This preprint has not undergone peer review or any post-submission improvements or corrections. The Version of Record of this contribution is published in
Applied cryptography and network security workshops, M. Andreoni, Ed., Cham: Springer Nature Switzerland, 2024, pp. 8–28.

and is available online at *https://doi.org/10.1007/978-3-031-61489-7_2*

Quantum Circuit Design for the Lee-Brickell based Information Set Decoding

Simone Perriello[1][0000−0001−9656−7252], Alessandro Barenghi[1][0000−0003−0840−6358], and Gerardo Pelosi[1][0000−0002−3812−5429]

Department of Electronics, Information and Bioengineering - DEIB
Politecnico di Milano, 20133 Milano, Italy
{simone.perriello,alessandro.barenghi,gerardo.pelosi}@polimi.it

Abstract. In the race for quantum-safe cryptography, fostered by the ongoing National Institute of Standards and Technology (NIST) post-quantum standardization process, it is crucial to assess the security of the emerging cryptoschemes. In this work, we propose a fully quantum algorithm to accelerate the Lee-Brickell variant of the Information Set Decoding (ISD) on binary error correcting codes — one of the main cryptanalytic techniques used for assessing the security of code-based cryptoschemes. Our solution relies on a careful scheduling of the quantum gates included in the circuit design, coupled with a strategy that applies multiple times the oracle-reflection, from a Grover-like search, within a single Grover iteration. Compared with the state-of-the-art alternatives, our solution shows a reduction of the circuit depth ranging between 2^3 and 2^{26}, when considering the parameters sets for code-based cryptosystems advanced to the fourth round of the NIST process. Denoting as t and $t-p$ the two sets of bit flips tackled by the Lee-Brickell's strategy, as an additional noteworthy fact we show that our solution exhibits 1 as the best value for p instead of 2 as it is the case for the classic ISD.

Keywords: code-based cryptography· post-quantum cryptography· quantum computing· Information Set Decoding· ISD

1 Introduction

Shor's quantum algorithm [1], with its ability to compute the order factorization and discrete logarithm computations exponentially faster than any classical algorithm, threatens widely used public-key cryptosystems like RSA and ECC. Acknowledging this threat, prominent organizations like the *National Institute of Standards and Technology (NIST)* are actively engaged in standardizing cryptographic primitives for *Post-Quantum Cryptography (PQC)*, aiming to safeguard classical data against quantum-accelerated attacks. Among the PQC options, *Code-Based Cryptography (CBC)* has emerged as a promising alternative. Its security hinges on the *Syndrome Decoding Problem (SDP)*, whose decision version is known to be NP-complete [2]. Notably, NIST's report on the third (latest at the time of writing) round [3] of its standardization call [4] suggests that all the

CBC systems will be advanced from the third to the fourth round, underscoring their potential as sound alternatives to lattice-based cryptosystems. Given the continuing relevance of CBC, particularly evident in the proposals submitted to the first (latest at the time of writing) round of NIST's additional call for digital signature schemes [5], it is essential to meticulously assess the computational complexity needed to undermine the security of CBC schemes, allowing designers to precisely tune their parameters to meet security requirements without compromising efficiency. Classically, the most effective cryptanalytic tool relies on the *Information Set Decoding (ISD)* technique, dating back to Prange's original proposal [6] that was later slightly improved in several ways [7–15]. On the quantum front, attacks relying on Grover's framework or quantum-walk frameworks exhibit only an asymptotic quadratic speed-up compared to classical ISD approaches.

Related works. The first work showing a reduction in the computational complexity of a quantum-accelerated adaptation of the Prange's ISD variant with respect to its classical counterpart was proposed in [16]. The proposal, relying on Grover's framework, paved the way to a few practical implementations of quantum circuits designs of such an attack, based both on Prange's ISD variant [17–19], and Lee-Brickell's variant [18–20], with the design presented in [21] offering, to the best of our knowledge, the best figures in terms of all the relevant cost metrics related to quantum circuits (namely, number of qubits, number of gates, depth and depth×width). In [21], the authors additionally argue that the Lee-Brickell's variant of the ISD is unlikely to produce significant improvements on the plain Prange's variant of the ISD, providing evidences of their assumption based on the analysis of the quantum circuits presented in both [20], providing a design of a hybrid classical-quantum approach based on Lee-Brickell, and [18], proposing instead a full-quantum approach for both Prange's and Lee-Brickell's variants. On a parallel research path, the quantum-walk based approaches [22, 23], aiming to accelerate more advanced ISD variants, were shown to provide a theoretical speed-up with respect to the Grover's based approaches. However, to the best of our knowledge, the literature still lacks a concrete gate-based design for these approaches, making therefore challenging a precise assessment of their complexity in the finite regime.

Contribution. In this work, we describe a quantum circuit to accelerate the Lee-Brickell's ISD variant [7] based on Grover's quantum framework [24]. By relying on a computational model akin to [25], in which the quantum unit acts as a memory peripheral controlled by the classical unit, we show how a careful design of the classical algorithm generating the quantum circuits is capable of reducing the complexity metrics. Disproving the conjectures made in [21], our quantum circuit design shows a significantly reduced computational complexities compared to the state-of-the-art designs, with gains ranging from 2^4 to 2^{22} in the depth and depth $\times$ width metrics with respect to all the CBC schemes advanced to the fourth round of NIST's standardization call [3]. In detailing all of our circuit components, we additionally show an implementation of Grover's oracle operator diverging from the traditional compute-uncompute pattern employed in its implementation. Finally, as an interesting side result, our proposal additionally

shows that our quantum adaptation of the Lee-Brickell's variant shows that the optimal value of the parameter p used in such variant is equal to 1. This result is in contrast with the classical results, for which the optimal value of such parameter was proven to be equal to 2 [26].

2 Background

In the following, we introduce our notation and provide a summary of the concepts on Information Set Decoding (ISD) techniques. We denote sets using calligraphic capitalized letters, as in $\mathcal{S}$, and their cardinality as $|\mathcal{S}|$. The notation $[x]$ represents the set of all integers within the range $[0, x-1]$. For sets composed of all the size-y subsets containing integers within $[x]$, we use the notation $\binom{[x]}{y}$.

Vectors are denoted with lowercase bold letters, as in $\boldsymbol{v}$, and intended as column vectors. Matrices are denoted with uppercase bold letters, as in $\boldsymbol{M}$. Elements within vectors and matrices belong to $\boldsymbol{F}_2$. The notation $\boldsymbol{v}_{|i}$ denotes the element at index i of the vector $\boldsymbol{v}$, while $\boldsymbol{v}_{|\mathcal{S}}$ denotes the projection of the elements of $\boldsymbol{v}$ on the indexes of $\mathcal{S}$. The number of elements of $\boldsymbol{v}$ is denoted as $|\boldsymbol{v}|$, while its Hamming weight as $\mathsf{Hw}(\boldsymbol{v})$. Similarly, the subscript $\boldsymbol{M}_{|i,j}$ denotes the element of $\boldsymbol{M}$ at row i and column j, while $\boldsymbol{M}_{|\mathcal{S}_1,\mathcal{S}_2}$ denotes a selection of the rows indexed by the elements in $\mathcal{S}_1$ and of the columns indexed by the element in $\mathcal{S}_2$. In this context, we use the symbol $*$ to represent the set of all rows or columns, as in $\boldsymbol{M}_{|i,*}$, representing the selection of all the elements at row i. Given $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$, the notation $[\boldsymbol{v_1}\ \boldsymbol{v_2}]$ denotes their row-wise concatenation. On the other hand, given $\boldsymbol{A_1}$ and $\boldsymbol{A_2}$ of appropriate sizes, the notation $[\boldsymbol{A_1}\ \boldsymbol{A_2}]$ denotes their column-wise concatenation. Finally, we denote as $\mathsf{Hw}(\boldsymbol{v})$ the Hamming weight of $\boldsymbol{v}$, i.e., the number of non null entries in $\boldsymbol{v}$, and we denote as $\mathcal{B}_y^x$ the set of all the length-x binary vectors having Hamming weight y. To conclude, we extend the vector notation to array objects by employing Greek letters as labels, as in, $\boldsymbol{\alpha}$, and a 0-based indexing, as in $\boldsymbol{\alpha}_{|i}$, with $0 \leq i < |\boldsymbol{\alpha}|$.

2.1 Code-based cryptography

Code-based cryptography relies on computationally hard problems involving objects from the theory of error-correcting codes. A binary linear code, denoted as $\boldsymbol{\mathcal{C}}$, is a linear mapping from the vector space $\boldsymbol{F}_2^k$ into the vector space $\boldsymbol{F}_2^n$, with $n>k$, used to encode a k-bit *message*, represented as the vector $\boldsymbol{m}\in\boldsymbol{F}_2^k$, into an n-bit *codeword*, represented as the vector $\boldsymbol{c}\in\boldsymbol{\mathcal{C}} \subset \boldsymbol{F}_2^n$. The encoding adds $r=n-k$ redundant bits to the message $\boldsymbol{m}$ through a linear transformation described by a generator matrix $\boldsymbol{G} \in \boldsymbol{F}_2^{k\times n}$, that is such that $\boldsymbol{\mathcal{C}} = \{\boldsymbol{c} \in \boldsymbol{F}_2^n : \boldsymbol{c} = \boldsymbol{G}^T\boldsymbol{m}\}$. Alternatively, the linear code $\boldsymbol{\mathcal{C}}$ can be characterized by a parity-check matrix $\boldsymbol{H} \in \boldsymbol{F}_2^{r\times n}$, that is such that $\boldsymbol{\mathcal{C}} = \{\boldsymbol{c} \in \boldsymbol{F}_2^n : \boldsymbol{H}\boldsymbol{c} = \boldsymbol{0}\}$, being $\boldsymbol{H}\boldsymbol{G}^T = \boldsymbol{0}_{r\times k}$. The definition of $\boldsymbol{H}$ provides a straightforward test for the integrity of a potentially corrupted codeword $\boldsymbol{y} = \boldsymbol{c} \oplus \boldsymbol{e}$, where $\boldsymbol{e} \in \boldsymbol{F}_2^n$ represents the error vector. Indeed, since $\boldsymbol{H}\boldsymbol{y} = \boldsymbol{H}\boldsymbol{c} \oplus \boldsymbol{H}\boldsymbol{e} = \boldsymbol{H}\boldsymbol{e}$, if the product $\boldsymbol{H}\boldsymbol{y}$ is not a null vector, the fact indicates the presence of an error. The vector $\boldsymbol{s} = \boldsymbol{H}\boldsymbol{e}$ is known as the

syndrome of the error $\boldsymbol{e}$. If no restriction is placed on the values of $\boldsymbol{e}$, many different values may satisfy $\boldsymbol{He} = \boldsymbol{s}$ for a given $\boldsymbol{H}$, $\boldsymbol{s}$ pair. In the following we will consider only error vectors with a Hamming weight equal to t, where t is chosen to be small enough that a single error vector satisfies the equation above. While, for appropriately chosen parity check matrices, it is possible to obtain $\boldsymbol{e}$ given $\boldsymbol{He} = \boldsymbol{s}$ and $\boldsymbol{H}$, a computation known as *syndrome decoding*, there is no known polynomial time algorithm solving the Syndrome Decoding Problem (SDP) for a random $\boldsymbol{H}$. The decision version of the SDP (i.e.,determining if a value for $\boldsymbol{e}$ exists) was shown to be NP-complete in [2], while its search version is NP-hard with a straightforward search to decision reduction algorithm (as all NP-complete problems are endowed with it) [26]. From now on, we consider the computational variant of the SDP, which is the one solved by ISD based cryptanalytic approaches, defined as follows:

