} = \emptyset$ so that Algorithm 5.8 can terminate early.*

Example 5.11. *$n = 5$, $\mathcal{F} = \{x_1 + x_3 + x_5, x_2 + x_3 + 1, x_1x_4 + x_2 + 1, x_3x_5\}$ When we use Algorithm 4.2 to solve $\mathbb{V}(\mathcal{F}, \mathbb{H}_{\mathbb{X}})$, we have the* $\mathrm{Sdeg}(\mathcal{F} \cup \mathbb{H}_{\mathbb{X}}) = 8$*, the Macaulay matrix is of dimension* 3894×1286*, and its condition number is $\kappa = 15.44$.*

The following lemma shows that we can find Boolean solutions over certain finite fields.

Corollary 5.12. *For an equation system $\mathcal{F} = \{f_1, \ldots, f_r\}$ over a finite field $\mathbb{F}_p$ for a prime number p, if all solution of $\mathcal{F}$ is Boolean* (0 *or* 1)*, we can also use Algorithm 5.8 to compute a solution of $\mathcal{F}$ by replacing $F_i = \prod_{k=f_i(\mathbf{0})}^{\lfloor t_i/2 \rfloor}(f_i - 2k)$ with $F_i = \prod_{k=f_i(\mathbf{0})}^{\lfloor t_i/p \rfloor}(f_i - pk)$ in Step 2.*

The following theorem gives the exact complexity for solving Boolean equations, which will be used in Section 6.

Theorem 5.13. *Let $\mathcal{F} = \bigcup_{s=1}^{t}\{f_{s1}, \ldots, f_{sr_s}\} \subset \mathcal{R}_2[\mathbb{X}]$ be a Boolean equation system such that $s = \#f_{sj}$, $T = \sum_{s=1}^{t} sr_s$, $r = \sum_{s=1}^{t} r_s$. Then we can find a solution of $\mathcal{F} = 0$ in time $\sqrt{2}c((n + T - 3r + 2r_1 + r_2)\log_2 e + \log_2(n + 2T - 5r + 4r_1 + 2r_2))(n + T - 3r + 2r_1 + r_2)^{1.5}(3n + 9T - 21r + 12r_1 + 5r_2 + 1)\kappa^2 \lceil \log_2 1/\epsilon \rceil$, where c is the complexity constant of the HHL algorithm defined in Corollary 2.4.*

Proof. By Corollary 5.7, $C(S(\mathcal{F}, 3))$ consists of r_1 monomials, $n + T - 3r + 2r_1 + 2r_2$ binomials, and $T - 2r + r_1$ polynomials of sparseness 6, and the number of indeterminates is $n + T - 3r + 2r_1 + r_2$. Thus the total sparseness for $C(S(\mathcal{F}, 3))$ is $T_P = r_1 + 2(n + T - 3r + 2r_1 + 2r_2) + 6(T - 2r + r_1) = 2n + 8T - 18r + 10r_1 + 4r_2$. By Lemma 4.8, the exact complexity for Algorithm 5.8 to find a solution is $\sqrt{2}c(\log_2(r_1 + n + T - 3r + 2r_1 + 2r_2 + T - 2r + r_1) + (n + T - 3r + 2r_1 + r_2)\log_2 e)(n + T - 3r + 2r_1 + r_2)^{1.5}((n + T - 3r + 2r_1 + r_2) + 1 + (2n + 8T - 18r + 10r_1 + 4r_2))\kappa^2 \lceil \log_2 1/\epsilon \rceil = \sqrt{2}c(\log_2(n + 2T - 5r + 4r_1 + 2r_2) + (n + T - 3r + 2r_1 + r_2)\log_2 e)(n + T - 3r + 2r_1 + r_2)^{1.5}(3n + 9T - 21r + 12r_1 + 5r_2 + 1)\kappa^2 \lceil \log_2 1/\epsilon \rceil$. □

5.3 Application to 3-satisfiability problem

Let $\mathbb{X} = \{x_1, \ldots, x_n\}$ be Boolean indeterminates. A 3-SAT problem is to check the satisfiability of the propositional logic formula $y_{i1} \vee y_{i2} \vee y_{i3} = 1$ for $i = 1, \ldots, r$, where $y_{ij} = x_k$ or $\neg x_k$ for some k. Decision of 3-SAT is NPC. The 3-SAT problem is equivalent to solve the Boolean equation system

$$\mathcal{F} = \{\bar{y}_{i1}\bar{y}_{i2}\bar{y}_{i3} : i = 1, \ldots, r\} \cup \{x_k + \bar{x}_k + 1 : k = 1, \ldots, n\}\}$$

in $\mathbb{R}_2[\mathbb{X},\overline{\mathbb{X}}]$, where $\overline{\mathbb{X}} = \{\bar{x}_1,\dots,\bar{x}_n\}$.

Proposition 5.14. *For a 3-SAT with r clauses, there is a quantum algorithm to decide its satisfiability with probability $1-\epsilon$ and with complexity $\widetilde{O}((n^{2.5}(n+r)\kappa^2\log 1/\epsilon)$.*

Proof. It is easy to see that solving the Boolean system $\mathcal{F}$ is equivalent to find the Boolean solutions for the polynomial system in $\mathbb{C}[\mathbb{X},\overline{\mathbb{X}}]$,

$$\mathcal{F}_1 = \{\bar{y}_{i1}\bar{y}_{i2}\bar{y}_{i3} : i=1,\dots,r\} \cup \{x_k+\bar{x}_k-1 : k=1,\dots,n\} \cup \{x_k^2-x_k : k=1,\dots,n\},$$

that is, the 3-SAT problem is satisfiable if and only if $\mathbb{V}_{\mathbb{C}}(\mathcal{F}) \neq \emptyset$. Also note that $x+\bar{x}-1=0$ and $x^2-x=0$ imply $\bar{x}^2-\bar{x}=0$. $\mathcal{F}_1$ consists of n binomials, n trinomials and r monomials. Thus $T=5n+r$, by Lemma 4.8, we can use Algorithm 4.2 to find a Boolean solution in time $\widetilde{O}(n^{2.5}(n+5n+r)\kappa^2\log 1/\epsilon) = \widetilde{O}(n^{2.5}(n+r)\kappa^2\log 1/\epsilon)$. □

The best classic probabilistic algorithm for 3-SAT was $(\frac{3}{4})^n$ given in [22]. In order for our quantum algorithm to perform better n should be ≥ 64, if κ is not too big.

6 Solving Boolean quadratic equations and cryptanalysis

Cryptanalysis of stream ciphers, block ciphers, certain hash functions, and MPKC can be reduced to the solving of multivariate Boolean quadratic equations (BMQ). In this section, we will apply our quantum algorithm to the analysis of these cryptosystems.

6.1 Quantum algebraic attack against AES

The Advanced Encryption Standard (AES), also known by its original name Rijndael, is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology in 2001 [12].

