Improving the Efficiency of Quantum Circuits for Information Set Decoding

SIMONE PERRIELLO, ALESSANDRO BARENGHI, and GERARDO PELOSI, Department of Electronics, Information and Bioengineering, Politecnico di Milano, Italy

Code-based cryptosystems are a promising option for Post-Quantum Cryptography, as neither classical nor quantum algorithms provide polynomial time solvers for their underlying hard problem. Indeed, to provide sound alternatives to lattice-based cryptosystems, U.S. National Institute of Standards and Technology (NIST) advanced all round 3 code-based cryptosystems to round 4 of its Post-Quantum standardization initiative. We present a complete implementation of a quantum circuit based on the Information Set Decoding (ISD) strategy, the best known one against code-based cryptosystems, providing quantitative measures for the security margin achieved with respect to the quantum-accelerated key recovery on AES, targeting both the current state-of-the-art approach and the NIST estimates. Our work improves the state-of-the-art, reducing the circuit depth by 2^{19} to 2^{30} for all the parameters of the NIST selected cryptosystems, mainly due to an improved quantum Gauss–Jordan elimination circuit with respect to previous proposals. We show how our Prange's-based quantum ISD circuit reduces the security margin with respect to its classical counterpart. Finally, we address the concern brought forward in the latest NIST report on the parameters choice for the McEliece cryptosystem, showing that its parameter choice yields a computational effort slightly below the required target level.

CCS Concepts: • **Security and privacy** → **Public key (asymmetric) techniques**; **Cryptography**; **Cryptanalysis and other attacks**; • **Theory of computation** → *Quantum computation theory*; **Computational complexity and cryptography**; **Circuit complexity**; • **Computer systems organization** → *Quantum computing;*

Additional Key Words and Phrases: Post-quantum cryptography, public-key encryption, code-based cryptography, Grover, Prange, Lee–Brickell, ISD, Information Set Decoding

ACM Reference format:
Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. 2023. Improving the Efficiency of Quantum Circuits for Information Set Decoding. *ACM Trans. Quantum Comput.* 4, 4, Article 25 (August 2023), 40 pages.
https://doi.org/10.1145/3607256

This work is an extension of the conference paper [54].
This research activity was partially funded by the Italian "Centro Nazionale di Ricerca in HPC, Big Data and Quantum computing — SPOKE 10".
Authors' address: S. Perriello, A. Barenghi, and G. Pelosi, Department of Electronics, Information and Bioengineering, Politecnico di Milano, Piazza Leonardo da Vinci 32, Milan, Italy, 20133; emails: {simone.perriello, alessandro.barenghi, gerardo.pelosi}@polimi.it.

2643-6817/2023/08-ART25 $15.00
https://doi.org/10.1145/3607256

1 INTRODUCTION

The work of Shor [62], detailing a quantum algorithm to attack the RSA cryptosystem exponentially faster than any known classic algorithm—later adapted to attack cryptosystems based on elliptic curves [57]—created the need for cryptographic schemes capable of withstanding both classical computing and quantum computing attacks.

The quest for cryptographically sound primitives spurred the **U.S. National Institute of Standards and Technology (NIST)** [51] to run a standardization initiative program for **Post-Quantum Cryptography (PQC)**, which started in 2016. The E.U. European Telecommunications Standards Institute, however, established in 2020 a working group on quantum safe cryptography [27], with the aim of assessing and making recommendations for quantum-safe cryptographic primitives.

Error-correcting codes, formalized by Shannon [61] and originally thought as a way of detecting and correcting corrupted piece of information over a noisy communication channel, have seen an increasing interest in their use as the backbone of asymmetric cryptosystems, as shown by NIST decision that arrived in mid-2022 to advance all code-based cryptographic schemes to the fourth stage of the competition [49]. Indeed, to have an alternative to lattice-based cryptosystems, NIST suggests in the same report that one of the three code-based candidates, namely Classic McEliece [12], BIKE [3], and HQC [47], will be standardized with high probability. The only other alternative cryptosystem advanced to the fourth round, SIKE [35], an efficient polynomial-time algorithm has just recently been found [18], making its advancement to the standardization stage unlikely. It is therefore crucial to accurately assess the computational complexity of an attack to code-based cryptosystems accelerated through the help of a quantum computer to precisely tune the parameters for these cryptographic schemes.

Code-based cryptographic schemes, dating back to the proposal by Robert McEliece [46], build their security on the hardness of the (search-)**syndrome decoding problem (SDP)**, proven to be NP-hard in Reference [10]. The problem can be summarized as finding a solution to an underconstrained set of simultaneous linear equation, having a fixed amount of non-zero elements. The best-known classical algorithm to solve SDP, called **Information Set Decoding (ISD)**, still runs in exponential time, while theorized quantum-aided attacks do not go beyond the quadratic speed-up provided by Grover's framework [31]. The first work theorizing a potential speed-up in the ISD attack accelerated by a quantum computer [11] showed indeed a considerable decrease in the amount of computation required when Grover's framework is used. The work, however, did not present any concrete circuit, offering only asymptotic estimates in terms of number of gates and number of qubits. To the best of our knowledge, the first concrete implementation of a full quantum circuit using the ISD strategy to solve the SDP problem was reported in Reference [54], in which the authors report a computational effort smaller by a factor of 2^4 with respect to the asymptotic effort estimated in Reference [11] for cryptographically relevant sizes of the problem instance. In the same work, they also show how the parameter choices made for BIKE and Classic McEliece results in a larger computational effort than the one required by NIST, which was set as a comparison to the effort of breaking symmetric cryptosystems. Recently, the authors of Reference [26] reported a similar implementation of a quantum SDP solver based on the ISD strategy and proposed a strategy trading off number of qubits with circuit depth.

Contribution. In this article, we present a quantum circuit implementing the Prange variant of the ISD strategy to solve the SDP, extending and improving the work [54], as well as improving the current state of the art. We provide exact measures for the gate count (size), number of required qubit (width), and depth of our quantum circuit design. Our measures depend only on the linear code parameters, allowing code designers to fine-tune them to match the desired security level.

- We improve the depth × width figure with respect to the previous theoretical proposals presented in Reference [11], the estimates provided in Reference [25], and the implementations detailed in Reference [54] and Reference [26]. The improvements are mainly due to an adaptation of the classical *decoding one out of many* strategy [60] to our quantum circuit and to a series of optimizations to the state-of-the-art proposal for a **quantum version of the Gauss–Jordan elimination (QGJE)** circuit presented in Reference [54].
- We evaluate, according to the criteria specified by NIST, the complexity of carrying out attacks against all the code-based cryptosystems advanced to the fourth round of the NIST Post-Quantum standardization initiative, namely Classic McEliece, BIKE, and HQC, extending the previous analysis presented in Reference [54], limited only to the first two of the three candidates. To retrieve our measures, we target both a reversible gate set extended with uncontrolled phase-rotation gates (i.e., R_y), and the Clifford + T gate set, which is the most promising candidate for fault-tolerant quantum computation. The latter gate set allows a direct comparison against the best-known quantum circuits designed to bruteforce the AES cryptographic scheme [44, 67].
- By constraining the depth of our circuits, and hence parallelizing our algorithm across multiple instances, we show that the parameter choice for the Classic McEliece intended to match the robustness of AES-192 yields an ISD strategy that can be solved with $\approx 2^4$ less computational effort than the expected bar, a concern already brought forward in Reference [25] and by NIST itself in its latest report [49].
- We show that the most convenient way to attack code-based post-quantum cryptosystems relying on quasi-cyclic random codes (i.e., HQC and BIKE) is to employ our quantum Prange ISD approach. Indeed, taking the symmetric cipher AES-128 as the bar for the parameters' selection, the code length required to ward off a quantum cryptanalytic attack exceeds the one required to ward off the classic attack (considering quantum and classic attacks against AES as the bars, respectively). This fact shows that the design of the parameters for cryptosystems such as BIKE and HQC should take into account the results derived from our quantum circuit. By contrast, for the Classic McEliece cryptosystem, the estimates on the parameters required to match the computational effort of breaking AES-128 either classically or not yield substantially the same figures.

2 BACKGROUND

In this section, we provide a summary of the basics of code-based cryptography and the ISD technique. We recall the framework proposed by Grover [31], and we conclude the section providing a background on two classical algorithmic techniques, sorting networks and Hamming weight computation via adder trees, for which we employed the quantum counterparts in our quantum circuits.

Notation. Vectors in this work are denoted with lowercase bold letters, e.g., $\mathbf{v}$. They have a finite dimension, and their elements are in the set $\{0, 1\}$, i.e., a vector having d components is said to belong to the finite, binary field $\in \mathbb{F}_2^d$. A subscript to the column vector of the form $\mathbf{v}_i$ denotes element at index i of vector $\mathbf{v}$, $0 \leq i \leq d-1$, while $\mathbf{v}_{\mathcal{S}}$ denotes the projection of the elements of $\mathbf{v}$ on the indexes of the set $\mathcal{S}$, i.e., $\mathbf{v}_{\mathcal{S}} = \{\mathbf{v}_i | i \in \mathcal{S}\}$.

Finite, binary matrices are denoted as uppercase boldface letters, e.g., $\mathbf{M} \in \mathbb{F}_2^{d_1 \times d_2}$, where d_1 and d_2 denote the number of rows and columns, respectively. Similarly to the vector case, we denote with $\mathbf{M}_{i,j}$ the element of the matrix at row i and column j, $0 \leq i \leq d_1 - 1$, and $0 \leq j \leq d_2 - 1$. The notation $\mathbf{M}_{\mathcal{S}_1, \mathcal{S}_2}$ denotes a selection of the rows indexed by the elements in $\mathcal{S}_1$ and of the columns indexed by the element in $\mathcal{S}_2$, i.e., $\mathbf{M}_{\mathcal{S}_1, \mathcal{S}_2} = \{\mathbf{M}_{i,j} | i \in \mathcal{S}_1, j \in \mathcal{S}_2\}$. In this context, we use the

colon to represent the set of all rows, i.e., $\mathbf{M}_{i,:}$ denotes the vector obtained taking all the elements of matrix $\mathbf{M}$ in row i. By extension, $\mathbf{M}_{\mathcal{S}_1,:}$ is the matrix obtained selecting the rows indexed by the elements in $\mathcal{S}_1$, while keeping the whole set of columns. The colon notation is extended to the selection of one or multiple rows, denoted as $\mathbf{M}_{:,j}$ or $\mathbf{M}_{:,\mathcal{S}_2}$ respectively.

Finally, we use the notation $\mathrm{WT}(\mathbf{v})$ to denote the Hamming weight of the vector $\mathbf{v}$, i.e., the number of its elements different from 0.

2.1 Linear Error-correcting Codes

A binary linear code C is a linear subspace of the vector space $\mathbb{F}_2^n$ having dimension k. A linear code is used to encode a message, represented as a k-length binary vector $\mathbf{m} \in \mathbb{F}_2^k$, into another binary vector $\mathbf{c} \in \mathbb{F}_2^n$, called *codeword*, belonging to C. The encoding is done through the addition of $r = n - k$ bits, called redundancy bits, to $\mathbf{m}$. Denoting as $\mathbf{y}$ a corrupted codeword, we can express it as the bitwise sum of two binary vectors $\in \mathbb{F}_2^n$, namely the original codeword $\mathbf{c}$ and an error vector $\mathbf{e}$, i.e., $\mathbf{y} = \mathbf{e} + \mathbf{c}$, with $\mathrm{WT}(\mathbf{e}) = t$.

The encoding process can be described through a standard matrix-vector multiplication between a matrix $\mathbf{G} \in \mathbb{F}_2^{k \times n}$, called *generator matrix*, and the original message $\mathbf{m}$, i.e., $C = \{\mathbf{c} \mid \mathbf{c} = \mathbf{G}^T\mathbf{m}\}$. In our work, we will use the alternative, though equivalent, formulation of the code through the matrix $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, called *parity-check* matrix, such that $\mathbf{H}\mathbf{G}^T = \mathbf{0}_{r \times k}$ and therefore $C = \{\mathbf{c} \mid \mathbf{H}\mathbf{c} = \mathbf{0}_{r \times 1}\}$. The decoding process, however, consists in removing the r redundancy bits from the (possibly) corrupted codeword $\mathbf{y} = \mathbf{c} + \mathbf{e}$. The **codeword decoding problem (CDP)** consists in retrieving the original codeword $\mathbf{c}$ starting from $\mathbf{y}$. Given the linearity of the code, the two challenges of retrieving either $\mathbf{c}$ or $\mathbf{e}$ starting from $\mathbf{y}$ are equivalent. In this work, we thus consider the SDP, which asks to retrieve the error $\mathbf{e}$ given only its *syndrome* vector $\mathbf{s} = \mathbf{H}\mathbf{y} = \mathbf{H}(\mathbf{c} + \mathbf{e}) = \mathbf{H}\mathbf{e}$.

2.2 Code-based Cryptosystems

The use of linear codes as a way to build trapdoor functions for cryptographically safe primitives is justified by the result in Reference [10], showing that the decision version of both CDP and SDP are NP-complete if the matrix $\mathbf{H}$ is randomly chosen among the ones in $\mathbb{F}_2^{r \times n}$ having rank r. The first public-key cryptographic scheme based on linear codes was proposed by McEliece [46]. The proposal uses as a private key a parity-check matrix $\mathbf{H}'$ for which an efficient decoding algorithm correcting up to weight t errors is known, together with a random non-singular matrix $\mathbf{A}_{r \times r}$ and a random permutation matrix $\mathbf{P}_{n \times n}$. If the matrix $\mathbf{H} = \mathbf{A}\mathbf{H}'\mathbf{P}$ is indistinguishable from a random, full-rank one, then this allows its use as a public key. To encrypt a message, the sender encodes the message in an error vector $\mathbf{e}$ of weight t, computes the syndrome $\mathbf{s} = \mathbf{H}\mathbf{e}$, and sends it through an insecure channel. An attacker willing to recover $\mathbf{e}$ is then forced to find a solution to the SDP for a random-looking $\mathbf{H}$, while the legitimate receiver can simply compute $\mathbf{s}' = \mathbf{A}^{-1}\mathbf{s}$ and efficiently solve the SDP for the non-random $\mathbf{H}$.

The McEliece cryptosystem withstood deep scrutiny over the years, and, since no efficient classical or quantum attack undermining its security has been found up to now, it was promoted to the fourth round as a finalist in the NIST Post-Quantum Cryptography initiative [49]. New families of codes yielding on smaller key sizes have been proposed, such as the **Quasi-Cyclic Moderate-Density Parity-Check codes (QC-MDPC)** presented for the first time in Reference [48]. Their design builds upon the theory of quasi-cyclic codes. More specifically, for the proposed codes, denoting the code length n as the product of two positive integers, $n = n_0 r$, the parity-check matrix $\mathbf{H}$ of a quasi-cyclic code is described with n_0 circulant matrices of size $r \times r$. Each of the circulant matrices, called a *circulant block*, is defined starting from a single row, generating all the other $r-1$ ones by cyclically shifting it by an amount in $\{1, \ldots, r-1\}$. This allows a significant reduction in

the size of the public key, as the public quasi-cyclic parity check matrix can be reconstructed from a single row. The main drawback of QC-MDPC codes is that the cyclic structure of the parity-check matrix can be exploited to parallelize the solution to the SDP. Indeed, by shifting $r-1$ times the given syndrome $\mathbf{s}$, we can derive a set of r distinct associated syndromes, corresponding to the same cyclic shifts in the error vector $\mathbf{e}$ that we want to retrieve. All of these instances are associated to the same parity-check matrix $\mathbf{H}$ and the same weight t and, acting on r distinct syndromes, can be run in parallel. Finding the solution for any of the r syndromes allows us to immediately retrieve the required error vector. Depending on the desired degree of parallelization, we can tune the number of instances M, with $1 \leq M \leq r$.

2.3 Information Set Decoding

The most efficient method known to solve the SDP generically relies on the fact that knowing the k error-free positions of the corrupted codeword $\mathbf{y}$ allows the retrieval of the corresponding information word $\mathbf{m}$, as it uniquely identifies it.

The first work employing the Information Set as a decoding technique is due to Prange [56]. Prange's ISD technique considers a set $\mathcal{I} \subseteq \{0,\dots,n-1\}$ of size k, the *Information Set*, or, equivalently, the complement set $\mathcal{J} = \{0,\dots,n-1\} \setminus \mathcal{I}$ of size $n-k=r$. After randomly guessing $\mathcal{J}$, the proposed algorithm proceeds by assuming that the error vector $\mathbf{e}$, with $\mathrm{WT}(\mathbf{e}) = t$, has all its t asserted bits in the positions indexed by $\mathcal{J}$. As a consequence, all the bits of $\mathbf{e}$ in the positions indexed by $\mathcal{I}$ are 0. We can express the previous desiderata as $\mathbf{e}_{\mathcal{I}} = \mathbf{0}_{k\times 1}$, while $\mathrm{WT}\left(\mathbf{e}_{\mathcal{J}}\right) = t$. We can split the syndrome computation, originally obtained as $\mathbf{s} = \mathbf{He}$, as the bitwise addition between two distinct vectors, i.e., $\mathbf{s} = \mathbf{H}_{:,\mathcal{J}}\mathbf{e}_{\mathcal{J}} \oplus \mathbf{H}_{:,\mathcal{I}}\mathbf{e}_{\mathcal{I}}$. When the assumption made by Prange is true, that is, $\mathbf{e}_{\mathcal{I}} = \mathbf{0}_{k\times 1}$, the value of the syndrome is $\mathbf{s} = \mathbf{H}_{:,\mathcal{J}}\mathbf{e}_{\mathcal{J}}$, which can be seen as a system of linear equations with unknown $\mathbf{e}_{\mathcal{J}}$. If $\mathbf{H}_{:,\mathcal{J}}$ is invertible, then we can derive $\mathbf{e}_{\mathcal{J}} = (\mathbf{H}_{:,\mathcal{J}})^{-1}\mathbf{s}$ and, from there, reconstruct the whole error vector $\mathbf{e}$. If either the submatrix $\mathbf{H}_{:,\mathcal{J}}$ is not invertible or $\mathrm{WT}\left(\mathbf{e}_{\mathcal{J}}\right) = (\mathbf{H}_{:,\mathcal{J}})^{-1}\mathbf{s} \neq t$, then the algorithm proceeds to guess another random $\mathcal{J}$ and restarts.

This approach is summarized in Algorithm 1. At line 2, we associate the random choice of $\mathcal{J}$ to a random permutation matrix $\mathbf{P}\in\mathbb{F}_2^{n\times n}$, used in the next line to bring the columns of $\mathbf{H}$ indexed by $\mathcal{J}$ to its leftmost $r\times r$ portion, leading to a new matrix $\widehat{\mathbf{H}} = \mathbf{HP} = [\mathbf{H}_{:,\mathcal{J}} \mid \mathbf{H}_{:,\mathcal{I}}]$. To note that, from $\mathbf{s} = \mathbf{He}$, we also derive $\mathbf{s} = (\mathbf{HP})(\mathbf{P}^{\intercal}\mathbf{e})$, which, by renaming the two parts of the equation, can be seen as the new system of linear equations $\mathbf{s} = \widehat{\mathbf{H}}\hat{\mathbf{e}}$, with unknown vector $\hat{\mathbf{e}} = [\mathbf{e}_{\mathcal{J}} \mid \mathbf{e}_{\mathcal{I}}]$. The condition on the invertibility on the $r\times r$ matrix $\mathbf{H}_{:,\mathcal{J}}$ is restated as a check on the matrix $\widetilde{\mathbf{H}} = \mathrm{GJE}(\widehat{\mathbf{H}})$, in which $\mathrm{GJE}(\cdot)$ denotes the **Gauss–Jordan elimination (GJE)** algorithm: If, after the GJE procedure, $\widetilde{\mathbf{H}}$ has an identity in its rightmost $r\times r$ submatrix, then the initial matrix $\mathbf{H}_{:,\mathcal{J}}$ was invertible. While performing the GJE algorithm, we also obtain the vector $\tilde{\mathbf{s}} = (\widehat{\mathbf{H}}_{:,\{0,\dots,r-1\}})^{-1}\mathbf{s}$ by building the augmented matrix $\mathbf{L}_{r\times(n+1)} = [\widehat{\mathbf{H}}|\mathbf{s}]$ and then performing $\mathrm{GJE}(\mathbf{L})$. If the leftmost $r\times r$ submatrix of the result is an identity matrix and $\tilde{\mathbf{s}}$ has weight t, then we can trivially retrieve $\mathbf{e}$.