Definition 1 (Computational Syndrome Decoding Problem (CSDP)). *Given a parity-check matrix* $\boldsymbol{H} \in \boldsymbol{F}_2^{r \times n}$*, a syndrome* $\boldsymbol{s} \in \boldsymbol{F}_2^r$*, and an integer* $t > 0$*, find the error vector* $\boldsymbol{e} \in \boldsymbol{F}_2^n$ *such that* $\mathsf{Hw}(\boldsymbol{e}) = t$ *and* $\boldsymbol{s} = \boldsymbol{He}$*.*

The complexity of the SDP paved the way to the development, in the mid-20th century, of two asymmetric cryptosystems: McEliece [27] and Niederreiter [28]. In these systems, the trapdoor function relies on an obfuscated version of a non-random, efficiently decodable parity-check matrix $\boldsymbol{H}'$. The obfuscation is achieved through the selection of a random secret, non-singular matrix $\boldsymbol{A}$, along with a random permutation matrix $\boldsymbol{P}$, obtaining an apparently random parity-check matrix $\boldsymbol{H} = \boldsymbol{AH}'\boldsymbol{P}$ to be used as a public key. In the Niederreiter cryptosystem, the matrix $\boldsymbol{H}$ enables any party wishing to transmit a message, encoded as an error vector $\boldsymbol{e}$ of Hamming weight t, to compute its syndrome $\boldsymbol{s} = \boldsymbol{He}$ and transmit it over a public channel. An adversary aiming to recover $\boldsymbol{e}$ must then find a solution to the SDP. In contrast, the legitimate receiver, knowing $\boldsymbol{A}$ and $\boldsymbol{P}$, can efficiently decode $\boldsymbol{e}$ from $\boldsymbol{s}$.

2.2 Information Set Decoding

Information Set Decoding is a technique to solve the CSDP, with a significant speedup with respect to a straightforward exhaustive search for the value of $\boldsymbol{e}$. Lee-Brickell variant [7] of the ISD, shown in Alg. 1, builds upon Prange's initial proposal [6]. The algorithm starts by guessing uniformly at random a set from $\binom{[n]}{k}$, known as *Information Set* and denoted as $\mathcal{I}$, representing a random guess on the k indexes of the elements of the error vector $\boldsymbol{e}$ containing only p bits set to 1. The remaining r integers, which belong to the complement of $\mathcal{I}$, $\overline{\mathcal{I}} = [n] \setminus \mathcal{I}$, represent the indexes of $\boldsymbol{e}$ for which the Hamming weight is equal to $t - p$.

The set $\overline{\mathcal{I}}$ (line 3) is employed to obtain a permutation matrix $\boldsymbol{P}$ (line 4), that is used to bring the matrix obtained packing the columns of $\boldsymbol{H}$ indexed by $\overline{\mathcal{I}}$, $\boldsymbol{H}_{|*,\overline{\mathcal{I}}}$, to the leftmost part of $\boldsymbol{H}$ (line 5), resulting in the permuted matrix $\widetilde{\boldsymbol{H}}$. Using the original equation of the syndrome $\boldsymbol{s} = \boldsymbol{He}$, the permutation produces the equivalent formulation $\boldsymbol{s} = (\boldsymbol{HP})\left(\boldsymbol{P}^T\boldsymbol{e}\right)$ that, by renaming the two components

Algorithm 1: Lee-Brickell Algorithm

Input : $\boldsymbol{H}$: a $r \times n$ binary parity-check matrix
$\boldsymbol{s}$: a r-bit long syndrome (column vector)
t: the weight of the error vector to be recovered
p: the weight of the first k bits of $\widehat{\boldsymbol{e}}$, $0 \leq p \leq t$

Output : $\boldsymbol{e}$: $n \times 1$ column vector s.t. $\boldsymbol{He} = \boldsymbol{s}$, $\mathsf{Hw}(\boldsymbol{e}) = t$

1 **repeat**
2 **repeat**
3 $\overline{\mathcal{I}} \xleftarrow{\$} \binom{[n]}{r}$
4 $\boldsymbol{P} \leftarrow \mathsf{GetPermutation}(\mathcal{I})$
5 $\widehat{\boldsymbol{H}} \leftarrow \boldsymbol{HP}$ // implies $\widehat{\boldsymbol{e}}=\boldsymbol{P}^T\boldsymbol{e}$
6 $\left[\widetilde{\boldsymbol{H}} \mid \tilde{\boldsymbol{s}}\right] \leftarrow \mathsf{GJE}\Big(\left[\widehat{\boldsymbol{H}} \mid \boldsymbol{s}\right]\Big)$
7 $[\boldsymbol{W}_{r\times r} \mid \boldsymbol{V}_{r\times k}] \leftarrow \widetilde{\boldsymbol{H}}$
8 **until** $\boldsymbol{W} = \boldsymbol{I}_r$
9
10 **foreach** $\mathcal{J} \in \binom{[k]}{p}$ **do**
11 $\boldsymbol{e_1} \leftarrow \tilde{\boldsymbol{s}} \oplus \left(\bigoplus_{j\in\mathcal{J}} \boldsymbol{V}_{|*,j}\right)$
12 **if** $\mathsf{Hw}(\boldsymbol{e_1}) = t - p$ **then**
13 $\boldsymbol{e_2} \leftarrow \boldsymbol{0}_{k\times 1}$
14 **foreach** $j \in \mathcal{J}$ **do**
15 $\boldsymbol{e_2}_{|j} \leftarrow 1$
16 **return** $\boldsymbol{P}\left[\boldsymbol{e_1}\, \boldsymbol{e_2}\right]$

enclosed in parenthesis, describes a system of linear equations $\boldsymbol{s}=\widehat{\boldsymbol{H}}\widehat{\boldsymbol{e}}$, with $\widehat{\boldsymbol{e}}=\boldsymbol{P}^T\boldsymbol{e}$. We denote the topmost $r\times 1$ portion of $\widehat{\boldsymbol{e}}$ as $\boldsymbol{e_1}$, and the bottommost $k\times 1$ portion as $\boldsymbol{e_2}$; that is, $\widehat{\boldsymbol{e}} = \left[\boldsymbol{e_1}\, \boldsymbol{e_2}\right]$. Lee-Brickell's assumption can be restated as $\mathsf{Hw}(\boldsymbol{e_1})=t-p$, and $\mathsf{Hw}(\boldsymbol{e_2})=p$.

The next step of Alg. 1 executes a *Gauss-Jordan Elimination (GJE)* (line 6) on the augmented matrix $\left[\widehat{\boldsymbol{H}}\ \boldsymbol{s}\right]$, verifying if the resulting matrix $\left[\widetilde{\boldsymbol{H}}\ \tilde{\boldsymbol{s}}\right]$ contains an identity submatrix in its leftmost $r\times r$ portion. If the condition is not met, the algorithm iterates the random guess of $\overline{\mathcal{I}}$. Conversely, if the condition is met, we can rewrite the equation of the syndrome as:

$$\tilde{\boldsymbol{s}} = [\boldsymbol{I}\ \boldsymbol{V}]\ \left[\boldsymbol{e_1}\, \boldsymbol{e_2}\right] = \boldsymbol{e_1} \oplus \boldsymbol{V}\boldsymbol{e_2} \leftrightarrow \boldsymbol{e_1} = \tilde{\boldsymbol{s}} \oplus \boldsymbol{V}\boldsymbol{e_2}$$

The last product represents the sum of the columns of $\boldsymbol{V}$ indexed by the p asserted bits of $\boldsymbol{e_2}$. For this reason, the algorithm performs an exhaustive search through all the sets belonging to $\binom{[k]}{p}$ (line 10). If, for one such set $\mathcal{J}$, the sum of $\tilde{\boldsymbol{s}}$ to the p columns of $\boldsymbol{V}$ indexed by the elements of $\mathcal{J}$ results in a vector of Hamming weight $t-p$, the algorithm reports a success.