Murphy and Robshaw [21] proposed a method to construct a Boolean equation system, solving of which consists of an algebraic attack against AES. We will use this approach to establish a BMQ.

Denote the 32-bit key length of AES as N_k and the number of rounds as N_r. Denote p, $c \in \mathbb{F}_2^{4N_k\times 8}$ as the plaintext and the ciphertext of AES, $w_0 \in \mathbb{F}_2^{4N_k\times 8}$ as the key of AES, $w_i \in \mathbb{F}_2^{4N_k\times 8}$ as the expanded key of AES, $\bar{w}_i \in \mathbb{F}_2^{4N_k\times 8}$ as the image of w_i under the S-box map in the key expansion step, $x_i \in \mathbb{F}_2^{4N_k\times 8}$ as the state after the AddRoundKey step of AES, and $y_i \in \mathbb{F}_2^{4N_k\times 8}$ as the state after the InvSubBytes step of AES, where $x_i(j,m)$ means the m-th bit at the j-th word of state x for round i. In the key expansion step, several states $\bar{w}_i$ are obtained as the image of w_i under the S-box. Then, an algebraic attack on the N_r-rounds AES with key length N_k is to solve the following BMQ, denoted as AES-(N_k,N_r):

$$\begin{aligned}
0 &= x_0(j,m)+p(j,m)+w_0(j,m); \\
0 &= x_i(j,m)+w_i(j,m)+\sum_{j',m'}\alpha(j,m,j',m')y_{i-1}(j',m') && \text{for } i=1,\cdots,N_r-1; \\
0 &= c(j,m)+w_{N_r}(j,m)+y_{N_r-1}(5j \bmod 16, m); \\
\mathbf{0} &= \mathcal{S}(x_i(j,0),\dots,x_i(j,7),y_i(j,0),\dots,y_i(j,7)) && \text{for } i=0,\cdots,N_r-1;
\end{aligned}$$

$$
\begin{aligned}
&\mathbf{0} = \mathcal{S}(w_i(\bar{\jmath},0),\ldots,w_i(\bar{\jmath},7),\bar{w}_i(\bar{\jmath},0),\ldots,\bar{w}_i(\bar{\jmath},7)) && \text{for } \bar{\jmath} = 4N_k-4,\ldots,4N_k-1;\\
&0 = w_i(\bar{\jmath},m) + w_{i-1}(\bar{\jmath},m) + \bar{w}_{i-1}(\bar{\jmath}+13,m) + \chi(m,i) && \text{for } \bar{\jmath} = 0,1,2;\\
&0 = w_i(3,m) + w_{i-1}(3,m) + \bar{w}_{i-1}(12,m) + \chi(m,i).\\
&\ \text{For } N_k \le 6:\\
&0 = w_i(\bar{\jmath},m) + w_{i-1}(\bar{\jmath},m) + w_i(\bar{\jmath}-4,m) && \text{for } \bar{\jmath} = 4,\ldots,4N_k-1.\\
&\ \text{For } N_k > 6:\\
&\mathbf{0} = \mathcal{S}(w_i(\bar{\jmath},0),\ldots,w_i(\bar{\jmath},7),\bar{w}_i(\bar{\jmath},0),\ldots,\bar{w}_i(\bar{\jmath},7)) && \text{for } \bar{\jmath} = 12,\ldots,15;\\
&0 = w_i(\bar{\jmath},m) + w_{i-1}(\bar{\jmath},m) + \bar{w}_i(\bar{\jmath}-4,m) && \text{for } \bar{\jmath} = 16,\ldots,19;\\
&0 = w_i(\bar{\jmath},m) + w_{i-1}(\bar{\jmath},m) + w_i(\bar{\jmath}-4,m) && \text{for } \bar{\jmath} = 4,\ldots,15,20,\ldots,4N_k-1,
\end{aligned}
$$

where j runs from 0 to $(4N_k - 1)$, and m runs from 0 to 7. $\bar{w}_i(j,m)$, $x_i(j,m)$, and $y_i(j,m)$ are state variables, $w_i(j,m)$ are key variables, $\mathcal{S}$ is a set of 39 BMQ in $\mathbb{F}_2[x_0,\ldots,x_7,y_0,\ldots,y_7]$ representing the Rijndael S-box, which can be found in the Appendix of this paper. χ is the round constant. Thus x, y, w and $\bar{w}$ are Boolean indeterminates and other alphabets are known constants. In the second group of equations, exactly 640 of $\alpha(j,m,j',m')$ are 1 for a given i.

The equation set $\mathcal{S}$ of the S-box is given in the Appendix (Section 8), which can be simplified as follows. The original $\mathcal{S}$ is a BMQ with total sparseness 1688. By doing Gaussian elimination, we obtain a BMQ $\mathcal{S}$ with with total sparseness 1192. We can introduce 1075 new indeterminates u_{ij} to split $\mathcal{S}$ into 3-sparse BMQ. Thus $P(\mathcal{S},3)$ consists of 1331 quadratic binomials, 115 quadratic polynomials, 989 cubic polynomials, and 10 quartic polynomials over $\mathbb{C}$.

Totally, the AES-(N_k, N_r) can be represented by a BMQ with number of indeterminates $n = 96N_kN_r+32N_k+32N_r$ (if $N_k \le 6$) or $n = 96N_kN_r+32N_k+64N_r$ (if $N_k > 6$), number of equations $r = 220N_rN_k+64N_k+156N_r$ (if $N_k \le 6$) or $r = 220N_rN_k+64N_k+312N_r$ (if $N_k > 6$), and total sparseness $T = 4928N_rN_k + 192N_k + 5440N_r$ (if $N_k \le 6$) or $T = 4928N_rN_k + 192N_k + 10208N_r$ (if $N_k > 6$). By Theorem 5.13, we have

Proposition 6.1. *There is a quantum algorithm to obtain a solution of AES-(N_k, N_r) with complexity $(4364N_kN_r+32N_k+5004N_r)^{2.5}(40020N_kN_r+480N_k+45780N_r+1)\sqrt{2}\log_2 ec\kappa^2 \log_2 1/\epsilon$ (if $N_k \le 6$), or $(4364N_kN_r + 32N_k + 9336N_r)^{2.5}(40020N_kN_r + 480N_k + 85512N_r)\sqrt{2}\log(4e)$ $c\kappa^2 \log_2 1/\epsilon$) (if $N_k > 6$) with probability $> 1-\epsilon$, where κ is the condition number of $\mathcal{F}$ and c is the complexity constant of the HHL algorithm.*

Set $N_k = 4, 6, 8$, $N_r = 4, 6, 8, 10, 12, 14$, and $\epsilon = 1\%$. We have the following complexities on quantum algebraic attack on various AESes. From Table 2, we can see that AES is secure under quantum algebraic attack only if the condition number κ is large.