Prange's algorithm is a Las Vegas algorithm, i.e., a randomized algorithm outputting the correct result—the value of $\mathbf{e}$—using a probabilistic amount of computational time. Its runtime can be computed starting from the probability of success, $\Pr_{succ-PR}$, of a single iteration of the outer loop. Since for each loop iteration the matrix $\mathbf{P}$, corresponding to a specific choice of $\mathcal{J}$, is independent of all other iterations, the probability of success of a single loop iteration is independent of the others.

The first factor of this probability is obtained by dividing the number of permuted error vectors matching Prange's requirements, i.e., all the vectors of length r and weight t, $\binom{r}{t}$, divided by the total number of error vectors, i.e., all the vectors of length n and weight t, $\binom{n}{t}$.

ALGORITHM 1: Prange algorithm

Input : $\mathbf{H}$: $r \times n$ parity-check matrix
$\mathbf{s}$: $r \times 1$ syndrome
t: weight of $\mathbf{e}$
Output : $\mathbf{e}$: $n \times 1$ column vector s.t. $\mathbf{He} = \mathbf{s}$, $\mathrm{W_T}(\mathbf{e}) = t$
Data : $\mathbf{P}$: $n \times n$ permutation matrix
$\widehat{\mathbf{H}}$: $\mathbf{H}$ after permutation
$\widetilde{\mathbf{H}}$: $\widehat{\mathbf{H}}$ after GJE
$\tilde{\mathbf{s}}$: $\mathbf{s}$ after GJE

1 **repeat**
  // random choice of $\mathcal{J}$
2 $\mathbf{P} \leftarrow \mathrm{RandomPermutation}(n)$
  // error vector is $\hat{\mathbf{e}} = \mathbf{P}^\intercal \mathbf{e}$
3 $\widehat{\mathbf{H}} \leftarrow \mathbf{HP}$
4 $[\widetilde{\mathbf{H}} \mid \tilde{\mathbf{s}}] \leftarrow \mathrm{GJE}\left([\widehat{\mathbf{H}} | \mathbf{s}]\right)$
5 $[\mathbf{W}_{r\times r} \mid \mathbf{V}_{r\times k}] \leftarrow \widetilde{\mathbf{H}}$
6 **until** $W_T(\tilde{\mathbf{s}}) \neq t$ **or** $\mathbf{W} \neq \mathbf{I}_r$
7 $\hat{\mathbf{e}} \leftarrow [\tilde{\mathbf{s}} \mid \mathbf{0}^\intercal_{1\times k}]$
8 **return** $\mathbf{e} = \mathbf{P}\hat{\mathbf{e}}$

ALGORITHM 2: Gauss-Jordan elimination

Input : $\mathbf{L}$: $r \times n + 1$ matrix, $n > r$
Output : $\mathbf{L}$ in reduced-row echelon form, or failure

1 **for** $x \leftarrow 0$ **to** $r - 1$ **do**
2 $i \leftarrow x$
  // **Part 1.**
  // Put 1 in pivot position if it contains a 0
3 **while** $i < r$ **and** $\mathbf{L}_{i,x} = 0$ **do**
4 $i \leftarrow i + 1$
5 **if** $\mathbf{L}_{i,x} = 0$ **then**
6 **return** $\perp$
7 **if** $x \neq i$ **then**
8 swapRow $(\mathbf{L}_{x,:}, \mathbf{L}_{i,:})$
  // **Part 2.**
  // Put 0 in all elements above and below pivot
9 **for** $i \leftarrow 0$ **to** $r - 1$ **do**
10 **if** $i \neq x$ **and** $\mathbf{L}_{i,x} = 1$ **then**
11 $\mathbf{L}_{i,:} \leftarrow \mathbf{L}_{i,:} \oplus \mathbf{L}_{x,:}$
12 **return** $\mathbf{L}$

Pseudo-code for the classical implementation of Prange's algorithm, calling in its loop body the GJE function acting on the augmented matrix $\mathbf{L} = [\widehat{\mathbf{H}} \mid \mathbf{s}]$.

To note that this part of the probability can also be expressed as the division between $\binom{n-t}{r-t}$ and $\binom{n}{r}$ since, among all the $\binom{n}{r}$ choices of distinct $\mathcal{J}$, we need the one placing all the $n - t$ zeros in exactly $r - t$ positions of the r-length vector $\mathbf{e}_{\mathcal{J}}$. The second and last contribution to the probability of success of a single iteration is given by the probability that the leftmost $r \times r$ submatrix is invertible, which corresponds to finding an identity matrix after the GJE procedure. We recall that the probability that a random $r \times r$ binary matrix is non-singular is $\prod_{i=1}^{r}(1 - 2^{-i})$, a value quickly converging to ≈ 0.288 for increasing values of r.

The cost C_{iter} of a single iteration of Prange's algorithm is dominated by the GJE algorithm, detailed in Algorithm 2, applied to the augmented matrix $\mathbf{L} = [\widehat{\mathbf{H}} \mid \mathbf{s}]$. The procedure iterates on the r rows of $\mathbf{L}$ and, for each such a row, denoted by x, makes two different kinds of operations. The first one (Part 1 in the algorithm) corresponds to finding the first row $\mathbf{L}_{i,:}$, $x \leq i < r$, such that $\mathbf{L}_{i,x}$ is not null (lines 2–4). If no such row exists, then the random choice of $\mathcal{J}$ selected a singular $r \times r$ submatrix of $\mathbf{H}$, and the procedure aborts. Indeed, for all ISD algorithms, since we want to check on the invertibility of the submatrix $\mathbf{H}_{\mathcal{J}}$, it is not sufficient to only have the matrix in row-reduced echelon form—we specifically need the leftmost $r \times r$ submatrix of $\mathbf{L}$ matrix to be an identity matrix. When, instead, such a row $\mathbf{L}_{i,:}$ is found, the algorithm swaps it with $\mathbf{L}_{x,:}$, ensuring that $\mathbf{L}_{x,x} = 1$ (line 8). The second operation (Part 2 in the algorithm) sets to 0 all the elements above and below the pivot by adding the row $\mathbf{L}_{x,:}$ to all the other rows $\mathbf{L}_{i,:}$, $i \neq x$, whenever $\mathbf{L}_{i,x} = 1$. The computational complexity of Algorithm 2, as reported in Reference [54], is $\mathrm{C}_{GJE} = O(\frac{3nr^2}{4} + \frac{nr}{4} - \frac{n}{2} + \frac{3r^2}{4} - \frac{r}{2})$ bit operations. The remaining factor in the cost of Prange's algorithm, denoted as C_{HWCC}, is the cost of computing the Hamming weight of $\tilde{\mathbf{s}}$ and check that it has weight equal to t, which is in $O(r)$.

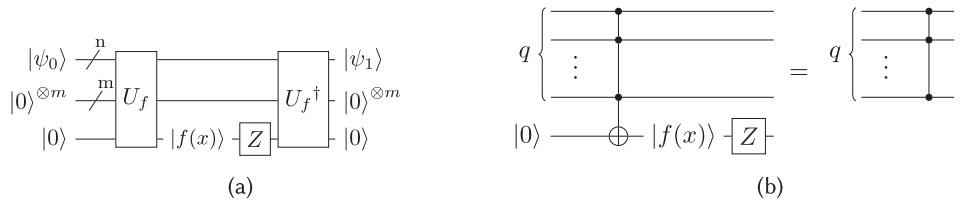


Fig. 1. (a) Implementation of Grover's oracle circuit U_O through the U_f. Starting from state $|\psi_0\rangle$, we obtain, at the end of the circuit, a state $|\psi_1\rangle$ that has the same basis states of $|\psi_0\rangle$ but a different sign on the amplitude of the searched basis state $|x^*\rangle$. (b) Two equivalent circuits to perform the sign flip on the amplitude of the basis state $|1^q\rangle$ stored on q qubits. While the first one requires a multi-controlled X gate and a Z gate using one additional ancilla qubit, the other requires only a multi-controlled Z without ancillae. The multi-controlled Z is represented in the rightmost quantum circuit without an explicit Z box as it alters the phase of the entire basis component if all controlling qubits are set to one.

Overall, the cost of Prange's algorithm is therefore

$$\frac{\mathrm{C}_{iter}}{\mathrm{Pr}_{succ-PR}} \approx \frac{\binom{n}{t}}{0.288\binom{r}{t}} \left(\mathrm{C}_{GJE} + \mathrm{C}_{HWCC}\right) \equiv \frac{\binom{n}{r}}{0.288\binom{n-t}{r-t}} \left(\mathrm{C}_{GJE} + \mathrm{C}_{HWCC}\right). \tag{1}$$

2.4 Grover's Algorithm

Grover's algorithm [31] solves the following search problem: Given a function $f : \{0,1\}^n \to \{0,1\}$ to which a black-box access is provided, find the unique argument x^* such that $f(x^*) = 1$. In the query-model, which is used to evaluate Grover's algorithm and to show its speedup, we are given a black-box access to a quantum circuit U_f implementing f. The goal is to use as few applications as possible of U_f to find x^*. In the classical computing paradigm, we have to query the function f—or, equivalently, the classical circuit implementing it—2^n times in the worst case and 2^{n-1} times on average. In contrast, the quantum algorithm proposed by Grover, after encoding all the bitstrings composing the domain of f as quantum basis states, performs only $\approx 2^{n/2}$ queries to retrieve x^*, achieving therefore a quadratic speedup. Grover's algorithm can be expressed using three main stages: (1) input preparation, (2) oracle, and (3) diffusion. By repeating (2) and (3) $\approx 2^{n/2}$ times, the probability of observing the basis state $|x^*\rangle$ encoding the wanted state x^* upon measurement get close to 1.

Input preparation stage. In the input preparation stage, a uniform superposition of all 2^n basis states belonging to the domain $\{0,1\}^n$ of f is prepared. Such a superposition is obtained through a layer of H gates applied to n input qubits, initially in state $|0\rangle$. After the application, we obtain

$$|\psi_0\rangle = H^{\otimes n} |0^n\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle .$$

If the domain of the function f is not composed by all the Boolean strings of length n, but it is instead $D \subset \{0,1\}^n$, then the input preparation stage is implemented with a dedicated quantum circuit U_D preparing a uniform superposition of only the basis states labeled as bitstrings belonging to D, i.e.,

$$|\chi_0\rangle = U_D |0^n\rangle = \frac{1}{\sqrt{|D|}} \sum_{x \in D} |x\rangle . \tag{2}$$

Oracle stage. In Grover's original framework, the oracle is considered as a black-box circuit implementing f and additionally changing the sign of the amplitude associated to the basis state

$|x^*\rangle$ encoding the input bitstring x^* we are looking for. The goal of this circuit, denoted as U_O, is to obtain a new state

$$|\psi_1\rangle = U_O\,|\psi_0\rangle = \frac{1}{\sqrt{2^n}}\left(-|x^*\rangle + \sum_{x\in\{0,1\}^n, x\neq x^*} |x\rangle\right),$$

which, for our proposed circuit, becomes

$$|\chi_1\rangle = U_O\,|\chi_0\rangle = \frac{1}{\sqrt{|D|}}\left(-|x^*\rangle + \sum_{x\in D, x\neq x^*} |x\rangle\right). \tag{3}$$

It is easy to see, from the previous equations, that the oracle operator can be expressed as $U_O = I - 2\,|x^*\rangle\,\langle x^*|$, a notation used to highlight that the unitary matrix denoting the quantum circuit U_O only changes the amplitude associated to $|x^*\rangle$, leaving all other basis states untouched. This operator, indeed, can be seen as the operator performing an inversion around the set of basis states orthogonal to $|x^*\rangle$.

The main ingredient of the oracle circuit is the quantum circuit U_f implementing the Boolean function f. The U_f circuit, after computing $f(x)$, stores the results on an additional qubit, initialized to $|0\rangle$, as $|f(x)\rangle$. For non-trivial functions, U_f also involves additional m qubits to carry on the computation in a reversible way. As shown in Figure 1(a), the flip in the sign of the amplitude of the wanted state performed by the oracle can be computed applying a Z gate on the qubit storing the result of the computation.

Usually, for the circuit implementing U_f, there is a multi-controlled X gate right to the end that, if the state of some set of q qubits is equal to $|1^q\rangle$, $1 \leq q \leq m+n$, sets the output qubit to $|1\rangle$. This part of the circuit, which we denote as $\mathrm{C}^q(\mathrm{X})$, is shown on the left side of Figure 1(b). However, since the goal of the overall U_O circuit is to perform a phase flip on the amplitude of $|x^*\rangle$, we can avoid the use of the additional ancilla qubit storing the result, directly using a multi-controlled Z gate, denoted as $\mathrm{C}^q(\mathrm{Z})$, involving only the q qubits, as shown on the right side of Figure 1(b). This alternative approach can be seen as using a different function $f' : D \rightarrow \{0,1\}^q$, that is such that $f : D \xrightarrow{f'} \{0,1\}^q \xrightarrow{g} \{0,1\}$, in which g computes the logic **and** of the bits composing the output of f'. In other words, f' produces an all 1's bitstring of length q if the same conditions of f are satisfied. The circuit $U_{f'}$ implementing f' will need therefore one less qubit.

As a final remark, we notice that we can express the whole oracle circuit as $U_O = U_{f'} U_{x^*} U_{f'}^\dagger$, in which U_{x^*} corresponds to applying the $\mathrm{C}^q(\mathrm{Z})$ gate involving the q qubits storing the results, while the quantum circuit implementing $U_{f'}^\dagger$ has the same gates of $U_{f'}$ but applied in reverse order and with complex conjugated parameters with respect to the $U_{f'}$.

Diffusion stage. The diffusion stage of Grover's algorithm is expressed as the operator $U_{\psi_0} = \mathrm{H}^{\otimes n} U_0 \mathrm{H}^{\otimes n}$ applied to the first n qubits. Operator U_0, expressed as $I - 2\,|0^n\rangle\,\langle 0^n|$, is analogous to the U_{x^*} of the oracle stage, performing an inversion around the state $|0^n\rangle$. As before, to implement it we have to use a multi-controlled Z gate, this time acting on all the n qubits storing the input of our circuit and therefore defined as $\mathrm{C}^n(\mathrm{Z})$. Generalizing the formulation of this stage to adapt it to our algorithm, we defined the diffusion stage as $U_{\chi_0} = {U_D}^\dagger U_0 U_D$, in which U_D is the quantum circuit we used to prepare our input.

Since the standard $\mathrm{C}^n(\mathrm{Z})$ gate flips the sign of the amplitude if and only if all the input qubits are in state $|1\rangle$, while we want instead a rotation if and only if all of them are in state $|0\rangle$, we need to apply a wall of X gates on all the n qubits before and after the $\mathrm{C}^n(\mathrm{Z})$ gate. We denote this circuit as $U_1 = \mathrm{X}^{\otimes n} U_0 \mathrm{X}^{\otimes n}$.

Number of iterations. Grover observed that, at each iteration, the amplitude of the sought state $|x^*\rangle$ increases by $\approx 1/\sqrt{2^n}$—which becomes $\approx 1/\sqrt{|D|}$ in the case of our modified algorithm—while at the same time the amplitudes of all other states decrease. After applying the oracle and diffusion stage $\approx O(\sqrt{2^n})$, we have a probability close to 1 to observe the wanted state $|x^*\rangle$ upon measurement. In Reference [16] it is shown how executing half of the said number of repetitions provides a success probability close to 50%. Moreover, they also show how the optimal number of iterations is indeed close to $0.58278 \cdot \sqrt{2^n}$, obtaining a success probability close to 0.84458. Finally, they also show how if, instead of having a single state $|x^*\rangle$ for which our function f evaluates to 1, we have M of them, the number of repetitions decreases by $\sqrt{M}$.

2.5 Sorting Networks

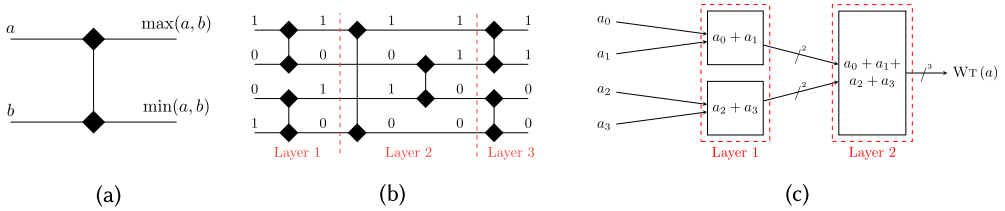
A key component of our oracle implementation is a restructuring of the classical circuit used to sort an n-length bitstring into its reversible variant. Different alternatives of such circuits, known as *sorting networks*, are extensively analyzed in Reference [43, Chap. 5.3.4]. All of them have as their building block a comparator element, depicted in Figure 2(a), that upon receiving two bits on its two input wires a and b outputs the maximum between them on the top wire and the minimum on the bottom one. The comparator can analogously be thought as a device that, taking as input two wires a and b, swaps them only if $a < b$, while it does nothing in all other cases. The idea underlying a sorting network is to use a fixed sequence of such comparators, organized in layers as shown in Figure 2(b), to obtain at the end of the circuit a bitstring with all the 1's on top and all 0's on the bottom. The values on the wires at the end of the circuit, read from bottom to top, represent the sorted version of the input bitstring.

All comparators in a layer act on different pairs of wires. A comparator object can produce an output value only when both of the values on its input wires are ready. If we assume that the computation of each comparator takes a fixed amount of time, which is realistic given that all the comparators have the same identical structure, then the running time of the overall sorting network is given by the largest number of comparators acting on any wire, a quantity known as depth. Since inside each layer all the comparators can be run in parallel, the total depth of the network is given by the number of layers, and it is not directly influenced from the total number of comparators. Distinct amounts and configurations of comparator layers can be employed to achieve the sorting of the n values present on the input wires. The two main factors to consider when evaluating the efficiency of a sorting network are the number of comparators used and the degree of parallelization that they can reach. Indeed, there exist several sorting networks specialized for fixed n, minimizing either the amount of comparators or the depth. Since our primary objective is to embed the sorting network in a quantum circuit in the most general way, we focused on generalizable proposals minimizing the overall depth.

For bitstrings having $n \approx 2^{10}$, as it happens in our scenario, [1] reports a design having an asymptotically optimal depth, i.e., $O(\log(n))$. However, Knuth reports [43, Chap. 5.3.4] that this network design is not of practical interest, since the constants hidden by the asymptotic notation are significant. We employ therefore the sorting network topology detailed in Reference [19, Chap. 27.5], where a careful analysis of the network gives a total number of comparators equal to $(n-1)\log_2(n)(\log_2(n)-1)$ and a total depth of $\frac{1}{2}\log_2(n)(\log_2(n)+1)$. Figure 2(b) shows an example of such a network applied to an $n = 4$-length input bitstring.

2.6 Hamming Weight Computation

In our quantum SDP solver, we need to compute the Hamming weight of a binary string. This computation can be efficiently carried out by a binary tree composed of integer adders with a



(a) (b) (c)

Fig. 2. (a) Comparator element employed in a sorting network. (b) A 4-wires sorting network, used to sort a 4-bits string. (c) A binary adder tree computing the Hamming weight of a 4-bits string.

suitable operand width. The underlying concept behind the classical circuit known as binary adder tree is to use a cascade of adders, organized in layers composed of adders of increasing sizes, to perform the sum of two input bitstrings. We exploit the binary adder tree design to compute the Hamming weight of an r-length bitstring, with r being a power of two.

The number of layers of an adder tree circuit is $\log_2(r)$. The first layer is composed by $r/2$ adders adding together two distinct single bits of the original bitstring. All the following layers compute a sum based on the output of the previous layers, adding their results in pairs and hence involving addends whose bitsize increases by 1 with respect to the previous layer. In other words, at layer i, each adder of the ith layer takes as input two distinct length-i bitstrings and produces in output an $i+1$ bitstring, $1 < i \leq \log_2 r$. An example of such a circuit is shown in Figure 2(c).

In the ith layers, $r/(2^i)$ adders are employed, leading to a total count of $\sum_{i=1}^{\log_2(r)} r/2^i = r - 1$ adders. The overall number of gates required is given by

$$\sum_{i=1}^{\log_2(r)} \frac{r}{2^i} \cdot \textsc{AdderGates}(i), \tag{4}$$

where $\textsc{AdderGates}(i)$ denotes the gate cost of an adder taking as inputs two i-length bitstrings.

The tree structure allows taking advantage of parallelism, reducing the overall depth of the circuit. All the adders acting at the same level have an identical structure and, acting on different qubits, can be run in parallel. For this reason, the overall depth of the circuit is

$$\sum_{i=1}^{\log_2(r)} \textsc{AdderDepth}(i), \tag{5}$$

where $\textsc{AdderDepth}(i)$ denotes the depth of an adder taking as inputs two i-length bitstrings.

