





# Faster Quantum Algorithms for MQ2 and Applications

Quentin Edme<sup>a,1</sup>, Pierre-Alain Fouque<sup>2</sup>  and André Schrottenloher<sup>2</sup> 

<sup>1</sup> Independent researcher, France

<sup>2</sup> Univ Rennes, Inria, CNRS, IRISA, Rennes, France

**Abstract.** We study quantum algorithms for multivariate quadratic Boolean equation systems by focusing on their precise gate count. While better asymptotic algorithms are known, currently gate counts were only computed for exhaustive search (Schwabe and Westerbaan, SPACE 2016) and a variant of Grover’s search using preprocessing (Pring, WAIFI 2018). This limits the applicability of Boolean equation solving to cryptanalysis, which considers relatively small numbers of variables (from 40 to 200) and is concerned with the exact complexity of the solver.

In this paper, we introduce two new quantum algorithms. The first algorithm is an optimized quantum exhaustive search which amortizes the cost of polynomial evaluation at each quantum search iterate. The second algorithm adapts a method of Bouillaguet et al. (SOSA 2022) which proceeds by linearization of the system. In both cases, we implement the quantum circuits, study their complexity, and obtain significant improvements over previous results.

Next, we apply these new algorithms to the cryptanalysis of the block ciphers LowMC and RAIN in the single-data setting. By adapting attacks from Liu et al. (ToSC 2022) and Liu et al. (ToSC 2023) we obtain the first quantum cryptanalysis results on these ciphers.

**Keywords:** Quantum cryptanalysis · Boolean equation systems · Multivariate quadratic systems · Quantum search · LowMC · RAIN

## 1 Introduction

Solving multivariate equation systems in finite fields is a well-studied hard problem. Being intractable for quantum computers, it now underlies the design of many post-quantum public-key cryptosystems, including 10 signature designs which were submitted to the ongoing NIST post-quantum signature standardization process [NIS22].

In this paper, we will focus on the simpler case of *Boolean quadratic* equation systems, denoted the  $MQ_2$  problem. Besides being of importance for post-quantum cryptography, it is also useful in symmetric cryptography, as many algebraic attacks require to solve Boolean quadratic equations. This happens notably in designs which aim at minimizing the amount of non-linear operations, like LowMC [ARS<sup>+</sup>15]. As an example, some preimage attacks on reduced-round Keccak [WWF<sup>+</sup>21] and key-recovery attacks on some instances of LowMC [Din21a] used Boolean equation solving as a building block. Even on more recent primitives such as AIM [KHS<sup>+</sup>23] and RAIN [DKR<sup>+</sup>22], which are defined on larger finite fields  $\mathbb{F}_{2^n}$ , attacks based on Boolean equation solving have been given in [LMØM23], which rely on alternative representations of the ciphers with low algebraic degree, combined with the isomorphism between the large finite field  $\mathbb{F}_{2^n}$  and  $\mathbb{F}_2^n$ .

E-mail: [pierre-alain.fouque@irisa.fr](mailto:pierre-alain.fouque@irisa.fr) (Pierre-Alain Fouque), [andre.schrottenloher@inria.fr](mailto:andre.schrottenloher@inria.fr) (André Schrottenloher)

<sup>a</sup>This work was done while the author was at Orange Labs, Caen



Among the many algorithms available to solve Boolean quadratic systems, one can distinguish the *algebraic approaches* from the *polynomial method*. Algebraic approaches include computing Gröbner bases of polynomial ideals [Fau02]. The polynomial method [LPT<sup>+</sup>17] offers nowadays the best asymptotic complexity for the  $MQ_2$  problem at  $\mathcal{O}(2^{0.6943n})$  bit operations [Din21b].

For small number of variables, simpler algorithms such as the fast exhaustive search (FES) [BCC<sup>+</sup>10], the Crossbred algorithm [JV17] and the algorithm of [BDT22] (which all belong to the family of algebraic approaches) appear to be quite efficient, and are often used in cryptanalysis.

While the complexity of  $MQ_2$  is well studied in the classical setting, in the quantum setting, the available set of algorithms is quite sparse. Schwabe and Westerbaan [SW16] computed the cost of Grover’s exhaustive search on quadratic systems, which was improved by Pring [Pri18] using a precomputation trade-off. Concurrently, improved asymptotic complexities were given by Faugère et al. [FHK<sup>+</sup>17] and Bernstein and Yang [BY18] using combinations of classical techniques (respectively BooleanSolve [BFSS13] and XL [CKPS00]) with Grover’s search. However, due to large polynomial factors and the lack of concrete estimates, these algorithms do not seem applicable in cryptanalysis for small numbers of variables.

Despite this lack of tools, there is a clear motivation to study quantum attacks on the aforementioned primitives such as LowMC [ARS<sup>+</sup>15], AIM [KHS<sup>+</sup>23] and RAIN [DKR<sup>+</sup>22]. These primitives are used in post-quantum signature schemes based on the MPC-in-the-head paradigm. By reducing the security to a symmetric cryptography problem (e.g., finding the key of a LowMC cipher given a single plaintext-ciphertext pair), it is expected that a quantum attacker can only obtain up to a quadratic speedup on classical attacks, using Grover’s quantum search [Gro96] and other algorithms based on it. However, in order to give a concrete quantum security level for these schemes, a more precise study of quantum attacks is necessary.

**Contributions.** In this paper, we study the cost of quantum algorithms for the  $MQ_2$  problem in the (logical) quantum circuit model, where an algorithm is decomposed as a sequence of gates. Our primary optimization target is the (exact) number of gates, and the depth of the circuit.

We consider two algorithms based on quantum search. Our first algorithm is a quantum variant of exhaustive search. Our second algorithm is a quantum variant of the algorithm of Bouillaguet, Delaplace and Trimoska (BDT) [BDT22]. They will be denoted as “Quantum Fast Exhaustive Search (FES)” and “Quantum BDT” respectively.

A comparison between these algorithms and the previous ones is given in Table 1, in which we observe that the gate count decreases by a factor more than  $2^{10}$ . We also obtain a similar reduction in depth; meanwhile, the number of qubits increases with respect to exhaustive search [SW16], but at most ten-fold (a more precise comparison is given in Table 2). This means that for  $n$  in the hundreds, the number of (logical) qubits is in the thousands, which is similar to Grover’s exhaustive search on AES [JNRV20], when the circuit is optimized for gate count and depth.

Our Quantum FES and Quantum BDT algorithms are obtained by combining several layers of quantum search, with classical reversible sub-circuits performing operations on polynomials and / or linear algebra. We use a framework developed in [SS24] to handle the analysis of their complexity and probability of success. Besides an asymptotic analysis, we use numerical optimization to determine the number of iterates in these searches. We implement the required classical sub-circuits using Qiskit [Qis23], obtaining precise counts of gates, qubits and circuit depth. Our code is available at <https://gitlab.inria.fr/capsule/quantum-algorithms-for-mq2>.

**Table 1:** Comparison of Clifford+T gate counts to solve Boolean quadratic systems with  $m = n$  equations (in  $\log_2$ , rounded). The parameters  $n$  and corresponding gate counts for [SW16] and [Pri18] are taken from Table 1 in [Pri18]. (\*) gate count estimates obtained using the upper bounds of Equation 12 and Equation 17.

$n$	Exhaustive search [SW16]	Preprocessing [Pri18]	Quantum FES Section 4	Quantum BDT Section 5
117	80.99	78.38	73.2	69.9
209	129.4	126.26	119.7	114.6
457	256.71	252.93	244.3 (*)	236.3 (*)

**Applications.** Our algorithms do not challenge the parameter choices of post-quantum signature designs based on multivariate quadratic equations (e.g., [BFR24]). Indeed, the parameters of these designs will typically take asymptotic complexities and under-estimate the additional factors. Most often, they also use a larger field size on which our algorithms either do not apply, or become inefficient.

We give several applications in symmetric cryptanalysis, with indirect impact on the security of post-quantum signature schemes. We focus on the ciphers LowMC and RAIN, when used inside an MPC-in-the-head signature scheme (e.g., Picnic [CDG+20] or Rainier [DKR+22]). In this scenario, one must recover the secret key from a single known plaintext-ciphertext pair.

On LowMC, we adapt the linearization strategy of Liu et al. [LMSI22] and use the Quantum BDT algorithm as a black-box. In particular, we show that 3-round variants of LowMC with full S-Box layers can be attacked faster than exhaustive search of the key with Grover’s algorithm, whether in total gate count or circuit depth. Previously, only exhaustive key search had been studied by Jaques et al. [JNRV20].

On 2-round RAIN, we apply an attack from Liu et al. [LMØM23] which relies on a simplified algebraic representation of the cipher, and also reduces the key-recovery problem to a quadratic Boolean system.

For both applications, the complexities are much better than what one would obtain with previous algorithms [SW16, Pri18], and this is crucial to obtain an actual speedup with respect to Grover’s search of the key.

**Outline.** The paper is organized as follows. Section 2 gives some notation and background on quantum computing and the  $MQ_2$  problem. In Section 3, we specialize the quantum search framework of [SS24] to the case of a two-level search problem, which is the only one needed in the paper. The generic analysis performed here is subsequently reused for our two algorithms, presented in Section 4 (Quantum FES) and Section 5 (Quantum BDT) respectively. The applications to LowMC and RAIN are given in Section 6 and Section 7 respectively.

## 2 Preliminaries

In this section, we introduce our notation, preliminaries on quantum computing and on the number of solutions of polynomial systems.

### 2.1 Problem and Notation

We are interested in solving *multivariate Boolean quadratic* systems of equations, a problem that is named “ $MQ_2$ ” and formulated as follows.

**Problem 1** (Multivariate Boolean quadratic ( $MQ_2$ )). *Given  $m$  quadratic Boolean functions  $f_0, \dots, f_{m-1}$  on  $n$  variables, find  $\mathbf{x} = (x_0, \dots, x_{n-1})$  such that  $\forall i, f_i(\mathbf{x}) = 0$ .*

Most often we consider a “planted” version of the problem in which a system with a solution  $\mathbf{x}_0$  is generated at random. That is, we sample random polynomials  $f_i$  such that  $f_i(\mathbf{x}_0) = 0$ . Then, as we show below in Subsection 2.3, if  $m \geq n$  the system admits no other solution with constant probability. In this paper we work only with such systems.

**Notation.** Since we work exclusively in  $\mathbb{F}_2$  and  $\mathbb{F}_{2^n}$  Boolean addition (XOR) will be denoted  $+$ , and this should be clear from context. We use bold lowercase letters ( $\mathbf{x}, \mathbf{y}, \mathbf{z}$ ) for Boolean vectors,  $|\mathbf{x}|$  for the length of  $\mathbf{x}$ , and  $\mathbf{x}||\mathbf{y}$  for the concatenation of two vectors.

## 2.2 Quantum Gates and Cost Metrics

We refer to [NC02] for an introduction to quantum computing notions such as quantum states and unitary evolution. The quantum algorithms studied in this paper are written in the quantum circuit model, in which unitary operators are decomposed in elementary quantum gates. Our basic gate set is the following: Hadamard gate (H), NOT (or X), Controlled-NOT (CX), Toffoli (CCX).

Indeed, the X + CX + CCX gate set is universal for classical reversible computing, and the addition of the Hadamard gate is sufficient to define quantum algorithms based on Grover’s search [Gro96]. We name this set the “Clifford+CCX” gate set, because H, X and CX belong to the Clifford set.

In this paper, our primary cost metric is the Clifford+CCX gate count, where CCX gates are also considered more costly than Clifford. We also give counts of space (number of qubits) and circuit depth (i.e., its runtime if gates can be applied concurrently). The Clifford+T gate set has also been widely used to benchmark the cost of quantum circuits for cryptanalysis [Pri18, JNRV20]. In order to convert the Clifford+CCX estimates to the Clifford+T gate set, we simply decompose generically all CCX gates into 17 gates (2 Hadamard, 7  $T$  and  $T^\dagger$ -gates, 8 Clifford gates) following [NC02].

## 2.3 Number of Solutions of Random Systems

The algorithms that we consider solve random systems of quadratic polynomial equations. Inside the algorithms, we will consider many sub-systems, which are expected to behave like randomly drawn systems (planted or not).

Following [Mas98], we expect that the number of solutions of a random Boolean equation system with  $m$  equations in  $n$  variables ( $m \leq n$ ), which is consistent, is approximated by a Poisson distribution of parameter  $\lambda = 2^{n-m}$ . That is, the probability that there are  $k$  solutions is approximately  $e^{-\lambda} \lambda^k / k!$ .