AES	N_k	N_r	#Vars	#Eqs	T-Sparseness	Complexity
AES-128	4	4	1792	4400	101376	$2^{64.63}c\kappa^2$
AES-128	4	6	2624	6472	151680	$2^{66.68}c\kappa^2$
AES-128	4	8	3456	8544	201984	$2^{68.13}c\kappa^2$
AES-128	4	10	4288	10616	252288	$2^{69.26}c\kappa^2$
AES-192	6	12	7488	18096	421248	$2^{71.83}c\kappa^2$
AES-256	8	14	11904	29520	696384	$2^{74.38}c\kappa^2$

Table 2: Complexities of the quantum algebraic attack on AES

6.2 Quantum algebraic attack against Trivium

Trivium is a synchronous stream cipher designed by Canniére and Preneel [7] in 2005 to provide a flexible trade-off between speed and gate count in hardware, and reasonably efficient software implementation, which has been specified as an International Standard under ISO/IEC 29192-3. Trivium can be represented by the following nonlinear feedback shift registers (NFSR) which can also be considered as BMQ [26] $\mathcal{F}$:

$$\begin{aligned} A(t+93) &= A(t+24)+C(t+45)+C(t)+C(t+1)C(t+2), & 0 \le t \le N_r-67;\\ B(t+84) &= B(t+6)+A(t+27)+A(t)+A(t+1)A_2(t+2), & 0 \le t \le N_r-70;\\ C(t+111) &= C(t+24)+B(t+15)+B(t)+B(t+1)B(t+2), & 0 \le t \le N_r-67;\\ z(t) &= A(t+27)+A(t)+B(t+15)+B(t)+C(t+45)+C(t), & 0 \le t \le N_r-1, \end{aligned} \tag{14}$$

where A, B, C are state variables and z is the output. For an initial state $Z_0 = (A(0), \dots, A(92), B(0), \dots, B(83), C(0), \dots, C(110)) \in \mathbb{F}_2^{288}$, we can generate the key sequence $z(0), z(1), \dots, z(N_r -1)$ with the above NFSR. Thus, for the N_r-round Trivium, $\mathcal{F}$ consists of $(3N_r - 201)$ quadratic polynomials of sparseness 5, and N_r linear polynomials of sparseness 7, and with $(3N_r + 87)$ indeterminates. Thus, $T = 5(3N_r - 201) + 7N_r = 22N_r - 1005$.

The algebraic attach on the N_r-round Trivium is to solve the BMQ (14), where $z(0), z(1), \dots, z(N_r - 1)$ are constants. It is generally believed that for $N_r > 288$, (14) has a unique solution.

Proposition 6.2. *There is a quantum algorithm to find a solution for the N_r-round Trivium equation system in time $2^{17.22}N_r^{3.5}c\kappa^2 \log 1/\epsilon$ with probability $> 1-\epsilon$, where κ is the condition number of $\mathcal{F}$ and c is the complexity constant of the HHL algorithm.*

Proof. By Theorem 5.13, the complexity is $\sqrt{2}c(\log_2((3N_r{+}87){+}2(22N_r{-}1005){-}5(4N_r{-}201)){+}((3N_r{+}87){+}(22N_r{-}1005){-}3(4N_r{-}201))\log_2 e)((3N_r{+}87){+}(22N_r{-}1005){-}3(4N_r{-}201){+}2r_1{+}r_2)^{1.5}(3(3N_r{+}87){+}9(22N_r{-}1005){-}21(4N_r{-}201){+}12r_1{+}5r_2{+}1)\kappa^2\lceil\log_2 1/\epsilon\rceil = \sqrt{2}c(\log_2(27N_r - 918) + (13N_r-315)\log_2 e)(13N_r-315)^{1.5}(123N_r-4562)\kappa^2\lceil\log_2 1/\epsilon\rceil \le 2^{17.22}N_r^{3.5}c\kappa^2\lceil\log_2 1/\epsilon\rceil$. □

In Table 3, we give the complexities for several N_r assuming $\epsilon = 1\%$. From Table 3, we can see that Trivium is secure under quantum algebraic attack only if the condition number κ is large.

Round	#Vars	#Eqs	T-Sparseness	Complexity
288	951	951	5331	$2^{48.11}c\kappa^2$
576	1815	2103	11667	$2^{51.88}c\kappa^2$
1152	3543	4407	24339	$2^{55.50}c\kappa^2$
2304	6999	9015	49683	$2^{59.06}c\kappa^2$

Table 3: Complexities of the quantum algebraic attack on Trivium

6.3 Quantum algebraic attack against Keccak

Keccak [5], the winner of SHA-3, is the latest member of the Secure Hash Algorithm family of standards, released by NIST on August 5, 2015. For Keccak-$[N_h, b, N_r]$, we denote N_h, b, and N_r as the output bit size, the state bit size, and the number of rounds. Let $A_0(x, y, z)$ be the message, $A_i(x, y, z)$ be the state variable after applying the τ function for i-times, and $B_i(x, y, z)$ be the state variable after applying the π function for i-times, where $x, y \in \mathbb{Z}/5\mathbb{Z}$, $z \in \mathbb{Z}/w\mathbb{Z}$, and $w = b/25 = 1, 2, 4, \dots, 64$ is the bit length of each word. Then for Keccak-$[N_h, b, N_r]$, we have the following BMQ $\mathcal{F}$ [27]:

$$\begin{aligned}
&B_i(3y+x,x,z-r(3y+x,x)) = A_{i-1}(x,y,z) + \sum_{j=0}^{4} A_{i-1}(x-1,j,z) + \sum_{j=0}^{4} A_{i-1}(x+1,j,z-1);\\
&A_i(x,y,z) = B_i(x,y,z) + (1-B_i(x+1,y,z))B_i(x+2,y,z), \text{ for } x \neq 0 \text{ or } y \neq 0;\\
&A_i(0,0,z) = B_i(0,0,z) + (1-B_i(1,0,z))B_i(2,0,z) + RC(z)
\end{aligned}$$

for $i = 1,\dots,N_r$, $x,y = 0,\dots,4$, $z = 0,\dots,w$. In the preimage attack on Keccak, $r(3y+x,x)$ and $RC(z)$ are known constants, the first N_b of $A_{N_r}(x,y,z)$ are the known Hash output, and $A_i(x,y,z)$ $(i < N_r)$ and $B_i(x,y,z)$ are indeterminates. Thus we have $n = 2bN_r$ indeterminates and $r = (2b-1)N_r + N_h$ Boolean quadratic equations with total sparseness $T = 401N_rw + 101N_h/25 - 101w$.