3 A QUANTUM CIRCUIT FOR PRANGE'S ISD

Notation. We will denote a set of qubits composing a quantum register as `qreg`. Since, in our work, a quantum register is used to hold either a binary matrix $\mathbf{M}$ or a binary vector $\mathbf{v}$, we will use the same notation for matrices and vectors introduced at the beginning of Section 2. Thus, $\mathtt{v}_i$ will denote the qubit of $\mathtt{v}$ corresponding to the element at position i in the vector $\mathbf{v}$, while $\mathtt{M}_{i,j}$ the qubit corresponding to element $\mathbf{M}_{i,j}$ of the matrix. We extend the colon notation to the quantum registers as well, therefore using $\mathtt{M}_{i,:}$ ($\mathtt{M}_{:,j}$) to denote the set of qubits corresponding to all the elements of $\mathbf{M}_{i,:}$ ($\mathbf{M}_{:,j}$). Analogously, $\mathtt{M}_{\mathcal{S}_1,:}$ ($\mathtt{M}_{:,\mathcal{S}_2}$) is used to express the elements belonging to $\mathbf{M}_{\mathcal{S}_1,:}$ ($\mathbf{M}_{:,\mathcal{S}_2}$).

3.1 Adapting Grover's Framework to the ISD Technique

To employ Grover's framework to accelerate the Prange ISD solver, we rephrase the ISD strategy as a search procedure for a solution of the Boolean function $f' : D \rightarrow \{0,1\}^q$, in which $D \subset \{0,1\}^n$ is

defined as the set of all binary vectors of length n having weight t, $D = \{\mathbf{v} \mid \mathbf{v} \in \{0,1\}^n, \mathrm{Wt}(\mathbf{v}) = t\}$. The size of our domain is $|D| = \binom{n}{t}$, since this is the number of distinct vectors of length n having weight t. The choice of such a domain is motivated by the use of the vectors belonging to it as a way to represent the Information Set complement $\mathcal{J}$. That is, $\mathcal{J} = \{j \mid \mathbf{v}_j = 1, \mathbf{v} \in D\}$, which expresses that the set of all the indexes of the asserted bits of a vector belonging to the domain corresponds to $\mathcal{J}$. The function f' will output the bitstring 1^q if and only if (i) the given choice of $\mathcal{J}$ selected a matrix $\mathbf{H}_{:,\mathcal{J}}$ that is invertible and (ii) the syndrome vector $\tilde{\mathbf{s}}$ has weight equals to t.

In the following, we also detail how to adapt our quantum circuit to the case in which we have $M > 1$ syndromes at our disposal, all associated to the same parity-check matrix $\mathbf{H}$ and error-weight t, which is useful to accelerate the attack against the circulant codes presented in Section 2.2. We can indeed adapt the algorithm performing GJE presented in Algorithm 1 by building an augmented matrix $\mathbf{L}$ of size $r \times (n + M)$, at the cost of an increased number of qubits.

As common in the literature, we assume that all the qubits of our quantum circuit are initially in state $|0\rangle$.

Input circuit: superposition of all permutations. The first challenge in adapting Grover's algorithm to the ISD problem is to have a circuit U_D capable of generating a uniform superposition of all the $\binom{n}{r}$ bitstrings of length n and weight r, thought as labels of $\binom{n}{r}$ distinct basis states. Such a superposition, that will be stored on a quantum register called `inp` of size n, can be expressed as

$$|D_r^n\rangle = \frac{1}{\sqrt{\binom{n}{r}}} \sum_{\mathrm{Wt}(x)=r} |x\rangle\,,\ x \in \{0,1\}^n. \tag{6}$$

Many proposals result in a quantum circuit capable of producing such a state, known as the Dicke state. To the best of our knowledge, the most efficient circuit producing a Dicke state in a deterministic way is the one reported in Reference [50]. This work improved the previous best result provided in Reference [7] both in terms of number of gates and depth. The overall circuit for the input preparation stage, which we denoted as U_D, requires r X, $5nr - 5r^2 - 2n$ CNOTs and $4nr - 4r^2 - 2n + 1$ $\mathrm{R_y}$ gates. By analyzing the circuit carefully and by using the same considerations made for the depth evaluation in Reference [7], we can derive an upper bound on the total depth of the circuit as the sum of the CNOTs and $\mathrm{R_y}$ gates, divided by $\lfloor \frac{r+1}{3} \rfloor - 1$. The resulting depth for the input preparation circuit is therefore $\leq \frac{27nr-12n-27r^2+3}{r-2} \in O(n)$.

Oracle operator. In this section, we present the components composing our oracle circuit, denoted as a whole as U_O. As explained in Section 2.4, we can describe the unitary representing this circuit as $U_O = U_{f'} U_{x^*} U_{f'}^\dagger$. The first four components compute $U_{f'}$ and produce the state $|1^q\rangle$ on a set of q qubits, contained in a quantum register named `out`. Since the circuit corresponding to $U_{f'}$ does not contain phase rotations, but only classical reversible gates, the circuit corresponding to $U_{f'}^\dagger$ corresponds to the application of the gates belonging to $U_{f'}$ in reverse order. Finally, the circuit corresponding to U_{x^*}, which can be implemented by simply applying a $\mathrm{C}^q(\mathrm{Z})$ gate on the set of qubits belonging to `out` for $M = 1$, has a slightly different structure when we are encoding a number of syndromes M greater than 1.

Data encoding: H and s as quantum states. The first subcircuit of the oracle encodes the parity-check matrix $\mathbf{H}$ and the syndrome $\mathbf{s}$ in a quantum state. Since the input matrices only contains bits, it is straightforward to put them in one-to-one correspondence with qubits basis states $|0\rangle$ and $|1\rangle$. We denote the qubits representing $\mathbf{H}$ as `H` and the ones representing $\mathbf{s}$ as `s`. Since those set of qubits are in state $|0\rangle$ at the beginning of the circuit, we need to apply an X gate each time our input binary data contains a 1. Given that $\mathbf{H}$, with size $r \cdot n$, and $\mathbf{s}$, with size r, are both random-looking, we

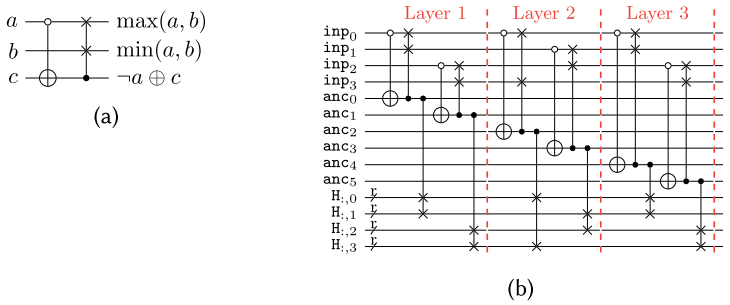


Fig. 3. (a) The quantum comparator used for the circuit of Section 3.1, swapping the two qubits a and b only if a is in state $|0\rangle$ and b in state $|1\rangle$. (b) The quantum sorting network corresponding to the example of Figure 2(b). Each of the qubits of inp is used to address a set of r qubits of H, corresponding to a single column of the parity-check matrix, allowing to swap its columns together with the inp register.

expect on average to have $M(rn + r)/2$ bits set to 1, and therefore the same number of X gates in their quantum encoding. Since all the gates can be applied in parallel, the depth of this stage is 1.

Column permutation: from H to $\widehat{\mathbf{H}}$. The superposition obtained on the inp register will be composed by basis states labeled as bitstrings of length n and weight r. The indexes of the qubits of inp having state $|1\rangle$ correspond to the Information Set complement $\mathcal{J}$, and we logically bind each qubit of inp to a set of qubits of H corresponding to a column of $\mathbf{H}$. This stage, corresponding to line 3 of Algorithm 1, moves the columns of $\mathbf{H}$ indexed by $\mathcal{J}$ to the left of the matrix $\mathbf{H}$, obtaining the matrix $\widehat{\mathbf{H}} = [\mathbf{H}_{\mathcal{J}} \mid \mathbf{H}_{I}]$ on the qubits of H. This is equivalent to sorting the bitstring stored in inp in descending order, simultaneously moving the associated qubits of H corresponding to the columns of $\mathbf{H}$. For this reason, we designed a quantum circuit to sort the bitstring stored in inp, for which we can exploit the reversible sorting network circuit presented in Section 2.5. Figure 3(a) depicts our proposal of a quantum circuit implementing a comparator element, denoted as *quantum comparator*. It uses one additional qubit, denoted c, that, starting from state $|0\rangle$, will be put in state $|1\rangle$ if the qubit denoted as a is in state $|0\rangle$. Only in this case do we swap the amplitudes of the quantum state associated to a and b.

The gate count of a quantum column permutator, of which a circuit example is given in Figure 3(b), can be trivially derived by adding together the gate count of a single quantum comparator together with the r additional swaps needed to swap the r elements of the column, i.e., 2 X gates (to convert the positive to a negative control), 1 CNOT gate, and $r + 1$ CSWAP gates. Analogously to the classical case, the gates belonging to distinct quantum permutators acting at the same layer can be executed in parallel, as they compare and swaps different sets of qubits. Therefore, for the depth computation, we can focus only on a single quantum column permutator per layer. A significant reduction in the circuit depth is achieved rescheduling the CSWAP gates acting on H. Indeed, the CSWAP gates inside a single quantum column permutator are sequentially locked as they employ the same (ancilla) control qubit. By contrast, CSWAP gates from different quantum permutators—acting on different pairs of elements of H—can be executed in parallel, as they employ different (ancillae) controls. Since each permutator requires r CSWAPs for the H register, we can conclude that the number of CSWAPs required by a single quantum column permutator is less than the number of overall permutators, $(n-1)\log_2(n)(\log_2(n)-1)$. As a result, we can interleave the swaps associated to different permutators, circumventing the dependence of the CSWAPs from the same control qubit. This scheduling leads to a total depth of $\log_2^2(n) + \log_2(n) + r - 1$.

Quantum Gauss–Jordan elimination: from $[\widehat{\mathbf{H}} \mid \mathbf{s}]$ to $[\widetilde{\mathbf{H}} \mid \tilde{\mathbf{s}}]$. One of the major challenges in porting an ISD algorithm in its quantum form is the ability to find a reversible variant of the

ALGORITHM 3: Reversible Gauss–Jordan Elimination

Input : $\mathbf{L} \in \mathbb{F}_2^{r \times n+1}$: augmented matrix, $n > r$
Data : $\mathbf{B} \in \mathbb{F}_2^{r(r-1)/2}$: auxiliary vector initialized to all 0's
$\mathbf{C} \in \mathbb{F}_2^{r(r-1)}$: auxiliary vector initialized to all 0's
Output : $\mathbf{L}$ in reduced-row echelon form

1 $b \leftarrow 0$, $c \leftarrow 0$
2 **for** $x \leftarrow 0$ **to** $r-1$ **do** // x-iteration
// **Part 1.** Put 1 in pivot position if it contains a 0
3 **if** $x \neq r-1$ **then**
4 **for** $i \leftarrow x+1$ **to** $r-1$ **do** // for all rows below pivot row
5 **if** $\mathbf{L}_{x,x} = 0$ **then** // swap operation must be performed
6 $\mathbf{B}_b = \mathbf{B}_b \oplus 1$ // $\equiv \mathbf{B}_b = 1$
7 **if** $\mathbf{B}_b = 1$ **then** // or equivalently, $\mathbf{L}_{x,x} = 0$
8 swapRow($\mathbf{L}_{x,:}, \mathbf{L}_{i,:}$)
9 $b \leftarrow b+1$
// **Part 2.** Put 0 in all elements above and below pivot
10 **for** $i \leftarrow 0$ **to** $r-1$ **do**
11 **if** $i = x$ **then continue**
12 **if** $\mathbf{L}_{i,x} = 1$ **then** // column addition operation must be performed
13 $\mathbf{C}_c = \mathbf{C}_c \oplus 1$ // $\equiv \mathbf{C}_c = 1$
14 **if** $\mathbf{C}_c = 1$ **then** // or equivalently, $\mathbf{L}_{i,x} = 1$
15 $\mathbf{L}_{i,:} \leftarrow \mathbf{L}_{x,:} \oplus \mathbf{L}_{i,:}$
16 $c \leftarrow c+1$

GJE. Indeed, as we saw in Section 2.3, the dominant cost of one iteration of Prange ISD variant is associated to this stage. At this point, we have a matrix $\widehat{\mathbf{H}} \in \mathbb{F}_2^{r \times n}$ and a vector $\mathbf{s} \in \mathbb{F}_2^r$, encoded in the state of the quantum registers H and S, respectively. The goal is to perform a set of row operations on the augmented matrix $\mathbf{L}_{r \times (n+1)} = [\widehat{\mathbf{H}} \mid \mathbf{s}]$, obtaining at the end a new matrix $\widetilde{\mathbf{H}} = [\widetilde{\mathbf{H}} \mid \tilde{\mathbf{s}}]$. In a later stage, we have to check that the obtained matrix $\widetilde{\mathbf{H}}$ has an $r \times r$ matrix in its leftmost portion. In the general case of $M > 1$ distinct syndromes, the augmented matrix $\mathbf{L}$ has size $r \times (n+M)$, since we want to represent all the possible M syndromes on the right portion of the augmented matrix. However, all the additional syndromes, being treated as just other columns of $\mathbf{L}$, have no impact on the description of the algorithm, and hence we will continue to describe it for the basic case in which $M = 1$.

The main challenges to the translation of Algorithm 2 in an appropriate reversible algorithm are as follows: (C1) a data-dependent early-abort (line 6), (C2) the presence of a non-countable loop to search for a pivot (line 3), and (C3) conditional operations acting on data also present in the condition calculation. Algorithm 3 is semantically equivalent to Algorithm 2, but it also takes care of removing all the previous challenges. Challenge (C1) is indeed common to all quantum algorithms. Since we act on a quantum state composed of a superposition of distinct basis states encoding both a failure and a success of the algorithm, we have no way to remove the basis states encoding a failure from the superposition without compromising the whole superposition. To overcome the problem, we need, later in our quantum circuit, an additional piece capable of detecting the failure and hence act accordingly without performing any external observation. In our specific case, the early abort of Algorithm 2 due to a singular submatrix in the leftmost $r \times r$ part of $\widehat{\mathbf{H}}$ is managed through an *a posteriori* check on the presence of an identity submatrix in this portion of $\widetilde{\mathbf{H}}$, and we therefore removed altogether the runtime check in Algorithm 3.

Challenge (C2), i.e., the presence of a non-countable loop to find the pivot (lines 3 of Algorithm 2), has to be tackled with a countable loop. Our strategy consists into having a loop sweeping over

all the rows i, with i starting from the one right below the pivot row x until the last row $r-1$, and swapping the ith row with the xth one only if the xth row does not contain a valid pivot. This is shown in line 4 of Algorithm 3.

The presence of conditional operations in our circuit, which was our challenge (C3), is overcome employing auxiliary vectors, namely $\mathbf{B}$ and $\mathbf{C}$. To explain why they are needed, we first analyze the conditional operations present in our classical algorithm. Classical conditional operations (i.e., if conditions) are generally translated in reversible operations through the help of controlled gates. For example, each swap operations at line 8 of Algorithm 3 can be directly implemented with n CSWAP gates, having as control one qubit belonging to $\mathbf{B}$ and target a pair of elements belonging to columns $\mathbf{L}_{x,:}$ and $\mathbf{L}_{i,:}$. In that operation, the goal is to apply the CSWAP gates only if the pivot of the row under analysis ($\mathbf{L}_{x,x}$) is in state $|0\rangle$. The use of an additional ancilla belonging to $\mathbf{B}$ is required because it is not possible to apply the CSWAP between the qubits containing $\mathbf{L}_{x,x}$ and $\mathbf{L}_{i,x}$, having $\mathbf{L}_{x,x}$ acting both as target and as control of the controlled gate. Therefore, if $\mathbf{L}_{x,x}$ is in state $|0\rangle$, then we set to $|1\rangle$ the value of the corresponding ancillary qubits containing $\mathbf{B}$ (line 5), using its value later to control the set of n CSWAPs. Overall, to take into account all the pivot checks, we need a number of ancillary qubit equal to the number of times we execute part 1 of Algorithm 3. Following the same line of reasoning, the additional vector $\mathbf{C}$ is required to store, in part 2, the results of checking if the elements below and above the pivot are equal to $|1\rangle$. The sizes of the new ancillary registers are therefore evaluated to be

$$|\mathbf{B}| = \sum_{x=0}^{r-2}\left(\sum_{i=x+1}^{r-1} 1\right) = \frac{1}{2}r(r-1) \qquad |\mathbf{C}| = \sum_{x=0}^{r-1}\left(\sum_{i=0}^{r-2} 1\right) = r(r-1),$$

and we will use $|\mathbf{B}|$ and $|\mathbf{C}|$ to denote as well the number of times we execute parts 1 and 2, respectively.

Another point worth stressing is related, once again, to line 8. Indeed, the condition of line 7, unlike the classic algorithm, does not check that the row i under analysis also contains a valid pivot but only that the current pivot of row x is not valid. This means that if row x does not have a valid pivot, then we will swap rows i and x even if row i does not contain an alternative valid pivot. While for the candidate pivot of row x nothing changes, since we are exchanging qubits in the same basis state, the rest of the row will change, since we are performing a swap on all the elements. However, the only goal of part 1 is to set the pivot to $|1\rangle$, while part 2 will put all the elements below and to the left of the pivot—and hence also the remaining elements of row x—to $|0\rangle$ regardless of their previous state.

The circuit corresponding to Algorithm 3 is shown in Figure 4, which is similar to the one reported in Reference [54]. In this circuit, we stored the two vectors $\mathbf{B}$ and $\mathbf{C}$ inside the two quantum registers denoted as b and c. Remember also that the values of the matrix $\mathbf{L}$, containing $[\widehat{\mathbf{H}} \mid \mathbf{s}]$ at the beginning of the computation and the matrix $[\widetilde{\mathbf{H}} \mid \tilde{\mathbf{s}}]$ at the end, are stored on the two quantum registers H and s.

The obtained quantum circuit requires an overall number of ancillae equals to $|\mathbf{B}| + |\mathbf{C}| = 3r(r-1)/2$. For what concerns the gate count, first we note that to translate each $|0\rangle$-control on the pivot elements of part 1 into $|1\rangle$-control, we have to use a number of X equals to $2|\mathbf{B}| = r(r-1)$. To check that the qubit of the candidate pivot is $|0\rangle$ in part 1 and the elements under pivot are $|1\rangle$ in part 2, we need to drive a CNOT targeting an ancilla (belonging to $\mathbf{B}$ or $\mathbf{C}$, respectively). The total number of CNOT required is therefore $|\mathbf{B}| + |\mathbf{C}| = 3r(r-1)/2$. To swap elements of each row in part 1 and to sum elements of each row in part 2, we need a total of $|\mathbf{B}| \cdot (n+M) = \frac{1}{2}r(r-1)(n+M)$ CSWAP and $|\mathbf{C}| \times (n+M) = r(r-1)(n+M)$ CCNOT, respectively.

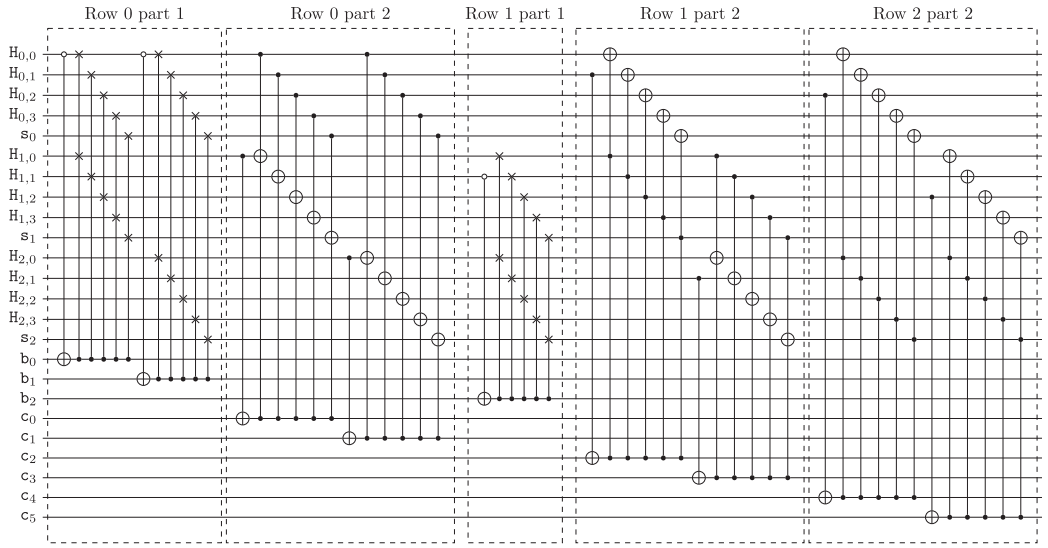


Fig. 4. Basic quantum circuit employed to perform the Gauss–Jordan elimination on the augmented matrix $[\widehat{\mathbf{H}} \mid \mathbf{s}]$ having $r = 3$ and $n = 4$. The quantum register containing the elements of the parity-check permuted matrix is denoted by H, while the one containing the elements of the syndrome is denoted by s. b and c are the registers associated to the two vectors $\mathbf{B}$ and $\mathbf{C}$ employed in Algorithm 3.