With numerical experiments, we observed that this statistic is very good even for small  $m$  and  $n$ , and that it still holds when  $m = n$  and  $m > n$ , leading to very good approximations of the probability that a random overdefined system admits a solution.

Consequently, let  $X$  be the number of solutions for a randomly drawn system of  $m$  equations with  $n$  variables.

- If  $m = n$  ( $\lambda = 1$ ) the system has no solution with probability  $e^{-1}$ , and using the Poisson cumulative density function we compute:

$$\Pr(X > 2) \leq 0.08 .$$

- If  $m > n$  ( $\lambda < 1$ ), the system has no solution most of the time:

$$\begin{cases} \Pr(X > 0) \leq 0.221 \text{ for } \lambda = 0.25 \text{ (i.e., } m - n = 2) \\ \Pr(X > 0) \leq 0.061 \text{ for } \lambda = 2^{-4} \text{ (i.e., } m - n = 4) . \end{cases} \quad (1)$$

## 2.4 Classical Algorithms for MQ2

The problem of solving multivariate polynomial systems on finite fields has been extensively studied. The most efficient algorithms in practice rely on computing a Gröbner basis of the ideal generated by the polynomials (e.g., the F4 or F5 algorithm [Fau02]). These Gröbner basis methods overtook the previous XL approach [CKPS00], even for overdefined systems of equations (see e.g. [AFI<sup>+</sup>04]).

In this paper we are interested in random Boolean systems (without any particular structure) and not necessarily overdefined. In that case, exhaustive search remains quite competitive for small parameters. Using Gray codes [BCC<sup>+</sup>10], one enumerates all values for a Boolean polynomial in  $n$  variables of degree  $d$  in about  $\mathcal{O}(d2^n)$  bit operations. Another efficient algorithm in practice is the Crossbred algorithm [JV17]; however, its time complexity is challenging to estimate in general. The most simple case of application of the Crossbred algorithm is the BDT algorithm [BDT22], a simplified algebraic approach which offers an asymptotic gain of order  $2^{\sqrt{m}}$  over exhaustive search.

Asymptotic complexities of the form  $2^{\alpha n}$  with  $\alpha < 1$ , guaranteed for random systems, are obtained by two families of algorithms: algebraic approaches (similar to those mentioned above) and the polynomial method. The former includes for example the BooleanSolve algorithm [BFSS13]. The latter [LPT<sup>+</sup>17] yields today the best known results with Dinur’s algorithm [Din21b], which solves  $MQ_2$  in time  $\mathcal{O}(2^{0.6943n})$  and also achieves speedups for larger  $d$ . Dinur also provided a version of this algorithm with an accurate complexity estimate at  $n^2 2^{0.815n}$  bit operations [Din21a], which has applications in cryptanalysis. However, no quantum version of this algorithm exists.

## 2.5 Quantum Algorithms for MQ2

In the quantum setting, Schwabe and Westerbaan [SW16] studied the cost to run Grover’s exhaustive search of the solution, which is of  $\mathcal{O}(n^2 m 2^{n/2})$  quantum gates when the system has  $m$  equations, since  $m$  quadratic polynomials need to be evaluated at each search iterate. They noticed that Grover’s search can run with an especially small number of qubits (around  $n + \log_2 m$  in the most space-optimized version).

Later, Pring used a preprocessing method to improve this gate count [Pri18]. Following [Pri18, Section 4.4], the method consists in performing a Grover’s search on  $n - b$  variables, followed by a classical exhaustive search on the  $b$  remaining variables. The classical search can be performed quite efficiently, in a way similar to the classical Fast Exhaustive Search. This gives an asymptotic gate count of:

$$2^{(n-b)/2} (m(n-b)^2 + 2^b m(n-b) + n) \ .$$

(See Eq. (22) and (23) in [Pri18]). Here, the term  $2^b m(n-b)$  is the classical exhaustive search term, while  $m(n-b)^2$  checks if the obtained solution is correct for the entire system. Optimizing this gives  $b = \log_2 n$  and a complexity of order  $\mathcal{O}(mn^{3/2} 2^{n/2})$ .

The situation in our paper is quite similar, though we obtain improved complexities. Indeed, both in Section 4 and Section 5, we guess the variables in two steps, and the number of variables in the second step is typically logarithmic. An important difference with [Pri18] is that we use quantum search in both layers.

Quantum variants of the BooleanSolve [FHK<sup>+</sup>17] and the XL algorithms [BY18] are known to improve asymptotically over exhaustive search, though their gate count on small instances has not been studied. The quantum Booleansolve [FHK<sup>+</sup>17] is quite relevant for our context since it was also analyzed from the point of view of total gate count, and has the same structure of a hybrid algebraic algorithm. It performs an exhaustive (quantum) search on a subset of variables, and then a sparse linear system-solving (a situation similar to Section 5).

We can also mention the study of Chen et al. [CHR<sup>+</sup>18], who estimated the complexities of quantum algorithms that combine classical algebraic approaches with Grover’s search, such as BooleanSolve [BFSS13] and the Crossbred algorithm [JV17]. The purpose of their approach was to estimate safe parameters for a signature scheme, and so, they used a conservative estimate (e.g., simplifying the cost of linear algebra) which cannot be reused as is for cryptanalysis.

In Table 2, we summarize the asymptotic complexities of known quantum algorithms, compared with ours.

**Table 2:** Quantum algorithms solving  $MQ_2$ , assuming  $m \geq n$  and a single solution.  $\mathcal{O}$  notation is omitted. (\*) GroverXL is a distributed algorithm: it computes in parallel on  $\tilde{\mathcal{O}}(2^{0.01467n})$  qubits. (\*\*) Due to non-negligible lower-order terms, these formulas are far from the actual number of qubits for small values of  $n$ .

Reference	Method	Time (gates)	Space (qubits)
[SW16]	Exhaustive search	$n^2 m 2^{n/2}$	$n + \log_2 m$
[Pri18]	Preprocessing	$mn^{3/2} 2^{n/2}$	$n + m$
[FHK <sup>+</sup> 17]	BooleanSolve	$\tilde{\mathcal{O}}(2^{0.462n})$	Not studied
[BY18]	XL algorithm	$\tilde{\mathcal{O}}(2^{0.45743n})$ (*)	$\tilde{\mathcal{O}}(2^{0.01467n})$ (*)
Section 4	FES	$(\log^3 n) 2^{n/2}$	$3n + \mathcal{O}((\log n)^2)$ (**)
Section 5	BDT	$m^{3/2} 2^{(n-\sqrt{2m})/2}$	$3n + 7m + o(n+m)$ (**)

### 3 Quantum Search Framework

The quantum algorithms that we design in this paper are specific instances of the *backtracking search* framework described in [SS24], which is built by nesting quantum search algorithms. In this section, we recall the results of [SS24], for the case of two levels of nesting (i.e.,  $\ell = 2$  in [SS24]). We restrict to two levels because we do not require more for the algorithms presented in this paper.

#### 3.1 Preliminaries: Quantum Amplitude Amplification

Quantum Amplitude Amplification (QAA) [BHMT02] is a generic process to increase the success probability of any quantum algorithm. Let  $\mathcal{A}$  be a quantum algorithm that, on input  $|0^n\rangle$ , returns any superposition of  $n$ -bit strings. Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be a Boolean function that recognizes certain of these  $n$ -bit strings as “good”, and let  $p$  be the *success probability* of  $\mathcal{A}$ , i.e., the probability that upon measurement of  $\mathcal{A}|0^n\rangle$ , the result  $x$  satisfies  $f(x) = 1$ . QAA relies only on  $\mathcal{A}$  and a *phase oracle*  $O_f : |x\rangle \mapsto (-1)^{f(x)} |x\rangle$ .

We define the *QAA iterate* as the unitary:

$$Q = -\mathcal{A}O_0\mathcal{A}^\dagger O_f, \quad (2)$$

where  $O_0|x\rangle = (-1)^{\delta_{x0}}|x\rangle$ , where  $\delta_{x0}$  is the Kronecker delta (1 if and only if  $x = 0$ ).

Let  $|\psi_G\rangle$  be the (normalized) part of the output of  $\mathcal{A}$  on which  $f$  is one, and  $|\psi_B\rangle$  be the other part. They are orthogonal (as they are supported by disjoint sets of basis vectors) and by definition:

$$\mathcal{A}|0^n\rangle = \sqrt{p}|\psi_G\rangle + \sqrt{1-p}|\psi_B\rangle. \quad (3)$$

The following is shown by Brassard et al. [BHMT02]:

$$\forall k \geq 0, Q^k \mathcal{A}|0^n\rangle = \sin[(2k+1)\arcsin\sqrt{p}]|\psi_G\rangle + \cos[(2k+1)\arcsin\sqrt{p}]|\psi_B\rangle. \quad (4)$$

The analysis of the algorithms detailed in this section follows entirely from this equality. Note that a priori,  $O_0$  needs to be applied on the entire state. However, qubits which are 0 when  $O_0$  is applied can be left out. The implementation of  $O_0$  relies on a multi-controlled X gate, which we implement from CCX gates using only a single ancilla qubit, thanks to a method of Gidney [Gid15] (detailed in our code). This gate is then converted to a phase flip using Hadamard gates.

**Lemma 1.** *There exists a circuit mapping:*

$$|x_0, \dots, x_{\ell-1}, b, 0\rangle \mapsto \left| x_0, \dots, x_{\ell-1}, b \oplus \prod_i x_i, 0 \right\rangle ,$$

using 1 CX gate if  $\ell = 1$ , 1 CCX gate if  $\ell = 2$ , 3 CCX gates if  $\ell = 3$ , 6 CCX gates if  $\ell = 4$ , and  $6(n-3) - 2((n+1) \bmod 2)$  CCX gates otherwise.

### 3.2 Definition of the Problem

Let  $C_1 = \{0, 1\}^{n_1}$ ,  $C_2 = \{0, 1\}^{n_2}$  be *choice sets*. Let  $W = \{0, 1\}^w$  be the *workspace* and  $F = F' = \{0, 1\}$  be *flags*. We will consider algorithms that act on  $C_1 \times C_2 \times W \times F \times F' = \{0, 1\}^{n_1+n_2+w+2}$ . By *register*, we mean a subset of the bits corresponding either to  $C_1, C_2, W, F$  or  $F'$ .

For ease of notation, we use the notation  $|_S$  for projecting the current state on the register  $S$  (resp. a subset of registers), for example  $s|_F$  is a single-bit flag.

We consider a triple of reversible algorithms  $A_1, A_2, D$  which form the basis for a *two-level backtracking search* which we will define later on. They have the following properties:

- $A_1$ : acts on  $C_1 \times W$  and modifies only  $W$
- $A_2$ : acts on  $C_1 \times C_2 \times W \times F$  and modifies only  $W$  and  $F$
- $D$ : acts on  $C_1 \times C_2 \times W \times F'$  and modifies only  $W$  and  $F'$ .

We are interested in finding a choice  $c_1, c_2$  which passes two tests:  $A_2$  (the test output is in the flag  $F$ ) and  $D$  (the test output is in the flag  $F'$ ). We assume that there is a *single solution*  $c_1^g, c_2^g$  such that:

$$D \circ A_2 \circ A_1(c_1^g, c_2^g, 0^w, 0, 0)|_{F, F'} = 1, 1 , \quad (5)$$

and our goal is to return it. Furthermore, we assume that  $c_2^g$  is the only value of  $c_2$  such that  $(c_1^g, c_2)$  passes the first test  $A_2$ . This will simplify the structure of the algorithm with respect to the more generic setting of [SS24].

### 3.3 Classical Backtracking Algorithm

One can find  $c_1^g, c_2^g$  by several nested search loops, as shown in Algorithm 1. The loops are iterated for a fixed number of times ( $k_1, k_2'$ ) and sample choices at random. As we might miss the solution, the success probability is not 1, and it depends on  $k_1, k_2'$ .

### 3.4 Description of the Quantum Algorithm

Intuitively, the classical ‘‘Repeat’’ loops with  $k_1, k_2'$  iterates in Algorithm 1 can be replaced by iterations of QAA operators. The framework described in [SS24] applies such a transformation, obtaining a quantum algorithm that solves the search problem.

In the quantum setting, we replace  $A_1, A_2$  and  $D$  by quantum circuits  $\mathcal{A}_1, \mathcal{A}_2, \mathcal{D}$  with the same constraints (acting reversibly and only modifying some subset of the registers).