Proposition 6.3. *For the BMQ Keccak-$[N_h, b, N_r]$, there is a quantum algorithm to find a preimage in time* $\sqrt{2}c(\log_2(802N_rw+5N_r+77N_h/25)+(401N_rw+3N_r+26N_h/25)\log_2 e)(401N_rw+3N_r+26N_h/25)^{1.5}(3609N_rw+21N_r+384N_h/25+1)\kappa^2\lceil\log_2 1/\epsilon\rceil < 2^{18.21}N_r^{3.5}b^{3.5}c\kappa^2\log_2 1/\epsilon$ *with probability $> 1-\epsilon$, where κ is the condition number of $\mathcal{F}$ and c is the complexity constant of the HHL algorithm.*

Proof. We have $n = 2bN_r$, $r = (2b-1)N_r + N_h$, and $T = 401N_rw + 101N_h/25 - 101w$. By Theorem 5.13, the complexity is $\sqrt{2}c(\log_2(2bN_r + 2(401N_rw + 101N_h/25 - 101w) - 5((2b-1)N_r + N_h)) + (2bN_r + (401N_rw + 101N_h/25 - 101w) - 3((2b-1)N_r + N_h))\log_2 e)(2bN_r + (401N_rw + 101N_h/25 - 101w) - 3((2b-1)N_r + N_h))^{1.5}(32bN_r + 9(401N_rw + 101N_h/25 - 101w) - 21((2b-1)N_r + N_h) + 1)\kappa^2\lceil\log_2 1/\epsilon\rceil \leq \sqrt{2}c(\log_2(802N_rw + 5N_r + 77N_h/25) + (401N_rw + 3N_r + 26N_h/25)\log_2 e)(401N_rw + 3N_r + 26N_h/25)^{1.5}(3609N_rw + 21N_r + 384N_h/25 + 1)\kappa^2\lceil\log_2 1/\epsilon\rceil \leq \sqrt{2}\log_2 ec(401N_rw + 26N_h/25)^{2.5}(3609N_rw + 384N_h/25)\kappa^2\lceil\log_2 1/\epsilon\rceil \leq 2^{18.21}N_r^{3.5}b^{3.5}c\kappa^2\lceil\log_2 1/\epsilon\rceil$. □

Setting $N_h = 224, 256, 384, 512$, $N_r = 24$, $b = 1600$ and $\epsilon = 1\%$, the complexities for various (N_h, b, N_r) are given in Table 4. From Table 4, we can see that Keccak is secure under quantum algebraic attack only if the condition number κ is large.

N_h	b	N_r	#Vars	#Eqs	T-Sparseness	Complexity
224	1600	24	76800	77000	610377	$2^{73.12}c\kappa^2$
256	1600	24	76800	77032	610506	$2^{73.12}c\kappa^2$
384	1600	24	76800	77160	611023	$2^{73.12}c\kappa^2$
512	1600	24	76800	77288	611540	$2^{73.12}c\kappa^2$

Table 4: Complexities of the quantum preimage attack on Keccak

The best known traditional attacks on Keccak were given in [25] and [17]. In [25], practical collision attacks against the 5-round Keccak-224 and an instance of the 6-round Keccak collision challenge were given. In [17], key recovery attacks were given for 4- to 8-round Keccak.

6.4 Quantum algebraic attack against MPKC

Multivariate Public Key Cryptosystem (MPKC) is one of the candidates for post-quantum cryptography [13]. An MPKC is generally constructed as follows

$$H = L \circ G \circ R = (h_1(\mathbb{X}), \dots, h_r(\mathbb{X})) \tag{15}$$

where $L \in GL(m, \mathbb{F}_2)$, $R \in GL(n, \mathbb{F}_2)$, and $G : \mathbb{F}_2^n \to \mathbb{F}_2^m$ is a quadratic map whose inversion can be efficiently computed. L and R are the secret keys and H is the public map. The direct algebraic attack against the MPKC is to solve the BMQ:

$$y_1 = h_1(\mathbb{X}), \ldots, y_r = h_r(\mathbb{X}) \tag{16}$$

where $\mathbb{X} = (x_1, \ldots, x_n)$ is the plaintext and $\mathbb{Y} = (y_1, \ldots, y_r)$ is the known ciphertext. Note that the BMQ in (16) are dense. We have

Proposition 6.4. *For dense BMQ $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathcal{R}_2[\mathbb{X}]$, there is a quantum algorithm to obtain a solution in time $\sqrt{2}c((n + (n^2 + 3n - 4)r/2)\log_2 e + \log_2(n + (n^2 + 3n - 3)r))(n + (n^2 + 3n - 4)r/2)^{1.5}(3n + (9n^2 + 27n - 24)r/2 + 1)\kappa^2\lceil\log_2 1/\epsilon\rceil = O(n^7 r^{3.5}\kappa^2 \log 1/\epsilon)$ with probability $> 1 - \epsilon$, where κ is the condition number of $\mathcal{F}$ and c is the complexity constant of the HHL algorithm.*

Proof. If $\mathcal{F}$ is a dense BMQ, $T = (n+1)(n+2)r/2$. By Theorem 5.13, we can find a solution of $\mathcal{F} = 0$ in time $\sqrt{2}c(\log_2(n + (n+1)(n+2)r - 5r) + (n + (n+1)(n+2)r/2 - 3r)\log_2 e)(n + T - 3r)^{1.5}(3n + 9(n+1)(n+2)r/2 - 21r + 1)\kappa^2\lceil\log_2 1/\epsilon\rceil = \sqrt{2}c(\log_2(n + (n^2 + 3n - 3)r) + (n + (n^2 + 3n - 4)r/2)\log_2 e)(n + (n^2 + 3n - 4)r/2)^{1.5}(3n + (9n^2 + 27n - 24)r/2 + 1)\kappa^2\lceil\log_2 1/\epsilon\rceil = \widetilde{O}(n^7 r^{3.5}\kappa^2 \log 1/\epsilon)$. □

Corollary 6.5. *Suppose $r = \gamma n$. Then there is a quantum algebraic attack against MPKC in time $\widetilde{O}(\gamma n^{10.5}\kappa^2 \log 1/\epsilon)$ with probability $> 1 - \epsilon$, where κ is the condition number of (16).*

Related to this problem, the BMQ Challenge is to solve a given random BMQ with $m = 2n$ or $n = 1.5m$ over the finite fields $\mathbb{F}_2, \mathbb{F}_{2^8}$ [28]. Considering the field $\mathbb{F}_{2^8} = \mathbb{F}_2[\alpha]/(\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1)$, each variable x over $\mathbb{F}_{2^8}$ is a sum of eight Boolean variables, that is $x = \sum_{i=0}^{7} x_i \alpha^i$. Then $\mathcal{F} = \{f_1, \ldots, f_r\} \subset \mathbb{F}_{2^8}[x_1, \ldots, x_n]$ can be rewritten as $\mathcal{F}_1 = \{f_{11}, \ldots, f_{18}, f_{21}, \ldots, f_{m8}\} \subset \mathcal{R}_2[x_{11}, \ldots, x_{18}, x_{21}, \ldots, x_{n8}]$, where x_{ij} and f_{ij} denote the j-th bit of x_i and f_i. As a consequence, equation solving over $\mathbb{F}_{2^8}$ has the same complexity as Boolean equation solving. For the BMQ challenge [28], $m = 2n$ or $n = 1.5m$ implies the complexity is $\widetilde{O}(T^{3.5}\kappa^2 \log 1/\epsilon) < \widetilde{O}(n^{10.5}\kappa^2 \log 1/\epsilon)$.