Compute Hamming weight of $\tilde{\mathbf{s}}$. At this stage, the s register contains the representation of $\tilde{\mathbf{s}}$. To check if condition (ii) is true, that is, the Hamming weight of this register is t, we have to check that the bitstring labeling the quantum basis states of s has weight equal to t. To perform such a computation, we use a reversible variant of the classical binary adder tree design presented in Section 2.6. Numerous papers have been published to show quantum circuits to perform integer additions between two x qubit operands on quantum computers. The proposal in Reference [23] exploits the Quantum Fourier Transform algorithm to perform quantum integer addition. The main drawback of this approach is the use of controlled arbitrary rotation gates, which are difficult to be translated into fault-tolerant gates, such as the Clifford+T gate set.

Subsequently, other approaches were proposed, porting techniques from the classical world into reversible circuits. These techniques all rely on the use of X, CNOT, and CCNOT gates, that have the main advantage of being less expensive to be implemented in a fault-tolerant fashion. For example, the Cuccaro adder presented in Reference [20] has a depth of $O(2x)$ and uses two single ancilla qubits, while the one proposed in Reference [63] increases the depth to $O(5x)$, while not using any additional qubit. The first adder to go under the linear depth threshold was proposed in Reference [24], which has the main drawback of requiring $2x$ additional qubits.

Since for all the proposed adders the depth of the adder circuit grows at most linearly with the number of input qubits, and given that the widest adder employed in the Hamming weight check circuit has a number of input qubits equal to $\log_2(r)$, choosing an adder with respect to the other does not have a significant impact on the depth of the global circuit, which will always stay $\in O(\log_2^2 r)$. However, since the number of adders required is $r - 1$, the use of a smaller number of qubits for each adder is expected to have a bigger impact on the overall measures. We decided therefore to use the adder presented in Reference [63], a reversible variant of the classical ripple carry adder; we will refer to it as TTK adder. The TTK adder stores the sum of its input a and b in the qubit where b is stored, with one additional qubit c for the carry-out. We need therefore

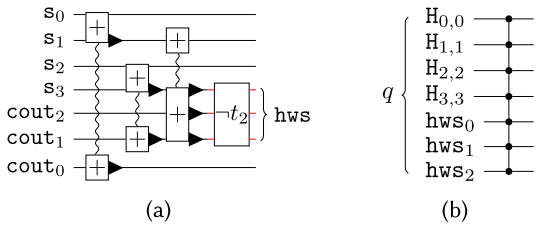


Fig. 5. (a) Hamming weight compute and check subcircuits for the s register. The result of the addition at each step is stored on the qubits marked with a black triangle. The final result is compared against the Boolean complement of the natural binary representation of the constant t. (b) The oracle phase flip, involving the qubits containing the diagonal of matrix $\widetilde{\mathbf{H}}$ and the ones storing the Hamming weight of s.

exactly $r-1$ carry-out ancillary qubits, and Figure 5(a) shows an example for the case in which $r=4$.

To determine the gate count for this subcircuit, we note that a TTK adder operating on i qubit inputs requires $2i-1$ CCNOT gates, $5i-5$ CNOT gates, and has a circuit depth of $5i-3$. Replacing these figures in Equations (4) and (5), we obtain the gate-count and depth reported in Table 1. Notice how the number of gates and additional qubits required by this stage have to be multiplied by M. However, since all the M syndromes are stored on distinct qubits belonging to $\tilde{\mathbf{s}}$, all the operations can be performed in parallel, and as such, the depth is not influenced.

At level i, the output of the sum of a single adder is stored on exactly $i+1$ qubits, as TTK's design reuses the qubits of one of the operands to store part of the result. Hence, in the final stage, the final sum of a single syndrome will be stored on a register composed of $\log_2(r)+1$ qubits, denoted as hws. In the case of $M>1$, the result of the M distinct adder trees will be stored on M distinct qubits registers, denoted as hws^i, each of which has size $\log_2(r)+1$. At this point, we should check if hws—or one of the hws^i if $M>1$—contains the binary representation of t. To this end, we XOR into the qubits of hws the Boolean complement of the natural binary representation of t. This is done to ensure that if a given state is such that $\mathrm{WT}(\mathsf{hws})=t$, then all the qubits contained in the hws register will become $|1\rangle$. In this way, they can be used as control qubits in the multi-controlled gate of the next stage. This operation is therefore performed via a set of X gates, as the number being added (at most $\log_2(r)+1$ if $t=0$) is smaller than r and can thus be represented on the same number of qubits. We need therefore to use $\log_2(r)+1-\log_2(t)=\log_2(r/t)-1$ X gates to perform the XOR. Thus, if the output of previous stage contains a state with the binary encoding of t, then hws will be in the basis state $|1^{\log_2(r)+1}\rangle$. Once again, for circulant codes, the figures for the X should be multiplied by M, while the depth stays unchanged.

Phase flip. To perform the phase flip of the oracle circuit, we need to put a set of q qubits in state $|1^q\rangle$ if both conditions (i) and (ii) are satisfied. Condition (i) can be verified by simply checking that the r qubits of H containing the main diagonal of $\widetilde{\mathbf{H}}$ are in state $|1\rangle$. Indeed, given the description of the QGJE circuit of Section 3.1, it is enough to check that all the diagonal elements are in state $|1\rangle$ to ensure that there is an identity stored in the portion of H containing the leftmost $r\times r$ part of $\mathbf{L}$, hence ensuring that the original matrix was invertible. Condition (ii), however, checks that the weight of the syndrome $\tilde{\mathbf{s}}$ is equal to t. The previous stage of the oracle ensured indeed that if this is the case, then the $\log_2(r)+1$ qubits of hws were all in state $|1\rangle$. As explained in Section 2.4, at this point, the flip in the sign of the amplitude associated to the basis state containing $|x^*\rangle$ in the first n qubits can be obtained by performing a multi-controlled Z gate, $\mathrm{C}^q(\mathrm{Z})$, applied on all the $q=r+\log_2(r)+1$ qubits storing the output of the oracle function, as shown in Figure 5(b).

For circulant codes, additional care must be taken. Indeed, at this point, we should check, as before, that the r qubits on the diagonal of $\widetilde{\mathbf{H}}$ are in state $|1\rangle$. Differently from before, however,

we should also check all the distinct M syndromes obtained at the end of the QGJE procedure to check if (at most) one of them has the correct weight t. Since the syndromes are obtained from the shift of the single original syndrome $\mathbf{s}$, we are sure that all the distinct M syndromes with the correct weight will be associated to a distinct choice of $\mathcal{J}$, corresponding to a simple (and known) shift of the original $\mathcal{J}$ found for $\mathbf{s}$. For this reason, instead of involving M distinct $\mathrm{C}^q(\mathrm{Z})$ gates, each one controlled by the same r qubits of the identity, but distinct $\log_2(r)+1$ qubits associated to the Hamming weight check results of the previous stage, we can use a single $\mathrm{C}^r(\mathrm{X})$ gate for the identity check, that has the aim of setting an additional ancilla to $|1\rangle$ if the identity check is positive. Then, we can perform M distinct $\mathrm{C}^{\log_2(r)+2}(\mathrm{Z})$ gates, having in common only the single ancilla of the previous stage, while using M distinct sets of $\log_2(r)+1$ qubits storing the weight of the M syndromes. The depth, for this reason, is equal to M.

Diffusion operator. The diffusion operator employed by our circuit, as already explained in Section 2.4, can be expressed as $U_D^\dagger U_1 U_D$, in which U_D is the quantum circuit preparing the Dicke state explained in Section 3.1 and $U_D^\dagger$ is the same sequence of gates applied in reverse order, with a sign change in the angles of the $\mathrm{R_y}$ gates. The circuit corresponding to U_1, once again, can be implemented as a $\mathrm{C}^n(\mathrm{Z})$ gate applied on the input qubits and two walls of X gate before and after it.

Number of repetitions. As already seen in Section 2.4, the Oracle and Diffusion operators must be repeated roughly $\sqrt{\frac{|D|}{0.288 \cdot M \cdot \binom{n-t}{r-t}}}$ times, with the numerator denoting the size of our domain—i.e., $\binom{n}{r}$—and the denominator the number of solutions. If $M = 1$, then the number of solutions is trivially $.288\binom{n-t}{r-t}$, as we saw in Section 2.3. For circulant cryptographic schemes, having the representation of $M > 1$ distinct syndromes allows us to solve M distinct instance of the syndrome decoding problems, each of which has the same starting parity-check matrix $\mathbf{H}$ and the same error weight t and, most importantly, a single solution. This strategy, known as *decoding one out of many*, or DOOM for short, and presented for the first time in Reference [60], translates into M distinct solutions for our function and a quantum speedup of $\sqrt{M}$. To conclude, following the observations highlighted in Section 2.4, we fix the number of repetitions of the Oracle and Diffusion stage to $0.58278 \cdot \sqrt{\frac{|D|}{0.288 \cdot M \cdot \binom{n-t}{r-t}}}$.

3.2 Reducing Width and Depth of the Quantum Circuit for QGJE

The authors of Reference [54] propose some optimizations to the basic implementation of Section 3.1. In this section, we propose a set of optimizations to the QGJE circuit that encompass the ones of Reference [54] and carry the savings further. Since all the optimizations focus on the leftmost $r \times k$ portion of the augmented matrix $\mathbf{L}$—i.e., the portion storing the parity-check matrix $\mathbf{H}$—we will describe them disregarding the rightmost part of $\mathbf{L}$, containing the M syndromes (see also Figures 6(a), 6(b), 7(a) and 7(b)). We take Algorithm 3 and its quantum circuit implementation shown in Figure 4 as starting points.

***Optimization 1*—CSWAP *to* CCNOT.** It replaces all the CSWAPs of part 1 with CCNOTs. Indeed, the computation of the swap control condition and the actual swaps (lines 7 and 8) can be replaced by the bitwise sum $\widetilde{\mathbf{H}}_{x,:} = \widetilde{\mathbf{H}}_{x,:} \oplus \widetilde{\mathbf{H}}_{i,:}$, conditioned on the value of $\widetilde{\mathbf{H}}_{x,x}$ being 0. This operation, although yielding a linear combination for row x containing the pivot element as opposed to a simple row swap, has nonetheless the effect of setting the pivot to $|1\rangle$, which is the goal of part 1 of the QGJE algorithm. The underlying assumption of this optimization is that, despite common universal quantum gate sets include neither CSWAP nor CCNOT, they offer better translations of the CCNOT in terms of both number of gates and depth with respect to the CSWAP.

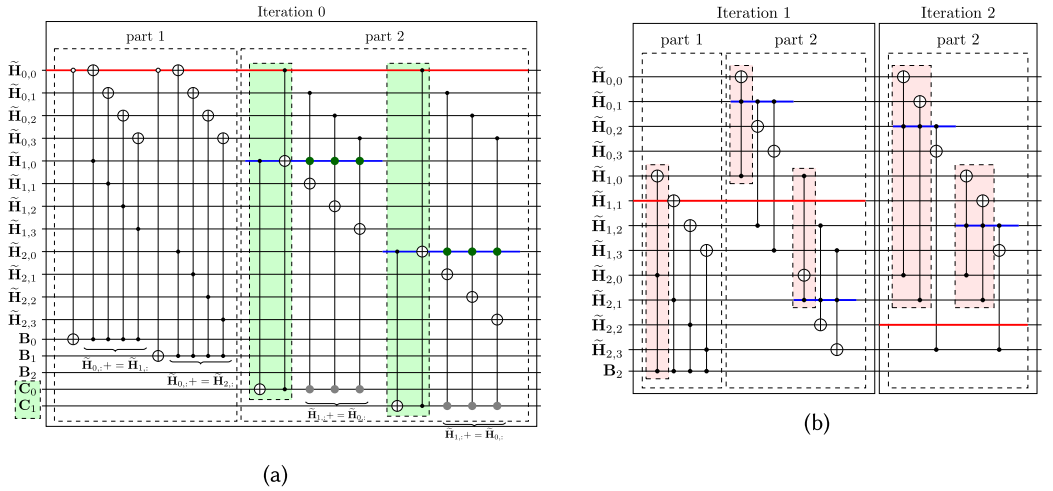


Fig. 6. Quantum subcircuits highlighting the first three optimizations. The red wires show the qubits corresponding to the pivot element of the row that we need to set to $|1\rangle$ (part 1), while the blue wires the qubit that we need to set to $|0\rangle$ (part 2). (a) The effect of optimization 1, which replaces all the swaps of part 1 with CCNOTs. Additionally, the green boxes highlight the gates and qubits that can be removed thanks to optimization 2. Building up on the previous optimizations, the red boxes in (b) show the gates that can be removed after optimization 3.

Optimization 2—Avoid clearing pivot column. It substantially improves part 2 by deleting all the CCNOTs involving the elements in the same column of the pivot ones. In part 2, when we find an element in position $\widetilde{\mathbf{H}}_{i,x}$ with value $|1\rangle$, we should add to row i under analysis the pivot row x, to set it to $|0\rangle$. However, this addition is not required for the element $\widetilde{\mathbf{H}}_{i,x}$ itself, since we will not use its value in the rest of the circuit, as, to check the identity later, we will just use the value of the r qubits containing the diagonal of $\widetilde{\mathbf{H}}$. As a final result, the whole vector $\mathbf{C}$ is now useless, since all the CCNOT gates can directly use as control qubit the one $\mathbf{H}_{i,x}$. This optimization leads therefore to a reduction of $|\mathbf{C}| = r(r-1)$ CCNOTs and the same number of qubits. For simplicity, we will still use $|\mathbf{C}|$ to refer to the total number of part 2.

Optimization 3—Removing 0-controlled gates. After part 2 of each outer iteration—that we call x-iteration—all the qubits corresponding to elements of the matrix at column x (except the one containing the pivot) will be in state $|0\rangle$. For this reason, all the CCNOTs involving them in the following x-iterations are useless.

Optimization 4—Skipping computation of last r columns. It is only valid for the Prange variant of ISD algorithm, in which we do not need the rightmost $r \times k$ submatrix of the row-reduced $\widetilde{\mathbf{H}}$ matrix at the end of the computation but only the leftmost $r \times r$ one. For this reason, we can completely avoid all the row additions (i.e., CCNOT gates) involving those columns. In other words, for each of the $|\mathbf{B}| + |\mathbf{C}| = \frac{3}{2}r(r-1)$ operations of parts 1 and 2, we can spare k CCNOT gates.

Optimization 5—Removing redundant X gates. In part 1 of Algorithm 3, to control if the pivot under analysis is 0, we need to check that the corresponding qubit is in state $|0\rangle$ by using it as control in a CNOT gate. Since quantum circuits usually work with $|1\rangle$-controlled gates and not $|0\rangle$-controlled ones, to have a semantically equivalent circuit we need, for each of the $|\mathbf{B}|$ pivot checks, an X gate applied on the control right before and right after the CNOT. In Reference [54],

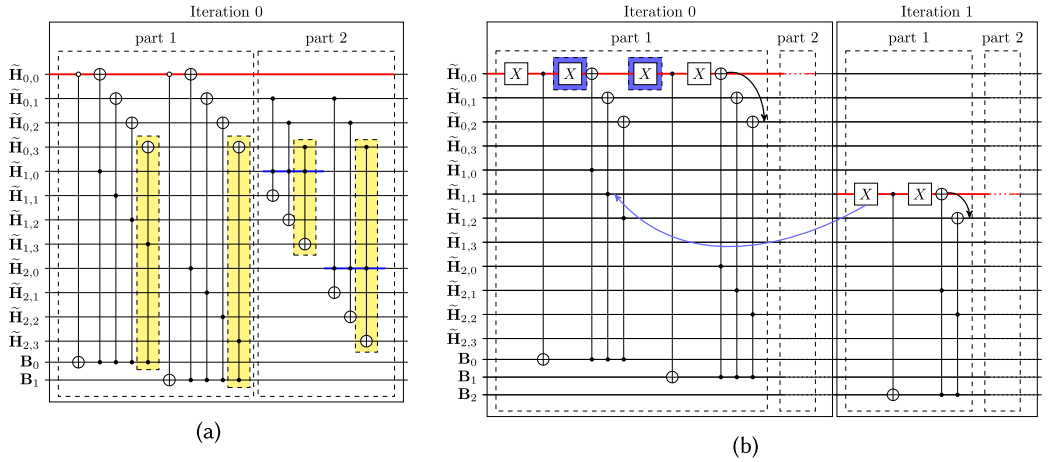


Fig. 7. Continuing Figure 6. The red wires show the qubits that we need to set to $|1\rangle$ (the pivots), while the blue wires represent the qubits that we need to set to $|0\rangle$. The yellow boxes of (a) show the effect of optimization 4, which allows excluding operations on the rightmost $\mathbf{V}$ submatrix of $\widehat{\mathbf{H}}$. The purple boxes of (b), instead, describe the X gate removed after optimization 5, with the arrows signaling the rearrangement of the remaining ones used to reduce the overall depth of the circuit. The black arrows, however, show the gate rearrangements of CCNOTs due to optimization 6, wich allows us to anticipate the first gates of part 2 and hence reduce the global depth.

the authors propose to reduce the number of X to just $2(r-1)$ by applying a single X gate on the candidate pivot only at the beginning and at the end of each part 1. With this optimization, we improve the proposal by anticipating the first X required to transform the $|0\rangle$ control into a $|1\rangle$ control right after the first row addition of the part 1 of the previous x-iteration. In this way, the overall depth is not affected by the X gates, except for a constant additional factor of 1 due to the first X of the first part 1, which, however, can be easily interleaved with previous subcircuits.

Optimization 6—Rescheduling the setting of the last pivot. It rearranges gates to reduce the overall depth of the circuit. In part 1, in the last sequence of swaps involving the last row of the matrix, we can postpone the CCNOT on the pivot until the last operation of the sequence of CCNOTs applied on the rightmost $r \times r$ matrix. The fundamental observation is that, differently from previous rows, we do not need the pivot qubit in the rest of the circuit. Furthermore, in part 2 of the same x-iteration, the other control qubit used in this CCNOT—i.e., the qubit in the same column of the pivot, but on row $r-1$—will be involved only in the last part 2 iteration of the same x iteration, while all the other CCNOTs of part 1 involve qubits that will be used earlier.

Finally, we remark that the whole QGJE circuit is extended in a straightforward fashion to include one or more syndromes, depending on if the starting code is a circulant one. As explained before, they can be treated just as other columns of the matrix, on which we cannot skip any operation, so it has an overall number of CCNOTs equals to $(|\mathbf{B}| + |\mathbf{C}|) \cdot M$, which is equal to $\frac{3}{2}r(r-1)M$.

The final gate count and depth for this improved version are reported in Table 1. The depth was experimentally confirmed for increasing value of r, for which we adapted the approach used in classical combinatorial circuits reported in Reference [15]. The idea is to represent the circuit as a directed acyclic graph, in which each node corresponds to a gate applied to a given (set of) qubit(s).