Computational complexity of Algorithm 1. The time complexity of Alg. 1 is

$$\frac{\mathrm{T_{Iter}}}{\mathrm{Pr_{Succ}}} \approx \frac{\binom{n}{t}}{0.288\binom{r}{t-p}\binom{k}{p}}\Big(\mathrm{T_{GJE}}(n,r) + \mathrm{T_{ES}}(k,p,r)\Big) \tag{1}$$

Lee-Brickell's algorithm is indeed a Las Vegas algorithm, that is, a randomized algorithm that ensures the correct result (the value of $\boldsymbol{e}$) with a probabilistic runtime. To compute the expected runtime, we focus on determining the probability of success $\mathrm{Pr_{Succ}}$ for a single iteration of the outer loop (line 1). Notably, each iteration operates independently, randomly selecting a set $\overline{\mathcal{I}}$, making the success probability for one iteration unaffected by others. The probability of

success, $\mathrm{Pr}_{\mathrm{Succ}}$, is split into two components. The first component is the probability that the submatrix $\boldsymbol{H}_{|*,\overline{\mathcal{I}}}$ is invertible, which converges to ≈ 0.288 as the value of r increases [26]. The second component is the probability of guessing a set $\overline{\mathcal{I}}$ generating, among all the total number of length-n error vectors having Hamming weight t (i.e., $\binom{n}{t}$), the ones for which $\boldsymbol{e}_{|\overline{\mathcal{I}}}$ has Hamming weight $t-p$ (i.e., $\binom{r}{t}$). The main factor influencing the cost $\mathrm{T}_{\mathrm{Iter}}$ of a single iteration of Alg. 1 is $\mathrm{T}_{\mathrm{GJE}}$, representing the cost of executing the `GJE` algorithm, for which [26] reports a bit complexity of $\mathrm{T}_{\mathrm{GJE}}(n,r) = \mathcal{O}\left(\frac{nr^2}{2} + \frac{nr}{2} - \frac{r^3}{6} + r^2 + \frac{r}{6} - 1\right)$. The next component of the cost, $\mathrm{T}_{\mathrm{ES}}(k,p,r)$, pertains to the exhaustive enumeration of all the $\binom{k}{p}$ possible sets that belong to $\binom{[k]}{p}$. In [26], the authors report a bit complexity of $\mathcal{O}\left(\binom{k}{p}\left((2k-p)\log_2^2\binom{k}{p} + pr\right)\right)$, obtained by generating, at each iteration, the combinadics representation corresponding to the iteration number (cost $(2k-p)\log_2^2$), followed by p sums of length-r columns (cost pr).

2.3 Quantum computing

In quantum computing, the state of a computation is a column vector belonging to a finite-dimensional Hilbert space — that is, a complex vector space equipped with an inner product — spanned by 2^n orthonormal basis states. A state of such a system is denoted as $|\alpha\rangle = \sum_{i\in\{0,1\}^n} a_i\,|i\rangle$, where each $|i\rangle$ is a basis column vector labelled with an n bit string; $a_i \in \mathbb{C}$ represents its *amplitude*. The state $|\alpha\rangle$ is thus described as a linear combination, called *superposition*, of the basis states. Each digit of a basis state $|i\rangle$ is called *qubit*. When the state is observed, the superposition collapses to one of the basis states $|i\rangle$ with probability $\|a_i\|^2$.

The state of a quantum system is manipulated using quantum *operators*, described through unitary matrices in $\mathbb{C}^{2^n\times 2^n}$. A matrix $\boldsymbol{U}$ is called unitary if $\boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{U}^\dagger\boldsymbol{U} = \boldsymbol{I}$, in which $\boldsymbol{U}^\dagger$ denotes the adjoint of $\boldsymbol{U}$. The quantum state obtained applying the operator $\boldsymbol{U}$ to a quantum state $|\alpha\rangle$ is described through the matrix-vector multiplication $\boldsymbol{U}|\alpha\rangle$. If we partition the qubits into k disjoint subsets, and have k operators $\boldsymbol{U}_1,\ldots,\boldsymbol{U}_k$ each acting on a subset, the global operator acting on the quantum state is represented as $(\boldsymbol{U}_1\otimes\ldots\otimes\boldsymbol{U}_k)|\alpha\rangle$, in which $\otimes$ denotes the Kronecker products between matrices. On the other hand, the sequential application of the operators $\boldsymbol{U}_{t_1},\ldots,\boldsymbol{U}_{t_k}$ to a quantum state $|\alpha\rangle$ is expressed as $(\boldsymbol{U}_{t_k}\ldots\boldsymbol{U}_{t_1})|\alpha\rangle$. Figure 1 (top left) shows an example of operators applied to a state $|\alpha\rangle$.

As an alternative to the algebraic formulation, a sequence of operators can be visualized using a *quantum circuit*, in which each wire corresponds to a qubit, and operators are seen as *quantum gates* sequentially applied, from left to right, to qubits. When considering a circuit implementing $\boldsymbol{U}$, deriving the circuit for $\boldsymbol{U}^\dagger$ involves reversing the order of the gates used for $\boldsymbol{U}$, additionally replacing each gate with its adjoint. Such a circuit is commonly seen as an *uncomputation* of $\boldsymbol{U}$. Figure 1 (top center) shows the circuit corresponding to the same sequence of operations described in the previous example.

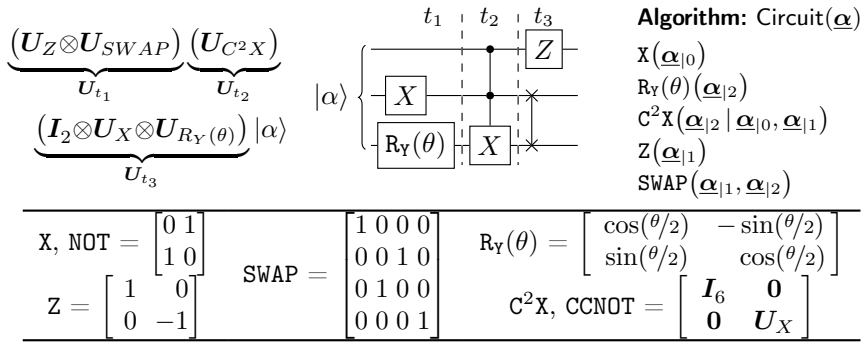

Fig. 1. (Top) Three different conventions to represent a sequence of operations applied to a quantum state: mathematical representation (left), gate-based circuit model (center), pseudocode (right). (Bottom) Unitary matrices associated to the gates in the top circuit. Note that $\boldsymbol{I}_k$ denotes the identity matrix of size $k \times k$.

Assuming the computational model of [25], in which the quantum unit is a memory peripheral whose operations are governed by a classical controller, we employ a pseudocode notation to accommodate both quantum and classical operations. To tighten the connection between classical and quantum computation, we refer to a set of qubits as *quantum register*, and we employ the same notation as the one for classical arrays, using an underline to remark the difference between their quantum behavior, as in $\underline{\boldsymbol{\alpha}}$ and $\underline{\boldsymbol{\alpha}}_{|i}$. Moreover, we use the function notation $\mathtt{G}(\underline{\boldsymbol{\alpha}})$ to express that the gate G operates on the quantum register $\underline{\boldsymbol{\alpha}}$. In contrast, classical routine names are expressed using camel case notation, as in, $\mathsf{GenerateCircuit}(\underline{\boldsymbol{\alpha}}, a, \boldsymbol{\alpha})$, denoting a classical routine taking as inputs a quantum register $\underline{\boldsymbol{\alpha}}$, a classical variable a and a classical vector $\boldsymbol{\alpha}$, using them to generate a sequence of quantum gates to be applied on $\underline{\boldsymbol{\alpha}}$. Figure 1 (top right) shows the sequence of pseudocode instructions corresponding to the same running example.

Finally, we say that a quantum gate — and, by extension, the classical routine generating it— has one or more control qubits to indicate that the gate acts on the target qubits only if all the control ones are 1. In the circuit model, the control qubits are denoted by a black circle, while in the pseudocode, they are separated from the target qubits by the symbol |. For a generic multi-controlled gate, we use the notation $\mathtt{C}^n\mathtt{G}$ to denote an arbitrary gate G having n control qubits and a single target qubit. When n=1 the exponent is omitted. Additionally, since the X gate is commonly interpreted as the quantum analogous of the NOT gate, it is common to refer to the CX and $\mathtt{C^2X}$ as CNOT and CCNOT respectively.

Reflection operators We use the notations $\boldsymbol{U}_{\mathrm{ref}(\alpha)}$ to represent the operator performing a reflection of a quantum state around the subspace defined by $|\alpha\rangle$. Its action is such that $\boldsymbol{U}_{\mathrm{ref}(\alpha)}|\alpha\rangle = |\alpha\rangle$, while $\boldsymbol{U}_{\mathrm{ref}(\alpha)}|\alpha^{\perp}\rangle = -|\alpha\rangle$, in which $|\alpha^{\perp}\rangle$ is any state orthogonal to $|\alpha\rangle$. Analogously, $\boldsymbol{U}_{\mathrm{ref}^{\perp}(\alpha)}$ is the operator performing a reflection around the subspace spanned by $|\alpha^{\perp}\rangle$. We define as $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ the

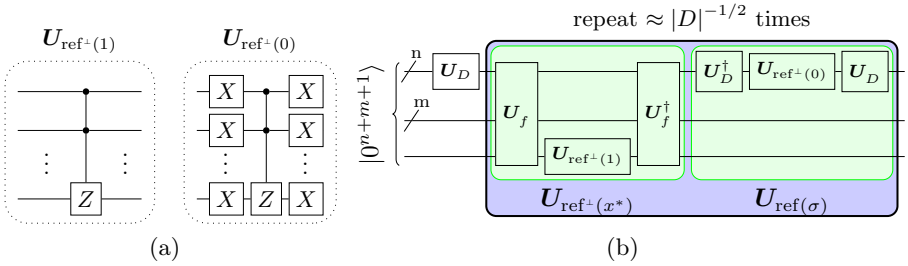

Fig. 2. (a) Circuit for the $\boldsymbol{U}_{\text{ref}^\perp(1)}$ and $\boldsymbol{U}_{\text{ref}^\perp(0)}$ reflection operators. (b) Circuit for Grover's framework in terms of the input preparation $\boldsymbol{U}_D$, and the reflections $\boldsymbol{U}_{\text{ref}^\perp(x^*)}$ and $\boldsymbol{U}_{\text{ref}(\sigma)}$. The reflections can be decomposed in terms of the simpler reflections $\boldsymbol{U}_{\text{ref}^\perp(1)}$ and $\boldsymbol{U}_{\text{ref}^\perp(0)}$, and the unitary $\boldsymbol{U}_f$, corresponding to the quantum circuit implementation of the function f, that stores in the last qubit the result of the function evaluation.

operator inverting the sign of the amplitude associated to the basis state labeled as the all-one's bitstring, while leaving all the other amplitudes unchanged. The operator corresponds to a $\texttt{C}^n\texttt{Z}$ gate involving all the n qubits describing the state $|1^n\rangle$. Similarly, $\boldsymbol{U}_{\text{ref}^\perp(0)}$ inverts the sign of the amplitude associated to the basis state labeled as the all-zero's bitstring. The operator can be implemented using a layer of `X` gates on each of the n qubits describing the basis state $|0^n\rangle$, followed by a $\texttt{C}^n\texttt{Z}$ gate, followed by another layer of `X` gates. Figure 2a shows both.