The best known deterministic algebraic algorithms to solve the BMQ are the Gröbner basis method [3] which has complexity $O(2^{0.841n})$ under certain regularity condition for the equation system, and the multiplication free characteristic set method [15] which has bit complexity $O(2^n)$ for general BMQ. Although exponential in n, these methods had been used to solve BMQ from cryptanalysis with $n = 128$.

Remark 6.6. *From the above discussion, we can see that AES, Trivium, Keccak, and MPKC are secure under quantum algebraic attack only if the condition numbers of the related Boolean equation systems are large. This suggests that a possible* new quantum criterion for cryptosystem design*: the Boolean equation system of the cryptosystem has a large condition number.*

7 Conclusion

We give two quantum algorithms to find the Boolean solutions of a polynomial system in $\mathbb{C}[\mathbb{X}]$ and to solve Boolean equations in $\mathcal{R}_2[\mathbb{X}]$ in any given probability, whose complexities are polynomial in the number of variables, the total sparseness of the equation system, and the condition number of the equation system. As a consequence, we achieved exponential speedup for sparse Boolean equation solving if the condition number of the equation system is small.

The main idea of the algorithm is to solve the Macaulay linear system of the equation system using the modified HHL algorithm and to obtain the Boolean solutions based on the properties of quantum solution state.

The new quantum algorithm is used to give quantum algebraic attack against major cryptosystems AES, Trivium, and SHA-3/Keccak and show that these ciphers are secure under quantum algebraic attack only if the condition numbers of their equation systems are large. Similar results hold for MPKC, which is a candidate for post-quantum cryptosystems.

We also use the quantum algorithms to three famous problems from computational theory: the 3-SAT problem, the graph isomorphism problem, and the subset sum problem and show that the complexities to solve these problems are polynomial in the input size and the condition number of their corresponding equation system.

One of the major problems for future study is on the condition number: either to estimate the condition number of the cryptosystems and the 3-SAT problem, or to find new quantum method to solve Boolean systems, which has less relation with the condition number. It is also interesting to extend the method proposed in this paper to more general equation systems, such as equation solving over the field of complex numbers.

References

[1] Aharonov, D. and Ta-Shma, A., Adiabatic quantum state generation and statistical zero knowledge, *Proc. STOC'03*, 20-29, ACM Press, New York, 2003.

[2] Ambainis, A., Variable time amplitude amplification and a faster quantum algorithm for solving systems of linear equations, *Proc. STACS*, 636-647, 2012.

[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] Berry, D.W., Childs, A.M., Kothari, R., Hamiltonian simulation with nearly optimal dependence on all parameters, *Proc. 56th FOCS*, 792-809, 2015.

[5] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Keccak sponge function family main document. Submission to NIST (Round 2) 3 (2009): 30.

[6] Caminata, A. and Gorla, E., Solving multivariate polynomial systems and an invariant from commutative algebra, arXiv1706.06319, 2017.

[7] Canniére, C.D., Preneel, B., Trivium, in *New Stream Cipher Designs: The eSTREAM Finalists*, LNCS, vol. 4986, 244-266, Springer, 2008.

[8] Cao, Y., Daskin, A., Frankel, S., Kais S., Quantum circuit design for solving linear systems of equations, *Journal of Molecular Physics*, 1675-1680, 2012.

[9] Childs, A.M., Quantum algorithms: equation solving by simulation, *Nature Physics*, 5(12), 861-861, 2009.

[10] Childs, A.M., Cleve, R., Deotto, E., Farhi, E., Gutmann, S., Spielman, D.A., Exponential algorithmic speedup by a quantum walk, *Proc. STOC'03*, 59-68, ACM Press, New York, 2003.

[11] Courtois, N., Klimov, A., Patarin, J., Shamir, A., Efficient algorithms for solving overdefined systems of multivariate polynomial equations, *Eurocrypt'00*, LNCS, vol. 1807, 392-407, Springer, 2000.

[12] Daemen, J. and Rijmen, V., *AES Proposal: Rijndael*, NIST, 1999.

[13] Ding, J., Gower, J.E., Schmidt, D.S., *Multivariate Public Key Cryptosystems*, Springer, 2006.

[14] Faugere, J.C., A new efficient algorithm for computing Gröbner bases (F4), *J. Pure Appl. Algebra*, 139, 61-88, 1999.

[15] Gao, X.S. and Huang, Z., Characteristic set algorithms for equation solving in finite fields, *Journal of Symbolic Computation*, 47, 655-679, 2012.

[16] Harrow, A.W., Hassidim, A., Lloyd, S., Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15): 150502, 2009.

[17] Huang, S., Wang, X., Xu, G., Wang, M., Zhao, J., Conditional cube attack on reduced-round Keccak sponge function, *EUROCRYPT 2017*, 259-288, Springer, 2017.

[18] Huang, Z., Sun Y., Lin D., On the efficiency of solving boolean polynomial systems with the characteristic set method, arXiv:1405.4596, 2016.

[19] Lazard, D., Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations, *Proc. Eurocal 83*, LNCS, vol. 162, 146-156, Springer, Berlin, 1983.

[20] Macaulay, F.S., Some formulas in elimination, *Proc. of the London Mathematical Society*, 35(1), 3-38, 1902.

[21] Murphy, S. and Robshaw, M., Essential algebraic structure within the AES. *CRYPTO'02*, 1-16, 2002.

[22] Schöning, U., A probabilistic algorithm for k-SAT and constraint satisfaction problems, *Proc. FOCS'99*, 410-414, 1999.

[23] Shewchuk, J.R., An introduction to the conjugate gradient method without the agonizing-pain, Tech. Rep. CMU-CS-94-125, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1994.

[24] Shannon, C.E., A mathematical theory of communication, *Bell Syst. Tech. J.*, 27(3), 379-423 and 623-656, 1948.

[25] Song, L., Liao, G., Guo, J., Non-full Sbox linearization: applications to collision attacks on round-reduced Keccak. *CRYPTO'17*, 428-451, Springer, 2017.

[26] Teo, S.G., Wong, K.K.H., Bartlett, H., Simpson L, Dawson, E., Algebraic analysis of Trivium-like ciphers. Proceedings of the Twelfth Australasian Information Security Conference-Volume 149. Australian Computer Society, Inc., 77-81, 2014.

[27] Wu, C.K. and Feng, D., *Boolean Functions and their Applications in Cryptography*, Springer, 2016.

[28] Yasuda, T., Dahan, X., Huang, Y.J., Takagi, T., Sakurai, K., MQ Challenge: hardness evaluation of solving multivariate quadratic problems, *The NIST Workshop on Cybersecurity in a Post-Quantum World*, 2015, https://www.mqchallenge.org/.

8 Appendix. Equations for the AES S-Box

We list the 39 Boolean quadratic polynomials for the AES S-Box used in this paper.