---

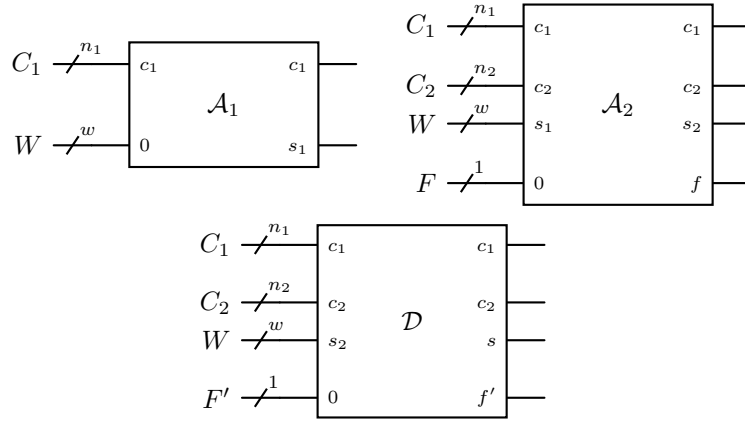
**Algorithm 1** Classical two-level backtracking search.

---

**Registers:**  $C_1, C_2, W, F, F'$  (initialized to zeroes)

**Returns:** the good choice  $c_1^g, c_2^g$

- 1: **Repeat**  $k_1$  **times**
  - 2:   Choose a random  $c_1$
  - 3:   Compute  $A_1$
  - 4:   **Repeat**  $k_2'$  **times**
  - 5:     Choose a random  $c_2$
  - 6:     Compute  $A_2$ . If  $f = 1$ , **break**
  - 7:   **EndRepeat**
  - 8:   Compute  $D$ . If  $f' = 1$ , **break**
  - 9: **EndRepeat**
- 



**Figure 1:** Quantum circuits for basic components in the two-level backtracking search.

They are represented in Figure 1. Here  $s_1$  denotes the state of the workspace after  $A_1$ ,  $s_2$  after  $A_2$ ,  $s$  after  $D$ .

Then, one constructs three algorithms using QAA, named  $\mathcal{A}'_2$ ,  $\mathcal{B}_2$  and  $\mathcal{B}_1$ .

**Algorithm  $\mathcal{A}'_2$  (Algorithm 2):**  $\mathcal{A}'_2$  acts on the registers  $C_1, C_2, W, F$  and modifies  $C_2, W, F$ . It assumes that  $C_2$  and  $F$  contain 0 at the start, and  $C_1, W$  contain a choice  $c_1$  and a corresponding workspace state  $s_1$ . Intuitively,  $\mathcal{A}'_2$  is responsible for finding a  $c_2 \in C_2$  such that  $A_2(c_1^g, c_2, w, 0)|_F = 1$  (that is,  $c_2^g$ ).

**Algorithm  $\mathcal{B}_2$ :**  $\mathcal{B}_2$  acts on the registers  $C_1, C_2, W, F, F'$  and modifies  $C_2, W, F, F'$ . It computes  $\mathcal{A}'_2$  followed by  $D$ .

**Algorithm  $\mathcal{B}_1$  (Algorithm 3):**  $\mathcal{B}_1$  acts on all registers. It is a QAA in which the amplified algorithm is  $\mathcal{B}_2 \circ \mathcal{A}_1$  and the test simply reads the flags.

### 3.5 Success Probability of the Quantum Algorithm

On input  $|0\rangle$ , running  $\mathcal{B}_1$ , measuring and projecting on  $C_1, C_2$  gives the good choice  $(c_1^g, c_2^g)$  with some *probability of success*. This probability can be expressed by a closed formula of  $k_1, k_2', n_1, n_2$  (Lemma 3 in [SS24]). We detail here how this formula is obtained in the specific case of the two-level search.



---

**Algorithm 2** Definition of  $\mathcal{A}'_2$ . Steps 3 to 5 are the test in the QAA, where the amplified algorithm is simply a Hadamard transform.

---

**Registers:**  $C_1, C_2, W, F, F'$   
**Modifies:**  $C_2, W, F$

- 1: Apply a Hadamard transform  $H$  on  $C_2$
- 2: **Repeat**  $k'_2$  **times**
- 3:     Compute  $\mathcal{A}_2$  ▷ Updates the workspace
- 4:     Flip the phase if  $F$  contains 1
- 5:     Uncompute  $\mathcal{A}_2$  ▷ Restores the workspace
- 6:     Apply  $H$ ,  $-O_0$  and  $H$  on  $C_2$
- 7: **EndRepeat**
- 8: Compute  $\mathcal{A}_2$

---

**Algorithm 3** Definition of  $\mathcal{B}_1$ .

---

**Registers:**  $C_1, C_2, W, F, F'$   
**Modifies:**  $C_1, C_2, W, F, F'$

- 1: Apply  $H$  on  $C_1$
- 2: Compute  $\mathcal{A}_1$
- 3: Compute  $\mathcal{B}_2 = \mathcal{D} \circ \mathcal{A}'_2$
- 4: **Repeat**  $k_1$  **times**
- 5:     Flip the phase if  $F$  and  $F'$  contain 1
- 6:     Uncompute  $\mathcal{B}_2$  and  $\mathcal{A}_1$
- 7:     Apply  $H$  on  $C_1$
- 8:     Apply  $-O_0$  on  $(C_1, C_2)$   
▷ We only need to apply  $O_0$  to  $C_1$  and  $C_2$  instead of the entire workspace  $C_1, C_2, W, F, F'$  because at this point of the algorithm,  $W, F, F'$  are always 0: they have been uncomputed by  $\mathcal{B}_2$  and  $\mathcal{A}_1$  respectively.
- 9:     Apply  $H$  on  $C_1$
- 10:     Compute  $\mathcal{A}_1$  and  $\mathcal{B}_2$
- 11: **EndRepeat**

---

**Success in  $\mathcal{A}'_2$ .** Consider  $\mathcal{A}'_2$  starting from  $|c_1^g, 0, s_1^g, 0, 0\rangle$ , where  $s_1^g$  is the workspace state after running  $\mathcal{A}_1$  on  $c_1^g$ .

Since the amplified algorithm merely samples  $c_2 \in C_2$  at random and computes  $\mathcal{A}_2$ , its probability of success is  $2^{-n_2}$ . By Equation 4, the probability of success of  $\mathcal{A}'_2$  is:

$$\nu_2'^2 := \sin^2((2k'_2 + 1) \arcsin 2^{-n_2/2}) .$$

This is the probability to measure an output of the form  $|c_1^g\rangle |c_2^g\rangle |s_2\rangle |1, 0\rangle$  if we run  $\mathcal{A}'_2 |c_1^g\rangle |0\rangle |s_1^g\rangle |0, 0\rangle$  (where  $s_2$  is the updated workspace corresponding to  $c_2$ ).

**Success in  $\mathcal{B}_2$ .** Consider  $\mathcal{B}_2 = \mathcal{D} \circ \mathcal{A}'_2$  starting from  $|c_1^g, 0, s_1^g, 0, 0\rangle$ . Its probability of success is the probability to measure  $|c_1^g\rangle |c_2^g\rangle |s^g\rangle |1, 1\rangle$  in the output, which is  $\nu_2'^2$  as given above. Starting from  $c_1 \neq c_1^g$ , we cannot measure 1, 1 in the flags, so the probability of success is 0.

**Success in  $\mathcal{B}_1$ .** Consider  $\mathcal{B}_2 \circ \mathcal{A}_1 \circ H$  starting from  $|0, 0, 0, 0, 0\rangle$ . The probability of success of this algorithm is:

$$2^{-n_1} \nu_2'^2 := 2^{-n_1} \sin^2((2k'_2 + 1) \arcsin 2^{-n_2/2}) ,$$

which corresponds to finding the right  $c_1$ , then succeeding in  $\mathcal{B}_2$ .

By Equation 4, the probability of success of  $\mathcal{B}_1$  is:

$$\nu^2 := \sin^2 \left[ (2k_1 + 1) \arcsin \left[ 2^{-\frac{n_1}{2}} \sin((2k'_2 + 1) \arcsin 2^{-\frac{n_2}{2}}) \right] \right]. \quad (6)$$

### 3.6 Complexity of the Quantum Algorithm

The number of qubits of  $\mathcal{B}_1$  is  $n_1 + n_2 + w + 3$  qubits (work qubits used locally are also included in the workspace). Let  $G$  be a cost metric such as the Clifford+CCX gate count (but we can also adapt it to be the depth). We assume given an implementation of  $O_0$  that for  $n$  bits, has cost  $G_0(n)$ . Then the costs of the algorithms can be expressed as:

$$\begin{cases} G(\mathcal{A}'_2) = (2k'_2 + 1)(G(\mathcal{A}_2) + n_2 G(H)) + k'_2 G_0(n_2) + k'_2 G(\text{CZ}) \\ G(\mathcal{B}_2) = G(\mathcal{A}'_2) + G(\mathcal{D}) \\ G(\mathcal{B}_1) = (2k_1 + 1)(G(\mathcal{A}_1) + n_1 G(H) + G(\mathcal{B}_2)) + k_1 G_0(n_1 + n_2) + k_1 G(\text{CCZ}) . \end{cases}$$

And the complexity of  $G(\mathcal{B}_1)$  is:

$$\begin{aligned} G(\mathcal{B}_1) = & G(\mathcal{A}_1) \left[ 2k_1 + 1 \right] + G(\mathcal{D}) \left[ (2k_1 + 1) \right] + G(\mathcal{A}_2) \left[ (2k_1 + 1)(2k'_2 + 1) \right] \\ & + G(H) \left[ (2k_2 + 1)(n_1 + n_2(2k'_2 + 1)) \right] + G_0(n_2) \left[ k'_2(2k_1 + 1) \right] + G_0(n_1 + n_2) \left[ k_1 \right] \\ & + G(\text{CZ}) \left[ k'_2(2k_1 + 1) \right] + G(\text{CCZ}) \left[ k_1 \right] . \quad (7) \end{aligned}$$

A decomposition of the algorithm using the sub-circuits  $\mathcal{A}_1, \mathcal{A}_2$  and  $\mathcal{D}$  is given in Figure 2. Here we highlight some intermediate states which are always 0.

**Choice of Iteration Numbers.** In order to guarantee  $\nu^2 = \mathcal{O}(1)$  in Equation 6, we need  $k'_2 = \mathcal{O}(2^{n_2/2})$  and  $k_1 = \mathcal{O}(2^{n_1/2})$ . Furthermore, the complexities of  $\mathcal{A}_1, \mathcal{D}$  and  $\mathcal{A}_2$  will always dominate in Equation 7, which we can simplify into:

$$G(\mathcal{B}_1) = \mathcal{O} \left( 2^{n_1/2} (G(\mathcal{A}_1) + G(\mathcal{D})) + 2^{(n_1+n_2)/2} G(\mathcal{A}_2) \right) . \quad (8)$$

In practice, we determine  $k_1$  and  $k'_2$  by optimizing numerically the quantity  $G(\mathcal{B}_1)/\nu^8$ . This gives typically a large probability of success ( $\nu^2 > 0.9$ ) and minimizes the complexity at the same time.

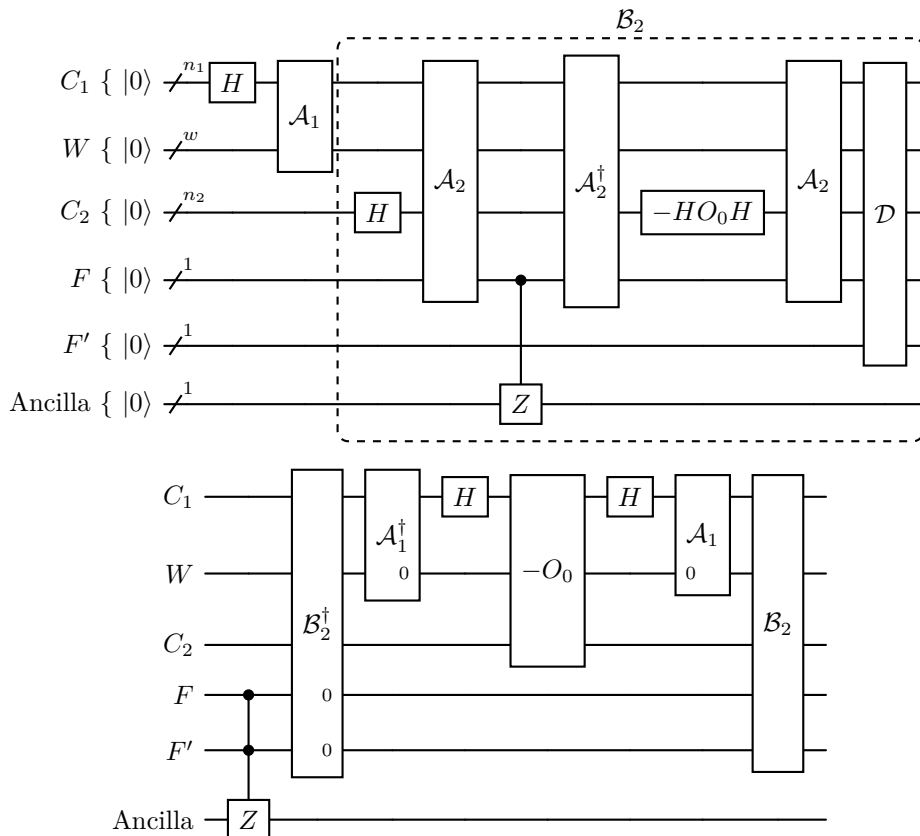
## 4 Fast Quantum Exhaustive Search

Our first algorithm is a quantum *fast exhaustive search* (FES). It differs significantly from the classical one [BCC<sup>+</sup>10], which explores its search space using Gray codes to minimize the time to evaluate a new point. Since this technique is inherently sequential, it does not admit a simple equivalent in the quantum setting. Instead, we will guess the variables in two steps.