2.4 Grover's framework

Grover's algorithm [24] can be phrased in terms of a quantum framework searching for the unique value x^* for which the Boolean function $f{:}D{\mapsto}\{0,1\}$ evaluates to 1, with $D \subseteq \{0,1\}^n$ representing a subset of all length-n bitstrings. The framework, whose circuit description is given in Fig. 2b, relies on three operators.

1) *Input preparation* ($\boldsymbol{U}_D$). It corresponds to a quantum circuit preparing a uniform superposition of all the basis states labeled as the bitstrings belonging to the function domain D; that is, $|\sigma\rangle = \sum_{i \in D} \frac{1}{\sqrt{|D|}} |i\rangle$.
2) *Oracle* ($\boldsymbol{U}_{\text{ref}^\perp(x^*)}$). It is a reflection operator that changes the sign of the basis state labeled as x^*. This operator is typically stated in terms of an auxiliary quantum operator $\boldsymbol{U}_f$, corresponding to a quantum implementation of the Boolean function f. For all the non-trivial functions, the circuit requires a number m of auxiliary qubits, plus an additional single qubit that will store the result of the evaluation of $f(x)$. The $\boldsymbol{U}_f$ operator is then followed by the application of the reflection $\boldsymbol{U}_{\text{ref}^\perp(1)}$ operator on the previous qubit — that is, a `Z` gate — that, by inverting the sign of the basis state containing a 1 in the output qubit, achieves the wanted reflection $\boldsymbol{U}_{\text{ref}^\perp(x^*)}$. Finally, following a compute-uncompute pattern, the operator $\boldsymbol{U}_f^\dagger$ restores all the qubit labels to their previous states, with the only difference being in their amplitudes.

Table 1. Overview of the main quantum registers used in our circuit.

Quantum register	Semantics of the content
$\underline{\boldsymbol{\delta}}$	Encodes the superposition of all the possible $\overline{\mathcal{I}}$
$\underline{\boldsymbol{\eta}}$	Encodes the parity-check matrix through the various stages ($\boldsymbol{H}$, $\widehat{\boldsymbol{H}}$, $\widetilde{\boldsymbol{H}}$)
$\underline{\boldsymbol{\sigma}}$	Encodes the syndrome through the various stages ($\boldsymbol{s}$, $\tilde{\boldsymbol{s}}$)
$\underline{\boldsymbol{\alpha}}$	Encodes $\|11\rangle$ iff both conditions (C0) and (C1) are true

3) *Diffusion* ($\boldsymbol{U}_{\mathrm{ref}(\sigma)}$). It is a reflection around the superposition state prepared by the $\boldsymbol{U}_D$ circuit, that can be decomposed as $\boldsymbol{U}_{\mathrm{ref}(\sigma)} = \boldsymbol{U}_D \boldsymbol{U}_{\mathrm{ref}^{\perp}(0)} \boldsymbol{U}_D^{\dagger}$.

By repeating steps 2) and 3) $\approx \frac{1}{\sqrt{|D|}}$, the probability of observing the basis state $|x^*\rangle$ is close to 1. In [29], the authors showed that the optimal number of iterations is $\approx \frac{0.58}{\sqrt{|D|}}$, leading to a probability of observing $|x^*\rangle$ close to 0.84.

3 Lee-Brickell circuit to Speed-up ISD Iterations

To accelerate the Lee-Brickell ISD variant, we frame it as a search procedure to find the unique solution x^* to a Boolean function $f : \mathcal{B}_r^n \mapsto \{0,1\}$, where the domain $\mathcal{B}_r^n$ is the set of all binary vectors of length n and Hamming weight r, representing all the possible choices for the Information Set complement $\overline{\mathcal{I}}$. Indeed, given a binary vector $\boldsymbol{x} \in \mathcal{B}_r^n$, the set $\left\{ i \;\middle|\; \boldsymbol{v}_{|i} = 1 \right\}$ is an admissible value for the set $\overline{\mathcal{I}}$. In line with the assumptions of Lee-Brickell's algorithm (see Sec. 2.2), the function f evaluates to 1 if and only if both the following conditions are met:

(C0) the corresponding set $\overline{\mathcal{I}} \in \binom{[n]}{r} = \left\{ i \;\middle|\; \boldsymbol{x}_{|i} \right\}$ selects a non-singular submatrix $\boldsymbol{H}_{|*,\overline{\mathcal{I}}}$ (line 8 of Alg. 1);

(C1) the sum of a selection of p columns of $\boldsymbol{V}$ and the vector $\tilde{\boldsymbol{s}}$, obtained after the `GJE` procedure, results in a vector having Hamming weight equal to $t-p$ (line 12 of Alg. 1).

Given such formulation, employing Grover's framework to speed in the Lee-Brickell algorithm requires a circuit generating a uniform superposition of all the basis states labeled as the bitstrings of $\mathcal{B}_r^n$, and another one implementing $\boldsymbol{U}_f$. In the following, we detail the design of the required subcircuits, mapping them to the operators introduced in Sec. 2.4 and highlighting the relevant differences when needed. Table 1 provides a summary of the quantum registers we employ and the semantics of their contents.

3.1 Input circuit: superposition of all $\overline{\mathcal{I}}$ on $\underline{\boldsymbol{\eta}}$

The $\boldsymbol{U}_D$ operator in Grover's algorithm creates, in our case, a uniform superposition of all the basis states labeled by bitstrings within the domain $\mathcal{B}_r^n$. This

superposition corresponds to the Dicke state $|D_r^n\rangle$, defined as the equal superposition of all n-qubit basis states labeled as bitstrings of Hamming weight r; that is, $|D_r^n\rangle = \binom{n}{r}^{-\frac{1}{2}} \sum_i |i\rangle$, with $i \in \mathcal{B}_r^n$.

The first deterministic quantum circuit to generate such a state was presented in [30]. The Dicke state is prepared with an inductive approach, building a Dicke state starting from one with one less qubit involved, with a dedicated procedure depending on whether the qubit being added is equal to 1 or 0. The authors of [31] reduced the gate count metric from [30], obtaining a circuit consisting of k `X` gates, $5nk-5k^2-2n$ `CX` gates, and $4nk-4k^2-2n+1$ $\mathtt{R_Y}$ gates. The whole circuit only requires n qubits to store the Dicke state. The complexity of the circuit by [30], was further analyzed in [20] obtaining a tighter upper bound on its depth equal to $\leq \frac{27nk-12n-27k^2+3}{k-2} \in \mathcal{O}(n)$.

3.2 A Grover oracle circuit for Lee-Brickell's ISD

We now describe the oracle circuit realizing Lee-Brickell's approach from Alg. 1, rewritten in terms a Boolean function f. Our design for the oracle operator $\boldsymbol{U}_{\mathrm{ref}^{\perp}(x^*)}$ diverges from the standard formulation of Grover's framework given in Sec. 2.4 in two ways. First of all, our reflection operator $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ does not act on a single qubit, but on a pair of qubits belonging to a register denoted as $\underline{\boldsymbol{\alpha}}$. The first qubit, $\underline{\boldsymbol{\alpha}}_{|0}$, will be put in state 1 by the subcircuits composing the oracle if and only if condition (C0) is true; the second one, $\underline{\boldsymbol{\alpha}}_{|1}$, will instead be put in state 1 if and only if condition (C1) is true. As a consequence, as explained in Sec. 2.3, the circuit implementation in terms of gates for $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ corresponds to a `CZ` gate involving the two qubits of $\underline{\boldsymbol{\alpha}}$.

Secondly, our oracle operator $\boldsymbol{U}_{\mathrm{ref}^{\perp}(x^*)}$ diverges from the standard compute-rotate-uncompute pattern shown in Sec. 2.4, and instead is implemented by splitting the single $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ subcircuit into a sequence of repeated $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ operator applications. In the following, describe all the subcircuits composing our proposal for the oracle implementation. The complexity metrics related to all the oracle subcircuits are reported in Table 2.

Basis encoding: $\boldsymbol{H}$ into $\underline{\boldsymbol{\eta}}$, $\boldsymbol{s}$ into $\underline{\boldsymbol{\sigma}}$ The first step of the oracle circuit involves the encoding of the parity-check matrix $\boldsymbol{H} \in \boldsymbol{F}_2^{r \times n}$ and the syndrome vector $\boldsymbol{s} \in \boldsymbol{F}_2^r$ within the circuit. This subcircuit maps them onto the qubits of two quantum registers, denoted as $\underline{\boldsymbol{\eta}}$ and $\underline{\boldsymbol{\sigma}}$, respectively. It does so by applying an `X` gate on the corresponding qubit whenever the value in the matrix or vector is equal to 1. This encoding technique, known as *basis encoding*, requires a number of qubits equal to the original data size, resulting in a total of $rn + r$ qubits. Given that both $\boldsymbol{H}$ and $\boldsymbol{s}$ are random looking, it is reasonable to anticipate that, on average, approximately half of them will be equal to 1. Consequently, this subcircuit needs a number of `X` gates approximately equal to $(r + rn)/2$. As all these gates can be applied simultaneously, the depth of this stage is 1.

Columns permutation: from $\boldsymbol{H}$ to $\widehat{\boldsymbol{H}}$ The second subcircuit composing the oracle uses the qubits of $\underline{\boldsymbol{\delta}}$, encoding a superposition of all the binary strings representing the possible sets $\overline{\mathcal{I}}$ The overall idea behind this subcircuit is to bind each of the n qubits of $\underline{\boldsymbol{\delta}}$ to a group of r qubits in $\underline{\boldsymbol{\eta}}$ encoding a single column of $\boldsymbol{H}$. The value encoded in the qubits of $\underline{\boldsymbol{\delta}}$ will represents a binary choice on where to put the qubits tied to them. Specifically, whenever $\underline{\boldsymbol{\delta}}_{|i}$ is equal to 1, the r qubits $\underline{\boldsymbol{\eta}}_{|*,i}$ encoding the column $\boldsymbol{H}_{|*,i}$ are brought in the positions of $\underline{\boldsymbol{\eta}}$ encoding the leftmost $r \times r$ submatrix of $\boldsymbol{H}$.