Table 1. Number of Quantum Gates as a Function of the Linear Code Parameters (n, k, $r = n - k$) and the Wanted Hamming Weight (t) for the Different Quantum Subcircuits Used to Solve Prange ISD

	State Preparation	Oracle					Diffusion
Cost metric	**Dicke state**	**Data preparation**	**Pack columns**	**Gaussian elimination**	**Hamming weight compute and check**	**Amplitude flip**	
X	r	$\frac{r+rn}{2}$	$2(n-1)\log_2(n)(\log_2(n)-1)$	$2(r-1)$	$M(4r-\log_2(r/t)-3)$	0	$n+2r$
CNOT	$5nr-5r^2-2n$	0	$(n-1)\log_2(n)(\log_2(n)-1)$	$\frac{1}{2}r(r-1)$	$M(\frac{9}{2}r-5\log_2(r)-11)$	0	$10nr-10r^2-4n$
CCNOT	0	0	0	$\frac{1}{6}r(r-1)(5r+9M-1)$	$M(3r-2\log_2(r)-3)$	0	0
CSWAP	0	0	$(n-1)\log_2(n)(\log_2(n)-1)(r+1)-\frac{n}{2}$	0	0	0	0
R_y	$4nr-4r^2-2n+1$	0	0	0	0	0	$8nr-8r^2-4n+2$
$C^r(X)$	0	0	0	0	0	1	0
$C^{\log_2(r)+2}(Z)$	0	0	0	0	0	M	0
$C^n(Z)$	0	0	0	0	0	0	1
Depth	$\leq \frac{27nr-12n-27r^2+3}{r-2}$	1	$\log_2^2(n)+\log_2(n)+r-1$	$\frac{3}{2}r^2-\frac{1}{2}r+M+2$	$\log_2^2 r+7\log_2(r)-4$	M	$\leq \frac{27nr-12n-27r^2+3}{r-2}$
Qubits	n	$rM+rn$	$(n-1)\log_2(n)(\log_2(n)-1)$	$\frac{1}{2}r(r-1)$	$M(\frac{5r}{4}-1)$	1	0

Except for the multi-controlled Z and X gates, all the gate figures of the Oracle phases should be multiplied by 2 to take into account the uncomputation stage. Each multi-controlled gate, denoted as $C^m(\cdot)$, involves m control qubits and a single target qubit. The M variable represents the number of syndromes represented in the circuit, which is r for circulant codes and 1 for the other ones.

This counting approach gave us a depth of $3/2r^2 - 1/2r + 2 + M$, which, since $M \leq r$, results in an asymptotic value of $O(r^2)$. Even more interestingly, in the general case in which we also need the right part of the matrix after QGJE and we cannot apply Optimization 4, the resulting depth only increase by a factor of k, keeping it in the order of $O(r^2)$.

4 QUANTITATIVE EVALUATION OF THE SOLUTION EFFICIENCY

In this section, we provide quantitative measures for all code-based cryptographic schemes that advanced to the fourth stage in the latest NIST report [49]. We provide a direct comparison between our proposal and all the relevant works solving the same problem, together with a comparison to the quantum algorithms used to attack AES. We additionally provide a translation of our quantum gates into the Clifford+T gate set and provide a definition of quantum security margin inspired from the one used in Reference [25], following the recommendation by NIST. Finally, we make an asymptotical analysis of our quantum circuit depth for increasing values of n.

4.1 Conversion to Canonical Gate Sets

In the previous sections, we used abstract multi-qubits quantum gates to model the logic of our quantum circuit. However, these abstract gates do not accurately reflect the capabilities of concrete quantum computers, which in turn offer only 1- and 2-qubits gates. Additionally, different implementations of quantum computers rely on different gate sets, making it harder to target a general architecture when designing a quantum algorithm. Finally, much work is spent into optimally decompose commonly used quantum gates, especially for 3-qubits gates (like the CCNOT and CSWAP), and arbitrary, optionally controlled, rotation gates (like the R_y or Z). Willing to obtain precise, concrete estimates for the quantum gate count required to compute our ISD solver, we analyze two different translations of the multi-qubit gates we employ into simpler ones, each one with a different focus.

Translating multi-controlled gates. We only translate multi-qubit gates acting on more than three qubits, namely the $C^m(Z)$ gate and $C^m(X)$ gates, with m being the number of controls, matching the current state of the art for complexity measures. Since such gates are repeated at each iteration of Grover's algorithm, their decomposition should not dominate the depth of the circuit, i.e., it should be smaller than the QGJE circuit having depth $O(r^2)$, and width smaller than the data representation portion of the circuit, which has depth $O(rn + r)$. The problem of decomposing a

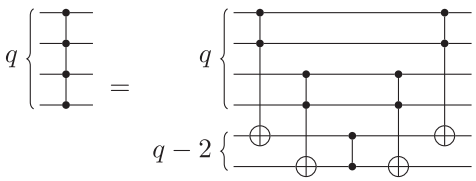


Fig. 8. Decomposition of an $C^3(Z)$ gate into CCNOT gates and a single CZ. The decomposition requires two additional qubits and has depth 3.

$C^m(Z)$ gate is indeed equivalent to the one of decomposing a $C^m(X)$ gate, since we can replace an $C^m(Z)$ with an $C^m(X)$, taking care to add an H gate before and after on the target qubit. Different approaches to decomposing multi-controlled gates were explored in literature, focusing on reducing width and depth with respect to the number of control qubits, using both approximate and exact decompositions. The work proposed in Reference [6] shows two exact decomposition for the multi-controlled Z requiring respectively $O(m)$ gates and $O(m)$ qubits or $O(m^2)$ gates and $O(m^2)$ depth without additional qubits. Our approach stems from both Reference [53] and Reference [6, Section 7.1], with a special focus on depth. For the case in which m is equal to a power of 2, minus 1, the decomposition requires $m-2$ additional qubits and resets all of them to their initial state. The gate count is of $2m-4$ CCNOT, plus 1 CZ. Such a decomposition, shown in Figure 8 for an $C^3(Z)$ gate, has a depth equal to $\log_2(m+1)+1$.

An interesting approach, described in Reference [58], shows a linear-depth decomposition of $C^m(X)$ gates using no additional qubits. The work uses a circuit in which all gates are controlled R_x rotation gates with arbitrary angles and precision. As described in Reference [41], controlled R_x gate can be further expanded into a single R_x and two CSWAP gates. This approach, however, did not show any practical advantage in our circuit, producing almost identical figures, since the main contribution to depth comes from the QGJE circuit.

Translation into a fault-tolerant gate set. The second gate basis considered in this work is the Clifford+T one, a universal set of gates introduced in Reference [17], and considered the most promising one for fault-tolerant quantum computation. The gates belonging to the Clifford group are generated starting from the set $\{H, CNOT, S\}$. As a consequence, all the Pauli gates employed in our work, namely X and Z, can be directly derived from them, as $Z = S^2$ and $X = HS^2H$. Additionally, we can derive the CZ employed in the two reflections required by Grover's algorithm at each iteration by applying a H gate on the target qubit, followed by a CNOT having the same control and target qubit as the CZ and a final H gate on the target qubit.

However, the CCNOT, CSWAP, R_y, and the multi-controlled Z gates employed in our circuits cannot be generated using the Clifford gates only. The inclusion of the T gate to the Clifford basis is indeed fundamental to reach the universality, as such inclusion allows us to approximate all the quantum gates up to an arbitrary small approximation factor. However, T requires extensively more resources to be implemented in a fault-tolerant fashion with respect to the Clifford gates [39], and for this reason most of the proposed algorithms in cryptography considers the number of T gates and the T-depth the most important measure, where the T-depth is defined as the number of stages in the circuit involving only non-Clifford group gates. To offer a comprehensive analysis, we use as well the additional measure, proposed in Reference [38], of the product of the number of qubits times the T-depth of the circuit.

Table 2 summarizes the translations used in our work. In fact, we are only concerned with the translation of the CCNOT and R_y gate, since the CSWAP gate can be immediately obtained using 1 CCNOT gate and two CNOT gates, while the multi-controlled Z gate can be decomposed into

Table 2. Translation of the Non-Clifford Gates Employed in Our Circuit

Gate to translate	Type & number of gates for translation				Additional Qubits	Equivalent T-depth
	H	CNOT	S	T		
CCNOT [37]	2	8	1	4	1	1
CCNOT† [37]	1	2	3	0	0	0
R_y [42]	151	0	82	149	0	149

The R_y gate is translated using the approximation algorithm presented in Reference [42] using a precision of 10^{-15}. For the CCNOT translation, we used the results of Reference [37], requiring shallower circuits with respect to all the other proposals.

multiple CCNOT gates plus 1 CZ, as shown in the previous section. Different techniques to efficiently decompose CCNOT gates have been explored in the literature, trading off number of qubits, number of T gates, and T-depth. For example, in Reference [2] the authors present a translation that, without any additional qubit, requires 2 H, 7 T, and 7 CNOT gates, with an overall T-depth of 3. The use of a single additional qubit, however, allows them to obtain a T-depth of 2. However, in Reference [37] the authors, by combining two previous techniques presented in References [28, 59], show a T-depth 1 decomposition of the CCNOT requiring only a single additional qubit, initialized in state $|0\rangle$, and restored to its value at the end. Their circuit is further detailed in Reference [36, Appendix C], in which they also show a circuit for the CCNOT† not relying on non-Clifford gates but only on an additional measurement operator and a classical bit.

For our quantum circuit, the technique proposed in Reference [37] leads to better T-depth · Width measures with respect to the others. To not alter the depth results presented in the previous sections, the number of additional qubits required to translate the CCNOT and CSWAP gates must still allow the same degree of parallelization between them. For the CSWAP gates involved in the column permutation of the oracle, we should have a number of additional qubits equal to the maximum number of CSWAPs that we can apply in parallel. Applying the same thoughts of Section 3.1, we obtain an additional number of qubits equal to r. On the same line of reasoning, the number of parallel CCNOT involved in the QGJE circuit is in the order of r^2. As a consequence, since we can reuse both set of qubits in different submodules, the total number of additional qubits required is the maximum between the two, i.e., r^2.

The only gate among the ones employed by us for which there is no straightforward translation is the R_y gate. An active field of research is indeed focusing on efficiently approximating single qubit gates with arbitrary rotations using only gates belonging to the Clifford+T. In Reference [42], they propose an exact synthesis algorithm showing how, fixing an arbitrary precision ϵ, on average $3.067 \log_2(1/\epsilon) - 4.322$ T are required to achieve a given quality of approximation. An $\epsilon = 10^{-15}$, a value sufficient for most applications [42], gives us a T-count of ≈149, a value confirmed by the extensive usage of their tool in our metrics. Since $R_y(\theta) = SHR_z(-\theta)HS^\dagger$, we can therefore say that the T-count is equivalent to ≈149. Note that although $S^\dagger$ does not belong to the Clifford basis, it is equivalent to S^3. Since the gates obtained from the decomposition of the R_z gate are sequentially applied to the same qubit, the T-depth of the translation is equivalent to the T-count.

4.2 Evaluating the Cryptanalytic Effort on NIST Post-quantum Code-based Cryptosystems Standardization Candidates

To assess the security of the different cryptographic schemes proposed, NIST defined in its call for proposal [52] a classification method based on security categories obtained through a comparison with existing NIST standards in symmetric cryptography. To this end, NIST defined three

Table 3. Number of Gates and Qubits Required by Our ISD Circuit Design for Three Code-based Cryptosystems: BIKE, HQC, and McEliece

Algorithm	Sec. Level	Code parameters			Grover Iter.s	Grover gates						Total			
		n	k	t		X	CNOT	CCNOT	CSWAP	RY	CZ	Gates	Depth	Qubits	D · W
BIKE (key)	L1	24,646	12,323	142	65	95	97	108	101	96	65	108	93	29	123
BIKE (key)	L3	49,318	24,659	206	96	129	131	142	135	129	96	142	127	31	158
BIKE (key)	L5	81,946	40,973	274	130	164	166	178	170	164	130	178	162	33	195
BIKE (message)	L1	24,646	12,323	134	61	91	93	104	97	92	61	104	89	29	119
BIKE (message)	L3	49,318	24,659	199	93	125	127	139	132	126	93	139	123	31	155
BIKE (message)	L5	81,946	40,973	264	125	159	161	173	165	159	125	173	157	33	190
HQC	L1	35,338	17,669	132	59	91	93	104	97	91	59	104	89	30	119
HQC	L3	71,702	35,851	200	93	126	128	140	133	127	93	140	125	32	157
HQC	L5	115,274	57,637	262	123	159	160	173	165	159	123	173	157	34	190
McEliece	L1	3,488	2,720	64	72	94	97	101	101	97	72	102	92	22	114
McEliece	L3	4,608	3,360	96	93	116	119	124	123	119	93	125	115	23	138
McEliece	L5	6,688	5,024	128	131	155	159	164	163	158	131	165	154	24	178
McEliece	L5	6,960	5,413	119	132	156	159	164	163	159	132	165	155	24	178
McEliece	L5	8,192	6,528	128	150	174	178	183	182	178	150	184	173	24	197

All the gates are considered to have the same impact on the overall measures. We report as well the overall depth of the circuit for our proposal, and we extend the analysis with the $D \cdot W$ measure, multiplying together the depth and the width (i.e., the number of qubits), which better captures the overall cost of our proposal. Except for the code parameters n, k, and t, all the values are expressed in base-2 logarithm.

distinct levels for public-key encryption, namely 1, 3, and 5, corresponding to a computational effort comparable to or greater than the one required for key search on the AES block cipher with a 128-bit key (AES-128), 192-bit key (AES-192), and 256-bit key (AES-256), respectively. Table 3 contains the parameter set proposed for each of the cryptographic schemes under scrutiny, with the submitters of McEliece cryptographic scheme proposing three distinct parameters for the level 5 security.

Starting from the closed-form equations reported in Table 1, and additionally converting the multi-controlled gates into simpler ones as explained in Section 4.1, we show in Table 3 the assessment of the effort needed to attack the three code-based cryptosystems being evaluated in NIST's post quantum standardization call: Classic McEliece [12], BIKE [3], and HQC [47]. In the table, we report the number of gates, divided by kind, required to build our quantum circuit to solve the SDP problem using Prange's ISD variant. In the same table, we also report the total number of gates required, considering an equal cost for all of them, as implied by NIST in Reference [52]. Last, for each cryptographic scheme, we report the depth, the number of qubits (also known as width) and the Depth · Width metric, as suggested in Reference [38].

4.3 Comparing Computational Efforts with Breaking Real AES

Since NIST call assess the cryptographic security of PQC proposals by relating them to the computational effort required to break AES with a quantum computer, we compare our proposal to the current state-of-the-art implementation for the acceleration of AES through the usage of Grover's framework [67], which extensively relies on the metrics related to the Clifford+T gate sets. The proposal improves on all the previous works [30, 37, 44] in terms of the CCNOT-Depth · Width metric and additionally highlights how the figures shown in Reference [37] arise from a bug in the Q# framework used in the implementation. We note that the work of Reference [44], although having slightly worse figures in terms of the CCNOT-Depth · Width metric, has nonetheless the best values in terms of all the gate counts and the CCNOT-Depth alone. Moreover, the work additionally reports the overall depth of the circuit and not only the CCNOT-depth. Later, two other works came out in Reference [32] and Reference [66]. However, both of them only target AES-128, and therefore we did not use their results in our comparison. In our analysis, we additionally take

Table 4. Computational Effort Required to Break AES via Grover-based Key Search Using the AES Implementation in References [67], [44], and [33] (Not Peer-reviewed) Compared to the Computational Effort to Solve the ISD Problem on BIKE, HQC, and Classic McEliece with Our ISD Implementation

Sec. Level	Algorithm	Qubits	Clifford + T-measures		
			T-Count	T-Depth	T-Depth · W
L1	AES [67]	9	80	76	86
L1	AES [44]	10	80	76	86
L1	AES [33]	13	80	71	84
L1	BIKE (key)	57	110	91	121
L1	BIKE (message)	57	106	87	117
L1	HQC	59	106	87	117
L1	McEliece	41	105	90	112
L3	AES [67]	10	113	108	118
L3	AES [44]	10	112	108	118
L3	AES [33]	13	112	103	116
L3	BIKE (key)	61	144	125	156
L3	BIKE (message)	61	141	121	153
L3	HQC	63	142	123	155
L3	McEliece	43	127	113	135
L5	AES [67]	10	145	140	150
L5	AES [44]	11	145	140	151
L5	AES [33]	13	144	135	148
L5	BIKE (key)	63	180	160	193
L5	BIKE (message)	63	175	155	188
L5	HQC	65	175	155	188
L5	McEliece	45	167	152	176
L5	McEliece	45	168	153	176
L5	McEliece	45	186	171	195

All the values are expressed in base-2 logarithm.

into account the work presented in Reference [33], that, while offering an extensive review of all the previous technique, additionally improves on all of them in all the metrics. To the authors' knowledge, this work has not been peer-reviewed yet. Table 4 reports all the relevant T measures for each of the cryptographic schemes listed in Table 1 and a comparison of our results with respect to the Grover-based AES key search in References [67], [44], and [33].

To have a fair comparison, we elaborated all the AES figures reported in the works under analysis. First, while, due to unicity distance constraints, the number of ciphertexts needed to univocally find the key with high probability is equal to 2, 2, and 3 for AES-128, -192, and -256, respectively [37]; nonetheless the work presented in Reference [21] shows that the computational complexity of AES is equivalent to one call to a single AES circuit. For this reason, we used a number of AES instances equal to 1 for all proposals. Furthermore, when using the AES circuit in Grover's framework, we need an uncomputation stage to restore all the qubits to their original values, effectively doubling the gate count of the oracle. Additionally, the oracle instance requires a multi-controlled X gate, with the number of controls equal to the block size of AES, that is, 128. Finally, while all the works on AES omit the diffusion stage from their analysis, we also take into

account the multi-controlled Z required by this stage, with the number of controls being equal to k for an AES-k variant, with $k = \{128, 192, 256\}$. For the multi-controlled gates, we used the same decomposition shown in 4.1. To conclude, we set the number of Grover's iterations for all AES proposals to the optimal value of $0.58278\sqrt{2^k}$, as discussed in Section 2.4.

To rephrase the CCNOT measures used by all the AES proposals in terms of the T measures used in our work, for both Reference [67] and Reference [44] we used the T-depth 4 conversion of Reference [2], since it results in better T-Depth · Width values. However, the proposal of Reference [33], due to the higher number of qubits and lower number of CCNOT gates with respect to the other proposals, shows better results when the T-depth 1 translation using one ancillary qubit [37] is used.

In the following, exploiting the figures obtained in Section 4.1, we detail how we retrieved the T measures for all the different subcircuits employed in our proposal. For the Dicke state, the only gates involving a T gate in their decomposition are the R_y ones, each of which requires ≈149 T gates. The same assumptions made in Reference [7] for the circuit depth, namely the possibility of parallelizing R_y gates involved in different subparts, can be used to derive a straightforward T-depth of $O(n)$. The T-count of the subcircuit to permute the columns of matrix $\mathbf{H}$ can be obtained by simply multiplying the number of CCNOT and CSWAP by 7. The T-depth can be obtained using a way of reasoning equal to the normal depth. Since indeed CSWAPs and CCNOTs belonging to different comparators can be interleaved, we can also parallelize the T gates involved in their decomposition, leading to a T-depth equal to $\log_2(n)(\log_2(n) + 1) + O(1)$. For the Q GJE circuit, the only gates involved in the relevant measures of this stage are the CCNOTs. The T-count is simply obtained by multiplying the overall CCNOT-count by 7. The T-depth, instead, can be taken as equal to the original depth since, once again, the assumptions originally made in the evaluation of the depth still holds. The only non-Clifford gate of the Hamming weight compute and check stage is the CCNOT gate, so the T-count is simply obtained by multiplying the figures by 7. The T-depth can be roughly approximated to the original depth, since the circuit provides a high degree of parallelization between layers. Finally, with respect to the multi-controlled gates, Reference [59] shows a decomposition requiring $8m$ T gates and a T-depth of $2\log_2(m)+1$, using the same amount of additional qubits of our previous decomposition.

We observe from our results that all the cryptographic schemes analyzed require considerably more effort to be broken than the corresponding symmetric ciphers employed as a gauge of their security level. These results indicate that, with respect to an attack conducted with our implementation of Prange's ISD, the choices made by the proposers of Classic McEliece, BIKE, and HQC are strongly conservative in terms of security. We highlight that the difference between the D · W measure of our proposal and the ones of the AES implementations is significantly lower for the McEliece cryptographic scheme at level 3 with respect to all the others. These findings are consistent with the ones reported in Reference [25].

4.4 Constraining the Depth of the Quantum Circuit

Motivated by the difficulty of running extremely long serial computations, in its original call for proposal [52] NIST suggests also taking into account quantum-accelerated attacks in which the quantum circuits are restricted to a maximum depth, called `MAXDEPTH`, with plausible values equal to $\{2^{40}, 2^{64}, 2^{96}\}$. Under depth constraints, it estimates the quantum gates needed to recover the key on AES to be equal to `QAES/MAXDEPTH`, with `QAES` equal to 2^{170} for AES-128, 2^{233} for AES-192, and 2^{298} for AES-256. To obtain such values, NIST likely adapted the figures for the quantum attack to AES proposed in Reference [30], the only one available at the time of the call for proposals, to a depth-constrained parallel version of Grover's algorithm.