$x_5x_7+x_5x_6+x_3x_7+x_3x_6+x_2x_4+x_1x_7+x_1x_6+x_1x_5+x_1x_3+x_1x_2+x_0x_7+x_0x_3+x_0x_2+x_6y_7+x_7y_6+x_6y_6+x_7y_5+x_5y_5+x_7y_4+x_1y_4+x_2y_3+x_0y_3+x_6y_2+x_4y_2+x_3y_2+x_0y_2+x_4y_0+x_2y_0+x_7+x_5+x_3+y_7+y_2+y_0+1,$

$x_6x_7+x_5x_7+x_4x_7+x_4x_6+x_4x_5+x_3x_4+x_2x_5+x_1x_7+x_1x_6+x_1x_5+x_1x_4+x_1x_3+x_1x_2+x_0x_5+x_0x_1+x_6y_6+y_5y_7+x_3y_4+y_4y_7+y_4y_5+x_5y_3+x_0y_3+y_3y_6+y_3y_4+x_3y_2+x_0y_2+y_2y_4+y_2y_3+x_5y_0+x_3y_0+x_1y_0+y_0y_7+y_0y_3+y_0y_1+x_5+x_3+x_0+y_2+1,$

$x_1y_7+x_0y_7+y_6y_7+x_7y_5+x_6y_5+y_5y_7+x_7y_4+x_5y_3+x_2y_3+y_3y_6+x_2y_2+x_0y_2+y_2y_5+x_6y_1+x_4y_1+x_1y_1+y_1y_2+x_6y_0+x_5y_0+x_4y_0+y_0y_7+y_0y_6+y_0y_5+y_0y_4+y_0y_3+x_3+x_1+y_3+y_2+y_1+1,$

$x_6x_7+x_4x_6+x_3x_7+x_2x_7+x_1x_4+x_0x_6+x_0x_3+x_6y_7+x_4y_7+x_3y_7+x_7y_6+x_3y_6+x_7y_5+x_7y_4+x_1y_4+x_5y_3+x_4y_3+x_1y_3+x_6y_2+x_2y_2+x_6y_1+x_5y_1+x_3y_1+x_1y_1+x_0y_1+x_7y_0+x_6y_0+x_5y_0+x_3y_0+x_2y_0+x_1y_0+x_6+x_1,$

$x_6y_7+x_5y_7+x_1y_7+x_0y_7+x_5y_6+x_4y_6+x_3y_6+x_4y_4+x_3y_4+x_2y_4+x_4y_3+x_3y_3+y_3y_5+x_2y_2+y_2y_7+y_2y_4+y_2y_3+x_7y_1+x_4y_1+x_3y_1+x_1y_1+y_1y_7+y_1y_6+y_1y_5+y_1y_2+x_7y_0+x_5y_0+x_4y_0+x_3y_0+y_0y_4+y_0y_3+y_0y_2+x_6+x_4+x_3+y_2,$

$x_2y_7+y_5y_6+x_1y_4+x_7y_3+x_2y_3+x_1y_3+x_0y_3+y_3y_6+y_3y_4+x_4y_2+x_2y_2+x_0y_2+y_2y_6+y_2y_5+y_2y_3+x_5y_1+x_3y_1+y_1y_7+y_1y_5+y_1y_4+y_1y_3+y_1y_2+x_5y_0+x_4y_0+x_3y_0+x_0y_0+y_0y_6+y_0y_1+x_7+x_6+x_3+x_2+x_1+y_4+y_2+y_1+y_0,$

$x_5y_7+x_3y_6+x_2y_6+x_0y_6+x_6y_5+x_5y_5+x_0y_5+x_6y_4+x_5y_4+x_3y_4+x_2y_4+x_0y_4+y_4y_7+y_3y_6+y_3y_5+x_3y_2+x_2y_2+x_0y_2+y_2y_7+y_2y_6+y_2y_5+x_3y_1+x_2y_1+y_1y_7+y_1y_6+y_1y_3+x_5y_0+x_4y_0+x_1y_0+y_0y_7+y_0y_2+x_6+x_1+y_4+y_3+y_1,$

$x_6y_7+x_3y_7+x_0y_7+x_2y_6+x_4y_4+x_2y_4+x_0y_4+x_7y_3+x_6y_3+x_3y_3+x_2y_3+x_1y_3+x_0y_3+x_5y_2+x_2y_2+x_1y_2+x_3y_1+x_2y_1+x_1y_1+x_3y_0+x_6+x_2+x_1+x_0+y_2,$

$x_7y_7+x_4y_7+x_1y_7+x_7y_6+x_6y_6+x_1y_6+y_6y_7+x_7y_5+x_6y_5+x_2y_5+x_0y_5+x_6y_4+x_4y_4+x_2y_4+y_4y_5+x_4y_3+x_3y_3+x_2y_3+x_1y_3+x_0y_3+y_3y_6+y_3y_5+x_3y_2+y_2y_4+x_6y_1+x_5y_1+x_4y_1+y_1y_5+y_1y_2+x_6y_0+x_2y_0+x_1y_0+y_0y_6+y_4+y_3,$

$x_4y_7+x_3y_7+x_4y_6+x_2y_6+x_1y_6+x_0y_4+x_3y_3+x_1y_3+x_0y_3+x_7y_2+x_3y_2+x_2y_2+x_1y_2+x_0y_2+x_7y_1+x_6y_1+x_5y_1+x_4y_1+x_3y_1+x_0y_1+x_4+x_2+x_1+y_2,$

$x_3x_6+x_2x_5+x_1x_4+x_1x_2+x_0x_4+x_0x_1+x_4y_6+x_2y_6+x_1y_6+x_7y_5+x_7y_4+x_2y_4+x_6y_3+x_4y_3+x_3y_3+x_2y_3+x_7y_2+x_0y_2+x_4y_1+x_3y_1+x_5y_0+x_2y_0+x_1y_0+x_4+x_1+y_4+y_3+y_0+1,$

$x_4x_7+x_2x_3+x_1x_5+x_1x_4+x_0x_7+x_0x_6+x_0x_5+x_0x_4+x_0x_1+x_7y_7+x_4y_7+x_1y_6+x_2y_4+x_1y_4+x_0y_4+x_7y_3+x_4y_3+x_3y_3+x_4y_2+x_3y_2+x_0y_2+x_7y_1+x_5y_1+x_2y_1+x_1y_1+x_0y_1+x_1y_0+x_0y_0+x_7+x_1+x_0+y_2,$

$x_6x_7+x_4x_5+x_3x_7+x_3x_5+x_2x_5+x_2x_4+x_2x_3+x_1x_7+x_0x_6+x_0x_4+x_2y_7+x_0y_7+x_1y_6+x_6y_5+x_2y_4+x_6y_3+x_5y_3+x_2y_3+x_6y_2+x_4y_2+x_3y_2+x_2y_2+x_1y_2+x_7y_1+x_5y_1+x_6y_0+x_5y_0+x_4y_0+x_3y_0+x_0y_0,$