To compute this operation, we use the quantum sorting network, initially proposed in [17] and later expanded in [21]. Its design relies on a quantum adaptation of the classical bitonic sorting network [32], and operates sorting the n qubits labeling a quantum state. As per its classical counterpart, the quantum sorting network employs a fixed number of compare-and-swap units. The comparator required inside the compare-and-swap unit checks if the values of two qubits are not sorted in e.g., increasing order. The authors of [21] show a quantum implementation for this component using 1 `CX` gate, 2 `X` gates, and one auxiliary qubit, that is set to 1 if the label of the first qubit is smaller than the one of the second. The result qubit is then used as the control qubit of a `CSWAP` gate having as target the pair of qubits previously being compared, that will be therefore swapped if and only if they are in the wrong order. The overall network requires $(n-1)\log_2(n)(\log_2(n)-1)$ comparators and the same amount of ancillary qubits, and it has a fixed depth of $\log_2(n)(\log_2(n)+1)$. Such a circuit can be used to reorder the qubits of $\underline{\boldsymbol{\eta}}$ in such a way that this register will encode, after its computation, the basis encoding of $\widehat{\boldsymbol{H}}$. To do so, while performing pairwise comparisons and swaps between the qubits of $\underline{\boldsymbol{\delta}}$, we additionally performs a set of r `CSWAP` gates, each one having as control the qubit storing the result of the comparison, and as targets the r pair of qubits of $\underline{\boldsymbol{\eta}}$ encoding the two matrix columns of $\boldsymbol{V}$. Since the `CSWAP` can be interleaved between layers, the overall depth for this circuit is $\log_2(n)(\log_2(n)+1)+r$. At the end of this stage, we obtain in $\underline{\boldsymbol{\eta}}$ the basis states encoding all the possible values of $\widehat{\boldsymbol{H}}$.

`Qgje`: from $\left[\widehat{\boldsymbol{H}}, \boldsymbol{s}\right]$ to $\left[\widetilde{\boldsymbol{H}}, \tilde{\boldsymbol{s}}\right]$; Check (C0) This stage performs a *Quantum Gauss Jordan Elimination (QGJE)* on the qubits of $\underline{\boldsymbol{\eta}}$ encoding, in superposition, the values of all the possible $\widehat{\boldsymbol{H}}$. To the best of our knowledge, the most efficient circuit in terms of depth, number of gates and qubits for a QGJE is shown in [21]. The plain version of the proposal is based on a quantum circuit first presented in [17], which relies on the classical generation procedure described in Alg. 2.

The QGJE generation algorithm, that relies on two auxiliary quantum registers denoted $\underline{\boldsymbol{\beta}}$ and $\underline{\boldsymbol{\gamma}}$, performs r distinct iterations. Each iteration, which we denote as x-iteration, focuses on the qubit $\underline{\boldsymbol{\eta}}_{|x,x}$, corresponding to the candidate pivot element $\widehat{\boldsymbol{H}}_{|x,x}$, and is split into two distinct phases. Phase 1 checks if the value of the qubit encoding the pivot is 0 and, in that case, sets it to 1. Therefore, the loop starting at line 4 iterates on all the qubits encoding the rows of $\widehat{\boldsymbol{H}}$ below

Algorithm 2: QGJE — classical generation procedure

Data : $\underline{\boldsymbol{\eta}}$: Encodes $\widehat{\boldsymbol{H}}$ at the start and $\widetilde{\boldsymbol{H}}$ at the end of the algorithm
$\underline{\boldsymbol{\sigma}}$: Encode $\boldsymbol{s}$ at the start and $\tilde{\boldsymbol{s}}$ at the end of the algorithm
$\underline{\boldsymbol{\beta}}$: Auxiliary quantum register initialized to $|\bar{0}\rangle$
$\underline{\boldsymbol{\gamma}}$: Auxiliary quantum register initialized to $|\bar{0}\rangle$

```
1  b ← 0      c ← 0
2  for x to r−1 do
       // Phase 1
3      if x ≠ r − 1 then
4          for i ← x + 1 to r−1 do
5              IsEqual(η_{|x,x}, 0, β_{|b})
6              for j ← 0 to r−1 do
7                  Add(η_{|i,j}, η_{|x,j}, | β_{|b})
8              Add(σ_{|i}, σ_{|x} | β_{|b})
9              b ← b + 1
10
       // Phase 2
11     for i ← 0 to r−1 do
12         if i ≠ x then
13             IsEqual(γ_{|c}, 1, η_{|i,x})
14             for j ← 0 to r−1 do
15                 Add(η_{|x,j}, η_{|i,j} | γ_{|c})
16             Add(η_{|x}, η_{|i} | γ_{|c})
17             c ← c + 1
```

the pivot row. At each iteration, the `IsEqual` function corresponds to a quantum circuit to check if the pivot encoded in $\underline{\boldsymbol{\eta}}_{|x,x}$ is equal to 0 and, in that case, set an auxiliary qubit belonging to register $\underline{\boldsymbol{\beta}}$ to 1 (line 5). Then, the same qubit of $\underline{\boldsymbol{\beta}}$ is used to control the addition of the qubits encoding the row under analysis to the ones encoding the pivot row through the `Add` procedure (line 7), which stores the result in the qubits encoding the pivot row. The same operation is performed on the qubits of $\underline{\boldsymbol{\sigma}}$ encoding the value of $\tilde{\boldsymbol{s}}$ (line 8).

The goal of phase 2 is to put all the qubits encoding the elements of $\widetilde{\boldsymbol{H}}$ below and under the pivot to 0. For this reason, at line 11 we iterate over all such elements. At each iteration, the `IsEqual` function (line 13) corresponds to a circuit to check if the element under analysis is 1 and, if this is the case, set the auxiliary qubit belonging to the quantum register $\underline{\boldsymbol{\gamma}}$ to 1. This qubit is then used to control the additions of the qubits encoding the pivot row to the ones encoding the row under analysis through the `Add` function (line 15). The same operation is performed on the qubits of $\underline{\boldsymbol{\sigma}}$ encoding the value of $\tilde{\boldsymbol{s}}$ (line 16).

In terms of standard quantum gates, the quantum circuit generated by the two functions `Add` and `IsEqual` is simplified by the fact that we are working in $\boldsymbol{F}_2$. The `IsEqual`$(\underline{\boldsymbol{\gamma}}_{|c}, 1, \underline{\boldsymbol{\eta}}_{|i,x})$ function can be implemented through a simple $\texttt{CX}\left(\underline{\boldsymbol{\gamma}}_{|c} \,|\, \underline{\boldsymbol{\eta}}_{|i,x}\right)$ gate. Similarly, the `IsEqual`$(\underline{\boldsymbol{\eta}}_{|x,x}, 0, \underline{\boldsymbol{\beta}}_{|b})$ circuit can be implemented through the sequence of gates $\texttt{X}\left(\underline{\boldsymbol{\eta}}_{|x,x}\right)$, $\texttt{CX}\left(\underline{\boldsymbol{\beta}}_{|b} \,|\, \underline{\boldsymbol{\eta}}_{|x,x}\right)$, and $\texttt{X}\left(\underline{\boldsymbol{\eta}}_{|x,x}\right)$. On the other hand, the circuit generated by the uncontrolled version of the `Add` procedure can be implemented as a `CX` gate having as control qubit the first argument of the function, and as target the second argument of the function. The controlled version of the `Add` requires therefore an additional control qubit, corresponding to the qubit of $\underline{\boldsymbol{\beta}}$ in phase 1 and $\underline{\boldsymbol{\gamma}}$ in phase 2. This plain version of the QGJE algorithm was later improved in [21], in which the authors reduced both

Algorithm 3: Exhaustive Search — classical generation procedure

Data : r: Number of rows of $\widetilde{\boldsymbol{H}}$ $\underline{\boldsymbol{\eta}}$: Encodes $\widetilde{\boldsymbol{H}}$
$\underline{\boldsymbol{\sigma}}$: Encodes $\tilde{\boldsymbol{s}}$
$\underline{\boldsymbol{\alpha}}$: Auxiliary, having $\underline{\boldsymbol{\alpha}}_{|0}$=1 if (C0) is satisfied, and $\underline{\boldsymbol{\alpha}}_{|0}$=0 before this procedure starts

1 $\mathcal{J}_p \leftarrow \emptyset$
2 **foreach** $i \in \binom{k}{p}$ **do**
3 $\quad \mathcal{J}_c \leftarrow$ `NextComb`(i)
4 $\quad \mathcal{D}_l \leftarrow \mathcal{J}_p \setminus \mathcal{J}_c$
5 $\quad \mathcal{D}_r \leftarrow \mathcal{J}_c \setminus \mathcal{J}_p$
6 $\quad$ **foreach** $j \in \mathcal{D}_l$ **do**
7 $\quad\quad$ `Sub`$(\underline{\boldsymbol{\eta}}_{|*,j+r}, \underline{\boldsymbol{\sigma}})$
8
9 $\quad$ **foreach** $j \in \mathcal{D}_r$ **do**
10 $\quad\quad$ `Add`$(\underline{\boldsymbol{\eta}}_{|*,:j+r}, \underline{\boldsymbol{\sigma}})$
11 `IsHwEqual`$(\underline{\boldsymbol{\sigma}}, t-p, \underline{\boldsymbol{\alpha}}_{|1})$
12 `Reflection`$(\underline{\boldsymbol{\alpha}}_{|0}, \underline{\boldsymbol{\alpha}}_{|1})$
13 `IsHwEqual`$^{\dagger}(\underline{\boldsymbol{\sigma}}, t-p, \underline{\boldsymbol{\alpha}}_{|1})$
14 $\mathcal{J}_p \leftarrow \mathcal{J}_c$

the overall depth of the circuit and the number of additional qubits by carefully avoiding the generation of operations on qubits not significant for the final result, and by rearranging gates across each iteration. Additionally, the authors also remove all the QGJE operations on the rightmost portion of the register $\underline{\boldsymbol{\eta}}$, since they only need the leftmost portion of the register. This optimization cannot be applied to our algorithm, which needs instead the full matrix $\widetilde{\boldsymbol{H}}$. Adapting their proposal to our case, even if the number of $\mathtt{C}^2\mathtt{X}$ gates increases by a factor of $\mathcal{O}(r^2k)$, the overall depth only increases by a factor of k, as shown in Tab. 2.