### 4.1 Overview

Let  $f_0, \dots, f_{m-1}$  be  $m$  polynomials in variables  $\mathbf{x} := (x_0, \dots, x_{n-1})$ ; we consider the system  $\forall i, f_i(\mathbf{x}) = 0$ . When assigning a value  $v$  to a variable  $x$  in  $f$ , we obtain a *partially evaluated* polynomial with one variable less, denoted  $f(x = v)$ . We extend this notation to vectors, e.g.,  $f(\mathbf{x} = \mathbf{v})$ . We assume that there is a single solution.

Let  $\ell$  and  $n_1 + n_2 = n$  be integer parameters that we will choose later. Let  $\mathbf{x} = \mathbf{x}_2 \parallel \mathbf{x}_1$  where  $|\mathbf{x}_1| = n_1$  and  $|\mathbf{x}_2| = n_2$ . Our starting point is Algorithm 4.



**Figure 2:** Decomposition of  $\mathcal{B}_1$  when  $k_1 = k'_2 = 1$  (one iteration in  $\mathcal{A}'_2$ , one iteration in  $\mathcal{B}_1$ ). The sub-algorithm  $\mathcal{B}_2$  is decomposed only once in this drawing.

This is an instance of [Algorithm 1](#) where the first choice set (for  $\mathbf{v}_1$ ) is  $\{0, 1\}^{n_1}$  and the second choice set (for  $\mathbf{v}_2$ ) is  $\{0, 1\}^{n_2}$ . Furthermore:

- $A_1$  evaluates partially  $\ell$  polynomials in  $n_1$  variables. Thus, it has a complexity of  $\mathcal{O}(\ell \cdot (n_2(n - n_2) + (n - n_2)^2)) = \mathcal{O}(\ell n n_1)$  (we refer to [Equation 10](#) for the expression of the coefficients in the partial evaluation).
- $W$  stores the coefficients of  $f'_i$ , which are polynomials in  $n_2$  variables  $\mathbf{x}_2$ . As we will notice below, for each  $f'_i$  only  $n_2 + 1$  coefficients need to be stored (the coefficients of  $\mathbf{x}_2$ ), as the coefficients of the quadratic terms remain the same as in  $f_i$ . So  $W$  is of size  $\mathcal{O}(\ell n_2)$  bits.
- $A_2$  evaluates  $\ell$  polynomials in  $n_2$  variables and checks if they are 0. Thus, it has a complexity  $\mathcal{O}(n_2^2 \ell)$ .
- $D$  checks if the whole system is satisfied. There remains  $n - \ell$  polynomials in  $n$  variables to evaluate, so its complexity is  $\mathcal{O}((n - \ell)n^2)$ .

**Choice of Parameters.** We take  $\ell = n_2 + 4$ . The idea is to minimize the number of values for  $\mathbf{x}_2$  which pass the test  $A_2$ . In particular, when taking the good choice of variables, the discussion of [Subsection 2.3](#) implies that with probability 0.061, no other value passes the test. The success probability and complexity of the algorithm are then given by [Equation 6](#) and [Equation 7](#), respectively.

---

**Algorithm 4** A classical algorithm for exhaustive search.

---

- Input:** polynomials  $f_0, \dots, f_{m-1}$  in  $\mathbf{x}$
- 1: Choose a value  $\mathbf{v}_1$   $\triangleright$  First choice: the number of iterations of this step corresponds to  $k_1$  in Algorithm 1
  - 2: Evaluate  $f_0, \dots, f_{\ell-1}$  partially on  $\mathbf{x}_1$ : obtain  $f'_i(\mathbf{x}_2) = f_i(\mathbf{x}_1 = \mathbf{v}_1)$  .  $\triangleright A_1$
  - 3: Choose a value  $\mathbf{v}_2$   $\triangleright$  Second choice: the number of iterations of this step corresponds to  $k'_2$  in Algorithm 1
  - 4: Evaluate  $f'_i(\mathbf{v}_2)$
  - 5: If  $\exists i < \ell, f'_i(\mathbf{v}_2) \neq 0$ : return to step 3  $\triangleright A_2$  and test
  - 6: If all choices  $\mathbf{v}_2$  have been checked: return to step 1
  - 7: Evaluate  $f_\ell, \dots, f_{m-1}$  on  $\mathbf{v}$ . If all values are 0, **Return**  $\mathbf{v}$ .  $\triangleright D$  and test
- 

**Asymptotic Complexity.** Using Equation 8 we immediately have the asymptotic gate count of the algorithm:

$$G = 2^{n_1/2}(G(\mathcal{A}_1) + G(\mathcal{D})) + 2^{n/2}G(\mathcal{A}_2) = 2^{n_1/2}((n - \ell)n^2 + \ell n n_1) + 2^{n/2}n_2^3 . \quad (9)$$

This is optimized by choosing  $\ell = n_2 + 4 = \mathcal{O}(\log n)$  giving  $G = \mathcal{O}((\log^3 n)2^{n/2})$ . The space complexity is  $\mathcal{O}(n)$  since the additional storage of  $W$  is negligible.

*Remark 1.* In this algorithm, we could use more than two levels of search. Though this would improve the asymptotic complexity, the gain would be likely imperceptible at small scales – and we obtain a better algorithm for  $MQ_2$  in the next section anyway. Such an algorithm would be rather suited for systems of higher degree, which are out of scope of this paper.

## 4.2 Quantum Implementation of Components

The circuits  $\mathcal{A}_1, \mathcal{A}_2$  and  $\mathcal{D}$  perform partial or complete evaluation of polynomials. We implemented them to obtain reliable bounds on their gate counts. We detail below some key points of this implementation.

**Evaluating Polynomials.** First, following [SW16] we give an elementary way to evaluate polynomials. Let us write:

$$f(x_0, \dots, x_{n-1}) = a_0 + \sum_{i=0}^{n-1} \sum_{j=0}^i a_{ij} x_i x_j ,$$

where  $a_{ij} = 0$  if  $j > i$ . The case  $i = j$  corresponds to the linear terms. Then:

$$f(x_0, \dots, x_{n-1}) = a_0 + \sum_{i=0}^{n-1} x_i \left( a_{ii} + \sum_{j=0}^{i-1} a_{ij} x_j \right) .$$

**Lemma 2.** *There is a quantum circuit that computes:*

$$|x_0, \dots, x_{n-1}, 0, 0\rangle \mapsto |x_0, \dots, x_{n-1}, f(x_0, \dots, x_{n-1}), 0\rangle$$

using  $n - 1$  CCX,  $n$  X,  $n(n - 1) + 1$  CX gates.

*Proof.* For each  $i$  from 1 to  $n - 1$ , we first compute  $a_{ii} + \sum_{j=0}^{i-1} a_{ij} x_j$  into the ancilla bit using a series of  $\leq i$  CX gates and  $\leq 1$  X gate, depending on the  $a_{ij}$ . We use a single CCX to XOR  $x_i \left( a_{ii} + \sum_{j=0}^{i-1} a_{ij} x_j \right)$  into the output bit. Then, we uncompute the sum. For  $i = 0$ , the term in the sum is only  $a_{00}$  so we only have (at most) one CX to apply.

The total CCX count is  $n - 1$ . The total X count is at most  $n - 1 + 1$  (if  $a_0 = 1$  we also need an X gate). The total CX count is:  $\leq 2 \sum_{i=0}^{n-1} i = n(n - 1)$ .  $\square$

**Partial Evaluation ( $\mathcal{A}_1$ ).** Again we start from:

$$f(x_0, \dots, x_{n-1}) = a_0 + \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_{ij} x_i x_j$$

where  $a_{ij} = 0$  if  $j > i$ . We choose a value  $\mathbf{v}_1 = (v_{n_2}, \dots, v_{n-1})$ , so the partially evaluated polynomial will be on  $n_2$  variables:

$$f'(\mathbf{x}_2) = f(\mathbf{x}_1 = \mathbf{v}_1) = a_0 + \sum_{i=0}^{n_2-1} \sum_{j=0}^{n_2-1} a_{ij} x_i x_j + \sum_{i=n_2}^{n-1} \sum_{j=n_2}^{n-1} a_{ij} v_i v_j + \sum_{i=0}^{n_2-1} \sum_{j=n_2}^{n-1} a_{ij} x_i v_j + \sum_{i=n_2}^{n-1} \sum_{j=0}^{n_2-1} a_{ij} v_i x_j$$

So the coefficients of the new polynomial can be expressed as:

$$\begin{cases} f'(x_0, \dots, x_{n_2-1}) = \sum_{i,j} b_{ij} x_i x_j \\ b_0 = a_0 + \sum_{i=n_2}^{n-1} \sum_{j=n_2}^{n-1} a_{ij} v_i v_j \\ \forall i < n_2, b_{ii} = a_{ii} + \sum_{j=n_2}^{n-1} (a_{ij} + a_{ji}) v_j \\ \forall i \neq j, i < n_2, j < n_2, b_{ij} = a_{ij} \end{cases} \quad (10)$$

We notice that  $\mathcal{A}_1$  only needs to return  $n_2 + 1$  bits of output for each polynomial, which are the coefficients  $b_{ii}$ . The other coefficients remain the same. This reduces the number of qubits used by the algorithm.

**Complete Evaluation ( $\mathcal{A}_2$ ).** The circuit for  $\mathcal{A}_2$  is a modified evaluation circuit which takes as input the coefficients returned by  $\mathcal{A}_1$ . It has essentially the same structure. For more details on the implementation one can refer to our code.

**Depth Optimization.** Since we always compute with multiple polynomials in parallel, it is possible to reduce the depth of the circuits by parallelizing some CCX and CX layers. As an example, let us consider a *multi-evaluation* circuit which evaluates in parallel multiple polynomials:

$$|\mathbf{x}, 0, \dots, 0\rangle \mapsto |\mathbf{x}, f_0(\mathbf{x}), \dots, f_{\ell-1}(\mathbf{x})\rangle \quad (11)$$

Following the circuit structure of above, we initialize one ancilla qubit per polynomial, and we use this ancilla qubit to compute linear expressions of the input variables via CX gates (the exact number of which depends on the  $f_i$ ), followed by CCX gates to create the quadratic terms.

Assume that  $\ell \leq n$ . During the computation of the linear functions, the CX gates can be ordered in such a way that the (at most)  $\ell$  gates applied in parallel will always use different inputs  $x_i$ . By choosing an appropriate ordering of the linear functions, we can also apply the CCX gates in layers.

This reduces the total depth of the circuit by a factor  $\ell$ , and makes it asymptotically optimal.

**Space Usage.** Due to our implementations of the circuits, we use in total:

- $(n_2 + 2)\ell$  qubits for the workspace: it contains the coefficients computed by  $\mathcal{A}_1$ , and later the evaluation outputs computed by  $\mathcal{A}_2$ , which do not need to be uncomputed
- $n$  qubits for the choice registers
- 2 qubits for the flags
- $2(m - \ell)$  qubits for ancillas (used in  $\mathcal{D}$  and partially used by the previous algorithms)

**Controlled Version.** Some of our applications use the polynomial system solver as a test in a Grover search. In that case, the polynomial system itself is a side input of the circuits  $\mathcal{A}_1, \mathcal{A}_2$  and  $\mathcal{D}$ . We have adapted our implementation to this case as well. The total number of gates is slightly increased (before, some gates were only applied depending on the coefficients). Some X gates are changed into CX gates. Besides the increase in space (since the polynomial system has  $n^2m$  coefficients), the most visible change is that most CX gates are changed into CCX gates, and so, the CCX gate count becomes dominating.

### 4.3 Optimization Results

For a given value of  $n$ , we optimize the cost of the algorithm as follows. First, we sample a random polynomial system (with a solution). While this will introduce some variance in the gate counts (since the coefficients of the polynomials are hard-coded in the circuits), we can observe in practice that it is below 0.1 in the exponent for  $n \geq 40$ .

Second, we fix the value of  $n_2$ . We construct the quantum circuits for  $\mathcal{A}_1, \mathcal{A}_2, \mathcal{D}$  using Qiskit [Qis23]. Internally the circuits are represented as a sequence of gates, and this allows to count trivially the number of gates, qubits, and the circuit depth.

Third, we use numerical optimization to determine optimal values of  $k_1$  and  $k'_2$ . In theory, different cost metrics (depth, gates) could lead to different optimizations, but in practice, we observed that the difference between the results was negligible. Our objective function is  $G(\mathcal{B}_1)/\nu^8$  where  $G(\mathcal{B}_1)$  and  $\nu$  are given by Equation 7 and Equation 6 (the exponent on  $\nu$  ensures that the success probability is large; we accept only a success probability greater than 0.9).

Results for different values of  $n$  are reported in Table 3. From these experiments, we estimate an upper bound for the Clifford+T gate count when  $n \geq 50$ :

$$(\log_2 n)^3 2^{6.32+n/2} . \quad (12)$$

**Table 3:** Optimization results for Quantum FES. The optimization target was Clifford+CCX gate count. Gate counts and depth are in  $\log_2$  and rounded.

$n$	$m$	$n_2$	Prob. of success	Qubits	Clifford+CCX	CCX	Depth
40	40	10	0.984	270	31.9	29.2	29.5
60	60	13	0.981	411	42.5	39.7	39.9
80	80	15	0.991	502	53.0	50.0	50.3
100	100	16	0.993	630	63.3	60.3	60.5

## 5 Quantum BDT Algorithm

In this section, we detail our second quantum algorithm for  $MQ_2$  which is a quantum version of the algorithm of Bouillaguet et al. [BDT22]. The idea of this algorithm is to select  $\mathcal{O}(\sqrt{m})$  variables and to guess the remaining ones. Since the system now contains  $m$  equations in about  $\sqrt{m}$  variables, it can be solved by Gaussian elimination, which runs in time  $\mathcal{O}(m^{3/2})$ . Consequently the complexity of this method is of order  $2^{n-\mathcal{O}(\sqrt{m})}$ .

As mentioned in [BDT22, Section 5.1], this algorithm can be seen as a special case of the Crossbred algorithm, which first constructs a system of larger degree  $\geq 2$ , then guesses some of the variables, and solves a smaller system of smaller degree (here linear).

The overall structure of the quantum algorithm is also similar to the quantum Boolean-solve [FHK<sup>+</sup>17], which performs an exhaustive (quantum) search on a subset of variables, then solves a linear system to obtain the remaining ones (with a consistency check).

## 5.1 Description as a Backtracking Search

Let  $\mathcal{F} = \text{Span}(f_0, \dots, f_{m-1})$  be the vector space spanned by the  $m$  quadratic polynomials in  $n$  variables. We assume that they are linearly independent, so  $\mathcal{F}$  is of dimension  $m$ . We choose a parameter  $u = \lfloor \sqrt{2m} - 2 \rfloor$ , and partition the variables into  $\mathbf{x} = \mathbf{y} \parallel \mathbf{z}$  where  $|\mathbf{y}| = n - u, |\mathbf{z}| = u$ . Here  $\mathbf{y}$  will be enumerated, and  $\mathbf{z}$  deduced by Gaussian elimination.

In a precomputation, we compute all combinations of the equations which eliminate all quadratic terms in  $\mathbf{z}$ . We let  $\mathcal{L} \subseteq \mathcal{F}$  be the vector space of these combinations. Following the discussion in [BDT22], if the input polynomials are linearly independent, the vector space  $\mathcal{L}$  has dimension at least:

$$\begin{aligned} \ell &:= m - \frac{u(u-1)}{2} = m - \frac{1}{2} \left( \lfloor \sqrt{2m} \rfloor - 2 \right) \left( \lfloor \sqrt{2m} \rfloor - 3 \right) \\ &= -3 + \frac{5 \lfloor \sqrt{2m} \rfloor}{2} \geq \frac{5u}{2} . \end{aligned}$$

If the dimension of  $\mathcal{L}$  is greater than  $\ell$ , we discard the additional equations. This gives us a subspace  $\mathcal{L}' \subseteq \mathcal{L}$  spanned by *exactly*  $\ell$  equations. Let  $(g_i)_{0 \leq i \leq \ell-1}$  be a basis of  $\mathcal{L}'$ . We write each  $g_i$  as:

$$g_i(\mathbf{y} \parallel \mathbf{z}) = q_i(\mathbf{y}) + (\mathbf{y}B_i + C_i)\mathbf{z}^t , \quad (13)$$

where for  $0 \leq i < \ell$ ,  $q_i$  are precomputed quadratic polynomials,  $B_i$  are precomputed Boolean matrices of dimension  $(n - u \times u)$ , and  $C_i$  are precomputed vectors of dimension  $u$ .

Obviously, any solution of  $\mathcal{F}$  is also a solution of  $\mathcal{L}'$ , but the converse will not be true. Our goal is to find solutions of  $\mathcal{L}'$  and test them afterwards against the full system.

A value  $\mathbf{v}$  for the variables  $\mathbf{y}$  defines a linear system in  $\mathbf{z}$ , as  $g_i(\mathbf{y} = \mathbf{v}) = 0$  is a linear equation in  $\mathbf{z}$ :

$$\forall i, g_i(\mathbf{y} = \mathbf{v}) = q_i(\mathbf{v}) + (\mathbf{v}B_i + C_i)\mathbf{z}^t . \quad (14)$$

This gives a system of  $\ell$  equations in  $u$  variables, which we can assume to be random (this assumption is heuristic, as systems constructed from different values  $\mathbf{v}$  are not independent). The system has a solution if the vector formed by evaluating the  $q_i(\mathbf{v})$  belongs to the image of the matrix formed by evaluating the  $(\mathbf{v}B_i + C_i)$ , which is of dimension  $u$  in  $\mathbb{F}_2^\ell$ . Therefore, if we assume that these vectors behave as randomly drawn, the system is consistent with probability at most  $2^{u-\ell} = \mathcal{O}(2^{-3u/2})$ .

In order to simplify the algorithm, we assume that the system that actually leads to the solution admits a single solution. This means that we will first check consistency of the system, and if it is consistent, we will attempt at solving it and check only the first solution that comes out.

The most costly step in the evaluation of  $g_i(\mathbf{y} = \mathbf{v})$  is the computation of  $q_i(\mathbf{v})$ . In order to speed this up, we can separate the evaluation in two steps as we did before, by guessing first a subset of the  $\mathbf{y}$  variables. By doing so, we obtain an algorithm with a similar structure as in Section 4. We select new parameters  $n_2, n_1$  such that  $n_1 + n_2 = n - u$ , and we separate  $\mathbf{y} = \mathbf{y}_2 \parallel \mathbf{y}_1$ .

This is an instance of Algorithm 1 where the first choice set is  $\{0, 1\}^{n_1}$  and the second choice set is  $\{0, 1\}^{n_2}$ . The algorithm  $A_1$  performs the partial evaluation of the linear system, and  $A_2$  performs the complete evaluation of the system, solves it via Gaussian elimination and determines if it has a solution (first test). Then,  $D$  tests all the equations (second test).

We use the following parameters. First,  $u$  should be as large as possible, so we take  $u = \lfloor \sqrt{2m} \rfloor - 2$ , and we will have  $\ell = \frac{5 \lfloor \sqrt{2m} \rfloor}{2} - 3$  and  $n_1 + n_2 = n - u$ . We choose  $n_2$  small enough to ensure that, when we start from the right choice of variables, no other value passes the test  $A_2$ . This means that among the  $2^{n_2}$  choices at the second step, we do

---

**Algorithm 5** Classical BDT algorithm [BDT22].

---

- Input:** polynomials  $f_0, \dots, f_{m-1}$
- 1: Precomputation: compute  $q_i(\mathbf{y}), B_i, C_i, 0 \leq i \leq \ell - 1$
  - 2: Choose a value  $\mathbf{v}_1$  for  $\mathbf{y}_1$  ▷ First choice
  - 3: Evaluate partially the polynomials  $q_i(\mathbf{y}_1 = \mathbf{v}_1)$  and the linear functions  $\mathbf{y}B_i$  ▷ Algorithm  $A_1$
  - 4: Choose a value  $\mathbf{v}_2$  for  $\mathbf{y}_2$  ▷ Second choice
  - 5: Evaluate completely the polynomials  $q_i(\mathbf{v}_2, \mathbf{v}_1)$  and the linear functions  $\mathbf{y}B_i + C_i$ .  
Solve the system. ▷ Algorithm  $A_2$
  - 6: If the system is not consistent, return to step 4. If all  $\mathbf{v}_2$  have been tested, return to step 2.
  - 7: Let  $\mathbf{w}$  be the solution of the system. Evaluate all  $f_i(\mathbf{v} \parallel \mathbf{w})$ . ▷ Algorithm  $D$
  - 8: If all values are 0, **Return** this result.
- 

not expect any system to be consistent (other than the one corresponding to the solution). Since random systems of  $\ell$  equations in  $u$  variables are consistent with probability  $2^{u-\ell}$ , we should have:

$$1 - (1 - 2^{u-\ell})^{2^{n_2}} \ll 1 \iff 2^{n_2+u-\ell} \ll 1 .$$

We will choose  $n_2 \leq \ell - u - 4$ .

*Remark 2.* An algorithm combining only BDT and Grover's algorithms, and evaluating  $g_i(\mathbf{v})$  entirely, would have complexity of order  $(mn^2)2^{(n-\text{sqrt}(2m))/2}$ . Indeed, the term  $(mn^2)$  would correspond to evaluating  $g_i(\mathbf{y} = \mathbf{v})$  and would dominate over the linear system solving. This is why we use another level of search to amortize.

## 5.2 Detailed Complexity Analysis and Results

Our implementation uses partial and complete evaluation circuits which are very similar to [Subsection 4.2](#). The novelty is that we need a quantum circuit that solves a Boolean linear system of the form  $M\mathbf{z} = C$  where both  $M$  and  $C$  are given as input.

To do so, we reuse a circuit from [BJ22]. Initially, its goal is to determine if a Boolean system is full rank, and to find an orthogonal vector. We can notice that:  $M\mathbf{z} = C$  is equivalent to  $(M|C)\begin{pmatrix} \mathbf{z} \\ 1 \end{pmatrix} = 0$ . Furthermore, by our analysis, we expect only a single solution if there is one. Thus the circuit [BJ22] can be reused to determine if a solution  $\mathbf{z}$  indeed exists, and to compute it.

Asymptotically the circuit contains  $\mathcal{O}(u^2\ell)$  gates when the matrix has  $\ell$  lines and  $u$  columns ( $\ell \geq u$ ). It requires  $\mathcal{O}(u\ell)$  space, and like the circuits in [Subsection 4.2](#) it can be made efficient in depth. For more precise estimates, we instantiated the circuit using our implementation and computed its costs.

**Asymptotic Gate Count.** We estimate the asymptotic gate count of the whole algorithm using [Equation 8](#):

$$G = 2^{n_1/2}(G(\mathcal{A}_1) + G(\mathcal{D})) + 2^{(n_1+n_2)/2}G(\mathcal{A}_2) . \quad (15)$$

Asymptotically we have  $\ell = u = \mathcal{O}(\sqrt{m})$ . The circuit  $\mathcal{A}_1$  contains  $\mathcal{O}(n_1n\ell) = \mathcal{O}(n^2\ell)$  gates. The circuit  $\mathcal{A}_2$  contains  $\mathcal{O}(n_2^2\ell + u^2\ell)$  gates. The circuit  $\mathcal{D}$  contains  $\mathcal{O}(n^3)$  gates, as we only need to check  $n + 4$  equations instead of  $m$ . This gives:

$$G = 2^{n_1/2}(n^3) + 2^{(n-\sqrt{2m})/2}(n_2^2\sqrt{m} + m^{3/2}) \quad (16)$$

This leads to the choice  $n_2 = \mathcal{O}(\log n)$ , which amortizes the first term, and gives us  $G = \mathcal{O}\left(m^{3/2}2^{(n-\sqrt{2m})/2}\right)$ . Contrary to the exhaustive search in [Section 4](#), this algorithm benefits from having more equations than unknowns, as we can eliminate more variables.



**Controlled Version.** When the polynomial system is given as input, the precomputation detailed in [Subsection 5.1](#) needs to be performed only once, before running the search. Most of the time, we can neglect its gate count. If not, we make the following estimation.

The  $q_i, B_i, C_i$  parameters of the algorithm are obtained by performing Gaussian elimination on the  $m \times n^2$  Boolean matrix of the polynomial system coefficients, by eliminating the quadratic terms in  $\mathbf{z}$ . Since there are  $u(u-1)/2$  such terms, one must perform  $u(u-1)/2 \times m$  row eliminations. Using Bonnetain and Jaques’ algorithm [[BJ22](#)], we expect asymptotically to use  $u(u-1)/2 \times m \times n(n+1)/2$  CCX gates. The space complexity is dominated by the storage of this matrix of dimension  $m \times n^2$ .

### 5.3 Optimization Results

Using the same optimization strategy as in [Section 4](#), we give some examples (with  $m = n$ ) in [Table 4](#). Though the number of qubits is typically larger, we observe significant improvements in all metrics compared to [Table 3](#). For example, when  $n = 100$ , we gain a factor  $2^{4.3}$  in Clifford+CCX gate count (respectively  $2^{5.3}$  in depth).

**Table 4:** Optimization results for quantum BDT. The optimization target was Clifford+Toffoli gate count. Complexities are in  $\log_2$  and rounded.

$n$	$m$	$u$	$n_2$	Prob. of success	Qubits	Clifford+CCX	CCX	Depth
40	40	6	6	0.965	407	30.6	28.3	27.3
60	60	8	9	0.983	669	40.2	38.2	36.6
80	80	10	12	0.992	985	49.5	47.9	45.8
100	100	12	15	0.992	1355	58.9	57.4	55.1

The following complexity estimate is an upper bound for the Clifford+T gate count (with decomposed CCX gates) of this algorithm when  $n \geq 50$ :

$$\begin{cases} u := \lfloor \sqrt{2m} - 2 \rfloor \\ \ell := \lfloor 5 \lfloor \sqrt{2m} \rfloor / 2 - 3 \rfloor \\ \text{Gate count} = \ell u^3 2^{6+(n-u)/2} \end{cases} \quad (17)$$

## 6 Application to LowMC

LowMC [[ARS<sup>+</sup>15](#)] is a block cipher aiming at minimizing the number of AND gates. Several variants were included in the NIST post-quantum candidate signature Picnic [[CDG<sup>+</sup>20](#)], which is based on the MPC-in-the-Head paradigm. This made LowMC a prominent target for cryptanalysis, especially in the Picnic scenario where the attacker must recover the secret key only from a single known plaintext-ciphertext pair.

Although Picnic did not reach standardization, the cryptanalysis of LowMC has remained very active [[BBVY21](#), [Din21a](#), [LMSI22](#), [BCV24](#)]. In the low-data scenario, these attacks are algebraic and rely on Guess-and-determine, Meet-in-the-middle and polynomial system solving. In this section, we will focus on examples of the latter.

Since it has applications in post-quantum cryptography, it seems natural and important to confront LowMC instances with quantum attacks. However, to the best of our knowledge, there has been no such attempt in the literature, apart from the analysis of Grover’s exhaustive search by Jaques et al. [[JNRV20](#)] and Jang et al. [[JBK<sup>+</sup>22](#)].

## 6.1 Specification

LowMC mainly relies on four adjustable parameters:  $r$  (number of rounds),  $n$  (block size, a multiple of 3),  $k$  (key length),  $s$  (number of S-Boxes used at each round). A single round of LowMC contains four operations:

1. Nonlinear layer: one applies  $s$  parallel instances of the LowMC S-Box to the first  $3s$  bits of the input. If the size of the input is bigger than  $3s$ , the remaining bits are left unchanged (thus LowMC allows for *partial S-Box layers*). The S-Box  $S$  transforms a 3-bit input  $a_1, a_2, a_3$  into  $S(a_1, a_2, a_3) = (a_4, a_5, a_6)$ , with:

$$\begin{cases} a_4 = a_1 \oplus a_2 a_3 \\ a_5 = a_1 \oplus a_2 \oplus a_1 a_3 \\ a_6 = a_1 \oplus a_2 \oplus a_3 \oplus a_1 a_2. \end{cases} \quad (18)$$

2. Linear layer: the block of length  $n$  is multiplied by a randomly generated  $n \times n$  invertible Boolean matrix. Each round uses a new matrix  $L_i$ . For the generation of these matrices, we can refer to the specification of LowMC [ARS<sup>+</sup>15]. The attacks studied here do not make use of their specificities.
3. Constant addition: the constant  $RC_i$  for round  $i$  is also randomly generated.
4. Round key addition: the round key  $K_i$  is obtained by multiplying the master key  $K$  (of size  $k$ ) by a matrix  $M_i$  of size  $n \times k$ , which is also randomly generated for every round  $i$ .

To summarize, a round of LowMC can be represented by the following function:  $A^{i+1} = L_i \circ S(A^i) \oplus RC_i \oplus K_i$ , where the input  $A^i$  is the state of round  $i$ . The first input is  $A^1 = P \oplus K_0$ , where  $P$  is the plaintext and  $K_0 = M_0 K$ . The ciphertext  $C$  after  $r$  rounds is  $A^{r+1}$ .

Parameters for LowMC in Picnic can be found in the specification document (Table 2 in [CDG<sup>+</sup>20]). With full S-Box layers, the designers used 4 rounds, and the block size  $n$  is respectively 129, 192 and 255 bits for NIST security levels 1, 3 and 5. With partial S-Box layers, the designers used  $s = 10$  and  $n, r = (128, 20), (192, 30)$  and  $(256, 38)$  respectively (in that case the block size does not need to be a multiple of 3).

### 6.1.1 Estimating the Cost of Grover's Search

We reuse part of the analysis done in [JNRV20] and [JBK<sup>+</sup>22] in order to estimate the cost of Grover's exhaustive search of the key for the LowMC variants we are interested in. The LowMC S-Box can be implemented in-place using 3 Toffoli gates and 2 CNOT gates. The affine layers used during the state update as well as the key expansion can be implemented *out of place* (increasing the number of qubits at each iteration) with about  $2^{13}$  Clifford gates and depth  $2^8$  for  $n = 129$  (resp.  $2^{14}, 2^{8.5}$  for  $n = 192$  and  $2^{15}, 2^9$  for  $n = 256$ ).

For a 3-round LowMC with full S-Box layers, there are 2 linear layers to apply and 3 linear operations in the key expansion. With the in-place S-Box circuit, we obtain the estimates of Table 5. For a LowMC with partial S-Box layers, the cost of the linear layers will largely dominate. In Grover's search one needs two computations of the cipher (forwards and backwards) per iteration.

## 6.2 Classical Attacks on LowMC in the Picnic Setting

In the Picnic setting, the length of the key is equal to the block size  $n$ , and the adversary must recover the key from a single known plaintext-ciphertext pair. We give a summary of

**Table 5:** Cost of a quantum circuit for different variants of LowMC (approximated).

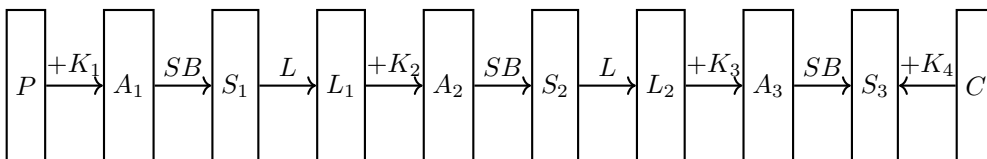
$n$	$s$	$r$	CCX	CCX + Clifford	Depth	Qubits
129	43	3	8.6	15.3	9.4	9.8
192	64	3	9.2	16.5	10.0	10.4
255	85	3	9.6	17.3	10.4	10.8
128	1	128	8.6	21.0	14.8	15.0
192	1	192	9.2	22.8	16.0	16.2
256	1	256	9.6	24.0	16.8	17.0
128	10	12	8.5	17.5	11.4	11.6
192	10	19	9.2	19.4	12.7	12.9
256	10	25	9.6	20.6	13.5	13.7

classical attacks in Table 6 for the variant with full S-Box layers and 3 to 7 rounds. To the best of our knowledge, no quantum attack (better than Grover’s search) is known for any of these variants.

In the following, we detail two attacks from [LMSI22]. The paper contains another attack on 4-round LowMC which is outside the scope of our analysis, as it relies on a degree-4 polynomial system solved with the polynomial method [Din21a].

### 6.3 Attack on 3-round LowMC with Full S-Box Layers

In this version (Section 3.2 in [LMSI22]) LowMC is reduced to three S-Box layers and two affine layers, as in Figure 3, and  $n = 3s$ .


**Figure 3:** Sequence of internal states in 3-round LowMC.

The reduction to a quadratic system of equations works as follows. First,  $P$  and  $C$  are known constants and all round keys  $K_1, K_2, K_3, K_4$  are linear in the master key  $K$ . Next, one guesses two bits in input of each S-Box in the first S-Box layer, i.e.,  $2s$  bits of  $K_1$ , which we note  $\mathbf{k}$ . There remains  $s$  unknown bits in  $K_1$ , which are denoted  $\mathbf{v}$ . All round keys  $K_2, K_3, K_4$  can be expressed as linear functions of  $\mathbf{v}$  (furthermore, these linear functions do not change, and can be precomputed).

For a single S-Box, let  $a_1^*$  and  $a_3^*$  be the two bits guessed; Equation 18 becomes:

$$\begin{cases} a_4 = a_1^* \oplus a_2 a_3^* \\ a_5 = a_1^* \oplus a_2 \oplus a_1^* a_3^* \\ a_6 = a_1^* \oplus a_2 \oplus a_3^* \oplus a_1^* a_2 \end{cases} \quad (19)$$

So the output  $a_4, a_5, a_6$  is linear in  $a_2$ . As a consequence,  $S_1$  in Figure 3 is a linear function of  $\mathbf{v}$ . After the next linear layer and round key addition,  $A_2$  is still linear in  $\mathbf{v}$ , and next, each bit of  $S_2$  is a quadratic polynomial in  $\mathbf{v}$ . This remains the case for  $L_2$  and  $A_3$ .

Next, one computes backwards from  $C$  (which is known). The state  $S_3$  is linear in  $\mathbf{v}$ , and consequently, each bit of  $A_3$  is a quadratic polynomial in  $\mathbf{v}$ . By equating these polynomials with those obtained in the forward computation, one obtains  $3s$  quadratic equations in the  $s$  variables  $\mathbf{v}$ .

**Table 6:** Summary of classical attacks on LowMC with full S-Box layers in the Picnic setting, taken from [BCV24]. Time is given in bit-operations and memory is given in bits. The success probability ranges from 0.5 to 1 for all attacks.

$r$	$n$	Reference	Time	Memory
3	129	FES [BCC+10]	135	
		[BBVY21]	140	53
		[Din21a]	125	104
		[LMSI22]	127.2	16.9
		[BCV24]	130.5	82.7
3	192	FES [BCC+10]	198	
		[BBVY21]	204	75
		[Din21a]	180	150
		[LMSI22]	186.2	18.6
		[BCV24]	188.9	119.3
3	255	FES [BCC+10]	261	
		[BBVY21]	267	97
		[Din21a]	235	197
		[LMSI22]	246.8	19.3
		[BCV24]	246.9	156.1
4	129	FES [BCC+10]	135	
		[Din21a]	130	113
		[LMSI22]	133.8	36.7
		[BCV24]	133.9	86.2
4	192	FES [BCC+10]	198	
		[Din21a]	188	113
		[LMSI22]	195.0	53.4
		[BCV24]	192.9	123.8
4	255	FES [BCC+10]	261	
		[Din21a]	245	218
		[LMSI22]	255.8	68.0
		[BCV24]	252.9	162.2
5	129	FES [BCC+10]	136	
		[BCV24]	135.7	88
5	192	FES [BCC+10]	199	
		[Din21a]	192	173
		[BCV24]	196.9	127.2
5	255	FES [BCC+10]	262	
		[Din21a]	254	228
		[BCV24]	256.9	166.0
6	192	FES [BCC+10]	199	
		[BCV24]	198	128.5
6	255	FES [BCC+10]	262	
		[BCV24]	259.7	168.5
7	255	FES [BCC+10]	263	
		[BCV24]	262.0	170.5

**Results of the Quantum Attack.** For this attack, we need to combine the Quantum BDT algorithm with a quantum search over  $\mathbf{k}$  ( $2s$  bits, hence  $\frac{\pi}{4}2^{s/2}$  search iterates). Therefore, the polynomial system depends on this guess, and the circuit for Quantum BDT is “controlled”. The “controlled” circuit uses mostly CCX gates, while the circuit for LowMC in Grover’s search is dominated by CX gates. Therefore, if we look at CCX gates only, the advantage becomes smaller.

The number of qubits is dominated by the storage of the quadratic system, which is constructed depending on  $\mathbf{k}$ . The time to construct the system remains negligible with respect to the Quantum BDT subroutine, since the number of variables ranges from 43 to 85. The full comparison is given in Table 7.

Regarding the depth, one can also parallelize Grover’s search, but a reduction of the depth by a factor  $S$  multiplies the qubit count by  $S^2$ . It can be seen that the quantum BDT still outperforms Grover’s search in this metric.

**Table 7:** Comparison between Grover’s search and the attack using the Quantum BDT algorithm, for 3-round LowMC with full S-Box layers. Complexities are given in  $\log_2$  and rounded. The probability of success is  $> 0.9$ .

Algorithm	$s$	CCX	CCX + Clifford	Depth	Qubits
Grover’s search	43	73.7	80.5	74.5	9.8
	64	105.8	113.1	106.6	10.4
	85	137.7	145.5	138.6	10.8
Quantum BDT	43	73.3	73.4	69.6	17.2
	64	104.1	104.2	100.1	18.9
	85	134.7	134.8	130.6	20.1

## 6.4 Attack on Reduced-round LowMC with Partial S-Box Layers

The other attack that we consider is from [LMSI22, Section 4], which targets a version of LowMC with  $s$  S-Boxes per round and  $r = \lfloor n/s \rfloor$  rounds. The idea is to guess a total of  $\lambda < n$  bits in the cipher, placed in a clever way, in order to reduce the key-recovery problem to an overdefined quadratic system.

Let us focus on the first  $r - 1$  rounds. One linearizes the first  $\lambda$  S-Boxes, but unlike the previous attack, one guesses one output bit rather than two input bits. Indeed, if one replaces  $a_4$  by a guess  $a_4^*$ , the expressions of  $a_4, a_5, a_6$  become linear in  $a_1, a_2, a_3$ :

$$\begin{cases} a_4 = a_4^* \\ a_5 = a_4^* \oplus a_2 \oplus a_3 a_4^* \\ a_6 = a_4^* \oplus a_2 \oplus a_3 \oplus a_2 a_4^* \end{cases} \quad (20)$$

It is important to note that  $\lambda$  is not necessarily a multiple of  $s$ , so this strategy can stop in the middle of a round, with some outputs guessed but not all of them. The value of  $\lambda$  will be a parameter of the attack, to optimize numerically.

The  $r - 1$  first rounds contain  $s(r - 1)$  S-Boxes. After the guesses, there are  $s(r - 1) - \lambda$  remaining S-Boxes. For all the three output bits of those, one introduces an intermediate variable. These  $3(s(r - 1) - \lambda)$  new variables are denoted  $\mu = (\mu_1, \mu_2, \dots, \mu_{3s(r-1)-3\lambda})$ .

The consequence is that each  $A^i$  is linear in  $(\mu, K)$ . Moreover, the states  $A^1$  (after first key addition) and  $A^{r+1}$  (before last key addition) are only linear in  $K$ .

If we focus on the last S-Box layer, both the input and output are linear in  $(\mu, K)$ , and furthermore, there are only  $s$  S-Boxes in the layer, meaning that we obtain  $n - 3s$  linear equations in  $(\mu, K)$  (since  $K$  is of  $n$  bits, this means  $3s(r - 1) - 3\lambda + n$  variables). By

Gaussian elimination, one reduces  $(\mu, K)$  to  $3s(r-1) - 3\lambda + n - (n-3) = 3(sr-\lambda)$  free variables, which are denoted  $\mathbf{v}$ .

Next, one constructs an overdefined system of quadratic equations in  $\mathbf{v}$ . For each linearized S-Box, in which the output bit  $a_4$  was guessed, one can obtain 3 equations in  $\mathbf{v}$ . Indeed, from Equation 18, replacing  $a_4$  by the guess and multiplying it by the inputs, one gets the following three quadratic equations in  $(a_1, a_2, a_3)$ :

$$\begin{cases} a_4^* = a_1 \oplus a_2 a_3 \\ a_4^* a_2 = a_1 a_2 \oplus a_2 a_3 \\ a_4^* a_3 = a_1 a_3 \oplus a_2 a_3 \end{cases} \quad (21)$$

Thus, the linearized S-Boxes yield  $3\lambda$  quadratic equations. Next, we focus on the  $s(r-1) - \lambda + s = sr - \lambda$  S-Boxes which are not linearized (including those of the last round). The authors of [LMSI22] show that for each S-Box, one can construct an overdefined system of 14 quadratic equations between the input and output bits, that are linearly independent. We refer to [LMSI22] for the complete system. These new equations give us  $14(sr-\lambda)$  quadratic equations in  $\mathbf{v}$ . In total, there are  $14sr - 11\lambda$  equations in  $3(sr-\lambda)$  variables.

**Results of the Quantum Attack.** The results for this attack are given in Table 8. Similarly to Subsection 6.3, the attack makes a guess of  $\lambda$  bits, then uses the quantum BDT algorithm to solve a quadratic system. The circuit here is “controlled” since the system varies depending on the guess. There are three important changes.

First, when choosing appropriate values of  $\lambda$  (similar to the ones in [LMSI22]), the quadratic systems become heavily overdefined. The cost of linear system solving dominates  $\mathcal{A}_2$  and  $\mathcal{A}_1$ . As a consequence, guessing the  $n-u$  variables in two steps does not bring any substantial gain, and we can take  $n_2 = 0$  for the cost estimates.

Second, the time to construct the polynomial system must be taken into account (in the other attack, it was negligible w.r.t. the cost of solving it). As mentioned in [LMSI22], the free variables are obtained by solving a linear system with about  $(n-3s)^2(n+3s(r-1)-3\lambda)$  bit operations. In the quantum setting, using Bonnetain and Jaques’ algorithm [BJ22], we have asymptotically the same gate count (mostly CCX gates), and we use an additional  $(n-3s)(n+3s(r-1)-3\lambda) + (n-3s)^2$  qubits.

Finally, since the system is large and very overdefined, the precomputation in the BDT algorithm is also not negligible anymore. When the parameter  $u = \mathcal{O}(\sqrt{14sr-11\lambda})$  is chosen, we need to eliminate  $u(u-1)/2$  quadratic terms in the polynomial system. This means  $u(u-1)/2 \times (14sr-11\lambda)$  row operations on a matrix whose rows have length  $(3(sr-\lambda))^2/2$ , and finally, approximately  $u(u-1)/2 \times (14sr-11\lambda) \times (3(sr-\lambda))^2/2$  CCX gates. Storing the quadratic system (which is the dominating space cost) occupies  $(14sr-11\lambda)(3(sr-\lambda))^2$  qubits.

As can be seen in Table 8, the advantage is less clear in this attack, especially since CCX gates dominate the time complexity. We only observe small gains in the total gate count (the depth-qubits trade-off is not advantageous). This is due to the larger costs of linear algebra, notably when creating and preprocessing the quadratic system.

## 7 Application to Reduced-round RAIN

RAIN is another block cipher suitable for MPC, which was designed as a building block of the post-quantum signature Rainier [DKR<sup>+</sup>22]. We will focus here on an attack proposed in [LMØM23], which leverages a particular algebraic representation of the cipher. The setting of the attack is the same as in Section 6: the key must be recovered from a single known plaintext-ciphertext pair. While the authors recommended 3 to 4 rounds, the attack targets a 2-round reduced version.

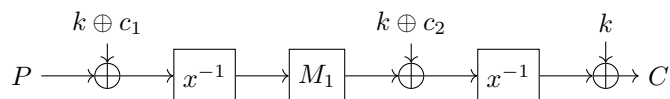
**Table 8:** Comparison between Grover’s search and the attack using the quantum BDT algorithm, for LowMC variants with partial S-Box layers. Complexities are given in  $\log_2$  and rounded. The probability of success is  $> 0.9$ .

Algorithm	$n$	$s$	$r$	$\lambda$	CCX	CCX + Clifford	Depth	Qubits
Grover’s search	128	1	128		73.2	85.6	79.5	15.0
	192	1	192		105.8	119.4	112.6	16.2
	256	1	256		138.2	152.6	145.5	17.0
	128	10	12		73.1	82.2	76.0	11.6
	192	10	19		105.8	116.0	109.3	12.9
	256	10	25		138.2	149.3	142.1	13.7
Quantum BDT	128	1	128	114	85.4	85.4	81.5	19.0
	192	1	192	174	117.5	117.5	113.2	20.2
	256	1	256	236	149.5	149.5	145.1	20.9
	128	10	12	105	81.3	81.3	77.3	19.2
	192	10	19	172	116.5	116.5	112.2	20.2
	256	10	25	230	146.5	146.5	142.1	20.8

## 7.1 Specification

The state of RAIN is a single element of the field  $\mathbb{F}_{2^n}$ . The nonlinear operation is the multiplicative inverse in  $\mathbb{F}_{2^n}$ , extended to 0 by 0, or equivalently, the map:  $S : x \mapsto x^{2^n-2}$ .

Randomly generated round constants  $c_i$  are used for the key schedule, and the key addition is performed over  $\mathbb{F}_{2^n}$ . Finally, there is a linear layer which can be seen either as a binary matrix  $M_i$ , or an  $\mathbb{F}_2$ -linearized polynomial. The two-round reduced RAIN is represented in Figure 4.



**Figure 4:** RAIN block cipher reduced to two rounds.

## 7.2 Classical Attack

We explain here the strategy of the attack of Section 4.3 in [LMØM23], which reduces the key-recovery on  $n$ -bit two-round RAIN to solving a quadratic system of  $3n$  equations in  $n$  unknowns. A key property to consider in this attack is the isomorphism between  $\mathbb{F}_{2^n}$  and  $\mathbb{F}_2^n$ , which transforms an element  $v$  into an  $n$ -bit vector  $\mathbf{v}$ . Under this isomorphism, taking a power of  $v$  of the form  $v^{2^i}$  for  $i > 0$  is equivalent to a linear transformation on the coordinates of  $\mathbf{v}$  in  $\mathbb{F}_2$ .

Let  $v = S(k + c_1 + P)$  be the output of the first non-linear layer, where  $P$  is the plaintext and  $S$  is the inverse function. The case  $P + k + c_1 = 0$  is trivial, since it would immediately give us  $k$ , so we can suppose that this is not the case, and rewrite:

$$v = (k + c_1 + P)^{-1} . \quad (22)$$

If one obtains  $v$ , one obtains  $k$  by  $k = v^{-1} + P + c_1$ . Again, the case  $v = 0$  would be trivially solved.

In order to write quadratic equations in  $v$ , one starts from this equality:

$$C = (M_1(v) + k + c_2)^{-1} + k \iff M_1(v) = (C + k)^{-1} + k + c_2 \quad (23)$$

Now, one replaces  $k$  by its expression in  $v$ :

$$M_1(v) = (C + (v)^{-1} + P + c_1)^{-1} + (v)^{-1} + P + c_1 + c_2 \quad (24)$$

Both terms  $P + c_1 + c_2$  and  $C + P + c_1$  are known constants, that we denote respectively as  $t_0$  and  $t_1$ . The equation becomes:

$$M_1(v) = (v^{-1} + t_1)^{-1} + v^{-1} + t_0 = v^{-1} + t_0 + v(1 + vt_1)^{-1} \quad (25)$$

By multiplying both sides by  $v(1 + vt_1)$  and rearranging the terms, one obtains:

$$(v + t_1v^2)M_1(v) = 1 + t_1v + t_0v + t_0t_1v^2 + v^2 . \quad (26)$$

Recall that under the isomorphism between  $\mathbb{F}_2^n$  and  $\mathbb{F}_{2^n}$ , taking the square  $v^2$  is equivalent to a linear transformation on  $\mathbf{v}$ . As a consequence, Equation 26 is equivalent to  $n$  quadratic (Boolean) equations in the coordinates of  $\mathbf{v}$ .

In order to obtain more equations, the authors of [LMØM23] multiply Equation 26 by  $(v + t_1v^2)$  and  $M_1(v)$  respectively, obtaining a system:

$$\begin{cases} (v + t_1v^2)M_1(v) = 1 + t_1v + t_0v + t_0t_1v^2 + v^2 \\ (v + t_1v^2)M_1(v)^2 = (1 + t_1v + t_0v + t_0t_1v^2 + v^2)M_1(v) \\ (v + t_1v^2)^2M_1(v) = (1 + t_1v + t_0v + t_0t_1v^2 + v^2)(v + t_1v^2) . \end{cases} \quad (27)$$

Under the isomorphism, since all four terms  $(v + t_1v^2)$ ,  $M_1(v)$ ,  $(v + t_1v^2)^2$  and  $M_1(v)^2$  are linear in  $\mathbf{v}$ , this forms a system of  $3n$  equations in  $n$  variables. One can check that these equations are indeed linearly independent over  $\mathbb{F}_2$ .

### 7.3 Results of the Quantum Attack

The quantum attack is a single instance of the Quantum BDT algorithm, with a fixed polynomial system. The optimization results are given in Table 9. For Grover's search, we assume that the number of qubits used is  $2n$  and evaluating the cipher costs  $n^3$  CCX gates. (We also assume that they can be perfectly parallelized, in order to estimate the circuit depth). We obtain a significant improvement in total gate count, and also in depth and qubits when taking into account the parallelization of Grover's search.

As a comparison, the classical time of the attack with the polynomial method is estimated in [LMØM23] to be of  $2^{118}$ ,  $2^{172}$  and  $2^{225}$  respectively.

**Table 9:** Comparison between Grover's search and the attack using the Quantum BDT algorithm, for 2-round RAIN. Complexities are given in  $\log_2$  and rounded. The probability of success is  $> 0.9$ .

Algorithm	$n$	CCX	CCX + Clifford	Depth	Qubits
Grover's search	128	84.7	84.7	77.7	8.0
	192	118.4	118.4	110.8	8.6
	256	151.7	151.7	143.7	9.0
Quantum BDT	128	67.7	68.4	63.8	11.8
	192	97.5	98.2	93.3	12.4
	256	127.3	127.9	122.7	12.8



## 8 Conclusion

In this paper, we gave new quantum algorithms for the  $MQ_2$  problem, showing improvements in gate counts for small problem sizes. These algorithms followed from the adaptation of well-studied classical techniques to the quantum search framework. We implemented the basic components of these circuits in classical reversible logic, allowing for reliable cost estimates.

While they are immediately applicable to quantum cryptanalysis, as our applications show, there remains much more to be done in the adaptation of algebraic attacks to the quantum setting.

In particular, while we have focused in this work on degree-2 equations, several important applications require to solve equations of larger degree, and even further, to adapt the solving algorithm to the specificities of the system studied. For example, with an efficient quantum algorithm for degree-4 equations, one could immediately attack the 4-round version of LowMC with full S-Box layers, like it was done previously in the classical setting [Din21a, LMSI22].

Finally, algorithms for larger fields would be relevant to public-key cryptography. So far this case seems only to have been studied asymptotically [BY18].

### Acknowledgments.

The authors would like to thank Loïc Ferreira for helpful discussions, and the anonymous reviewers for helpful remarks. This work received funding from the France 2030 program, managed by the French National Research Agency under grant agreement No. ANR-22-PETQ-0008 PQ-TLS. A.S. has been supported by the French Agence Nationale de la Recherche through the QATS project under Contract ANR-24-CE39-7894.

## References

- [AFI<sup>+</sup>04] Gwénolé Ars, Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between XL and gröbner basis algorithms. In *ASIACRYPT*, volume 3329 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 2004. doi:10.1007/978-3-540-30539-2\\_24.
- [ARS<sup>+</sup>15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In *EUROCRYPT (1)*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015. doi:10.1007/978-3-662-46800-5\\_17.
- [BBVY21] Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Hailun Yan. New attacks on LowMC instances with a single plaintext/ciphertext pair. In *ASIACRYPT (1)*, volume 13090 of *Lecture Notes in Computer Science*, pages 303–331. Springer, 2021. doi:10.1007/978-3-030-92062-3\\_11.
- [BCC<sup>+</sup>10] Charles Bouillaguet, Hsieh-Chung Chen, Chen-Mou Cheng, Tung Chou, Ruben Niederhagen, Adi Shamir, and Bo-Yin Yang. Fast exhaustive search for polynomial systems in  $\mathbb{F}_2$ . In *CHES*, volume 6225 of *Lecture Notes in Computer Science*, pages 203–218. Springer, 2010. doi:10.1007/978-3-642-15031-9\\_14.
- [BCV24] Subhadeep Banik, Andrea Caforio, and Serge Vaudenay. New attacks on LowMC using partial sets in the single-data setting. *IACR Commun. Cryptol.*, 1(1):22, 2024. URL: <https://doi.org/10.62056/ayzobjkrz>, doi:10.62056/AYZOBJKRZ.

- [BDT22] Charles Bouillaguet, Claire Delaplace, and Monika Trimoska. A simple deterministic algorithm for systems of quadratic polynomials over  $\mathbb{F}_2$ . In *SOSA*, pages 285–296. SIAM, 2022. doi:10.1137/1.9781611977066.22.
- [BFR24] Ryad Benadjila, Thibault Feneuil, and Matthieu Rivain. MQ on my mind: Post-quantum signatures from the non-structured multivariate quadratic problem. In *EuroS&P*, pages 468–485. IEEE, 2024. doi:10.1109/EUROSP60621.2024.00032.
- [BFSS13] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Pierre-Jean Spaenlehauer. On the complexity of solving quadratic boolean systems. *J. Complex.*, 29(1):53–75, 2013. URL: <https://doi.org/10.1016/j.jco.2012.07.001>, doi:10.1016/J.JCO.2012.07.001.
- [BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002. doi:10.1090/conm/305/05215.
- [BJ22] Xavier Bonnetain and Samuel Jaques. Quantum period finding against symmetric primitives in practice. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):1–27, 2022. URL: <https://doi.org/10.46586/tches.v2022.i1.1-27>, doi:10.46586/TCHES.V2022.I1.1-27.
- [BY18] Daniel J. Bernstein and Bo-Yin Yang. Asymptotically faster quantum algorithms to solve multivariate quadratic equations. In *PQCrypto*, volume 10786 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2018. doi:10.1007/978-3-319-79063-3\_23.
- [CDG<sup>+</sup>20] Melissa Chase, David Derler, Steven Goldfeder, Jonathan Katz, Vladimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Xiao Wang, and Greg Zaverucha. The Picnic signature algorithm. *Submission to NIST Post-Quantum Cryptography project, version 3.0*, 2020. URL: <https://raw.githubusercontent.com/microsoft/Picnic/master/spec/spec-v3.0.pdf>.
- [CHR<sup>+</sup>18] Ming-Shing Chen, Andreas Hülsing, Joost Rijneveld, Simona Samardjiska, and Peter Schwabe. SOFIA: MQ-based signatures in the QROM. In *Public Key Cryptography (2)*, volume 10770 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018. doi:10.1007/978-3-319-76581-5\_1.
- [CKPS00] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000. doi:10.1007/3-540-45539-6\_27.
- [Din21a] Itai Dinur. Cryptanalytic applications of the polynomial method for solving multivariate equation systems over  $\text{GF}(2)$ . In *EUROCRYPT (1)*, volume 12696 of *Lecture Notes in Computer Science*, pages 374–403. Springer, 2021. doi:10.1007/978-3-030-77870-5\_14.
- [Din21b] Itai Dinur. Improved algorithms for solving polynomial systems over  $\text{GF}(2)$  by multiple parity-counting. In *SODA*, pages 2550–2564. SIAM, 2021. doi:10.1137/1.9781611976465.151.
- [DKR<sup>+</sup>22] Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofneger, and Greg Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. In *CCS*, pages 843–857. ACM, 2022. doi:10.1145/3548606.3559353.

- [Fau02] Jean-Charles Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, 2002. doi:[10.1145/780506.780516](https://doi.org/10.1145/780506.780516).
- [FHK<sup>+</sup>17] Jean-Charles Faugère, Kelsey Horan, Delaram Kahrobaei, Marc Kaplan, Elham Kashefi, and Ludovic Perret. Fast quantum algorithm for solving multivariate quadratic equations. *IACR Cryptol. ePrint Arch.*, page 1236, 2017. URL: <http://eprint.iacr.org/2017/1236>.
- [Gid15] Craig Gidney. Constructing large controlled nots. <https://algassert.com/circuits/2015/06/05/Constructing-Large-Controlled-Nots.html>, 2015.
- [Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219. ACM, 1996. doi:[10.1145/237814.237866](https://doi.org/10.1145/237814.237866).
- [JBK<sup>+</sup>22] Kyungbae Jang, Anubhab Baksi, Hyunji Kim, Hwajeong Seo, and Anupam Chattopadhyay. Improved quantum analysis of SPECK and lowmc. In *INDOCRYPT*, volume 13774 of *Lecture Notes in Computer Science*, pages 517–540. Springer, 2022. doi:[10.1007/978-3-031-22912-1\\_23](https://doi.org/10.1007/978-3-031-22912-1_23).
- [JNRV20] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. Implementing grover oracles for quantum key search on AES and LowMC. In *EUROCRYPT (2)*, volume 12106 of *Lecture Notes in Computer Science*, pages 280–310. Springer, 2020. doi:[10.1007/978-3-030-45724-2\\_10](https://doi.org/10.1007/978-3-030-45724-2_10).
- [JV17] Antoine Joux and Vanessa Vitse. A crossbred algorithm for solving boolean polynomial systems. In *NuTMiC*, volume 10737 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2017. doi:[10.1007/978-3-319-76620-1\\_1](https://doi.org/10.1007/978-3-319-76620-1_1).
- [KHS<sup>+</sup>23] Seongkwang Kim, Jincheol Ha, Mincheol Son, ByeongHak Lee, Dukjae Moon, Joohee Lee, Sangyub Lee, Jihoon Kwon, Jihoon Cho, Hyojin Yoon, and Jooyoung Lee. AIM: symmetric primitive for shorter signatures with stronger security. In *CCS*, pages 401–415. ACM, 2023. doi:[10.1145/3576915.3616579](https://doi.org/10.1145/3576915.3616579).
- [LMØM23] Fukang Liu, Mohammad Mahzoun, Morten Øyegarden, and Willi Meier. Algebraic attacks on RAIN and AIM using equivalent representations. *IACR Trans. Symmetric Cryptol.*, 2023(4):166–186, 2023. URL: <https://doi.org/10.46586/tosc.v2023.i4.166-186>, doi:[10.46586/TOSC.V2023.I4.166-186](https://doi.org/10.46586/TOSC.V2023.I4.166-186).
- [LMSI22] Fukang Liu, Willi Meier, Santanu Sarkar, and Takanori Isobe. New low-memory algebraic attacks on LowMC in the picnic setting. *IACR Trans. Symmetric Cryptol.*, 2022(3):102–122, 2022. URL: <https://doi.org/10.46586/tosc.v2022.i3.102-122>, doi:[10.46586/TOSC.V2022.I3.102-122](https://doi.org/10.46586/TOSC.V2022.I3.102-122).
- [LPT<sup>+</sup>17] Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In *SODA*, pages 2190–2202. SIAM, 2017. doi:[10.1137/1.9781611974782.143](https://doi.org/10.1137/1.9781611974782.143).
- [Mas98] VI Masol. Limit distribution of the number of solutions of a system of random boolean equations with a linear part. *Ukrainian Mathematical Journal*, 50(9):1389–1404, 1998. doi:[10.1007/BF02525245](https://doi.org/10.1007/BF02525245).

- [NC02] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.
- [NIS22] NIST. Call for additional digital signature schemes for the post-quantum cryptography standardization process. <https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf>, 2022.
- [Pri18] Benjamin Pring. Exploiting preprocessing for quantum search to break parameters for  $MQ$  cryptosystems. In *WAIFI*, volume 11321 of *Lecture Notes in Computer Science*, pages 291–307. Springer, 2018. doi:10.1007/978-3-030-05153-2\_17.
- [Qis23] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023. doi:10.5281/zenodo.2573505.
- [SS24] André Schrottenloher and Marc Stevens. Quantum procedures for nested search problems: with applications in cryptanalysis. *IACR Commun. Cryptol.*, 1(3):9, 2024. URL: <https://doi.org/10.62056/aee0fhbmo>, doi:10.62056/AEE0FHBMO.
- [SW16] Peter Schwabe and Bas Westerbaan. Solving binary  $MQ$  with grover’s algorithm. In *SPACE*, volume 10076 of *Lecture Notes in Computer Science*, pages 303–322. Springer, 2016. doi:10.1007/978-3-319-49445-6\_17.
- [WWF<sup>+</sup>21] Congming Wei, Chenhao Wu, Ximing Fu, Xiaoyang Dong, Kai He, Jue Hong, and Xiaoyun Wang. Preimage attacks on 4-round Keccak by solving multivariate quadratic systems. In *ICISC*, volume 13218 of *Lecture Notes in Computer Science*, pages 195–216. Springer, 2021. doi:10.1007/978-3-031-08896-4\_10.