$x_5x_7+x_3x_6+x_1x_7+x_1x_2+x_0x_4+x_0x_3+x_1y_7+x_2y_6+x_1y_6+x_6y_5+x_4y_5+x_2y_5+x_0y_4+x_5y_3+x_2y_3+x_6y_2+x_5y_2+x_1y_2+x_0y_2+x_7y_1+x_6y_0+x_5y_0+x_0y_0+x_6+x_1+y_6+y_2+y_1,$

$x_1y_7+x_0y_7+x_2y_6+x_6y_5+x_2y_5+x_4y_4+x_3y_4+x_2y_4+x_1y_4+x_0y_4+x_5y_3+x_2y_3+x_1y_3+x_7y_2+x_4y_2+x_3y_2+x_1y_2+x_6y_1+x_3y_1+x_2y_1+x_1y_1+x_0y_1+x_5y_0+x_0y_0+x_3+y_2,$

$x_5x_7+x_3x_6+x_1x_7+x_1x_2+x_0x_4+x_0x_3+x_6y_7+x_3y_7+x_0y_7+x_4y_6+x_5y_5+x_2y_5+x_4y_4+x_3y_3+x_1y_2+x_0y_2+x_7y_1+x_5y_1+x_4y_1+x_2y_1+x_7y_0+x_6y_0+x_1y_0+x_6+x_5+x_4+x_2+x_0+y_7,$

$x_7y_7+x_7y_6+x_6y_6+x_4y_6+x_2y_6+x_1y_6+x_7y_5+x_2y_5+x_7y_4+x_1y_4+x_0y_4+y_4y_6+x_7y_3+$

$x_3y_3 + y_3y_5 + y_3y_4 + x_7y_2 + x_4y_2 + x_3y_2 + y_2y_4 + y_2y_3 + y_1y_6 + x_3y_0 + x_2y_0 + x_1y_0 + y_0y_6 + y_0y_4 + y_0y_2 + y_0y_1 + x_5 + x_4 + x_3 + x_1 + y_4 + y_2 + y_1,$

$x_7y_6 + y_6y_7 + x_7y_5 + x_3y_5 + y_5y_7 + x_7y_4 + x_0y_4 + y_4y_7 + y_4y_5 + x_6y_3 + x_4y_3 + x_3y_3 + x_1y_3 + y_3y_7 + y_3y_6 + y_3y_5 + y_3y_4 + x_6y_2 + x_5y_2 + x_4y_2 + x_1y_2 + x_0y_2 + y_2y_5 + x_6y_1 + x_5y_1 + x_4y_1 + x_0y_1 + y_1y_7 + y_1y_4 + y_1y_2 + x_5y_0 + x_4y_0 + y_0y_7 + y_0y_5 + y_0y_4 + x_0 + y_0,$

$x_6y_6 + x_1y_6 + x_5y_5 + x_3y_4 + x_0y_4 + x_7y_3 + x_6y_3 + x_5y_3 + x_1y_3 + x_0y_3 + x_7y_2 + x_7y_1 + x_6y_1 + x_4y_1 + x_3y_1 + x_0y_1 + x_4y_0 + x_2y_0 + x_7 + x_6 + x_4 + x_3 + x_0 + y_2,$

$x_6x_7+x_5x_7+x_4x_6+x_3x_7+x_2x_7+x_2x_5+x_1x_7+x_0x_6+x_0x_1+x_7y_7+x_3y_7+x_1y_7+x_5y_6+x_0y_4+x_6y_3+x_1y_3+x_7y_2+x_6y_2+x_5y_2+x_4y_2+x_3y_2+x_4y_1+x_2y_1+x_5y_0+x_3y_0+x_5+x_2+x_1+x_0+y_3+y_2,$

$x_0y_7 + x_2y_6 + x_0y_6 + x_5y_5 + x_0y_5 + x_6y_4 + x_3y_4 + x_2y_4 + x_0y_4 + x_7y_3 + x_6y_3 + x_4y_3 + x_3y_3 + x_2y_3 + x_4y_2 + x_3y_2 + x_4y_1 + x_2y_1 + x_7y_0 + x_2y_0 + x_0y_0 + x_7 + x_4 + x_2 + x_1 + y_4 + y_1 + y_0,$

$x_5x_7+x_5x_6+x_4x_6+x_3x_5+x_2x_6+x_2x_5+x_1x_7+x_1x_6+x_0x_7+x_0x_6+x_0x_3+x_0x_1+x_6y_7+x_3y_6+x_3y_4+x_0y_4+x_7y_3+x_4y_3+x_3y_3+x_5y_2+x_4y_2+x_6y_1+x_4y_1+x_2y_1+x_1y_1+x_1y_0+x_5+x_0+y_4+y_2,$

$x_5x_7 + x_5x_6 + x_3x_7 + x_3x_4 + x_2x_6 + x_2x_4 + x_1x_4 + x_1x_3 + x_1x_2 + x_0x_6 + x_7y_7 + x_6y_7 + x_4y_7 + x_3y_7 + x_1y_6 + x_1y_4 + x_7y_3 + x_3y_3 + x_2y_3 + x_6y_2 + x_2y_2 + x_0y_2 + x_7y_1 + x_5y_1 + x_4y_1 + x_3y_1 + x_6y_0 + x_3y_0 + x_1y_0 + x_6 + x_3 + x_2 + y_4,$

$x_5x_7 + x_4x_6 + x_4x_5 + x_3x_5 + x_2x_7 + x_2x_4 + x_2x_3 + x_0x_4 + x_0x_1 + x_4y_6 + x_2y_6 + x_7y_5 + x_7y_4 + x_6y_4 + x_1y_4 + x_6y_3 + x_5y_3 + x_2y_3 + x_1y_3 + x_7y_2 + x_5y_2 + x_7y_1 + x_6y_1 + x_0y_1 + x_4y_0 + x_3y_0 + x_1y_0 + x_3 + x_1 + y_4 + y_3,$

$x_5x_7 + x_4x_6 + x_3x_7 + x_3x_5 + x_3x_4 + x_2x_4 + x_1x_6 + x_1x_3 + x_1x_2 + x_0x_7 + x_4y_7 + x_3y_7 + x_0y_7 + x_4y_6 + x_2y_6 + x_0y_6 + x_5y_5 + x_0y_5 + x_6y_4 + x_6y_3 + x_6y_2 + x_4y_2 + x_3y_2 + x_1y_2 + x_5y_1 + x_4y_1 + x_5y_0 + x_1y_0 + x_0y_0 + x_7 + x_5 + y_1 + y_0,$

$x_6x_7+x_4x_6+x_4x_5+x_2x_6+x_2x_5+x_2x_3+x_1x_7+x_1x_5+x_1x_4+x_1x_3+x_0x_2+x_1y_7+x_2y_6+x_1y_6+x_6y_5+x_2y_5+x_2y_4+x_3y_3+x_7y_2+x_3y_2+x_2y_2+x_7y_1+x_5y_1+x_1y_1+x_7+x_3+x_0+y_6+y_4+y_2+y_1,$

$x_7y_5 + x_7y_4 + x_2y_4 + x_7y_3 + x_6y_3 + x_0y_3 + x_5y_2 + x_4y_2 + x_0y_2 + x_7y_1 + x_3y_1 + x_2y_1 + x_0y_1 + x_7y_0 + x_5y_0 + x_4y_0 + x_2y_0 + x_1y_0 + x_2 + x_1 + x_0 + y_4 + y_2 + y_1 + 1,$

$x_5x_6 + x_3x_7 + x_3x_6 + x_2x_5 + x_2x_4 + x_1x_6 + x_1x_5 + x_1x_4 + x_1x_3 + x_1x_2 + x_0x_7 + x_0x_2 + x_0x_1 + x_5y_7 + x_1y_6 + x_4y_4 + x_2y_4 + x_0y_4 + x_5y_3 + x_4y_3 + x_1y_3 + x_0y_2 + x_5y_1 + x_2y_1 + x_1y_1 + x_3y_0 + x_0y_0 + x_6 + x_5 + x_1 + y_3 + y_2,$

$x_6y_7 + x_3y_7 + x_1y_7 + x_6y_6 + x_5y_6 + x_1y_6 + x_5y_5 + x_1y_5 + x_6y_4 + x_6y_3 + x_5y_3 + x_4y_3 + x_3y_2 + x_7y_1 + x_6y_1 + x_1y_1 + x_6y_0 + x_3y_0 + x_2y_0 + x_5 + y_7 + y_3 + y_2 + y_1 + y_0,$

$x_5x_6 + x_4x_6 + x_3x_6 + x_3x_5 + x_3x_4 + x_1x_7 + x_1x_5 + x_0x_3 + x_0x_2 + x_5y_7 + x_1y_6 + x_6y_5 + x_2y_5 + x_2y_4 + x_0y_4 + x_6y_2 + x_4y_2 + x_2y_2 + x_1y_2 + x_6y_1 + x_5y_1 + x_4y_1 + x_2y_1 + x_5y_0 + x_3y_0 + x_7 + x_6 + x_4 + x_0 + y_6 + y_4 + y_2 + 1,$

$x_5y_7 + x_4y_7 + x_3y_7 + x_1y_7 + x_0y_7 + x_7y_6 + x_4y_6 + x_7y_5 + x_3y_5 + x_7y_4 + x_4y_4 + x_3y_4 + x_7y_3 + x_6y_3 + x_4y_3 + x_3y_3 + x_1y_3 + x_6y_2 + x_5y_2 + x_2y_2 + x_2y_1 + x_7y_0 + x_3y_0 + x_2y_0 + x_1y_0 + y_6 + y_2 + y_0,$

$x_7y_7 + x_4y_7 + x_1y_7 + x_6y_6 + x_5y_6 + x_5y_5 + x_2y_3 + x_1y_3 + x_0y_3 + x_5y_2 + x_4y_2 + x_2y_2 + x_1y_2 + x_7y_1 + x_0y_1 + x_6y_0 + x_5y_0 + x_4y_0 + x_0y_0 + x_6 + x_2 + x_1 + x_0 + y_7,$

$x_7y_7 + x_4y_7 + x_2y_7 + x_4y_6 + x_6y_5 + x_5y_5 + x_1y_5 + x_6y_4 + x_4y_4 + x_7y_3 + x_6y_3 + x_6y_2 + x_5y_2 + x_0y_2 + x_6y_1 + x_2y_1 + x_6y_0 + x_4y_0 + x_5 + x_4 + x_3 + x_1 + x_0 + y_7 + y_4 + y_2,$

$x_4x_6 + x_4x_5 + x_3x_5 + x_3x_4 + x_2x_7 + x_2x_6 + x_2x_4 + x_2x_3 + x_1x_6 + x_1x_5 + x_1x_4 + x_1x_2 + x_0x_7 + x_0x_6 + x_0x_2 + x_0x_1 + x_0y_5 + x_6y_4 + x_3y_4 + x_0y_4 + x_6y_3 + x_3y_3 + x_2y_3 + x_6y_2 + x_5y_2 + x_4y_2 + x_0y_2 + x_7y_1 + x_5y_1 + x_0y_0 + x_7 + y_1 + 1,$

$x_6y_7 + y_6y_7 + x_7y_5 + y_5y_7 + x_7y_4 + y_4y_7 + x_7y_3 + x_6y_3 + x_2y_3 + x_1y_3 + y_3y_5 + y_3y_4 + x_7y_2 + x_6y_2 + x_0y_2 + y_2y_3 + x_4y_1 + x_3y_1 + x_2y_1 + x_1y_1 + x_0y_1 + y_1y_5 + y_1y_2 + x_7y_0 + x_1y_0 + y_0y_7 + y_0y_6 +$

$y_0y_3 + y_0y_1 + x_2 + y_5 + y_4,$

$x_4y_7 + x_3y_7 + x_1y_7 + x_5y_6 + x_4y_6 + x_1y_6 + x_4y_5 + x_0y_5 + x_7y_4 + x_6y_4 + x_4y_4 + x_2y_4 + x_1y_4 + x_5y_3 + x_4y_3 + x_3y_3 + x_0y_3 + x_3y_2 + x_0y_1 + x_6y_0 + x_1y_0 + x_6 + x_4 + x_2 + y_6 + y_2 + y_1,$

$x_5x_7+x_4x_7+x_2x_5+x_2x_3+x_1x_7+x_1x_5+x_0x_7+x_0x_6+x_0x_5+x_0x_4+x_0x_3+x_4y_5+x_6y_4+x_5y_3+x_4y_3+x_0y_3+x_7y_2+x_3y_2+x_2y_2+x_5y_1+x_3y_1+x_0y_1+x_5y_0+x_3y_0+x_0y_0+x_7+x_5+x_1+x_0+y_4+y_0+1,$

$x_7y_7 + x_4y_7 + x_0y_7 + x_6y_6 + x_5y_6 + x_0y_6 + x_0y_5 + x_6y_4 + x_5y_4 + x_3y_4 + x_7y_3 + x_6y_3 + x_0y_3 + x_7y_2 + x_3y_2 + x_2y_2 + x_6y_1 + x_3y_1 + x_1y_1 + x_5y_0 + x_4y_0 + x_3y_0 + x_1y_0 + x_3 + y_3,$

$x_4x_5+x_3x_7+x_3x_6+x_3x_4+x_2x_7+x_2x_5+x_2x_3+x_1x_6+x_1x_4+x_1x_3+x_0x_7+x_1y_7+x_2y_6+x_1y_6+x_6y_5+x_2y_5+x_4y_4+x_3y_4+x_7y_3+x_3y_2+x_5y_1+x_2y_1+x_0y_1+x_4y_0+x_3y_0+x_0y_0+x_3+y_3+y_2+y_1+y_0.$