At the end of the QGJE stage, we obtain the basis encoding of $\boldsymbol{I}_r$ on the $r \times r$ qubits within $\underline{\boldsymbol{\eta}}$ if and only if the quantum register $\underline{\boldsymbol{\delta}}$ contains the encoding of a set $\overline{\mathcal{I}}$ for which the matrix $\boldsymbol{H}_{|*,\overline{\mathcal{I}}}$ is invertible. This condition, which corresponds to condition (C0), can be verified by utilizing a multi-controlled gate $\mathtt{C}^r\mathtt{X}$, having as control qubits all the ones within $\underline{\boldsymbol{\eta}}$ that encode the leftmost diagonal of $\widetilde{\boldsymbol{H}}$, while employing the ancillary qubit $\underline{\boldsymbol{\alpha}}_{|0}$ as the target.

Exhaustive sum of $\binom{k}{p}$ columns of $\boldsymbol{V}$; Check (C1); apply $\boldsymbol{U}_{\mathbf{ref}^{\perp}(\mathbf{1})}$ The goal of this stage is to find the group of p columns placed in the rightmost $r \times k$ portion of $\widetilde{\boldsymbol{H}}$ — corresponding to the submatrix $\boldsymbol{V}$ — and encoded in $\underline{\boldsymbol{\eta}}$ satisfying condition (C1). Since p is a fixed value known at circuit generation time, the classical controller, fixing in advance a strategy to generate all the $\binom{k}{p}$ sets belonging to $\mathcal{B}_p^k$, can generate, for each of them, the quantum gates performing the sum of the columns of $\boldsymbol{V}$ indexed by the set and encoded in $\underline{\boldsymbol{\eta}}$ to the vector $\tilde{\boldsymbol{s}}$ encoded in $\underline{\boldsymbol{\sigma}}$.

The pseudocode for the circuit generation strategy is given in Alg. 3. Starting from the empty set $\mathcal{J}_p$, each iteration of the enumeration strategy first generates the next combination of integers through the `NextComb` function, storing the result in $\mathcal{J}_c$ (line 3), and then computes two additional sets, $\mathcal{D}_l = \mathcal{J}_p \setminus \mathcal{J}_c$ and $\mathcal{D}_r = \mathcal{J}_c \setminus \mathcal{J}_p$. Using them, the algorithm can reuse, in the current iteration, the sum of the columns computed in the previous iteration, and only performs

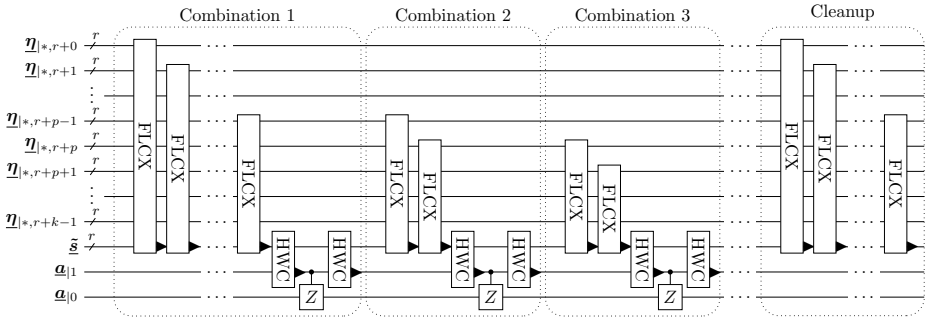

Fig. 3. Quantum circuit implementation of the exhaustive columns sum strategy of Lee-Brickell's variant (compare to Alg. 3).

the correct operations on the columns that changed across the two iterations. Hence, the algorithm first generates the quantum circuits to delete, from the sum computed in the previous iteration, the columns indexed by the integers that were present in the previous iteration but not in the current one (`Sub` procedure at line 7), and then generates the circuit to add the columns indexed by the integers that were not present in the previous iteration but are instead present in the current one (`Add` procedure at line 10). Note that, since we are working on $\boldsymbol{F}_2$, both the `Add` and the `Sub` procedure are equivalent to `XOR` operations. The next step of the algorithm, corresponding to the procedure `IsHwEqual` (line 11), generates a quantum circuit to compute the Hamming weight of the qubits of $\underline{\boldsymbol{\sigma}}$, setting the ancillary qubit $\underline{\boldsymbol{\alpha}}_{|1}$ to 1 if such a weight is equal to $t-p$. Later, the `Reflection` procedure (line 12) generates a circuit applying the reflection operator $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ to $\underline{\boldsymbol{\alpha}}$, whose qubit $\underline{\boldsymbol{\alpha}}_{|0}$ was set to 1 during the stage C) if and only if condition (C0) was satisfied. Finally, the `IsHwEqual`† procedure generates the quantum circuit corresponding to the adjoint of the circuit generated by the `IsHwEqual` procedure, restoring the qubit $\underline{\boldsymbol{\alpha}}_{|1}$ to 0 before the next iteration. This enumeration strategy, unlike what typically occurs in a standard adaptation of a Boolean function f to the Grover's framework, necessitates $\binom{k}{p}$ applications of the reflection operator $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ at each iteration of Grover's, as opposed to just one. Given the unicity of the solution, the strategy still ensures that only one among the $\binom{k}{p}$ combinations actually sets the qubit $\underline{\boldsymbol{\alpha}}_{|1}$ to 1, and therefore the sign of the amplitude is changed only once during a single Grover's iterations.

A naive enumeration strategy does not guarantee any pattern in two consecutive combinations of integers generated, resulting, in the worst case, in $|\mathcal{D}_l| = p$ and $|\mathcal{D}_r| = p$. That is, all the indices of the columns changed between two consecutive iterations of the enumeration strategy, and therefore we have to execute p times the procedure `Sub` and p times the procedure `Add`. Aiming to reduce both the gate count and the depth of this stage, we rely instead on a different enumeration strategy known as *revolving door algorithm* and detailed in [33, chap. 7.2.1.3]. This algorithm generates all the combinations of the subsets belonging to $\binom{[k]}{p}$ in such a way that, apart from the very first iteration, two consecutive subsets

Table 2. Complexity metrics for the subcircuits composing $\boldsymbol{U}_{\mathrm{ref}^{\perp}(x^*)}$, as a function of the linear code parameters (n, r), the wanted Hamming weight (t) and the Lee-Brickell parameter (p). Apart from D), all the other stages require an uncomputation at each Grover's iteration, and hence their values should be multiplied by 2.

Cost metric	A) Basis encoding	B) Columns Permutation	C) QGJE & Check (C0)	D) FLCX, HWCC & Check (C1)
`X`	$(r+rn)/2$	$2(n-1)\cdot\log_2(n)\cdot(\log_2(n)-1)$	$2(r-1)$	$\binom{k}{p}\cdot(4r-\log_2(r/t)-3)$
`CX`	0	$(n-1)\cdot\log_2(n)\cdot(\log_2(n)-1)$	$\frac{1}{2}r(r-1)$	$\binom{k}{p}\cdot\left(\frac{9}{2}r-5\log_2(r)-9\right)+2r-2p$
`C`^{2}`X`	0	0	$\frac{r}{6}\cdot(r-1)\cdot(9n+9-1)$	$\binom{k}{p}\cdot\left(3r-2\log_2(r)-3\right)$
`C`^{a}`X`$^{\diamond}$	0 0	1	$\binom{k}{p}$	
`CZ`	0	0	0	$\binom{k}{p}$
`CSWAP`	0	$(n-1)\cdot\log_2(n)\cdot(\log_2(n)-1)(r+1)$	0	0
Depth	1	$\log^2(n)+\log_2(n)+r-1$	$\frac{3}{2}r^2-\frac{1}{2}r+2$	$\binom{k}{p}\cdot\left(\log_2^2(r)+7\log_2(r)-2\right)$
Qubits	$r+rn$	$(n-1)\cdot\log_2(n)\cdot(\log_2(n)-1)$	$\frac{1}{2}r(r-1)$	$r-1$

$^{\diamond}$ a is equal to r in stage C) and $\log_2(r)$ in stage D).

differ by only a single element. During the first iteration, on the other hand, $\mathcal{D}_r$ has size p, while $\mathcal{D}_l$ is empty. The procedures `Add` and `Sub`, therefore, are called only once each for all the $\binom{k}{p}-1$ iterations following the first one.

Figure 3 shows the quantum circuit generated by using such a strategy. In the circuit, the quantum circuits generated by the `Sub` and `Add` procedures of Alg. 3 are both denoted as First to Last `CX` (`FLCX`). Both of them apply r `CX` between the qubits of $\underline{\boldsymbol{\eta}}$ and $\underline{\boldsymbol{\sigma}}$, using the former as controls and the latter as targets. The two `FLCX` applied at each iteration following the first one results in a constant depth of 2, since all the r `CX` gates composing a `FLCX` acts on distinct pair of qubits. The Hamming Weight Check (`HWC`) abstract gate, on the other hand, computes the Hamming weight of the qubits of $\underline{\boldsymbol{\alpha}}$, setting $\underline{\boldsymbol{\alpha}}_{|1}$ to 1 if it is equal to $t-p$. The quantum circuit to perform the Hamming weight count can be realized through a quantum implementation of a fast population count circuit, similar to the one used in [21], in which the authors report a logarithmic depth circuit requiring $r-1$ quantum adder, each one requiring an additional ancillary qubit for the carry. By using the adder of [34], the authors report a gate count of $5r-\log_2(r/t)-3$ `X`, $9r/2-5\log_2(r)-11$ `CX` and $3r-2\log_2(r)-3$ `C`^{2}`X`, with an overall depth of $\log_2^2(r)+7\log_2(r)-4$. Moreover, the additional component setting the qubit $\underline{\boldsymbol{\alpha}}_{|1}$ to 1 if the weight is equal to $t-p$ can be realized by first performing a basis encoding of the value of the binary complement of $t-p$ on the $\log_2(r)$ qubits storing the result of the Hamming weight computation, and then applying the `C`$^{\log_2(r)}$`X` gate having all of them as controls, and the qubit $\underline{\boldsymbol{\alpha}}_{|1}$ as target. Finally, the `CZ` gate performs the reflection $\boldsymbol{U}_{\mathrm{ref}^{\perp}(1)}$ using both the qubits of $\underline{\boldsymbol{\alpha}}$, whose qubit $\underline{\boldsymbol{\alpha}}_{|0}$ was set to 1 during stage C) if condition (C0) was true.