Considering a partition of the whole search space into S disjoint sets, a parallel version of Grover's algorithm assigns each set to a distinct quantum circuit, hence allowing the execution of all the circuits in parallel. Denoting as N the number of Grover iterations required by a sequential Grover's algorithm, this value is reduced to $N/\sqrt{S}$ in its parallel version. At the same time, both the depth D_{it} and number of gates G_{it} required by a single Grover iteration remain approximately equal. Therefore, by parallelizing Grover search across S quantum instances, we reduce the total depth of a single quantum circuit from the original $N{\cdot}D_{it}$ down to $N{\cdot}D_{it}/\sqrt{S}$. Analogously, the total number of quantum gates required by a single circuit is equal to $N{\cdot}G_{it}/\sqrt{S}$.

Nonetheless, the overall computational effort required by the parallel version of the algorithm is computed by considering a sequential execution of all the quantum circuits, i.e., by multiplying by S the results obtained for a single instance, obtaining therefore the following expressions for the overall depth and number of gates, respectively:

$$D_p = \sqrt{S}{\cdot}N{\cdot}D_{it}, \tag{7}$$

$$G_p = \sqrt{S}{\cdot}N{\cdot}G_{it}. \tag{8}$$

Once we constrain the depth of a single quantum circuit to the depth values required by NIST, we derive $\texttt{MAXDEPTH} = N{\cdot}D_{it}/\sqrt{S}$, and hence $D_p = S{\cdot}\texttt{MAXDEPTH}$. By using the values of $\sqrt{S}$ obtained from the previous equation, we can restate Equation (8) as

$$G_p = \frac{N^2{\cdot}G_{it}{\cdot}D_{it}}{\texttt{MAXDEPTH}}, \tag{9}$$

which can be employed to derive the plausible definition of QAES used by NIST as

$$\texttt{QAES} = N^2{\cdot}G_{it}{\cdot}D_{it}. \tag{10}$$

Additionally, we can derive the number of instances as

$$S = (N{\cdot}D_{it}/\texttt{MAXDEPTH})^2.$$

Finally, note that the previous equation immediately produces an alternative formulation of the overall depth as

$$D_p = \frac{N^2{\cdot}D_{it}^2}{\texttt{MAXDEPTH}}. \tag{11}$$

Using this parallel approach, the QAES values obtained by replacing the results of Reference [30] in Equation (10) are almost identical to the ones estimated by NIST in Reference [52], as we show in Table 5, and not much different to the ones obtained using the results in Reference [44]. However, if we consider the implementation of the quantum circuit for AES in Reference [67], then the results show a reduction of roughly 2^{10}. However, Reference [67] only reports the figures related to the T gate, avoiding the discussion on the overall depth, making the comparison with the other works only partial. Finally, the gap with the NIST estimates increases to roughly 2^{25} if we consider the proposal presented in Reference [33], which, however, has not been peer-reviewed yet.

Relying on the same kind of parallelization technique and the QAES values specified in the original NIST requirements [52], in Reference [25] the authors derive a parameter called quantum security margin, denoting the ratio between the overall depth D_p of the parallelized version of Grover's framework adapted to Prange algorithm, and the number of gates QAES/MAXDEPTH estimated by NIST to recover the key of AES at the same level. By using Equation (11), we can observe that this ratio will be given by

$$\frac{N^2D_{it}^2}{\texttt{MAXDEPTH}} \Big/ \frac{\texttt{QAES}}{\texttt{MAXDEPTH}} = \frac{N^2D_{it}^2}{\texttt{QAES}},$$

Table 5. QAES Values for the Parallelized Version of the Quantum Implementation of AES for All Three Computational Tasks

Quantum AES implementation	QAES		
	AES-128	AES-192	AES-256
[52]	170	233	298
[30]	169	232	298
[44]	167	233	298
[67]$^{\diamond}$	159	223	288
[33]	154	219	283

$^{\diamond}$Values computed considering only the T-count and T-depth.

The value proposed by NIST in Reference [52] are almost identical to the ones obtained substituting the results in Reference [30] in Equation (10). We show as well the QAES values obtained using the results in Reference [44], the most up-to-date peer-reviewed implementation of a quantum circuit for AES showing also metrics related to the overall depth and number of gates. However, Reference [67] do not report values related to the overall depth, hence the computation of the QAES value is based only on the number of T gates and the T-depth. Finally, we report the QAES values obtained using the results in Reference [33], which, to the authors' knowledge, has not been peer reviewered yet.

whose value is independent of the MAXDEPTH parameter. Table 6 shows a comparison between Reference [25], which estimates the depth of a single Grover iteration to $r^{2.5}$, and our exact depth measures obtained using the depth values taken from Table 3 and the T-depth values taken from Table 4. The security margin is obtained therefore by subtracting from the $\log_2$ of the number of quantum gates required by AES, the $\log_2$ of the depth of the code-based cryptographic scheme at the same level. We note that the number of iterations in our work is almost the same as in Reference [25]. As explained in Section 2.3, we indeed take into account also the probability that a random $r \times r$ matrix is non-singular and as well we set the success probability of Grover to .84458, producing a negligible increase of $.58278/\sqrt{.288} \approx 1.09$ in the number of iterations. With our proposed implementation, the security margin is reduced for all the cryptographic schemes, with the considerable decrease for BIKE and HQC due to the $\sqrt{r}$ reduction in the number of Grover iterations achievable for circulant codes. Most importantly, the already low security margin reported in Reference [25] for the parameters choice of the level 3 version of Classic McEliece, also taken into account by NIST in Reference [49], is further reduced in our circuit implementation by 7 bits, failing to pass the required security with respect to its AES counterpart. Moreover, considering the T-depth values, the security margin is further reduced by 4 bits.

Since NIST, in the original submission requirements [52], defines estimates for AES in terms of the number of quantum gates, in the last four columns of Table 6 we also report additional measures expressing the ratio between the overall number of gates G_p of the parallelized version of Grover's framework adapted to Prange algorithm and the ratio QAES/MAXDEPTH. In this case, it is not possible to ignore the MAXDEPTH values as we did before, and for this reason we report in Table 6 figures related to each suggested value of MAXDEPTH. Note that, to have up-to-date figures, we use the values of QAES obtained in Reference [44], and reported in Table 5, since it is the most up-to-date work showing measures related to the overall depth and number of gates. The observations on the security of McEliece parameters proposed for level 3 is confirmed, with the number of security bits falling behind the other candidates for the same level, though still above

Table 6. Results of the Parallelization of Our Proposed Algorithm, Required to Comply with the Upper Bounds on the Values of the MAXDEPTH Suggested by NIST for a Single Quantum Circuit

Algorithm	Sec. Level	Depth margin [25]	Depth margin Table 3◊	Depth margin Table 4♦	Circuit Instances	Overall Depth	Overall Gates	Gates margin
BIKE (key)	L1	41	17	13	{107, 59, 0}	{147, 123, 93}	{161, 137, 108}	{34, 34, 37}
BIKE (key)	L3	47	21	17	{174, 126, 62}	{214, 190, 158}	{229, 205, 173}	{36, 36, 36}
BIKE (key)	L5	53	26	22	{244, 196, 132}	{284, 260, 228}	{300, 276, 244}	{42, 42, 42}
BIKE (message)	L1	32	9	5	{99, 51, 0}	{139, 115, 89}	{153, 129, 104}	{26, 26, 33}
BIKE (message)	L3	40	14	10	{167, 119, 55}	{207, 183, 151}	{222, 198, 166}	{29, 29, 29}
BIKE (message)	L5	43	16	12	{234, 186, 122}	{274, 250, 218}	{290, 266, 234}	{32, 32, 32}
HQC	L1	33	8	4	{98, 50, 0}	{138, 114, 89}	{153, 129, 104}	{26, 26, 33}
HQC	L3	43	16	12	{169, 121, 57}	{209, 185, 153}	{225, 201, 169}	{32, 32, 32}
HQC	L5	44	15	11	{233, 185, 121}	{273, 249, 217}	{290, 266, 234}	{32, 32, 32}
McEliece	L1	21	15	11	{105, 57, 0}	{145, 121, 92}	{154, 130, 102}	{27, 27, 31}
McEliece	L3	3	−4	−8	{149, 101, 37}	{189, 165, 133}	{200, 176, 144}	{7, 7, 7}
McEliece	L5	18	11	7	{229, 181, 117}	{269, 245, 213}	{279, 255, 223}	{21, 21, 21}
McEliece	L5	18	11	7	{229, 181, 117}	{269, 245, 213}	{280, 256, 224}	{22, 22, 22}
McEliece	L5	56	48	44	{266, 218, 154}	{306, 282, 250}	{317, 293, 261}	{59, 59, 59}

◊Depth margin obtained by using the gate set {X, CNOT, CCNOT, CSWAP, RY, CZ}.
♦T-depth margin obtained by using the Clifford+T gate set.
To have a comparison against Reference [25, Table 5], in which the authors estimate a depth of $r^{2.5}$ for each Grover's iterations, we report the base-2 logarithmic difference between the depth measures and the estimates given by NIST [52, Section 4.A] on the number of quantum gates required to break AES for the same level, a quantity referred to as depth margin. The results are independent of the MAXDEPTH values.
Additionally, in the last four columns, we compare the number of quantum gates required by our circuit with respect to the ones required to break AES for the same level, using, however, the results of Reference [44]. Since in this case the results depend on the value chosen as the maximum depth of a single quantum circuit, we use a shorthand notation in which the set of three exponents correspond to setting this depth to the three values of $\{2^{40}, 2^{64}, 2^{96}\}$, suggested by NIST as reference for MAXDEPTH. All the values are expressed in base-2 logarithm.

the requirements. Additionally, we can see that each of the cryptographic scheme under analysis does not require any parallelization for level 1 security and $\texttt{MAXDEPTH} = 2^{96}$, since the depth of a single quantum circuit is already below the 2^{96} depth indicated by NIST to be the approximate number of gates that quantum computers could perform in a millennium.

As an additional point, we give here a clue on the different strategies that can be adopted to constrain the depth of a single circuit to a fixed value. A naive way to implement the partition strategy hinted in this section would consist in setting up $x < r$ qubits of the n representing $\mathcal{J}$ to $|1\rangle$, while generating a Dicke state $|D^{n-x}_{r-x}\rangle$ for the remaining portion. With respect to a serial computation, the domain of our oracle function will be reduced by a factor of $\binom{n}{r}/\binom{n-x}{r-x}$, which allows tuning the correct x giving the required MAXDEPTH.

Another strategy to reduce the depth of a single quantum circuit is to use a hybrid classical-quantum algorithm. In Reference [55], the authors use a hybrid version of Lee–Brickell algorithm, in which the classical part performs the column permutation and the Gauss–Jordan elimination, obtaining the matrix $\widetilde{\mathbf{H}} = [\mathbf{W} \mid \mathbf{V}]$. If $\mathbf{W} = \mathbf{I}_r$, then the quantum circuit will check if p columns of the matrix $\mathbf{V}$, added to the syndrome, results in a vector of weight $t - p$. While in Reference [55] the authors tune p to reduce the overall gate count, the parameter may be easily adapted to obey the requirements on MAXDEPTH.

4.5 Consequences from a Code-based Cryptosystem Design Perspective

Designing post-quantum code-based cryptosystem targeting a specific security level is a major challenge, requiring to choose the right set of parameter n, k, and t so that it provides just the

required security level, as designing larger-than-needed parameters would penalize the efficiency of the cryptosystem without need. For this reason, designers of cryptographic schemes have to perform a computational complexity analysis of their proposals, later deciding which is the best choice of parameters for a specific security level.

The choice of parameters is usually based only on two parameters, namely the n and the ratio k/n, known as rate of the code, and denoted by R. By analyzing the parameters of the code-based cryptosystems advanced to the final stages of NIST competition, reported in Table 3, we note that, while the rate for HQC and BIKE is fixed to $\frac{1}{2}$, a value mandated by the nature of the chosen code family, it assumes different values for McEliece. In our analysis, we assumed a ratio of $\frac{3}{4}$, obtained by approximating the average ratio of this family of cryptosystems.

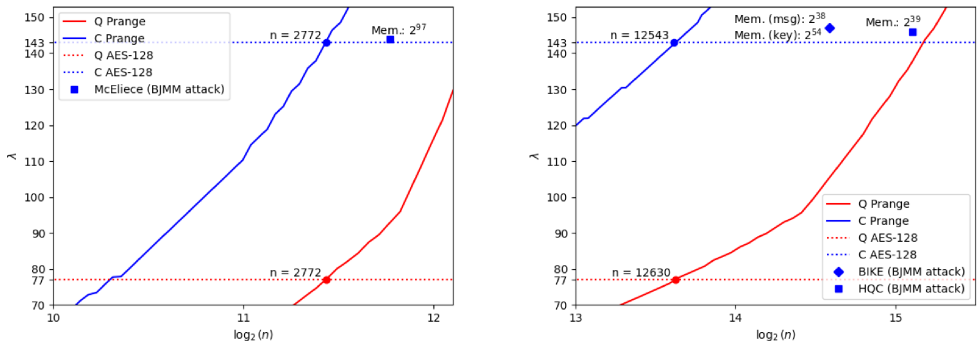
When studying the asymptotic performance of random codes, the value of the maximum error weight t that can be corrected determined from n by using the **Gilbert–Varshamov (GV)** bound. This approximation, found independently by the two authors in References [29] and [65], represents the largest value of t for which the SDP has a unique solution with overwhelming probability. Denoting as $H(x)$ the binary entropy function $H(x) = -x\log_2(x) - (1-x)\log_2(1-x)$, the value of t is obtained as $t = \lfloor H^{-1}(1-R)\frac{n}{2} \rfloor$. While the error weight used in the McEliece cryptosystem is quite close to the GV bound, BIKE and HQC use a different value of $t = O(\sqrt{n})$. In both cases, the error weight t is sublinear with respect to the code length n, and all the advantages of techniques improving on the basic Prange's strategy have been shown to asymptotically vanish [64]. For this reason, all the designers of code-based cryptosystems base their choice of parameters on the Prange's basic strategy, with a further heuristic polynomial factor to keep into account for polynomial speed-ups obtainable when applying more advanced ISD algorithms.

Figure 9 shows the results of our analysis under the previous assumptions using a Prange attack, both in its classical form and its quantum one. To obtain the figures for the Prange's classical attack, we relied on the tool obtainable at Reference [9], made freely available by the authors of Reference [25]. For the quantum analysis, instead, we imposed a value for the maximum depth allowed for a quantum circuit (introduced in Section 4.4) of 2^{96}.

Both figures show, for increasing values of n, two distinct plots related to a Prange attack: The depth obtained using our quantum implementation (solid red line); the classical number of gates required estimated using Reference [9] (solid blue line). In both plots, we also represent the number of classical gates assumed by NIST to be needed to break AES-128 (dotted horizontal blue line), together with the depth of the best quantum circuit in terms of overall depth [44] (dotted horizontal red line), as explained in Section 4.3.

Figure 9(a) shows the results obtained for codes with $R = \frac{3}{4}$ and t matching the one correctable at the GV-bound, i.e., an approximation of the values used by McEliece. We note how the code lengths required to target the AES-128 security level are almost identical between the classical and quantum version of the Prange's attack. Figure 9(b), however, shows the results obtained for codes with $R = \frac{1}{2}$ and $t = \sqrt{n}$ relying on quasi-cyclic random codes, i.e., BIKE and HQC. Unlike what happens in the previous case, we note instead that if we choose AES-128 as a bar, then our quantum Prange's attack mandates a greater code length n with respect to its classical counterpart, suggesting that designers of this family of cryptosystems should take into account also the results derived from our proposal when calibrating their parameters.

To conclude, in Figure 9, we report the classical computational cost of the most advanced ISD algorithm, i.e., BJMM [8], for all the code-based cryptosystems targeting AES-128, pointing out also their requirements in terms of memory. This attack, indeed, trades off the execution time and memory usage, and implies a memory usage of 2^{38}, 2^{39}, and 2^{97} for BIKE, HQC, and McEliece respectively [25]. At the best of our knowledge, quantum circuit designs related to advanced versions of the ISD algorithm (such as BJMM) have still not been investigated, as quantum memory



(a) $R = \frac{3}{4}$, t from GV-bound (b) $R = \frac{1}{2}$, $t = \sqrt{n}$

Fig. 9. Computational cost of a classical implementation of the Prange's ISD strategy, in terms of $\lambda = \log_2$ (no. of gates) (solid blue line) as a function of the code length n, and the classical computational cost of bruteforcing AES-128 (horizontal dotted blue line). The blue point highlights the minimum code length required to match the classical AES-128 security. Quantum computational cost of our quantum implementation of the Prange's ISD strategy, in terms of $\lambda = \log_2(depth)$ (solid red line) as a function of the code length n, and the depth of the quantum circuit implementing AES-128 (horizontal dotted red line). The red point highlights the minimum code length required to match the quantum AES-128 security. Panel (a) refers to QC-MDPC codes with rate 1/2 and no. of errors $t = \sqrt{n}$, e.g., the ones employed in HQC and BIKE cryptosystems; panel (b) refers to algebraic codes with rate 3/4 with no. of error equal to the GV-bound, e.g., the Classic McEliece cryptosystem. From the perspective of a post-quantum code-based cryptosystem designer, panel (a) shows that the code length of QC-MDPC-based ciphers is bounded by the quantum Prange's ISD; panel (b) shows that the code length of the McEliece cryptosystem is equally bounded by both classical and quantum attacks. The plots additionally show the security level for the BIKE, HQC (in (a)) and McEliece (in (b)) proposals when performing a BJMM attack, additionally highlighting their memory requirements.

models does not still have a physical counterpart. Furthermore, the NIST requirements for the cryptanalytical resistance of post-quantum candidates do not consider quantum memories.

5 COMPARISON TO THE STATE OF THE ART

In this section, we offer a comparison of our proposal for QGJE with respect to the state-of-the-art proposals. We also extensively compare our full quantum circuit proposal with respect to the literature, highlighting the differences and similarities in their respect. We conclude this section with a comparison toward other ISD variant, discussing how, although providing improvements in the classical paradigm of computation, do not offer any advantage in the quantum one.

5.1 Alternate Quantum Gauss–Jordan Elimination Approaches

The first work showing a quantum version of the Gauss–Jordan elimination algorithm is Reference [22]. In their proposal, the authors sketch an algorithm adapting Grover's algorithm to find, for each candidate pivot (part 1 of our algorithm), one non-zero entry in the matrix rows following the pivot one. However, for an $r \times r$ matrix, the algorithm requires a number of gates $\in O(2^{r/2})$, that is, exponential in the size of the square matrix.

To the best of our knowledge, the first work proposing a polynomial version of a QGJE is Reference [54], in which the authors report, for the typical $r \times n$ parity-check matrix, a depth $\in O(nr^2)$ and a number of additional qubits $\in O(r^2)$, plus the additional one $r \times r$ needed to represent the matrix.

However, in Reference [34] the authors propose a different implementation of the QGJE, targeting the ISD parity-check matrix. They report an additional number of qubits in the order of $O(k^2)$, but they do not provide exact complexity measures for the number of gates and depth of the circuit. Furthermore, they employ a large number of multi-controlled gates, in which the number of control qubits is sequentially increasing each time the algorithm selects a new pivot. No gate decomposition in the Clifford+T or other implementable gate set is provided.

The authors report the measures of the QGJE for a small example of an 8×8 matrix. Using a generic, multi-qubit gate set, they show a width of 88 and a depth of 1,404. For the same matrix, we report a width of 92 qubits—mainly due to the translation of our multi-controlled gates in one- and two-qubit gates—and a $\approx 14\times$ lower width of 92.

5.2 Comparison to Previous Circuit Design

Bernstein [11] provided the first high-level description of an attack to the classical McEliece cryptographic scheme, showing an asymptotic speedup with respect to the classical case. The work adapts Grover's framework to find the random choice of the Information Set complement $\mathcal{J}$ satisfying the conditions of Prange's variant of ISD and shows how the complexity of an attack to McEliece cryptosystem drops from $c^{(1+o(1))n/\log(n)}$ to $c^{(1/2+o(1))n/\log(n)}$, where $c = 1/(1-R)^{1-R}$, with R being the code rate k/n. The proposal, though, does not provide a quantum circuit and reports only asymptotic complexity values, stating that the oracle would need a number of gates in the order of $O(n^3)$. Our proposal, as we can see from Table 1, cuts the number of gates down to $O(r^3)$, a value obtained from the gate-demanding Gauss–Jordan elimination circuit. With respect to the number of qubits, Reference [11] gives an estimate of $n^{O(1)}$, i.e., polynomial in n. Since the theoretical algorithm proposed represents the whole generator matrix of size $k\times n$ and given that ancillary qubits are most likely needed, it is safe to assume a number of qubits in the order of n^2, which is indeed close to the results obtained by our quantum algorithm based on the parity-check matrix. Finally, we note that the asymptotic analysis in Reference [11] does not consider in the gate counts the cost of the input preparation and, consequently, of the diffusion subcircuits. However, to have a number of basis states with non-zero amplitude in the input superposition exactly equal to $\binom{n}{r}$, which yet is used in Reference [11], it is not possible to use the depth 1 layer of Hadamard gates as per Grover original proposal and hence to neglect the cost in the overall measures. A more complex input preparation subcircuit is needed, such as the one presented in Section 3.1, that produces a non-negligible number of gates, that are in the order of $O(nr)$. Despite the fact that Reference [11] aims at providing a conservative estimate of the cost, our realization has a number of gates smaller by a factor of $\approx 2^4$, as shown in Table 7.

To the best of our knowledge, the first practical implementation of a full quantum circuit based on the theoretical work proposed by Bernstein was presented in Reference [54]. The authors report indeed a number of gates and qubits quite similar to the one reported in Reference [11] but also used additional metrics based on depth that better capture the computational power required by quantum circuits.

This proposal significantly improves the previous work [54] in terms of gate count and, more importantly, total depth. The optimizations made to the QGJE circuit shown in Section 3.2 enabled indeed a depth reduction by a factor of $O(n)$ and a width reduction of $O(r^2)$. The adaptation of the *DOOM* strategy for quasi-cyclic codes to our quantum circuit made also possible a depth reduction of $\sqrt{r}$ at the expense of number of qubits increase of $O(r^2)$. However, since we already use $r\cdot n$ qubits to represent the parity-check matrix, and $O(r^2)$ ancillary qubits (cf. Table 1), this width increase does not have a relevant effect on the overall width. Indeed, as Table 7 confirms, we report a more pronounced reduction for BIKE than for McEliece with respect to Reference [54]. We also report

Table 7. Comparison of Our Figures against Different Works: The Asymptotical Estimates of References [11] and [25], the Results of the Previous Proposal of Reference [54], and a More Recent Work [26]

Algorithm	Sec. Level	Qubits				Number of gates			Depth				$D \cdot W$		
		Table 3	[54]	[26]	[11]	Table 3	[54]	[11]	Table 3	[54]	[26]	[25]	Table 3	[54]	[26]
BIKE (key)	L1	29	29	28	29	108	114	115	93	114	113	105	123	143	141
BIKE (key)	L3	31	31	30	31	142	149	150	127	149	148	140	158	180	178
BIKE (key)	L5	33	32	32	33	178	186	186	162	186	184	176	195	218	215
BIKE (message)	L1	29	29	28	29	104	110	111	89	110	109	101	119	139	137
BIKE (message)	L3	31	31	30	31	139	146	147	123	146	144	136	155	177	175
BIKE (message)	L5	33	32	32	33	173	181	181	157	181	179	171	190	213	210
HQC	L1	30	30	29	30	104	111	112	89	111	110	101	119	141	139
HQC	L3	32	32	31	32	140	148	149	125	148	146	138	157	180	177
HQC	L5	34	33	33	34	173	181	182	157	181	179	171	190	214	212
McEliece	L1	22	22	21	24	102	104	107	92	104	111	95	114	126	133
McEliece	L3	23	23	22	24	125	127	129	115	127	134	118	138	150	156
McEliece	L5	24	24	23	25	165	167	169	154	167	174	158	178	191	198
McEliece	L5	24	24	23	26	165	167	170	155	167	175	158	178	191	198
McEliece	L5	24	24	24	26	184	186	189	173	186	194	177	197	210	218

We report that our proposal outperforms all the other ones in all the relevant metrics, except for a negligible overhead in terms of number of qubits with respect to Reference [26]. The $D \cdot W$ metrics multiplies depth and width (i.e., number of qubits), giving a more reasonable measure of the computational effort required by our implementation. All the values are expressed in base-2 logarithm.

that, with respect to Reference [54], this work also extends the measures two other cryptosystems, namely HQC and the BIKE attack, to recover the private key of the cryptosystem.

Additionally, in the wake of Reference [25], we devoted Section 4.4 to the parallelization of our quantum circuit across distinct instances, reporting a comparison to our work in Table 6. While Reference [11] focuses on an estimation of the number of gates and the number of qubits, in Reference [25] the authors give instead an asymptotic value for the depth of a single Grover iteration, estimated to be in $O(r^{2.5})$. This is slightly above our value of $O(r^2)$ based on a practical quantum circuit implementation. We include their estimates in Table 7.

In Reference [26] the authors propose a circuit implementation similar to the one presented in Reference [54]. Since it is quite close to our proposal, we provide a more detailed comparison in the next section.

5.3 Quantitative Comparison with the State-of-the-art Circuits

In Reference [26], the authors propose an implementation of a quantum circuit for Prange ISD using the same structure of Reference [54]. From the analysis of the work and the inspection of the provided source code, we report better figures in our proposal for all the relevant metrics, as reported in Table 8. The circuit to generate the superposition of $\binom{n}{r}$, i.e., the Dicke state, is similar to Reference [7], which we used in our proposal. However, while Reference [7] obtains a depth of $O(n)$, with no other qubits being used except for the original n containing the superposition, the authors of Reference [26] report a depth of $O(nr \log(\log(r)))$ and a number of qubits equal to $n + 2\lceil \log(r+1) \rceil$. The quantum circuit used to move all the columns of the Information Set complement $\mathcal{J}$ at the beginning of $\mathbf{H}$, in Reference [26] has a depth of $O(rn^2)$. The proposal indeed relies on the direct application of CSWAPs controlled by the qubits storing the Dicke state and has to unroll in the quantum circuit all the possible controlled swaps of elements of the columns in advance. For this reason, they do not need additional qubits. Our proposal, however, stemming from the classical sorting network algorithm introduced in Section 2.5, has a depth in the order of

Table 8. Comparison of Our Figures against the Work Proposed in Reference [26]

Source	Cost metric	Dicke state	Pack columns	Gaussian elimination	Hamming weight compute and check
Table 1	Depth	$O(n)$	$O(\log_2^2(n))$	$O(r^2)$	$O(\log_2^2(r))$
[26]	Depth	$O(nr\log(r)\log(\log(r)))$	rn^2	$O(n^3\log(n))$	Not Reported
Table 1	Qubits	n	$(n-1)\log_2(n)(\log_2(n)-1)$	$\frac{1}{2}r(r-1)$	$M(\frac{5r}{4}-1)$
[26]	Qubits	$n+2\lceil\log(k+1)\rceil$	0	$n-2$	Not Reported

While the detailed depth cost of our different subcircuits is detailed in Table 1, here we only provide their asymptotic version for the sake of clarity. We note that Reference [26] does not report the figures for the Hamming weight check and computation figures, but, analyzing the provided source code, we deduced that both width and depth are at least in the order of $O(n)$.

$O(\log^2(n))$, additionally requiring $O(n\log_2^2(n))$ qubits that, since we already require $n \cdot r$ to store $\mathbf{H}$, do not constitute a significant overhead. The Gauss–Jordan elimination circuit, which is the most expensive circuit of our proposal, has both depth and width in the order of $O(r^2)$. In Reference [26] the equivalent circuit has instead depth $O(n^3\log(n))$, while additionally using $n-2$ qubits. Since $r \leq n/2$, considering only the figure of merit equal to the depth $\times$ width of the circuit, both approaches are comparable. In the Grover's iteration in Reference [26], there is already another subcircuit with depth of $O(rn^2)$, and hence the global depth of the iteration is not significantly increased. However, introducing this variant of the QGJE in our Grover's iteration would produce a significant increase of the depth from $O(r^2)$ to $O(n^3)$. The reduction on the number of qubits, although significant, does not make the tradeoff convenient in our design. Finally, in Reference [26], the authors do not report the depth and width of the Hamming Weight computation and check procedure. From the analysis of the associated source code, their proposal uses an accumulator, adding one by one the qubits containing the syndrome and storing the result in a separate accumulator register. For this reason, the depth is at least linear in r, since the accumulation is strictly sequential, while our implementation has a depth of $O(\log_2(r))$.

In Reference [26], the authors additionally propose a way to reduce the required number of qubits of the circuit by applying, before performing the quantum computation, a classical Gauss–Jordan elimination procedure to the parity-check matrix $\mathbf{H}$ to obtain its systematic form $\mathbf{H}' = [\mathbf{I}_r \mid \mathbf{A}_{r\times k}]$. The main reported advantage is that, by using this approach, we can avoid the representation of the elements of the identity submatrix, effectively sparing r^2 qubits, and hence materialize in the quantum circuit only the $r\times k$ right submatrix $\mathbf{A}$. In the classical case the preprocessing of the matrix allows reducing the number of operations needed in the Gauss–Jordan elimination stage. In the quantum case, however, this approach does not produce any improvement. The authors of Reference [26] report indeed an additional cost of $O(n^2)$ additional row swaps, each one involving $O(n)$ qubits. Even if they do not report the depth of such a circuit, it is plausible that this optimization will have a depth in the order of $O(n^2)$, since swaps of elements belonging to different rows can be interleaved. Since their most demanding subcircuits have already depth in the order of $O(n^3)$, this algorithm will not affect much the depth figures, while providing a reduction of r^2 in terms of used qubits. However, since they report an overall number of qubits in the order of n^2, this improvement will bring only negligible results. Considering an adaptation of this improvement to our proposal, since our Gauss–Jordan elimination algorithm is already in $O(r^2)$, and it is the deepest subcircuit of our total quantum circuit, applying this idea would only increase by one order of magnitude the depth at each iteration.

5.4 Comparing to Other ISD Variants

More advanced classical algorithms for ISD are known, such as the Lee–Brickell variant [45]. For this reason, Bernstein [11] suggested that quantum ISD solvers could indeed provide a speed-up

with respect to the classic Prange variant. However, practical implementations do not seem to offer a tangible advantage through a direct translation, since they rely either on an increase in the number of quantum states—and consequently an increase in the number of Grover's iteration—or into a deeper and wider circuit. Even hybrid approaches that tried to offload some of the computation onto a classical processor (see, for example, Reference [45] or Reference [26]), showed no practical advantage. We refer the reader to Appendix A for a detailed cost analysis of different implementation strategies for the quantum version of a Lee–Brickell attack.

Concerning improved ISD variants, a potentially promising line of research stemmed from Reference [40], where the authors showed how, by generalizing Grover's search framework into a quantum walk framework, a theoretical improvement on the quantum version of the ISD problem is possible. However, to now, no quantum circuit design was proposed to concretize this approach.

6 CONCLUSION

In this work, we present a full quantum circuit implementing an attack against code-based cryptosystems based on Prange variant of the ISD problem accelerated through Grover's framework. We translate our high-level gates in the Clifford+T gate set, the major candidate for noiseless quantum computation and widely employed in cryptographic proposals, comparing our results against the state-of-the-art implementation of a quantum attack against symmetric cryptography. Based on our results, we can state that the large majority of the parameter choices made by the NIST finalist cryptographic scheme s provide conservative margins with respect to the desired target security.

We also compared our measures with the estimates in terms of number of quantum gates given by NIST for attacking AES circuits, as well as to the estimates provided in a more recent proposal. In this case, the upper limit on the maximum depth required to run a single quantum circuit required a parallelization of our algorithm. We found that when the overall depth of our quantum attack is compared to the estimates provided by NIST, the McEliece cryptographic scheme at level 3 falls slightly below the requirements, while it is tightly above the requirements if we consider instead the overall number of gates of our attack. Further studies are needed to understand the effect of the parallelization on the quantitative measures for all the cryptographic schemes and provide more concrete figures.

In implementing our proposals, we developed reversible circuits to sort bitstrings—adapted in our circuit to perform matrix column permutations—to compute the Hamming weight of a given bitstring, and, most notably, a circuit performing Gauss–Jordan elimination on a matrix, which can be of independent interest. Such an implementation, for example, may also be of interest in devising new quantum circuits to attack code-equivalence-based public-key digital signature primitives, such as LESS [14]. We note as well that the ISD strategy can be used to find solutions to the Permuted Kernel Problem [13]. In general, all the components we presented can be used in any quantum accelerated attack to code-based cryptosystems based on ISD, since all of them reuse the same general structures.

All the subcomponents composing our quantum circuit have been fully validated and tested using the Atos Quantum Learning Machine [4] simulator. The source code related to all the components of our circuit is available at https://github.com/paper-codes/2023-TQuantum.[1]

Finally, we report that our algorithm solves a general combinatorial problem, and it can be seen as a binary constraint satisfaction problem. We note as well that it may easily be adapted to compute the spark of a matrix, which can be of independent interest.

[1] https://doi.org/10.5281/zenodo.8039516.

APPENDIX

A ANALYSIS OF THE QUANTUM VERSION OF LEE–BRICKELL

In the following, we analyze the improvement proposed by Lee and Brickell [45], highlighting why a set of strategies for the transposition of the approach onto our quantum ISD solver does not provide speedups.

Lee and Brickell analyzed the costs of allowing a fixed number of asserted bits in the k bits indexed by $\mathcal{I}$. In other words, while in Prange we had $\text{WT}(\mathbf{e}_{\mathcal{I}}) = 0$ and $\text{WT}(\mathbf{e}_{\mathcal{J}}) = t$, in Lee and Brickell's variant we assume $\text{WT}(\mathbf{e}_{\mathcal{I}}) = p$ and $\text{WT}(\mathbf{e}_{\mathcal{J}}) = t - p$. For this reason, the number of admissible vectors increases from $\binom{r}{t}$ to $\binom{r}{t-p}\binom{k}{p}$, leading to a probability of success of a single iteration given by $\text{Pr}_{succ-LB} = \binom{r}{t-p}\binom{k}{p}/\binom{n}{t}$. As we did for Prange, we can alternatively rephrase the probability of success as $\binom{n-t}{r-(t-p)}\binom{t}{t-p}/\binom{n}{r}$, since we want to put, in the error vector indexed by $\mathcal{J}$, all the $n - t$ zeros in $r - (t - p)$ positions and all t ones in the remaining $t - p$ positions.

The increase in the probability of success of a single iteration leads to a reduction in the expected number of iterations, at the price of an increased cost of a single iteration of the algorithm, $\text{C}_{iter-LB}$. Indeed, after performing the GJE procedure (line 4 of Algorithm 1), we obtain the matrix $\widetilde{\mathbf{H}} = [\mathbf{W} \mid \mathbf{V}]$ and the corresponding vector $\tilde{\mathbf{s}}$ that we should use to retrieve the unknown error vector $\hat{\mathbf{e}}$ such that $\tilde{\mathbf{s}} = \widetilde{\mathbf{H}}\hat{\mathbf{e}}$. The previous equation can be expanded as $\tilde{\mathbf{s}} = [\mathbf{W} \mid \mathbf{V}][\mathbf{e}_{\mathcal{J}} \mid \mathbf{e}_{\mathcal{I}}]$ and can be seen as the bitwise addition of two distinct vectors, namely $\tilde{\mathbf{s}} = \mathbf{W}\mathbf{e}_{\mathcal{J}} \oplus \mathbf{V}\mathbf{e}_{\mathcal{I}}$. If, as per Prange's hypothesis, we have $\mathbf{W} = \mathbf{I}_r$, signalling that $\mathbf{H}_{\mathcal{J}}$ is invertible, then we can rewrite the previous equation as $\mathbf{e}_{\mathcal{J}} = \tilde{\mathbf{s}} \oplus \mathbf{V}\mathbf{e}_{\mathcal{I}}$. Lee–Brickell's algorithm assumes that $\text{WT}(\mathbf{e}_{\mathcal{J}}) = t - p$ and $\text{WT}(\mathbf{e}_{\mathcal{I}}) = p$, and hence the multiplication $\mathbf{V}\mathbf{e}_{\mathcal{J}}$ can be seen as the sum of the p columns of $\mathbf{V}$ indexed by the position of the p asserted bits of $\mathbf{e}_{\mathcal{I}}$. Given this, in Lee–Brickell algorithm we insert, after line 5 of Algorithm 1, an additional iteration over all the $\binom{k}{p}$ possible picks of p columns of the matrix $\mathbf{V}$, reporting a success if, for one of these picks, adding together the p columns of $\mathbf{V}$ to $\tilde{\mathbf{s}}$ gives a vector of weight $t - p$. The cost of a single iteration for classical Lee–Brickell's algorithm is

$$\frac{\text{C}_{iter-LB}}{\text{Pr}_{succ-LB}} \approx \frac{\binom{n}{t}}{0.288\binom{r}{t-p}\binom{k}{p}}\left(\text{C}_{GJE} + \binom{k}{p}(\text{C}_{SUM} + \text{C}_{HWCC})\right), \tag{12}$$

in which C_{SUM} is the cost of adding p vectors of length r to the syndrome at each iteration, having a classical cost $\in O(rp)$, while C_{HWCC} is the cost of computing, each time, the Hamming weight of the resulting vector. By comparing the probabilities of success of Lee–Brickell and Prange variants, we obtain

$$\frac{\text{Pr}_{succ-LB}}{\text{Pr}_{succ-PR}} = \frac{\binom{r}{t-p}\binom{k}{p}}{\binom{n}{t}}\frac{\binom{n}{t}}{\binom{r}{t}} = \binom{k}{p}\frac{\binom{r}{t-p}}{\binom{r}{t}} = \binom{k}{p}\frac{t!}{(t-p)!}\frac{(r-t)!}{(r-(t-p))!}, \tag{13}$$

which exponentially increases for increasing $p \ll t$. Consequently, the expected number of iterations, obtained as the inverse of the probability of success, shows an exponential decrease, which is balanced by the $\binom{k}{p}$ number of distinct sums of columns required, at each iteration, to sum p columns of $\mathbf{V}$ to the syndrome. For this reason, while the admissible values for p are in $\{1, \ldots, t-1\}$, the tradeoff between the number of iterations and the computational cost of a single iteration shows its optimal value for the classic Lee–Brickell solver at $p = 2$, as shown in Reference [5].

When switching from Prange's ISD variant to Lee–Brickell's in a quantum setting, the number of required Grover iterations decrease of a quantity proportional to the square root of the inverse of Equation (13), as a consequence of the incremented number of admissible solutions. This comes at the cost of adapting the circuit to perform the inner $\binom{k}{p}$ column additions, for which different

strategies can be explored. Before analyzing them, we note that in Lee–Brickell's approach we need the entire result of the QGJE on the $\mathbf{H}$ matrix, and hence Optimization 4 cannot be applied.

Adaptation strategy 1. The idea of *Strategy 1* is to implement a reversible circuit capable of generating all the $\binom{k}{p}$ possible choices of the columns of $\mathbf{V}$ and check the result of their addition to $\tilde{\mathbf{s}}$. If the sum is, as required, equal to $t - p$, then we set an additional ancilla to $|1\rangle$ to signal a success. This strategy requires a reversible circuit capable of enumerating, on an auxiliary register `comb` of length k, all the $\binom{k}{p}$ choices of vectors of length k and Hamming weight p, one at a time. After that, we can use the qubits of `comb` to conditionally sum the columns of $\mathbf{V}$ to the quantum register storing $\tilde{\mathbf{s}}$ through CCNOT gates. Any possible algorithm will have at least depth $O(k)$ to generate each item of the enumeration and has to be performed $\binom{k}{p}$ times. Even with $p = 1$, i.e., the smallest possible p value, we obtain a circuit depth of $O(k^2)$. Given that the deepest circuit in our plain Prange's algorithm has a depth in $O(r^2)$ and that $k \geq r$, the depth of the overall circuit will be substantially increased, nullifying the benefits of a decreased number of iterations.

Adaptation strategy 2. Given the complexity of the previous quantum circuit and the fact that p is fixed, one can think of obtaining the combinations of columns sums needed for each $\binom{k}{p}$ at circuit generation time, i.e., classically, and hence translate the obtained sequence of sums into the corresponding CNOT quantum gates. In other words, *Strategy 2* will not use an additional quantum register to drive the sum of columns of $\mathbf{V}$ in the quantum circuit, but, while classically generating the quantum circuit, will decide which gates to apply based on classical results. This is the approach sketched in Reference [26], where the authors report an additional cost of $O(\binom{k}{p}p)$ for this circuit and no additional qubits. Indeed, each of the $\binom{k}{p}$ combination has to add together p columns. However, each column addition involves r elements sum, done through `CNOT` gates, and since $p \ll r$, there is not much room for parallelization between elements belonging to different columns, giving therefore a depth that is more realistically equal to $O(\binom{k}{p}r)$. Additionally, Reference [26] does not consider the cost of computing and checking the Hamming weight for each pick of p columns. Using our approach for this subcircuit would require $O(r)$ qubits and $O(\log_2^2 r)$ depth. Finally, each sum of p columns must be uncomputed before proceeding to the next combination. We argue therefore that the overall depth will be in the order of $O(\binom{k}{p}(2 \cdot r \cdot \log_2^2 r))$. Once again, even the smallest value for p, that is, $p = 1$, will give us a depth equal to $O(kr)$, making it again at least as deep as the QGJE circuit, and hence bringing no concrete benefit.

An important drawback of Strategies 1 and 2 is that, at each Grover iteration, the inner circuit performing the column addition of all the p columns of $\mathbf{V}$ is uncomputed, meaning that, upon measurement, we will not have any way to tell which of the possible $\binom{k}{p}$ pick of columns of $\mathbf{V}$ generated the right result. In other words, upon measurement, we will measure multiple choices of $\mathcal{J}$ that generated an invertible matrix $\mathbf{H}_{\mathcal{J}}$ and for which some pick of p columns of the matrix $\mathbf{V}$ gave the correct result. After the execution of the whole quantum circuit, we have to introduce an additional classical step, performing a permutation and a GJE step on $\mathbf{H}$ based on the value of $\mathcal{J}$ found through the quantum circuit, followed by an exhaustive search on the obtained rightmost submatrix $\mathbf{V}$ to detect which of its columns produces the correct result.

Adaptation strategy 3. The idea behind *Strategy 3* is to generate, alongside the original Dicke state generating a superposition of all the $\binom{n}{r}$ possible choices of $\mathcal{J}$, another one generating a superposition of all the $\binom{k}{p}$ pick of columns of $\mathbf{V}$. By using this approach, we still need the CCNOTs for the sum, but we need a single Hamming weight compute and check circuit at each Grover iteration. The additional subcircuit will have therefore a depth in the order of $O(k)$, since $k \cdot r$ CCNOT gates must be computed among all the qubits representing the elements of the matrix and,

Table 9. Comparison of the Different Strategies to Translate the Lee–Brickell ISD Variant into a Quantum Circuit with Respect to the Prange ISD Variant Shown in This Work

ISD variant	Qubits	Grover iterations	Grover iteration depth	Classical iterations
Prange	$rn + r^2 + n$	$\sqrt{\frac{\binom{n}{t}}{\binom{r}{t}}}$	$O(r^2)$	—
L-B Strategy 1	$rn + r^2 + n + k$	$\sqrt{\frac{\binom{n}{t}}{\binom{r}{t-p}\binom{k}{p}}}$	$O(r^2) + \binom{k}{p}O(k)$	—
L-B Strategy 2	$rn + r^2 + n$	$\sqrt{\frac{\binom{n}{t}}{\binom{r}{t-p}\binom{k}{p}}}$	$O(r^2) + \binom{k}{p}O(r)$	—
L-B Strategy 3	$rn + r^2 + n + k$	$\sqrt{\frac{\binom{n}{t}\binom{k}{p}}{\binom{r}{t-p}\binom{k}{p}}}$	$O(r^2) + O(k)$	—
L-B Strategy 4 [55]	$O(kr)$	$\sqrt{\binom{k}{p}}$	$O(k)$	$\frac{\binom{n}{t}}{\binom{r}{t-p}\binom{k}{p}}$

The number of classical iterations is required only in the hybrid approach of Strategy 4. The product of values on each row equals the D · W cost of the corresponding strategy.

since $r \leq k$, the CCNOT gates can be interleaved. This strategy has the main drawback that the number of input states increases by $\binom{k}{p}$, leading therefore to an additional $\sqrt{\binom{k}{p}}$ Grover iterations with respect to the other strategies, leaving therefore only a small reduction in the number of iterations with respect to Prange. Once again, no substantial improvement can be obtained with this approach.

Adaptation strategy 4. A final strategy, *Strategy 4*, explored in Reference [55], is to accelerate only one part of Lee–Brickell algorithm using a quantum circuit. In Reference [55], for example, the authors try to accelerate only the inner loop of the algorithm, by applying Grover's framework to find which pick of p columns of $\mathbf{V}$ among all the possible ones gives the correct solution. However, while they show that the quantum circuit alone has a reasonable depth and width, it has to be executed a number of times equal to the number of iterations required by the classical Lee–Brickell algorithm, and hence does not bring many benefits.

Table 9 summarizes the costs of the previous strategies to parallelize the Lee–Brickell approach, additionally comparing them to the plain Prange approach. In conclusion, translating the Lee–Brickell approach onto a quantum implementation does not appear to provide a significant improvement to the efficiency of the solver, as it is the case for the classical ISD solver counterpart.

REFERENCES

[1] Miklós Ajtai, János Komlós, and Endre Szemerédi. 1983. An O(n log n) sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas (Eds.). ACM, 1–9. https://doi.org/10.1145/800061.808726

[2] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. 2013. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 32, 6 (2013), 818–830. https://doi.org/10.1109/TCAD.2013.2244643

[3] Nicolas Aragon, Paulo S. L. M. Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, et al. 2017. BIKE: Bit Flipping Key Encapsulation. Retrieved from https://bikesuite.org.

[4] Atos. 2019. *Quantum Learning Machine*. Retrieved from https://atos.net/en/solutions/quantum-learning-machine.

[5] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. 2019. A finite regime analysis of information set decoding algorithms. *Algorithms* 12, 10 (2019), 209. https://doi.org/10.3390/a12100209

[6] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter. 1995. Elementary gates for quantum computation. *Phys. Rev. A* 52, 5 (Nov. 1995), 3457–3467. https://doi.org/10.1103/PhysRevA.52.3457

[7] Andreas Bärtschi and Stephan J. Eidenbenz. 2019. Deterministic preparation of dicke states. In *Proceedings of the 22nd International Symposium on the Fundamentals of Computation Theory (FCT'19) Lecture Notes in Computer Science*, Vol. 11651, Leszek Antoni Gasieniec, Jesper Jansson, and Christos Levcopoulos (Eds.). Springer, 126–139. https://doi.org/10.1007/978-3-030-25027-0_9

[8] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. 2012. Decoding random binary linear codes in 2 n/20: How 1 + 1 = 0 improves information set decoding. In *Proceedings of the 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'12), Lecture Notes in Computer Science*, Vol. 7237, David Pointcheval and Thomas Johansson (Eds.). Springer, 520–536. https://doi.org/10.1007/978-3-642-29011-4_31

[9] Emanuele Bellini and Andre Esser. 2021. Syndrome Decoding Estimator. Retrieved from https://github.com/Crypto-TII/syndrome_decoding_estimator.

[10] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. 1978. On the inherent intractability of certain coding problems (Corresp.). *IEEE Trans. Inf. Theory* 24, 3 (1978), 384–386. https://doi.org/10.1109/TIT.1978.1055873

[11] Daniel J. Bernstein. 2010. Grover vs. McEliece. In *Post-Quantum Cryptography*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, and Nicolas Sendrier (Eds.). Vol. 6061. Springer, Berlin, 73–80. https://doi.org/10.1007/978-3-642-12929-2_6

[12] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Misoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, and Wen Wang. 2020. Classic McEliece: Conservative Code-based Cryptography. Retrieved from https://classic.mceliece.org/nist/mceliece-20201010.pdf.

[13] Ward Beullens. 2020. Sigma protocols for MQ, PKP and SIS, and fishy signature schemes. In *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'20), Part III, Lecture Notes in Computer Science*, Vol. 12107, Anne Canteaut and Yuval Ishai (Eds.). Springer, 183–211. https://doi.org/10.1007/978-3-030-45727-3_7

[14] Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. 2020. LESS is more: Code-based signatures without syndromes. In *Proceedings of the 12th International Conference on Cryptology in Africa (AFRICACRYPT'20), Lecture Notes in Computer Science*, Vol. 12174, Abderrahmane Nitaj and Amr M. Youssef (Eds.). Springer, 45–65. https://doi.org/10.1007/978-3-030-51938-4_3

[15] Joan Boyar, Magnus Gausdal Find, and René Peralta. 2019. Small low-depth circuits for cryptographic applications. *Cryptogr. Commun.* 11, 1 (2019), 109–127. https://doi.org/10.1007/s12095-018-0296-3

[16] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. 1998. Tight bounds on quantum searching. *Fortschr. Phys.: Progr. Phys.* 46, 4-5 (1998), 493–505.

[17] P. Oscar Boykin, Tal Mor, Matthew Pulver, Vwani P. Roychowdhury, and Farrokh Vatan. 2000. A new universal and fault-tolerant quantum basis. *Inform. Process. Lett.* 75, 3 (2000), 101–107. https://doi.org/10.1016/S0020-0190(00)00084-3

[18] Wouter Castryck and Thomas Decru. 2022. An efficient key recovery attack on SIDH (Preliminary Version). Cryptology ePrint Archive, Paper 2022/975.

[19] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms, Second Edition.* The MIT Press and McGraw–Hill.

[20] Steven A. Cuccaro, Thomas G. Draper, Samuel A. Kutin, and David Petrie Moulton. 2004. A new quantum ripple-carry addition circuit. https://doi.org/10.48550/ARXIV.QUANT-PH/0410184

[21] James H. Davenport and Benjamin Pring. 2020. Improvements to quantum search techniques for block-ciphers, with applications to AES. In *Proceedings of the 27th International Conference on Selected Areas in Cryptography (SAC'20), Revised Selected Papers, Lecture Notes in Computer Science*, Vol. 12804. Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn (Eds.). Springer, 360–384. https://doi.org/10.1007/978-3-030-81652-0_14

[22] Do Ngoc Diep, Do Hoang Giang, and Nguyen Van Minh. 2017. Quantum gauss-jordan elimination and simulation of accounting principles on quantum computers. *Int. J. Theor. Phys.* 56, 6 (March 2017), 1948–1960. https://doi.org/10.1007/s10773-017-3340-8

[23] Thomas G. Draper. 2000. Addition on a quantum computer. https://doi.org/10.48550/ARXIV.QUANT-PH/0008033

[24] Thomas G. Draper, Samuel A. Kutin, Eric M. Rains, and Krysta M. Svore. 2006. A logarithmic-depth quantum carry-lookahead adder. *Quant. Inf. Comput.* 6, 4 (2006), 351–369. https://doi.org/10.26421/QIC6.4-5-4

[25] Andre Esser and Emanuele Bellini. 2022. Syndrome decoding estimator. In *Proceedings of the 25th IACR International Conference on Practice and Theory of Public-Key Cryptography (PKC'22), Part I, Lecture Notes in Computer Science*, Vol. 13177, Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe (Eds.). Springer, 112–141. https://doi.org/10.1007/978-3-030-97121-2_5

[26] Andre Esser, Sergi Ramos-Calderer, Emanuele Bellini, José I. Latorre, and Marc Manzano. 2021. An optimized quantum implementation of ISD on scalable quantum resources. Cryptology ePrint Archive, Paper 2021/1608.
[27] European Telecommunications Standards Institute (ETSI). 2020. Quantum-safe cryptography. Retrieved from https://www.etsi.org/technologies/quantum-safe-cryptography.
[28] Craig Gidney. 2018. Halving the cost of quantum addition. *Quantum* 2, 1 (June 2018), 74. https://doi.org/10.22331/q-2018-06-18-74
[29] E. N. Gilbert. 1952. A comparison of signalling alphabets. *The Bell Syst. Techn. J.* 31, 3 (1952), 504–522. https://doi.org/10.1002/j.1538-7305.1952.tb01393.x
[30] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. 2016. Applying Grover's algorithm to AES: Quantum resource estimates. In *Proceedings of the 7th International Workshop on Post-Quantum Cryptography (PQCrypto'16), Lecture Notes in Computer Science*, Vol. 9606, Tsuyoshi Takagi (Ed.). Springer, 29–43. https://doi.org/10.1007/978-3-319-29360-8_3
[31] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, Gary L. Miller (Ed.). ACM, 212–219. https://doi.org/10.1145/237814.237866
[32] Zhenyu Huang and Siwei Sun. 2022. Synthesizing quantum circuits of AES with lower T-depth and less qubits. In *Proceedings of the 28th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'22), Part III, Lecture Notes in Computer Science*, Vol. 13793. Shweta Agrawal and Dongdai Lin (Eds.). Springer, 614–644. https://doi.org/10.1007/978-3-031-22969-5_21
[33] Kyungbae Jang, Anubhab Baksi, Gyeongju Song, Hyunji Kim, Hwajeong Seo, and Anupam Chattopadhyay. 2022. Quantum analysis of AES. Cryptology ePrint Archive, Paper 2022/683.
[34] Kyungbae Jang, Hyunji Kim, and Hwajeong Seo. 2022. Quantum Gauss-Jordan elimination for code in quantum. In *Proceedings of the International Conference on Platform Technology and Service (PlatCon'22)*. IEEE, 44–47. https://doi.org/10.1109/platcon55845.2022.9932108
[35] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, et al. 2019. Supersingular Isogeny Key Encapsulation SIKE. Retrieved from https://sike.org.
[36] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. 2019. Implementing Grover Oracles for Quantum Key Search on AES and LowMC. https://doi.org/10.48550/ARXIV.QUANT-PH/0410184
[37] Samuel Jaques, Michael Naehrig, Martin Roetteler, and Fernando Virdia. 2020. Implementing grover oracles for quantum key search on AES and LowMC. In *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'20), Part II, Lecture Notes in Computer Science*, Vol. 12106, Anne Canteaut and Yuval Ishai (Eds.), Vol. 12106. Springer, 280–310. https://doi.org/10.1007/978-3-030-45724-2_10
[38] Samuel Jaques and John M. Schanck. 2019. Quantum cryptanalysis in the RAM model: Claw-finding attacks on SIKE. In *Proceedings of the 39th Annual International Cryptology Conference (CRYPTO'19), Part I, Lecture Notes in Computer Science*, Vol. 11692, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, 32–61. https://doi.org/10.1007/978-3-030-26948-7_2
[39] Cody Jones. 2013. Low-overhead constructions for the fault-tolerant toffoli gate. *Phys. Rev. A* 87, 2 (2013), 022328.
[40] Ghazal Kachigar and Jean-Pierre Tillich. 2017. Quantum information set decoding algorithms. In *Proceedings of the 8th International Workshop on Post-Quatnum Cryptography (PQCrypto'17), Lecture Notes in Computer Science*, Vol. 10346. Tanja Lange and Tsuyoshi Takagi (Eds.). Springer, 69–89. https://doi.org/10.1007/978-3-319-59879-6_5
[41] Taewan Kim and Byung-Soo Choi. 2018. Efficient decomposition methods for controlled-R n using a single ancillary qubit. *Sci. Rep.* 8, 1 (2018), 1–7.
[42] Vadym Kliuchnikov, Dmitri Maslov, and Michele Mosca. 2016. Practical approximation of single-qubit unitaries by single-qubit quantum clifford and T circuits. *IEEE Trans. Comput.* 65, 1 (2016), 161–172. https://doi.org/10.1109/TC.2015.2409842
[43] Donald Ervin Knuth. 1998. *The Art of Computer Programming, Volume III*, (2nd ed.). Addison-Wesley.
[44] Brandon Langenberg, Hai Pham, and Rainer Steinwandt. 2020. Reducing the cost of implementing the advanced encryption standard as a quantum circuit. *IEEE Trans. Quant. Eng.* 1, 1 (2020), 1–12. https://doi.org/10.1109/TQE.2020.2965697
[45] Pil Joong Lee and Ernest F. Brickell. 1988. An observation on the security of McEliece's public-key cryptosystem. In *Proceedings of theWorkshop on the Theory and Application of of Cryptographic Techniques (EUROCRYPT'88), Lecture Notes in Computer Science*, Vol. 330, Christoph G. Günther (Ed.). Springer, 275–280. https://doi.org/10.1007/3-540-45961-8_25
[46] Robert J. McEliece. 1978. A public-key cryptosystem based on algebraic. *Cod. Thv* 42, 43 (1978), 114–116.
[47] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loıc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and IC Bourges. 2018. Hamming quasi-cyclic (HQC). Retrieved from https://pqc-hqc.org/.

[48] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. 2013. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Proceedings of the IEEE International Symposium on Information Theory*. IEEE, 2069–2073. https://doi.org/10.1109/ISIT.2013.6620590

[49] Dustin Moody. 2022. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. Technical Report NIST IR 8413. National Institute of Standards and Technology, Gaithersburg, MD. NIST IR 8413. https://doi.org/10.6028/NIST.IR.8413

[50] C. S. Mukherjee, S. Maitra, V. Gaurav, and D. Roy. 2020. Preparing dicke states on a quantum computer. *IEEE Trans. Quant. Eng.* 1, 1 (2020), 1–17. https://doi.org/10.1109/TQE.2020.3041479

[51] National Institute of Standards and Technology. 2016. Announcing Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms. Retrieved from https://federalregister.gov/a/2016-30615.

[52] National Institute of Standards and Technology. 2016. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. Retrieved from https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[53] Michael A. Nielsen and Isaac L. Chuang. 2010. *Quantum Computation and Quantum Information*. Cambridge University Press.

[54] Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. 2021. A complete quantum circuit to solve the information set decoding problem. In *Proceedings of the IEEE International Conference on Quantum Computing and Engineering, (QCE'21)*, Hausi A. Müller, Greg Byrd, Candace Culhane, and Travis Humble (Eds.). IEEE, 366–377. https://doi.org/10.1109/QCE52317.2021.00056

[55] Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. 2021. A quantum circuit to speed-up the cryptanalysis of code-based cryptosystems. In *Proceedings of the 17th EAI International Conference on Security and Privacy in Communication Networks (SecureComm'21), Part II, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Vol. 399, Joaquín García-Alfaro, Shujun Li, Radha Poovendran, Hervé Debar, and Moti Yung (Eds.). Springer, 458–474. https://doi.org/10.1007/978-3-030-90022-9_25

[56] Eugene Prange. 1962. The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory* 8, 5 (1962), 5–9. https://doi.org/10.1109/TIT.1962.1057777

[57] John Proos and Christof Zalka. 2003. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quant. Inf. Comput.* 3, 4 (2003), 317–344. https://doi.org/10.26421/QIC3.4-3

[58] Mehdi Saeedi and Massoud Pedram. 2013. Linear-depth quantum circuits for n-Qubit Toffoli gates with no ancilla. *Phys. Rev. A* 87, 6 (2013), 062318.

[59] Peter Selinger. 2013. Quantum circuits of T-depth one. *Phys. Rev. A* 87, 4 (2013), 042302.

[60] Nicolas Sendrier. 2011. Decoding one out of many. In *Proceedings of the 4th International Workshop on Post-Quantum Cryptography (PQCrypto'11), Lecture Notes in Computer Science*, Vol. 7071. Bo-Yin Yang (Ed.). Springer, 51–67. https://doi.org/10.1007/978-3-642-25405-5_4

[61] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Techn. J.* 27, 3 (1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[62] Peter W. Shor. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE, 124–134. https://doi.org/10.1109/SFCS.1994.365700

[63] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. 2010. Quantum addition circuits and unbounded fan-out. *Quant. Inf. Comput.* 10, 9&10 (2010), 872–890. https://doi.org/10.26421/QIC10.9-10-12

[64] Rodolfo Canto Torres and Nicolas Sendrier. 2016. Analysis of information set decoding for a sub-linear error weight. In *Proceedings of the 7th International Workshop on Post-Quantum Cryptography (PQCrypto'16), Lecture Notes in Computer Science*, Vol. 9606, Tsuyoshi Takagi (Ed.). Springer, 144–161. https://doi.org/10.1007/978-3-319-29360-8_10

[65] Rom Rubenovich Varshamov. 1957. Estimate of the number of signals in error correcting codes. *Dokl. Akad. Nauk SSSR* 117, 1 (1957), 739–741.

[66] Ze-Guo Wang, Shi-Jie Wei, and Gui-Lu Long. 2022. A quantum circuit design of AES requiring fewer quantum qubits and gate operations. *Front. Phys.* 17, 4 (2022), 1–7.

[67] Jian Zou, Zihao Wei, Siwei Sun, Ximeng Liu, and Wenling Wu. 2020. Quantum circuit implementations of AES with fewer qubits. In *Proceedings of the 26th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'20), Part II, Lecture Notes in Computer Science*, Vol. 12492, Shiho Moriai and Huaxiong Wang (Eds.). Springer, 697–726. https://doi.org/10.1007/978-3-030-64834-3_24

Received 5 August 2022; revised 14 June 2023; accepted 16 June 2023