Table 3. Parameter sets for the code-based candidates advanced to round 4 of NIST standardization process [3].

Scheme	Security Level	Code parameters			Scheme	Security Level	Code parameters		
		n	k	t			n	k	t
BIKE (key)	L1	24,646	12,323	142	HQC	L1	35,338	17,669	132
	L3	49,318	24,659	206		L3	71,702	35,851	200
	L5	81,946	40,973	274		L5	115,274	57,637	262
BIKE (message)	L1	24,646	12,323	134	Classic McEliece	L1	3,488	2,720	64
	L3	49,318	24,659	199		L3	4,608	3,360	96
	L5	81,946	40,973	264		L5	6,688	5,024	128
						L5	6,960	5,413	119
						L5	8,192	6,528	128

We note that this stage, differently from all the others, does not require an uncomputation circuit to clean up the state of the registers before the next Grover's iteration. Indeed, after the last possible iteration of the enumeration strategy, the classical controller holds all the values of $\mathcal{J}_c$ that, `XOR`ed together, gives the result stored in σ. To restore this register to the state in which it was before the enumeration strategy, the controller can just delete, one by one, the p columns of $\boldsymbol{V}$ encoded in $\underline{\boldsymbol{\eta}}$ indexed by the elements of $\mathcal{J}_c$.

4 Experimental Evaluation

To evaluate the computational complexity of our solution with cryptographically relevant parameters, we focus on the CBC schemes that progressed to the fourth round (the ongoing one at the time of writing) of the NIST standardization process [3]: Classic McEliece [35], BIKE [36] and HQC [37]. According to NIST's original submission requirements [38], cryptographic schemes were classified into three security levels: L1, L3, and L5. These categories correspond to a computational effort comparable to or greater than that required for a quantum 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 displays the parameter sets proposed for all the three CBC proposals under analysis.

NIST's initial submission requirement [38] makes a conservative assumption of an equal weight for all quantum gates involved in a quantum circuit. However, recent literature advocates for converting gates operating on more than 3 qubits into ones operating on at most 3 qubits in order to allow a more robust assessment of the complexity metrics associated with a quantum circuit. Therefore, before delving into the analysis of relevant cryptographic schemes, we converted the multi-controlled `X` gates used at each Grover's iteration in both stages C) and D) to check for conditions (C0) and (C1), having a number of controls equal to r and $\log_2(r)$ respectively. While the $\mathtt{C}^r\mathtt{X}$ gate of stage C) is repeated once at each iteration of Grover's algorithm, the $\mathtt{C}^{\log_2(r)}\mathtt{X}$ gate of stage D) is instead

Table 4. Comparison of our full-quantum Lee-Brickell approach against the quantum Prange design of [18] and [21], and the hybrid Lee-Brickell design of [20]. G denotes the number of gates, D denotes the depth, W denotes the number of qubits, DW multiplies the depth by the width. All the values are expressed in base-2 logarithm.

Scheme	Sec. Level	Prange [18]			Prange [21]				Lee-Brickell [20]					This work				
		D	W	DW	G	D	W	DW	p	G	D	W	DW	p	G	D	W	DW
BIKE (key)	L1	113	28	141	108	93	29	123	3	161	159	15	174	1	105	90	29	119
	L3	148	30	178	142	127	31	158	3	228	225	16	241	1	140	123	31	154
	L5	184	32	215	178	162	33	195	3	295	291	17	308	1	175	158	33	191
BIKE (message)	L1	109	28	137	104	89	29	119	-	-	-	-	-	1	102	86	29	115
	L3	144	30	175	139	123	31	155	-	-	-	-	-	1	136	120	31	151
	L5	179	32	210	173	157	33	190	-	-	-	-	-	1	170	153	33	186
HQC	L1	110	29	139	104	89	30	119	-	-	-	-	-	1	102	86	30	116
	L3	146	31	177	140	125	32	157	-	-	-	-	-	1	138	121	32	153
	L5	179	33	212	173	157	34	190	-	-	-	-	-	1	171	153	34	186
Classic McEliece	L1	111	21	133	102	92	22	114	13	152	145	12	157	1	97	89	22	110
	L3	134	22	156	125	115	23	138	15	194	186	12	198	1	120	111	23	134
	L5	174	23	198	165	154	24	178	-	-	-	-	-	1	160	150	24	174
	L5	175	23	198	165	155	24	178	-	-	-	-	-	1	160	150	24	174
	L5	194	24	218	184	173	24	197	27	300	291	13	304	1	179	169	24	193

repeated $\binom{k}{p}$ times at each Grover's iteration. The logarithmic-depth translation of the multi-controlled `X` gate proposed in [21] proved to offer better results for our proposal. Such a translation requires a number of auxiliary qubits linear in the number of controls. Nevertheless, all of them are restored to their original value, and thus the overall number of qubits of the whole circuit is not affected, since it is still dominated by the rn qubits required to store $\boldsymbol{H}$.

Table 4 shows the assessment of the effort needed to attack the three code-based cryptosystems being evaluated in NIST's PQC standardization call. In the table, considering an equal impact of all the gates involved, we report the aggregate number of gates (G), the depth (D), the number of qubits (also known as width (W)) and the depth×width (DW) metric proposed in [25]. In the same table, we conduct a direct comparison between our proposal and the state-of-the-art works approaching the SDP using a quantum-accelerated attack based on Grover's framework. To the best of our knowledge, the first comprehensive design and implementation of a quantum circuit to accelerate the SDP problem through a quantum ISD approach was presented in [20]. This proposal, based on a quantum adaptation of Prange's variant of the ISD [6], was later refined in [21]. The work presented in [18] offers a similar adaptation of Prange's variant, and further sketches an algorithm for the Lee-Brickell's variant relying on an exhaustive search strategy similar to the one used in our proposal. The first difference between our strategy and the one employed in [18] is that the latter uses an additional accumulator register, which is increased by one, at each iteration of the enumeration, if the current sum of columns of $\boldsymbol{V}$ and $\tilde{\boldsymbol{s}}$ is equal to $t-p$.

The second and main difference is instead related to an unoptimized enumeration strategy in [18], leading to p applications at each Grover's iteration of the `Sub` and `Add` generation procedure of Alg. 3. Since $p \ll r$, the $2pr$ `CX` gates required for each set generated by the enumeration strategy to sum the p columns of $\boldsymbol{V}$ to $\underline{\boldsymbol{\sigma}}$ and then, after computing the resulting Hamming weight, delete them from $\underline{\boldsymbol{\sigma}}$, cannot be easily parallelized. Hence, the authors of [21] report, for such a strategy, a depth $\in \mathcal{O}(\binom{k}{p}2r)$. Since the proposal of [18] has already a depth for each Grover's iteration in the order of $\mathcal{O}(n^3 \log_2(n))$, the circuit performing the exhaustive strategy does not have a profound impact on the overall measures for $p \in \{1, 2\}$ with respect to the quantum Grover's proposal presented in the same work, while having an exponentially increasing worse complexity for $p > 2$.

Table 4 shows that our design improves the state-of-the-art in all the metrics, with a gain in the DW metric going from 2^4 to 2^{22} for all the Prange's based proposals. With respect to the Lee-Brickell's proposals made in [20], the gain in the depth and depth×width is still more pronounced. The substantially lower value in the number of qubits required by [20] is justified by the hybrid approach used in that work, that, in offloading only a portion of the Lee-Brickell's approach to the quantum unit, requires the basis encoding for only the matrix $\boldsymbol{V}$. Finally, we remark that our experimental results show that the value of p giving the best results in terms of all metrics is $p=1$, which is in contrast to the classical case, for which the optimal value was found to be $p=2$ in [26].

5 Concluding Remarks

Our work introduces a quantum circuit designed to accelerate the resolution of the syndrome decoding problem, specifically focusing on Lee-Brickell's variant of ISD within Grover's quantum framework. We provided a comprehensive breakdown of the circuit components, offering a detailed assessment of their computational complexities, considering factors such as width (number of qubits), depth, and depth×width. By targeting code-based candidates that advanced to the final stage of NIST's PQC standardization call, we compared the complexity metrics of our proposal with existing state-of-the-art circuits designed for the same problem.

Our results indicate substantial reductions in the computational effort required to attack these cryptosystems across key metrics. We highlight that our implementation of the quantum oracle, a crucial element in Grover's framework, diverges from traditional patterns, opting for a series of repeated rotations, closely mirroring the inner search procedure of Lee-Brickell's algorithm. This novel approach may be of independent interest.

A promising research path explored the adaptation of the quantum walk framework to the advanced ISD variants [22, 23], with theoretical enhancements with respect to Grover's framework. However, a concrete quantum circuit design for this approach is yet to be proposed.

References

[1] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*. IEEE Computer Society, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700. URL: https://doi.org/10.1109/SFCS.1994.365700.

[2] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. "On the Inherent Intractability of Certain Coding Problems (Corresp.)" In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 24.3 (1978), pp. 384–386. DOI: 10.1109/TIT.1978.1055873. URL: https://doi.org/10.1109/TIT.1978.1055873.

[3] Dustin Moody. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. NIST IR 8413. Gaithersburg, MD: National Institute of Standards and Technology, 2022, NIST IR 8413. DOI: 10.6028/NIST.IR.8413. URL: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf (visited on 07/06/2022).

[4] National Institute of Standards and Technology (NIST). "Announcing Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms". In: *Federal Register* 81.244 (Dec. 20, 2016), pp. 92787–92788. URL: https://federalregister.gov/a/2016-30615.

[5] National Institute of Standards and Technology (NIST). "Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process". In: (2022). URL: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.

[6] E. Prange. "The Use of Information Sets in Decoding Cyclic Codes". en. In: *IEEE Transactions on Information Theory* 8.5 (Sept. 1962), pp. 5–9. ISSN: 0018-9448. DOI: 10.1109/TIT.1962.1057777. URL: http://ieeexplore.ieee.org/document/1057777/ (visited on 09/08/2020).

[7] P. J. Lee and E. F. Brickell. "An Observation on the Security of McEliece's Public-Key Cryptosystem". en. In: *Advances in Cryptology — EUROCRYPT '88*. Ed. by D. Barstow et al. Red. by G. Goos and J. Hartmanis. Vol. 330. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 275–280. DOI: 10.1007/3-540-45961-8_25. URL: http://link.springer.com/10.1007/3-540-45961-8_25 (visited on 09/08/2020).

[8] Jeffrey S. Leon. "A Probabilistic Algorithm for Computing Minimum Weights of Large Error-Correcting Codes". In: *IEEE Transactions on Information Theory* 34.5 (1988), pp. 1354–1359. DOI: 10.1109/18.21270. URL: https://doi.org/10.1109/18.21270.

[9] Jacques Stern. "A Method for Finding Codewords of Small Weight". In: *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November 2-4, 1988, Proceedings*. Ed. by Gérard D. Cohen and Jacques Wolfmann. Vol. 388. Lecture Notes in Computer Science. Springer, 1988, pp. 106–113. DOI: 10.1007/BFb0019850. URL: https://doi.org/10.1007/BFb0019850.

[10] Ilya Dumer. "On Minimum Distance Decoding of Linear Codes". In: *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory*. Moscow, 1991, pp. 50–52.

[11] Matthieu Finiasz and Nicolas Sendrier. "Security Bounds for the Design of Code-Based Cryptosystems". In: *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Springer, 2009, pp. 88–105. DOI: 10.1007/978-3-642-10366-7_6. URL: https://doi.org/10.1007/978-3-642-10366-7_6.

[12] Alexander May, Alexander Meurer, and Enrico Thomae. "Decoding Random Linear Codes in $\tilde{\mathcal{O}}(2^{0.054n})$". In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. Lecture Notes in Computer Science. Springer, 2011, pp. 107–124. DOI: 10.1007/978-3-642-25385-0_6. URL: https://doi.org/10.1007/978-3-642-25385-0_6.

[13] Alexander May and Ilya Ozerov. "On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes". In: *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2015, pp. 203–228. DOI: 10.1007/978-3-662-46800-5_9. URL: https://doi.org/10.1007/978-3-662-46800-5_9.

[14] Leif Both and Alexander May. "Decoding Linear Codes with High Error Rate and Its Impact for LPN Security". In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*. Ed. by Tanja Lange and Rainer Steinwandt. Vol. 10786. Lecture Notes in Computer Science. Springer, 2018, pp. 25–46. DOI: 10.1007/978-3-319-79063-3_2. URL: https://doi.org/10.1007/978-3-319-79063-3_2.

[15] Anja Becker et al. "Decoding Random Binary Linear Codes in 2 n/20: How $1 + 1 = 0$ Improves Information Set Decoding". In: *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 520–536. DOI: 10.1007/978-3-642-29011-4_31. URL: https://doi.org/10.1007/978-3-642-29011-4_31.

[16] Daniel J. Bernstein. "Grover vs. McEliece". en. In: *Post-Quantum Cryptography*. Ed. by David Hutchison et al. Vol. 6061. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 73–80. DOI: 10.1007/978-3-642-12929-2_6.

URL: http://link.springer.com/10.1007/978-3-642-12929-2_6 (visited on 09/08/2020).

[17] Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. "A Complete Quantum Circuit to Solve the Information Set Decoding Problem". In: *IEEE International Conference on Quantum Computing and Engineering, QCE 2021, Broomfield, CO, USA, October 17-22, 2021*. Ed. by Hausi A. Müller et al. IEEE, 2021, pp. 366–377. DOI: 10.1109/QCE52317.2021.00056. URL: https://doi.org/10.1109/QCE52317.2021.00056.

[18] Andre Esser et al. "An Optimized Quantum Implementation of ISD on Scalable Quantum Resources". In: (2021). URL: https://eprint.iacr.org/2021/1608.

[19] Andre Esser et al. "Hybrid Decoding - Classical-Quantum Trade-Offs for Information Set Decoding". In: *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*. Ed. by Jung Hee Cheon and Thomas Johansson. Vol. 13512. Lecture Notes in Computer Science. Springer, 2022, pp. 3–23. DOI: 10.1007/978-3-031-17234-2_1. URL: https://doi.org/10.1007/978-3-031-17234-2_1.

[20] Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. "A Quantum Circuit to Speed-up the Cryptanalysis of Code-Based Cryptosystems". In: *Security and Privacy in Communication Networks - 17th EAI International Conference, SecureComm 2021, Virtual Event, September 6-9, 2021, Proceedings, Part II*. Ed. by Joaquín García-Alfaro et al. Vol. 399. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2021, pp. 458–474. DOI: 10.1007/978-3-030-90022-9_25. URL: https://doi.org/10.1007/978-3-030-90022-9_25.

[21] Simone Perriello, Alessandro Barenghi, and Gerardo Pelosi. "Improving the Efficiency of Quantum Circuits for Information Set Decoding". In: *ACM Transactions on Quantum Computing* (July 2023). ISSN: 2643-6809. DOI: 10.1145/3607256. URL: https://doi.org/10.1145/3607256.

[22] Ghazal Kachigar and Jean-Pierre Tillich. "Quantum Information Set Decoding Algorithms". In: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*. Ed. by Tanja Lange and Tsuyoshi Takagi. Vol. 10346. Lecture Notes in Computer Science. Springer, 2017, pp. 69–89. DOI: 10.1007/978-3-319-59879-6_5. URL: https://doi.org/10.1007/978-3-319-59879-6_5.

[23] Elena Kirshanova. "Improved Quantum Information Set Decoding". In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*. Ed. by Tanja Lange and Rainer Steinwandt. Vol. 10786. Lecture Notes in Computer Science. Springer, 2018, pp. 507–527. DOI: 10.1007/978-3-319-79063-3_24. URL: https://doi.org/10.1007/978-3-319-79063-3_24.

[24] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual {ACM} Symposium*

on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996. Ed. by Gary L. Miller. ACM, 1996, pp. 212–219. DOI: 10.1145/237814.237866. URL: https://doi.org/10.1145/237814.237866.

[25] Samuel Jaques and John M. Schanck. "Quantum Cryptanalysis in the RAM Model: Claw-finding Attacks on SIKE". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11692. Lecture Notes in Computer Science. Springer, 2019, pp. 32–61. DOI: 10.1007/978-3-030-26948-7_2. URL: https://doi.org/10.1007/978-3-030-26948-7_2.

[26] Marco Baldi et al. "A Finite Regime Analysis of Information Set Decoding Algorithms". In: *Algorithms* 12.10 (2019). ISSN: 1999-4893. DOI: 10.3390/a12100209. URL: https://www.mdpi.com/1999-4893/12/10/209.

[27] Robert J McEliece. "A Public-Key Cryptosystem Based On Algebraic Coding Theory". In: *The Deep Space Network progress report* (1978), pp. 114–116. URL: https://ntrs.nasa.gov/api/citations/19780016269/downloads/19780016269.pdf.

[28] Harald Niederreiter. "Knapsack-Type Cryptosystems and Algebraic Coding Theory". In: *Problems of Control and Information Theory* 15.2 (1986), pp. 157–166.

[29] Michel Boyer et al. "Tight Bounds on Quantum Searching". In: *Fortschritte der Physik: Progress of Physics* 46.4-5 (1998), pp. 493–505. DOI: 10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P. arXiv: quant-ph/9605034.

[30] Andreas Bärtschi and Stephan J. Eidenbenz. "Deterministic Preparation of Dicke States". In: *Fundamentals of Computation Theory - 22nd International Symposium, FCT 2019, Copenhagen, Denmark, August 12-14, 2019, Proceedings*. Ed. by Leszek Antoni Gasieniec, Jesper Jansson, and Christos Levcopoulos. Vol. 11651. Lecture Notes in Computer Science. Springer, 2019, pp. 126–139. DOI: 10.1007/978-3-030-25027-0_9. URL: https://doi.org/10.1007/978-3-030-25027-0_9.

[31] C. S. Mukherjee et al. "Preparing Dicke States on a Quantum Computer". In: *IEEE Transactions on Quantum Engineering* 1 (2020), pp. 1–17. DOI: 10.1109/TQE.2020.3041479.

[32] Kenneth E. Batcher. "Sorting Networks and Their Applications". In: *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1968 Spring Joint Computer Conference, Atlantic City, NJ, USA, 30 April - 2 May 1968*. Vol. 32. AFIPS Conference Proceedings. Thomson Book Company, Washington D.C., 1968, pp. 307–314. DOI: 10.1145/1468075.1468121. URL: https://doi.org/10.1145/1468075.1468121.

[33] Donald E Knuth. *The Art of Computer Programming, Volume IVa: Combinatorial Algorithms*. Addison-Wesley Professional, 2011.

[34] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. "Quantum Addition Circuits and Unbounded Fan-Out". In: *Quantum Information &*

Computation 10 (9&10 2010), pp. 872–890. DOI: 10.26421/QIC10.9-10-12. URL: https://doi.org/10.26421/QIC10.9-10-12.

[35] Daniel J. Bernstein et al. *Classic McEliece: Conservative Code-Based Cryptography*. 2020. URL: https://classic.mceliece.org/ (visited on 08/20/2023).

[36] Nicolas Aragon et al. *BIKE: Bit Flipping Key Encapsulation*. 2017. URL: https://bikesuite.org (visited on 08/20/2023).

[37] Carlos Aguilar Melchor et al. *Hamming Quasi-Cyclic (HQC)*. 2017. URL: https://pqc-hqc.org/ (visited on 08/20/2023).

[38] National Institute of Standards and Technology (NIST). *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. Dec. 2016. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf.