A Complete Quantum Circuit to Solve the Information Set Decoding Problem

Simone Perriello, Alessandro Barenghi and Gerardo Pelosi

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

***Abstract*—Providing strong security margins against cryptanalytic attackers equipped with quantum computers is a major research direction fostered by the US National Institute of Standards and Technology (NIST) Post-quantum Cryptography Standardization process. Among the viable candidates, code-based asymmetric cryptosystems are one of the prominent approaches. In this work, we propose the first fully detailed quantum circuit to compute the solution to the Information Set Decoding problem, the main cryptanalytic tool against such cryptosystems. We evaluate the cryptanalytic effort with our circuit design on actual parameters from cryptosystems admitted to the final stage of the NIST standardization process and compare it with the previous conservative asymptotic estimates. We show that the actual computational effort of our solution is smaller than the one estimated via asymptotics by a factor of 2^4. We also perform a comparison of our results with the quantum-computational effort of breaking the AES cipher, following the guidelines of the US NIST in evaluating the security of the ciphers. To do this, we translate our design on gates of the Clifford+T gate set only, one of the most promising candidate for fault-tolerant quantum computation, and report that the parameter choices for Classic McEliece and BIKE, two candidates admitted to the final round of the NIST standardization process provide an adequate security margin with respect to our ISD solution technique.**

***Index Terms*—Quantum cryptanalysis, post-quantum code-based cryptosystems, quantum circuit design**

I. Introduction

One of the most significant impact area of quantum computing is the ability to solve computationally hard problems underlying modern cryptography, such as factoring and discrete logarithms in cyclic groups [1]. As a consequence, a considerable amount of interest has been raised in the design and security evaluation of cryptographic primitives resistant to attacks supported by a quantum computer, also known as *post-quantum cryptosystems*. This interest is witnessed by the US National Institute of Standards and Technology (NIST) standardization process for post-quantum cryptosystems [2], and the EU ETSI working group on quantum-safe cryptography [3]. A consolidated practice to gauge the security of cryptosystems is to quantify the amount of computation required to break them through a given cryptanalytic technique, and tune the cryptosystem parameters so that such a computation is practically unfeasible [4]–[8]. This in turn requires the design and evaluation of efficient implementations of the said cryptanalytic techniques, a research direction which has attracted significant interest [9]–[12]. In particular, quantitative evaluations of the computational effort required to break factoring-based cryptosystems [13], [14] and discrete logarithm based cryptosystems based on prime cyclic groups [15], elliptic curve over prime fields [16], [17] and elliptic curves over binary fields [18], have proven that a quantum computer equipped with a few thousands reliable qubits will be able to break the aforementioned public-key cryptosystems (configured with most of the key-lengths currently in use) in little time.

Our purpose in this work is to provide the first detailed implementation of the so-called Information Set Decoding (ISD) technique, which is employed to cryptanalyze all the current code-based cryptosystems selected by NIST as the finalists in its standardization process. The current state of the art in the analysis of quantum ISDs sees a number of works which either provide asymptotic bounds on the complexity of ISDs (without explicit quantum circuits) [19]–[21] or perform a finite-regime analysis estimating the quantum speedup as a square root of the effort of the classic counterpart [22]. We note that the proposals in [20], [21] are relative to algorithms which require an exponential amount of qubits in the size of the input problem. In this work we focus on polynomial-space quantum ISD solvers. The quantum circuit proposed in our work allows us to provide a first concrete data point on the expected amount of qubits and quantum circuit size and depth that are required to break a code-based cryptosystem built on the binary *syndrome decoding* problem [23]. This falls in line also with the analyses on quantum circuits to solve their hard problems on other NIST competition finalists, i.e., the lattice sieving approach to solve the shortest vector problem [24] and the supersingular isogeny Diffie-Hellman problem [25]. Our results show that the parametrizations of code based cryptosystems provide more than adequate security margins when compared to the quantum effort of breaking AES, albeit we improve on the asymptotic estimates by $\approx 2^4\times$.

The paper is organized as follows: Section II provides a description code based cryptosystems and the ISD problem, a summary of Grover's algorithmic framework and background on sorting networks, which are employed as a building block in our quantum ISD approach. Section III details our quantum ISD circuit and evaluates its complexity in terms of gate count and circuit depth. Section IV reports quantitative results on the computational effort required to solve the ISD problem for Classic McEliece and BIKE, the finalist and one of the alternate cryptosystems selected by NIST in its final evaluation round, and Section V reports our conclusions.

II. Background

In this section, we recall the basic concepts of code-based asymmetric cryptosystems, the Information Set Decoding (ISD) technique, provide a summary on the Grover algorithmic framework, and recall the notions on the classical comparator networks used in our quantum circuit design.

A. Code based cryptosystems

A binary linear error correcting code $\mathcal{C}$ is a linear subspace of n-element (column) vectors with components over the finite field $\mathbb{Z}_2$. The purpose of a code is to encode a k-bit long information word $\boldsymbol{m}\in\mathbb{F}_2^k$ into a codeword $\boldsymbol{c}\in\mathcal{C}\subset\mathbb{F}_2^n$ by adding to it $r=n-k$ redundant bits. Codes are designed so that codewords $\boldsymbol{c} \in \mathcal{C}$ differ by at least d elements, a quantity known as *code distance*. This property allows correcting up to $\lfloor \frac{d}{2} \rfloor$ errors exploiting the fact that such a number of changes on a codeword cannot yield a result closer to a different codeword.

A binary linear code $\mathcal{C}$ is fully described bya parity-check matrix $\boldsymbol{H}\in\mathbb{F}_2^{r\times n}$ in such a way that $\mathcal{C}=\{\boldsymbol{c}\in\mathbb{F}_2^n : \boldsymbol{Hc}=\boldsymbol{0}\}$. Solving the *codeword decoding* problem corresponds to decoding a corrupted codeword $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{c}\in\mathcal{C}$ and $\boldsymbol{e}$ is an $n \times 1$ binary vector, also denoted as $\boldsymbol{e}_{n\times 1}$, with Hamming weight $\text{WEIGHT}(\boldsymbol{e}) = t$, $0<t<\lfloor \frac{d}{2} \rfloor$, through recovering either $\boldsymbol{c}$ or $\boldsymbol{e}$. To this end, many decoding algorithms exploit the *syndrome* of the corrupted codeword $\boldsymbol{y}$ through the parity-check matrix $\boldsymbol{H}$, i.e., the $r \times 1$ vector $\boldsymbol{s}$, also denoted as $\boldsymbol{s}_{r\times 1}$, computed as $\boldsymbol{s}=\boldsymbol{Hy}=\boldsymbol{H}(\boldsymbol{c}+\boldsymbol{e})=\boldsymbol{He}$, to recover the unknown error vector $\boldsymbol{e}$. The *syndrome decoding problem* is defined as the one of determining the binary error pattern $\boldsymbol{e}$, knowing a parity check matrix $\boldsymbol{H}$ and the syndrome $\boldsymbol{s}$ of $\boldsymbol{e}$ through $\boldsymbol{H}$.

Both the codeword decoding problem and the syndrome decoding problem, when the parity-check matrix $\boldsymbol{H}$ is randomly chosen among the ones in $\mathbb{F}_2^{r\times n}$ with $\text{rank}(r)$, were proven to be NP-hard by McEliece in [23]. This fact allows one to build asymmetric cryptosystems in which the trapdoor function relies on an obfuscated version of the parity-check matrix $\boldsymbol{H}'$ of a non-random and efficiently decodable code, obtained through multiplying $\boldsymbol{H}'$ by a random non-singular matrix $\boldsymbol{S}_{r\times r}$, which is kept secret, yielding $\boldsymbol{H}=\boldsymbol{SH}'$ as a public piece of information. The obfuscated parity-check matrix $\boldsymbol{H}$ allows anyone willing to send a message, encoded as an error vector $\boldsymbol{e}$ of weight t, to compute its syndrome $\boldsymbol{s}=\boldsymbol{He}$ and send it on a public channel. An adversary willing to recover the message $\boldsymbol{e}$ will then need to solve the syndrome decoding problem for an apparently random parity-check matrix $\boldsymbol{H}$, whilst the legitimate receiver, knowing $\boldsymbol{S}$, can compute $\boldsymbol{s}'=\boldsymbol{S}^{-1}\boldsymbol{s}$ and efficiently solve the syndrome decoding problem for a non-random $\boldsymbol{H}$ obtaining the original message/error $\boldsymbol{e}$.

B. Information Set Decoding

Information Set Decoding (ISD) techniques [22] are the most efficient methods to solve the *syndrome decoding* problem for a generic random parity-check matrix $\boldsymbol{H}$. The main idea of ISD techniques, as proposed by Prange in [26], is to consider the syndrome as obtained by the sum of t columns of $\boldsymbol{H}$ indexed by the positions of the t bits equal to one that are present in the (unknown) error vector $\boldsymbol{e}$. To locate these

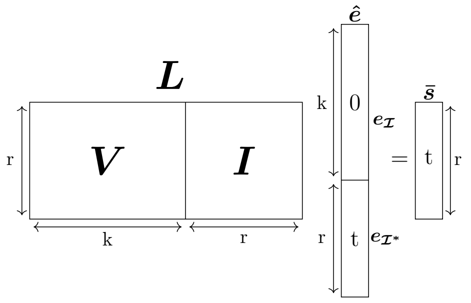


Fig. 1. Schematic view of the ISD strategy proposed by Prange, after that a column permuted and row echelon reduced parity-check matrix is obtained.

positions in $\boldsymbol{e}$, Prange suggests guessing a k-sized set of values $\mathcal{I} \subset \{1, \ldots, n\}$, called *information set*, hoping that it matches the indexes of k zero bits in $\boldsymbol{e}$. The remaining $r = n - k$ index values form the so-called *redundancy set*, denoted by $\mathcal{I}^*$.

Starting from the information set, a permutation matrix $\boldsymbol{P}_{n\times n}$ is derived in such a way that the k leftmost columns of $\widehat{\boldsymbol{H}} = \boldsymbol{HP}$, match the columns of $\boldsymbol{H}$ indexed by $\mathcal{I}$. $\widehat{\boldsymbol{H}}$ it is then brought in a reduced row echelon form $[\boldsymbol{V}_{r\times k} \mid \boldsymbol{I}_r]$ applying a Gaussian elimination algorithm. This procedure is equivalent to finding out the linear transformation $\boldsymbol{U}$ such that $\boldsymbol{U}\widehat{\boldsymbol{H}} = [\boldsymbol{V}_{r\times k} \mid \boldsymbol{I}_r]$. After this normalization, the $r \times k$ binary matrix $\boldsymbol{V}_{r\times k}$ will be a random-looking matrix, while $\boldsymbol{I}_r$ will be an identity matrix of size $r \times r$. In case the reduced row echelon form cannot be obtained, i.e., $\boldsymbol{U}\widehat{\boldsymbol{H}}$ exhibits a singular $r \times r$ submatrix on its right side instead of an identity matrix, another information set $\mathcal{I}$ is randomly guessed, a new permutation matrix $\boldsymbol{P}$ is computed, and a new attempt to obtain the reduced row echelon form $\widehat{\boldsymbol{H}}$ is made.

Prange observes in [26] that the syndrome $\boldsymbol{s} = \boldsymbol{He}$ can be seen as the computation of a syndrome associated to $\widehat{\boldsymbol{H}}$ if the original (unknown) error vector $\boldsymbol{e}$ is assumed to be permuted via the inverse of $\boldsymbol{P}$. Indeed, we have that $\boldsymbol{s} = \boldsymbol{He} = (\boldsymbol{HP})\boldsymbol{P}^{-1}\boldsymbol{e} = \widehat{\boldsymbol{H}}(\boldsymbol{P}^{-1}\boldsymbol{e}) = \widehat{\boldsymbol{H}}\hat{\boldsymbol{e}}$, with $\hat{\boldsymbol{e}}$, $\hat{\boldsymbol{e}} = \boldsymbol{P}^{-1}\boldsymbol{e}$. Considering the row echelon reduced version of $\widehat{\boldsymbol{H}}$, denoted as $\boldsymbol{L} = \boldsymbol{U}\widehat{\boldsymbol{H}}$, we have that $\left(\boldsymbol{U}\widehat{\boldsymbol{H}}\right)\hat{\boldsymbol{e}} = [\boldsymbol{V}_{r\times k} \mid \boldsymbol{I}_r]\hat{\boldsymbol{e}} = \bar{\boldsymbol{s}} = \boldsymbol{Us}$, as shown in Figure 1. Since the permutation is expected to pack all columns of $\boldsymbol{H}$ indexed by the bit-positions of the error vector corresponding to an asserted bit in the r rightmost columns, also the vector $\hat{\boldsymbol{e}}$ will include t asserted bits in its r trailing elements. Denoting as $\boldsymbol{e}_{\mathcal{I}}$ the binary (sub-)vector composed by a copy of the terms of $\boldsymbol{e}$ whose positions are in $\mathcal{I}$, $\hat{\boldsymbol{e}} = [\boldsymbol{e}_{\mathcal{I}}|\boldsymbol{e}_{\mathcal{I}^*}] = [\boldsymbol{0}_{k\times 1}|\boldsymbol{e}_{\mathcal{I}^*}]$. As a consequence, $\bar{\boldsymbol{s}}$ can be thought of as the syndrome of the permuted error $\hat{\boldsymbol{e}}$ through the permuted and row echelon reduced parity-check matrix $\boldsymbol{L} = \boldsymbol{U}\widehat{\boldsymbol{H}} = [\boldsymbol{V}_{r\times k} \mid \boldsymbol{I}_r]$, i.e.: $\bar{\boldsymbol{s}} = [\boldsymbol{V}_{r\times k} \mid \boldsymbol{I}_r][\boldsymbol{0}_{k\times 1}|\boldsymbol{e}_{\mathcal{I}^*}] = \boldsymbol{e}_{\mathcal{I}^*}$. Therefore, if the permutation $\boldsymbol{P}$ packs k positions of the original (and unknown) error vector $\boldsymbol{e}$ corresponding to zero bits on its top part, then multiplying the syndrome by $\boldsymbol{U}$ yields the non-zero part of the permuted error vector itself. To test if the permutation $\boldsymbol{P}$ did actually pack k-zero positions of the error in the first k rows, Prange's algorithm tests if the Hamming weight of $\bar{\boldsymbol{s}}$ matches the one of the error vector, t. If that is the case, the error vector

Algorithm 1: Gaussian Elimination

```
Input  : $\widehat{H}$: parity check matrix $\in \mathbb{F}_2^{r\times n}$
Output : $L$: row echelon reduced parity-check matrix $L = U\widehat{H}$
         $U$: non-singular matrix $\in \mathbb{F}_2^{r\times r}$
1  $k \leftarrow n-r, L \leftarrow \widehat{H}, U \leftarrow I_r$
2  for $i \leftarrow 0$ to $r-1$ do
3      $z \leftarrow i$
4      while $z < r$ and $L_{z,i+k} = 0$ do
5          $z \leftarrow z+1$
6      if $L_{z,i+k} = 0$ then
7          return $\langle \perp, \perp \rangle$
8      if $i \neq z$ then
9          SWAPROW($L_{i,:}, L_{z,:}$); SWAPROW($U_{i,:}, U_{z,:}$)
10     for $z \leftarrow 0$ to $r-1$ do
11         if $z \neq i$ and $L_{(z,i+k)} = 1$ then
12             $L_{z,:} \leftarrow L_{z,:} \oplus L_{i,:}$; $U_{z,:} \leftarrow U_{i,:} \oplus U_{z,:}$
13 return $\langle L, U \rangle$
```

is recovered computing $P\hat{e} = P[\mathbf{0}_{k\times 1}|\bar{s}] = P[\mathbf{0}_{k\times 1}|e_{\mathcal{I}^*}]$ and tested recomputing the syndrome as $s = He$. If the test fails, the algorithm restarts by picking another information set $\mathcal{I}$ (and the corresponding random permutation P).

To determine the complexity of Prange's algorithm, we observe that it is a randomized algorithm that always yields a correct output, i.e., the value of e or it informs about a *failure*, in an expected finite runtime. The expected runtime of the algorithm $C_{\text{Prange-ISD}}(n, r, t)$ is obtained given the probability that a run of the algorithm itself succeeds $\Pr_{succ}$, multiplying its inverse by the cost of an algorithm iteration $C_{\text{Prange-iter}}(n, r)$, under the assumption that $\Pr_{succ}$ is constant among all the iterations. In Prange's algorithm, $\Pr_{succ}$ is determined dividing the number of permuted error vector configurations having all the t asserted elements in the last r positions by the total number of possible error vector configurations, i.e., $\Pr_{succ} = \binom{r}{t}/\binom{n}{t}$. It is worth noting that there is a comparatively small factor contributing to the failure of a single iteration of the algorithm, namely the probability that the computation of the reduced row echelon form of $\widehat{H}$ fails. Since the probability that a random $r \times r$ binary matrix is non-singular is $\prod_{i=1}^{r}(1 - \frac{1}{2^i})$ [22], such a factor converges to ≈ 0.2887 for increasing values of r, contributing by factor of ≈ 4 to the number of repetitions to be done by Prange's ISD. The cost of performing a single repetition of the computation of Prange's algorithm is dominated by the computation of the row-echelon form reduction of $\widehat{H}$, for which the detailed classical algorithm is reported as Algorithm 1. The algorithm operates on a matrix L, initialized as a copy of $\widehat{H}$, which will contain the reduced row echelon form of $\widehat{H}$ at the end of the execution, and, in parallel, on a matrix U initialized as an $r \times r$ identity matrix, which will contain the linear transformation to be applied to map $\widehat{H}$ into L (line 1). The procedure iterates on the r rows of L, and for each of them, denoted as $L_{i,:}$, $0 \le i < r$, locates the first row $L_{z,:}$, $i \le z < r$, such that $L_{z,i+k}$ is not null (lines 3–5). If no such row exists, $\mathcal{I}^*$ selected a singular (right) $r \times r$ submatrix of H, and the procedure aborts. When such a row $L_{z,:}$ is found, the algorithm swaps it with $L_{i,:}$, ensuring that $L_{i,i+k} = 1$, and keeps track of the swap also on the rows of U (line 8–9). Finally, the row $L_{i,:}$ is added to all the other rows $L_{z,:}$, $i \neq z$, whenever $L_{z,i+k} = 1$, tracking the addition also on the rows of U (lines 10–12). The computational complexity of Algorithm 1 is $\mathrm{C}_{\mathrm{GE}} = \mathcal{O}\left(\frac{3nr^2}{4} + \frac{nr}{4} - \frac{n}{2} + \frac{3r^2}{4} - \frac{r}{2}\right)$ bit operations. The remaining portion of Prange algorithm requires $\mathcal{O}(r^2)$ and $7r + \log_2(r)$ bit operations to compute Us and the weight check on $\bar{s}$, respectively.

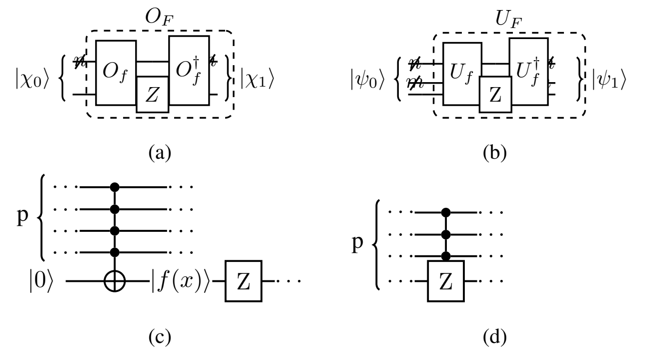


Fig. 2. (a) Standard implementation of Grover's oracle circuit. (b) Our realization of the Grover's oracle circuit. (c) Multicontrolled X gate in the circuit realization of U_f, with $p \le n+m$ controlled qubits and target qubit the ancilla used to store the result of $f(x)$ (d) Optimization of the U_F circuit that replaces the circuit in (c) with a multicontrolled Z gate, denoted as $\mathrm{C}^{p-1}(\mathrm{Z})$.

C. Grover's algorithm

The algorithmic framework proposed by Grover [27] finds the value of a vector of binary variables $x^* \in \mathbb{F}_2^n$ on which a given function $f : \mathbb{F}_{2^n} \mapsto \mathbb{F}_2$ evaluates to 1 (assuming that in $f(\cdot)$ only a single such value x^* exists), employing $\mathcal{O}(\sqrt{2^n})$ function computations. This provides a significant speedup over the classical alternative which needs $\mathcal{O}(2^n)$ function computations to derive the same result via exhaustive search.

The framework proposed by Grover relies on four steps, of which the second and the third are iterated $\mathcal{O}(\sqrt{2^n})$ times: *i)* preparation of an input quantum state, *ii)* computation of the so-called oracle function, *iii)* computation of the diffusion function, and *iv)* measurement of the solution. The purpose of the framework is to obtain a quantum state where the qubits representing the basis state of x^*, the sought value, will be measured with non-negligible probability.

State preparation. Grover's algorithm starts by building a quantum state obtained from the equal-amplitude superposition of all the basis states of n qubits, each labeled with a distinct binary string in $\{0,1\}^n$. Such a superposition is obtained through the joint application of n Hadamard gates acting simultaneously on the n qubits of the input quantum register $|00\ldots0\rangle$, i.e.: $\mathrm{H}^{\otimes n}|00\ldots0\rangle = \mathrm{H}^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle$. It is common to use an additional ancillary qubit, initialized to $|0\rangle$; the initial state will be therefore $|\chi_0\rangle = \mathrm{H}^{\otimes n}|0\rangle^{\otimes n}|0\rangle$.

In order to reduce the search space of the algorithm, we considered, in our adaption of Grover's algorithm, a proper subset of $\{0,1\}^n$ as our input domain. We also added $m \ge 1$ additional ancilla qubits, used to store temporary results. Specifically, denoting with $\mathbb{D} \subset \{0,1\}^n$ the said domain, with cardinality

$|\mathbb{D}|{=}d$, our input state $|\psi_0\rangle$ will be $|\psi_0\rangle = \mathrm{P}\,|0\rangle^{\otimes n}|0\rangle^{\otimes m}|0\rangle$, where the 2^n-dimensional operator P is such that

$$\mathrm{P}\,|0\rangle^{\otimes n} = \frac{1}{\sqrt{d}}\sum_{x\in\mathbb{D}}|x\rangle = \frac{1}{\sqrt{d}}\sum_{x\in\mathbb{D}\setminus\{x^*\}}|x\rangle + \alpha^*|x^*\rangle$$

and $\alpha^* = \frac{1}{\sqrt{d}}$ is the amplitude associated to state x^*.

Oracle function computation. This step aims at singling-out the basis state corresponding to the sought value x^* by changing the sign of its amplitude alone. To this end, the unitary operator O_F is applied to the input state, producing:

$$|\chi_1\rangle = O_F\,|\chi_0\rangle = \left(\frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n\setminus\{x^*\}}|x\rangle - \alpha^*|x^*\rangle\right)\otimes|0\rangle$$

and $\alpha^* = \frac{1}{\sqrt{2^n}}$ is the amplitude associated to state x^*. Figure 2a shows the standard implementation of Grover's oracle. It makes use of the O_f subcircuit, whose purpose is to implement the Boolean function $f(\cdot)$ and to store the binary result on the ancillary qubit. The goal of $\mathrm{Z} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$, instead, is to flip the sign of the input state only when the ancillary qubit is in state $|1\rangle$, i.e., when $|f(x)\rangle = |1\rangle$, which happens only when $x = x^*$. Finally, $O_f^\dagger$ restores all the input qubits to their starting state, apart from the phase of $|x^*\rangle$.

Figure 2b shows our implementation of the Grover's oracle circuit, which corresponds to applying the unitary operator $U_F = U_f^\dagger \circ (\mathrm{I}^{\otimes(n+m)}\otimes\mathrm{Z})\circ U_f$ to our input state $|\psi_0\rangle$, obtaining

$$|\psi_1\rangle = U_F\,|\psi_0\rangle = \left(\frac{1}{\sqrt{d}}\sum_{x\in\mathbb{D}\setminus\{x^*\}}|x\rangle - \alpha^*|x^*\rangle\right)\otimes|0\rangle^{\otimes m+1}$$

and $\alpha^* = \frac{1}{\sqrt{d}}$ is the amplitude associated to state x^*.

It is possible to perform a local optimization observing that the computation of U_f can always be written as a quantum circuit which never involves the bottom qubit until the last quantum gate, constituted by a multicontrolled $\mathrm{X} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$ gate (see Figure 2c for a graphical depiction of the last gate). Let us denote with p the number of control qubits of the said gate controlled X, $p{\leq}n{+}m$. Since the last qubit is subsequently fed into a Z gate, that changes the phase only if said qubit is in state $|1\rangle$, it is possible to build a multicontrolled Z gate taking one of the aforementioned p qubits as the target qubit and the remaining $p{-}1$ qubits as control qubits (see Figure 2d). In the next sections, we will denote the Z gate with one target qubit and $p{-}1$ control qubits as $\mathrm{C}^{p-1}(\mathrm{Z})$, considering it part of our Grover oracle.

Diffusion function computation. This step builds on the output of the Grover's oracle stage to obtain a superposition where the amplitude α^* of the state corresponding to x^* increases its modulus and its phase alone is flipped again. To do so, Grover observed that the output state of the oracle circuit $|\chi_1\rangle$ can be equivalently expressed as:

$$|\chi_1\rangle = (|\chi_0\rangle - 2\alpha^*|x^*\rangle)\otimes|0\rangle\,,$$

and that the application of the *diffusion operator* $\mathrm{D}{=}\,2\,|\chi_0\rangle\langle\chi_0|{-}\mathrm{I}^{\otimes n}{=}\ \mathrm{H}^{\dagger^{\otimes n}}(2\,|0^n\rangle\langle 0^n|{-}\mathrm{I}^{\otimes n})\mathrm{H}^{\otimes n}$ to the first n qubits of $|\chi_1\rangle$ yields

$$|\chi_2\rangle = (\mathrm{D}\otimes\mathrm{I})\,|\chi_1\rangle = \left(\frac{2^n-4}{2^n}|\chi_0\rangle + \frac{2}{\sqrt{2^n}}|x^*\rangle\right)\otimes|0\rangle$$

The state $|\chi_2\rangle$ is thus a non-uniform superposition of all basis states in $\{0,1\}^n$ and the quantum state associated to x^* exhibits an amplitude increased with respect to one resulting from uniform superposition. Our implementation of D traces the Grover's approach except for the use of our input state preparation step (which is represented by the 2^n-dimensional operator P), i.e., $\mathrm{D}_{\mathrm{our}} = \mathrm{P}^\dagger(2\,|0^n\rangle\langle 0^n| - \mathrm{I}^{\otimes n})\mathrm{P}$, yielding:

$$|\psi_2\rangle = \left(\mathrm{D}_{\mathrm{our}}\otimes\mathrm{I}^{\otimes m+1}\right)|\psi_1\rangle = \left(\frac{d-4}{d}|\psi_0\rangle + \frac{2}{\sqrt{d}}|x^*\rangle\right)\otimes|0\rangle^{\otimes m+1}$$

Operator $(2\,|0^n\rangle\langle 0^n|{-}\mathrm{I}^{\otimes n})$ can be implemented by a $\mathrm{C}^{n-1}(\mathrm{Z})$ acting on the n qubits involved in our input state preparation subcircuit, with an X gate before each of its input qubits and another one after each of its output qubits.

Measurement of the solution and number of repetition of the oracle and diffusion steps. Grover proved that the repeated application of the oracle and diffusion operators $(\mathrm{D}\circ O_F)$ keep increasing the amplitude of the sought basis state $|x^*\rangle$ $(f(x^*) = 1)$, while reducing the others. It can be shown that the optimal number of repetitions to have a probability close to 100% to measure a classic bit equal to one, acting on $|f(x)\rangle$, and consequently to measure the sought solution x^*, acting on $|x\rangle$, is $\approx \mathcal{O}(\frac{\pi}{4}\sqrt{2^n})$ or, in our case, $\approx \mathcal{O}(\frac{\pi}{4}\sqrt{d})$. Finally, [28] discusses two important variations to Grover's algorithm: how executing half of the aforesaid number of repetitions provides a success probability close to 50% and how having m distinct solutions reduces the number of iterations to $\approx \mathcal{O}(\frac{\pi}{4}\sqrt{\frac{2^n}{m}})$ or, in our case, $\approx \mathcal{O}(\frac{\pi}{4}\sqrt{\frac{d}{m}})$.

D. Sorting networks

The classical comparator network model of computation inspired the implementation of a crucial ingredient of our design for an ISD quantum circuit. In such a computational model a sorting algorithm can be performed in sublinear time by executing multiple comparison operations simultaneously. The implementation of sorting algorithms are denoted as *sorting networks* and are described drawing m "wires", each of which associated to one of the values to be sorted, and several comparator elements affixed on pairs of wires (see Figure 3a), which are in turn are organized in layers (see Figure 3b) in such a way that no cycles exist on the created graph.

As shown in Figure 3a, a classical comparator element receives two distinct values on its input wires, compares them and then outputs the minimum between them on the top output wire and the maximum on the bottom one. Under the assumption that each comparator performs its action in $O(1)$ (i.e., constant) time, the running time of a sorting network is the time it takes for all the output wires to receive their values once the input wires receive theirs. As a consequence, the largest number of comparators that any input element can pass through as it travels from an input wire to an output wire (a.k.a. *depth* of the network), defines the execution time of the whole network. An additional figure of merit is given by

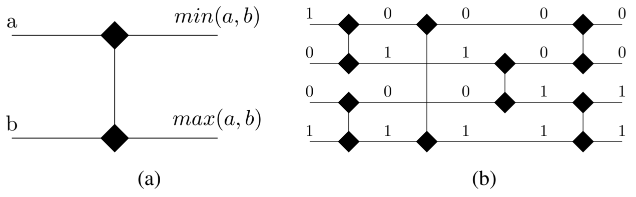


Fig. 3. (a) Comparator element. (b) A 4-bit input sorting network.

the number of non-overlapping comparisons done at the same time as this proves to influence also the depth of the network other than its total number of comparators.

Knuth [29, Chap. 5.3.4] offers a detailed review of several sorting networks design strategies. Since we want to reuse the sorting network in a quantum circuit, our focus has been to minimize the overall depth of the network. For values of m in the thousands, as it is the case in our scenario, only a single asymptotically optimal depth design (i.e., $\mathcal{O}(\log(m))$ depth) is known [30]. However, Knuth [29, Chap. 5.3.4] shows that this network design is not of practical interest, since the constants hidden by the asymptotic notation are significant, and the total number of comparators is increased significantly with respect to $\mathcal{O}(\log(m)^2)$ depth designs. In the following we employ the sorting network topology detailed in [31, Chap. 27.5], where a non-asymptotic analysis of the network complexity shows that the total number of comparators in this type of sorting network is $(m-1)\log_2(m)(\log_2(m)-1)$, while the overall depth is instead $\frac{1}{2}\log_2(m)(\log_2(m)+1)$. An example of such a network applied to m=4 binary values is shown in Figure 3b.

III. A Complete ISD Quantum Circuit

To employ the algorithmic framework proposed by Grover for Prange's algorithm, we rephrase the solution of the syndrome decoding problem into finding an r-sized set $\mathcal{I}^*$ that satisfies Prange's conditions, namely: *(i)* the columns of the parity check matrix $\boldsymbol{H}$ indexed by $\mathcal{I}^*$ form an invertible matrix; and *(ii)* the Hamming weight of $\bar{\boldsymbol{s}}=\boldsymbol{U}\boldsymbol{s}$ equals t.

This problem is equivalent to identify the input x^* to a Boolean function $f : \mathbb{D} \rightarrow \{0,1\}^p$, $p \geq 1$, constructed in a way that it outputs 1^p if and only if both the previous conditions are satisfied. Let $\mathbb{D} \subset \{0,1\}^n$ be the set of all the weight-r, length-n Boolean vectors. An element $x\in\mathbb{D}$, $x = (x_0, \ldots, x_{n-1})$ is a Boolean vector of Hamming weight r in which the indexes of asserted bits are elements of $\mathcal{I}^*$.

A. Preparing a superposition of all permutations

The input state of our algorithm is a superposition of states encoding all length-n, weight-r Boolean strings. This quantum state, encoded in an n-qubits register denoted sel, is known as the Dicke state $|D^n_r\rangle$ and it is defined as $|D^n_r\rangle = \frac{1}{\sqrt{\binom{n}{r}}} \sum_{\textsc{Weight}(x)=r} |x\rangle, \qquad x \in \{0,1\}^n.$

To the best of our knowledge, the most efficient algorithm in terms of both depth and number of gates to prepare $|D^n_r\rangle$ for a generic value of r is the one detailed in [32], requiring r X, $5nr - 5r^2 - 2n$ CNOTs and $4nr-4r^2-2n+1$ $\mathrm{R_y}$ gates.

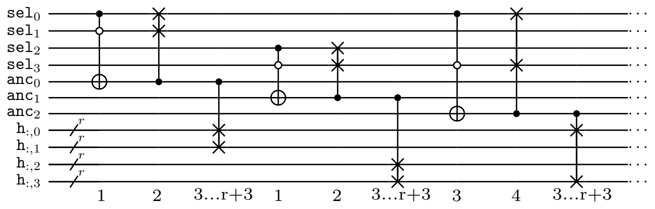


Fig. 4. The operations corresponding to the first three comparators in the sorting network of Figure 3b. Each classical comparator is translated into a CCNOT gate followed by a CSWAP gate, swapping the sel qubits. To move the columns of $\boldsymbol{H}_{r\times n}$ indexed by sel at the end of the matrix, we also have CSWAPs between the corresponding qubits of h register, here represented column-wise for clarity. Notice that each CSWAP involving the h register is composed by r distinct CSWAPs. The numbers at the bottom represent the stage (i.e., the depth) at which each gate can be executed.

Using the same assumptions for the depth evaluation of [33], we can straightforwardly derive 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 result for the depth is therefore $\leq \frac{27nr-12n-27r^2+3}{r-2}$.

B. Implementing our Grover oracle operator

The quantum circuit to compute Grover's oracle operator U_F for the Boolean function f can be described by 5 distinct steps which compute U_f (the first four) and required phase flip (the last one). The first four steps are repeated in reverse order to compute $U_f^\dagger$ after the phase flip is computed.

1) represent the matrix $\boldsymbol{H}$ and vector $\boldsymbol{s}$ on two quantum registers, denoted as h and syn; 2) prepare a superposition of all the columns indexed by all possible values of $\mathcal{I}^*$ on the right side of $\boldsymbol{H}$, obtaining a representation of all the possible $\widehat{\boldsymbol{H}}$ encoded in h; 3) apply the Gaussian elimination procedure to h, obtaining on this register the representation of $\boldsymbol{L}$; in parallel, apply the same operations to syn to obtain on this register the representation of $\bar{\boldsymbol{s}}$; 4) compute the Hamming weight of syn and check it against the integer value t; 5) flip the amplitude of the basis state if at step 2) the Gaussian elimination yielded for it an identity matrix on the qubits containing the $r \times r$ right submatrix of $\boldsymbol{L}$ and the weight comparison of step 4) was successful.

Step 1) Encoding $\boldsymbol{H}$ and $\boldsymbol{s}$. The first step of our oracle encodes $\boldsymbol{H}$ and $\boldsymbol{s}$ in the quantum circuit in two distinct quantum registers, h and syn, of rn and r qubits respectively. The registers are initialized to $|0^{rn}\rangle$ and $|0^r\rangle$ and we selectively apply X gates to the qubits representing asserted elements of $\boldsymbol{H}$ or $\boldsymbol{s}$. Since $\boldsymbol{H}$ is a random parity-check matrix, we expect half of its elements to be equal to 1, thus resulting in $\frac{rn}{2}$ X gates being used on average. Since also $\boldsymbol{s}$ has a random value, we expect to need $\frac{r}{2}$ X on average to initialize s. All the gates can be applied in parallel, resulting in a depth of 1.

Step 2) Compute all the $\widehat{\boldsymbol{H}}$ in superposition. We recall that the sel register contains $|D^n_r\rangle$ at the beginning of the execution of Grover's algorithm, and a non-uniform superposition of the weight-r length-n basis states in the subsequent iterations of the algorithm itself. Consider one of the basis states contained in sel, and the corresponding values of the qubits in h in the same basis state. Our goal is then to bring

Algorithm 2: Quantum Circuit Friendly Gauss. Elim.

```
Input  : $\widehat{\boldsymbol{H}}$: parity-check matrix $\in \mathbb{F}_2^{r\times n}$, $\boldsymbol{s}$: syndrome
Output : $\boldsymbol{L}$: parity check matrix $\boldsymbol{L} = \boldsymbol{U}\widehat{\boldsymbol{H}} = [\boldsymbol{V}|\boldsymbol{I}_r]$
         $\bar{\boldsymbol{s}}$: transformed syndrome, $\bar{\boldsymbol{s}} = \boldsymbol{U}\boldsymbol{s}$
1 $k \leftarrow n - r, \boldsymbol{L} \leftarrow \widehat{\boldsymbol{H}}$
2 for $i \leftarrow 0$ to $r - 1$ do
3     for $z \leftarrow i + 1$ to $r - 1$ do
4         if $\boldsymbol{L}_{i,i+k} = 0$ and $\boldsymbol{L}_{z,i+k} = 1$ then
5             $\boldsymbol{L}_{i,:} \leftarrow \textsc{Swap}(\boldsymbol{L}_{i,:}, \boldsymbol{L}_{z,:})$ $\boldsymbol{s}_i \leftarrow \textsc{Swap}(\boldsymbol{s}_i, \boldsymbol{s}_z)$
              /* the swaps can be replaced by
                 $\boldsymbol{L}_{i,:} \leftarrow \boldsymbol{L}_{i,:} \oplus \boldsymbol{L}_{z,:}$  $\boldsymbol{s}_i \leftarrow \boldsymbol{s}_i \oplus \boldsymbol{s}_z$        */
6     for $z \leftarrow 0$ to $r - 1$ do
7         if $z \neq i$ and $\boldsymbol{L}_{z,i+k} = 1$ then
8             $\boldsymbol{L}_{z,:} \leftarrow \boldsymbol{L}_{z,:} \oplus \boldsymbol{L}_{i,:}$ $\boldsymbol{s}_z \leftarrow \boldsymbol{s}_z \oplus \boldsymbol{s}_i$
9 return $\boldsymbol{L}, \bar{\boldsymbol{s}}$
```

all the r columns of h having the qubits in the basis state of sel in the corresponding position set to 1 in the portion of h corresponding to the rightmost $r \times r$ submatrix. Rearranging the columns of $\boldsymbol{H}$ according to the permutation encoded in sel is equivalent to sorting them lexicographically according to the value taken by the qubits in sel matching the positions of the columns in $\boldsymbol{H}$. We therefore sort both the columns of $\boldsymbol{H}$ and the qubits of sel using a quantum circuit inspired by the comparator network detailed in Section II-D, performing the comparisons required only among the qubits of sel. As such, we design a quantum circuit which swaps two sets of r qubits from h according to the values taken by two qubits of sel, together with swapping the qubits of sel themselves. This circuit is the analog of the comparator in a sorting network, we thus refer to it as *quantum comparator*. Figure 4 depicts the first three quantum comparators, placed according to the topology of the sample sorting network reported in Figure 3b. Each quantum comparator, consisting of $4 + r$ consecutive gates, is depicted in a compact form as 3 gates for the sake of space. The first quantum comparator acts on the first two qubits of sel, $\texttt{sel}_0$ and $\texttt{sel}_1$. The comparison $\texttt{sel}_0 > \texttt{sel}_1$ is computed with a CCNOT employing as controllers the value $\texttt{sel}_0$ and the negated value of $\texttt{sel}_1$ (depicted as a white circle instead of a filled one). We realize the negated control value through employing an X gate on $\texttt{sel}_1$ before and after the CCNOT. The CCNOT changes an ancilla qubit, $\texttt{anc}_0$, originally initialized to $|0\rangle$, to $|1\rangle$ in all the basis states where $\texttt{sel}_0 > \texttt{sel}_1$. The ancilla qubit is then employed as a controller for a single CSWAP gate performing the swap on $\texttt{sel}_0$ and $\texttt{sel}_1$, and for a set of r CSWAP gates acting on the two sets of r qubits representing the columns which have to be swapped, i.e., $\texttt{h}_{:,0}, \texttt{h}_{:,1}$.

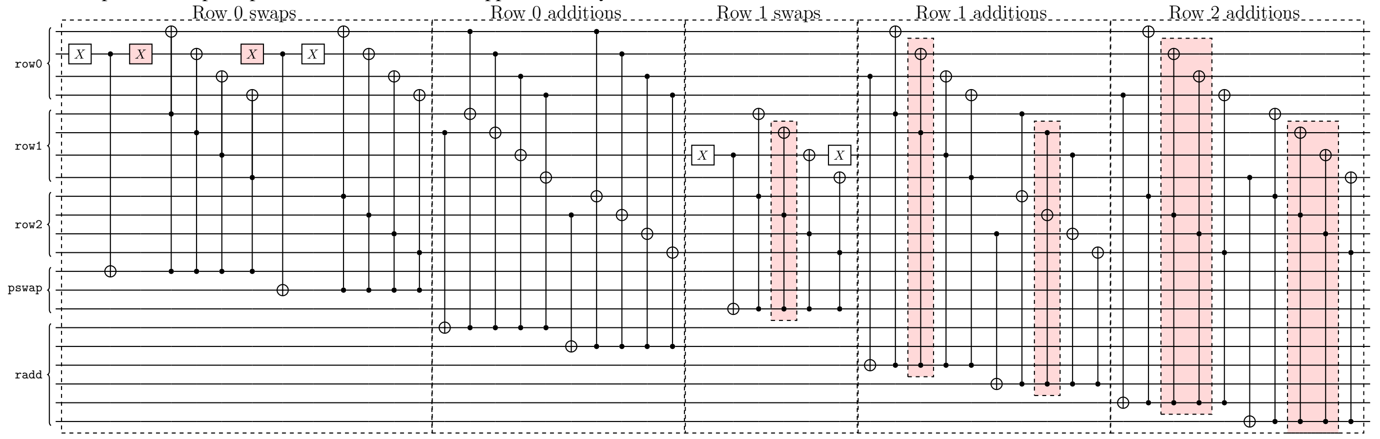
The gate count of a single quantum comparator is thus 2 X gates, 1 CCNOT and $r + 1$ CSWAPs. The overall cost of the computation of the superposition of all the values of $\widehat{\boldsymbol{H}}$ in h is obtained multiplying the cost of a single quantum comparator, by the number of comparators in the network, namely $(n-1)\log_2(n)(\log_2(n)-1)$ (see Section II-D). Scheduling whole quantum comparators in the same topology of a classical comparator network from Section II-D yields a total depth of $\frac{1}{2}\log_2(n)(\log_2(n)+1)$ multiplied by the depth of a single comparator, namely $4 + r$. To reduce the overall circuit depth, we observe that, at each layer of the sorting network, CCNOTs and CSWAPs belonging to different quantum comparators can be executed in parallel. Furthermore, the CSWAPs involving h can start in parallel with the execution of the next layer of the sorting network, which only requires that the CCNOT involving the sel register has already been executed. The total depth is therefore equal to twice the depth of the classical sorting network — since the first CCNOT and CSWAP are executed sequentially — plus an r factor due to the CSWAP gates involving the qubits in h. This scheduling leads to a total depth of $\log_2(n)(\log_2(n)+1) + r$.

Step 3) Gaussian elimination computation and $\bar{s}$ computation. h now contains in superposition all the column permutations which correspond to packing an r column set indexed by $\mathcal{I}^*$ on the right side of $\boldsymbol{H}$. Thus, we now compute the reduced row echelon form of this superposition, via Gaussian elimination, and check which of all the possible $r \times r$ rightmost submatrices are invertible. To this end, we translate the Gaussian elimination algorithm (Algorithm 1) in an appropriate quantum circuit. The hindrances to the translation in the Gaussian elimination are: 1) a data-dependent early-abort (lines 6–7) 2) the presence of non-countable loops (lines 4–5) 3) conditional operations which act on data also present in the condition calculation.

We start by rewriting Algorithm 1 into a semantically equivalent one, Algorithm 2, taking care of removing the first and second hindrances. We report an example of a complete circuit for a 4×3 element $\hat{\boldsymbol{H}}$ in Figure 5. Algorithm 2 also includes the computation of the transformed value of $\boldsymbol{s}$, namely $\bar{\boldsymbol{s}} = \boldsymbol{U}\boldsymbol{s}$. Indeed, this computation can be easily interleaved with the Gaussian elimination performing swaps and additions between single elements of $\boldsymbol{s}$. The first hindrance, i.e., the early abort of Algorithm 1 due to a potentially singular submatrix selected by $\mathcal{I}^*$, is managed through an a posteriori check on the presence of an identity submatrix in $\boldsymbol{L}$ when the check on the condition of the oracle function on the basis state is performed. We therefore removed altogether the runtime check in Algorithm 2. The second hindrance, i.e., the presence of a non-countable loop to find the pivot (lines 4–5, Alg. 1) is overcome with a countable loop strategy ensuring that the i-th row in the execution of the outermost loop contains a pivot. This strategy consists in computing a countable loop sweeping over the lines starting from the $i+1$-th to the $r-1$-th and swapping the row being swept with the i-th only if the i-th row does not contain a pivot. This approach, reported in Algorithm 2, lines 3–5 only employs a countable loop.

Managing the need to compute operations depending on the computation of a condition which involves also one of their operands (lines 4–5 and lines 7–8 of Algorithm 2) requires to encode the conditional operations as quantum circuits. We solve this third hindrance by storing the outcome of each condition computation on a dedicated ancilla qubit. These ancilla qubits are then employed as control qubits in appropriate control gates to perform the computation contained in the conditional statements. We thus introduce two quantum ancillae registers, pswap and radd, where the result of the computation of the condition driving row swaps (line 4), and the result of the computation of the condition determining row additions (line 7) are stored, respectively. To accommodate

Fig. 5. Quantum circuit employed for the reduced row echelon form of a 3×4 matrix, considering as pivot elements the ones belonging to the rightmost 3×3 submatrix. The gates in red, although present in the straightforward implementation of the algorithm, can be eliminated. Note that the last row does not have a pseudo swap subpart since it cannot be swapped with any other row.



all the values, the size of pswap is determined considering number of times the body of the loop at lines 3-5 is executed in a complete Gaussian elimination. Such a number is derived to be $\frac{1}{2}r(r-1)$ considering that the outer loop at lines 2–8 is computed r times. Through an analogous line of reasoning, the size of radd is determined to be $r(r-1)$.

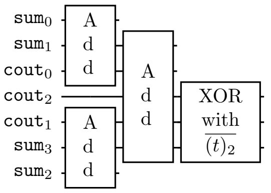
The computation of the swap control condition (line 4) and the actual swaps have the same effect of employing a CCNOT gate controlled by the two qubits in h storing $\boldsymbol{L}_{i,i+k}$ and $\boldsymbol{L}_{z,i+k}$, taking care to negate the first one, and acting on an ancilla in pswap. This CCNOT will toggle the ancilla qubit to $|1\rangle$ only if the i-th row does not contain a valid pivot and the z-th does. It is thus possible to employ the ancilla qubit to drive a sequence of CSWAP gates performing the swap between the rows in h and the appropriate bit in syn (line 5). In case CNOT gates are cheaper to synthesize than CCNOT gates, and CCNOT gates are cheaper than CSWAP, as it is the case with the commonly used and universal Clifford+T gate set, it is possible to optimize the computation of the swap control and the application of the swap itself as follows. Observe that substituting a row swap with the addition of the newly found pivot row (the z-th in Algorithm 2, line 5) to the one lacking it (the i-th one) will still yield a correct Gaussian elimination algorithm. Note that such an addition is necessary only when the i-th row lacks a pivot, leading to a simpler condition to be checked with respect to the one at line 4, allowing the test to be performed with a CNOT gate controlled by the negated qubit representing $\boldsymbol{L}_{i,i+k}$ in h instead of a CCNOT. The resulting optimized design of the compare-and-add functionality is reported in Figure 5, where it is also highlighted the fact that it is possible to uncompute only once the negation on $\boldsymbol{L}_{i,i+k}$ omitting some X gates.

The computation of the row addition control condition (line 7) can be done simply considering the $\boldsymbol{L}_{z,i+k} = 1$ portion, and abiding to the $z \neq i$ clause by not implementing the corresponding portion of the quantum circuit. The strategy to implement the comparison and addition is analog to the optimized one described above for the alternate strategy to row swapping. The row addition stage can be optimized observing that adding together the qubits representing values of $\boldsymbol{L}$ belonging to columns between k and $i+k$ is not required, as they will all contain zeroes by virtue of the Gaussian elimination actions performed. As a consequence, the corresponding additions, performed via CCNOT gates, can be omitted as depicted in Fig. 5 (removable gates in red).

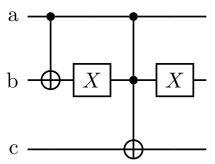
After performing the aforementioned optimizations, counting the number of gates required to perform the actual pseudo swaps and row additions on elements of h we obtain $\frac{1}{6}r(r-1)(3n-r+2)$ CCNOT for the pseudo swaps and $\frac{1}{2}r(r-1)(2n-r+1)$ CCNOT for the row additions, for a total of $\frac{1}{6}r(r-1)(9n-4r+5)$ CCNOTs. Considering our optimized proposal, i.e., the one employing CNOTs and CCNOTs instead of CCNOTs and swaps, the condition calculation portion of the circuit will need $\frac{3}{2}r(r-1)$ CNOTs, and $2(r-1)$ X gates. Concerning the gates required to apply the same operations on the contents of the syn quantum register to compute the corresponding $\bar{\boldsymbol{s}}$ superposition, we have a single CCNOT for each pseudo swap or row addition subcircuit. The total number of additional CCNOT is therefore $\frac{3}{2}r(r-1)$. The total depth of the circuit is given by the number of CCNOT gates involving rows, as the computation of the X gates required for the negated controls can be executed during the preceding row addition phase, and the CCNOT gates used for the computation of $\bar{\boldsymbol{s}}$ can be interleaved with the ones involving rows.

Step 4) Hamming weight compute and check To compute the Hamming weight of the syn register, that at this stage will contain the representation of $\bar{\boldsymbol{s}}$, we use two additional set of registers, cin and cout. Figure 6a reports the full Hamming weight circuit for a size 4 syn register.

We build our circuit as a binary adder tree where the results of two child adders are employed as addends in their father. We compute the values of the Hamming weights of our r qubit register syn employing a $\log_2(r)$ deep adder tree, where the first layer is constituted by 1 qubit operand adders employing a qubit from cout to store their carry-out. Following layers compute the sum on the output of the previous layers, and



(a) Hamming weigh computation circuit acting on sum



(b) Our optimized 1-qubit operand adder

Fig. 6. (a) shows the Hamming weight compute and check subcircuits for the sum register of the running example. The result of the computation, stored on $\texttt{cout}_2$, $\texttt{cout}_1$ and $\texttt{sum}_3$, is compared against the Boolean complement of the natural binary representation of the constant $t-p$. (b) shows our proposal for an adder taking as input two 1-qubit register. The sum of $a+b$ is stored in c and b, with c being the most significant bit.

they use adders involving addends whose qubit size increases by 1 with respect to the previous layer.

To realize multi-qubit adders we employed the design proposed by Cuccaro [34], a reversible variant of the ripple carry adder. The Cuccaro adder stores the sum of its inputs a and b in the qubits where b is stored, with one additional qubit for the carry-in and another one for the carry-out. The design by Cuccaro restores the initial state of the carry-in qubit, allowing its reuse. We employed this feature to use a number of carry-in qubits equal to the one of the first layer adders, and reusing the same carry-in qubits in all subsequent layers (which need a smaller number of them). To determine the gate count for this sub-circuit, we note that a Cuccaro adder operating on r qubit inputs requires $2r{-}1$ CCNOT gates, $5r{+}1$ CNOT gates and $2r$ X gates, and has a circuit depth of $2r{+}6$ gates. For $r{=}1$, we improved the proposal by Cuccaro, realizing an adder (Figure 6b) requiring 1 CCNOT, 1 CNOT and 2 X gates with depth 4. The number of layers of the adder tree circuit is $\log_2(r)$. In the i-th layer, $r/(2^i)$ adders are employed, leading to a total count of $\sum_{i=1}^{\log_2(r)} \frac{r}{2^i}{=}r{-}1$ adders. Therefore, we need exactly $\frac{r}{2}$ carry-in and $r-1$ carry-out ancillary qubits. Each adder of the i-th layer of the adder tree takes as input two distinct i-qubit strings. The overall number of gates required is given by $\sum_{i=1}^{\log_2(r)} \frac{r}{2^i}\textsc{Adder\_Cost}(i)$, where $\textsc{Adder\_Cost}(i)$ denotes the gate cost of an adder taking as inputs two i-qubit strings. Expanding the summation yields the gate-count in Table I. Since all the adder at the same level have identical depths and can be run in parallel, the overall depth is simply given by the sum of the depth of a single adder per each level. At level i, the result is stored on exactly $i+1$ qubits, as Cuccaro's design reuses the qubits of one of the operands to store part of the result. Hence, in the final stage, the final sum will be stored on a $\log_2(r)+1$ qubits register, denoted as hwreg.

At this point, we should check that hwreg contains the binary representation of t. To this end, we xor into the qubits of hwreg the boolean complement of the natural binary representation of t. This is done to ensure that, if a given state is such that $\textsc{Weight}(\texttt{syn}) = t$, then all the qubits contained in the hwreg 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 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, the basis state is transformed into a $\log_2(r)+1$ all-ones state contained in the hwreg register.

Step 5) Oracle phase flip As explained in Section II-C, we can apply a $\mathrm{C}^{p-1}(\mathrm{Z})$ to invert the phase of $|x^*\rangle$ in the oracle stage. We determine whether the basis state is $|x^*\rangle$ checking a collection of p qubits, belonging to both h, as we need to check if its right $r\times r$ submatrix is an identity, and to hwreg, for the Hamming weight value check. Given the Gaussian elimination procedure, it is sufficient to test whether the right submatrix of h has an all-one diagonal to test if it did succeed, in turn allowing us to test if the corresponding r qubits are asserted. Thus, the $\mathrm{C}^{p-1}(\mathrm{Z})$ gate acts on $p = r+\log_2(r)+1$ qubits.

C. Implementing our Grover diffusion operator

We recall from Section II-C that we implement Grover's diffusion operator $\mathrm{D}_{\mathrm{our}} = 2|\psi_0\rangle\langle\psi_0| - \mathrm{I}^{\otimes n} = \mathrm{P}^{\dagger}(2|0^n\rangle\langle 0^n| - \mathrm{I}^{\otimes n})\mathrm{P}$ realizing the $(2|0^n\rangle\langle 0^n| - \mathrm{I}^{\otimes n})$ function with a multi-controlled Z gate, $\mathrm{C}^{n-1}(\mathrm{Z})$, acting on the n qubits provided by our input state preparation subcircuit, with an X gate in front of each of its input qubits and another X gate following each of its output qubits. In our algorithm, those qubits are the n qubits of the sel register. Since we have to apply an X gates before and after the multi-controlled phase flip gate, in this stage we need $2n$ X gates and a single $\mathrm{C}^{n-1}(\mathrm{Z})$ gate. We complete the implementation of $\mathrm{D}_{\mathrm{our}}$ recalling that P corresponds in our case to the circuit for the preparation of the Dicke state on the sel register, which is computed twice: at the beginning of $\mathrm{D}_{\mathrm{our}}$ (in reverse) and at the end.

IV. Functional and Security Assessment

We validated the functionality of the proposed quantum circuit testing each one of its components on the Atos Quantum Learning Machine [35] simulator. All the components were validated as functional, and are freely available at https://github.com/paper-codes/2021-IQW.

A. Evaluating the cryptanalytic effort on NIST post-quantum code based cryptosystems standardization candidates

We employ the gate counts obtained for our quantum ISD implementation to quantify the amount of computation required to solve ISD instances for cryptographic grade code parameters. In particular, we consider one parameter sets for each of the NIST security levels for both the Classic McEliece [36] cryptosystem, the finalist in the NIST Post quantum standardization effort among code based cryptosystems, and BIKE [37], one of the alternate candidates. The NIST security levels are defined as the computational effort to break one of the three AES variants, and therefore correspond to a computational effort of about 2^{128} (level 1), 2^{192} (level 3) or 2^{256} (level 5) AES encryptions.

In order to provide a quantification of the computational effort in terms of elementary gates, we require a translation

TABLE I
NUMBER OF REQUIRED QUANTUM GATES AS A FUNCTION OF THE PRANGE ISD PARAMETERS FOR THE DIFFERENT SUBCIRCUITS. EXCEPT FOR THE MULTI-CONTROLLED Z GATES, ALL THE FIGURES OF THE ORACLE PHASES SHOULD BE MULTIPLIED BY 2 TO TAKE INTO ACCOUNT THE UNCOMPUTATION STAGE. EACH MULTI-CONTROLLED Z, DENOTED AS $C^m(Z)$, INVOLVES $m+1$ QUBITS, m OF WHICH ACTS AS CONTROLLERS AND ONE AS TARGET.

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

TABLE II
NUMBER OF GATES AND QUBITS REQUIRED BY OUR ISD CIRCUIT DESIGN COMPARED TO THE ASYMPTOTICAL ESTIMATES OF [19] FOR TWO CODE-BASED CRYPTOSYSTEMS: MCELIECE AND BIKE. WE REPORT THE OVERALL DEPTH OF THE CIRCUIT FOR OUR PROPOSAL

Algorithm	**AES Equiv.**	**Code parameters**			**Grover Iterations**	**Grover gates**						**Total gates**	**Asymp. gates**	**Total qubits**	**Asymp. qubits**	**Total depth**
		n	k	t		X	CNOT	CCNOT	CSWAP	RY	CZ					
BIKE	128	$24,466$	$12,233$	134	2^{68}	2^{97}	2^{99}	2^{111}	2^{105}	2^{98}	2^{69}	2^{111}	2^{112}	2^{29}	2^{30}	2^{111}
	192	$49,318$	$24,659$	199	2^{100}	2^{132}	2^{133}	2^{146}	2^{139}	2^{133}	2^{101}	2^{146}	2^{148}	2^{31}	2^{32}	2^{146}
	256	$81,946$	$40,973$	264	2^{133}	2^{166}	2^{167}	2^{181}	2^{174}	2^{166}	2^{134}	2^{181}	2^{183}	2^{32}	2^{33}	2^{181}
McEliece	128	$3,488$	$2,720$	64	2^{72}	2^{94}	2^{96}	2^{104}	2^{101}	2^{96}	2^{73}	2^{104}	2^{108}	2^{22}	2^{24}	2^{104}
	192	$4,608$	$3,360$	96	2^{93}	2^{116}	2^{119}	2^{127}	2^{124}	2^{118}	2^{94}	2^{127}	2^{130}	2^{23}	2^{25}	2^{127}
	256	$8,192$	$6,528$	128	2^{151}	2^{175}	2^{178}	2^{186}	2^{183}	2^{177}	2^{152}	2^{186}	2^{190}	2^{24}	2^{26}	2^{186}

of the m-controlled Z into a sequence of elementary gates. We chose the approach proposed in [38], which requires m additional qubits and $2m$ CCNOT, plus 1 CZ. While an alternate approach which does not require extra qubits was in [39], it requires 2^m-2 CNOT gates and 2^m-1 gates $\in SU(2)$. Given that the value of m is in the hundreds for cryptographically relevant parameters, employing the approach of [39] would imply a significant cost increase.

A second parameter to set is the success probability of each execution of Grover's algorithm, as this in turn determines the number of its iterations. We choose to reach the probability closest to 1 to observe one of the $|x^*\rangle$ upon measurement. Indeed, having a probability close to 50% only halves the number of iterations, a factor that, given the total computational effort, does not have a huge impact on the overall figures.

Table II reports the number of gates, split by kind, and the number of qubits needed to build the ISD circuit according to our design, comparing them with the asymptotic results for the same algorithm presented in [19]. In [19], the author analyzes the feasibility of computing Prange's ISD on a quantum computer, reporting that it is expected to employ $n^{\mathcal{O}(1)}$ qubits, i.e, polynomial in n. Since this asymptotic bound does not allow to have a direct comparison, we assumed a sensible bound of $\mathcal{O}(n^2)$, as the method described in [19] represents the generator matrix of the code, and will also require, when concretized in a quantum circuit, ancillary qubits. The estimate for the number of gates in [19] is $\mathcal{O}(n^3)$ for a single Grover iteration. The number of iterations of Grover's algorithm is the same in both our realization and Bernstein's asymptotic estimates. Indeed, our function has $\binom{n-t}{k}$ inputs satisfying the constraints of the ISD, out of $\binom{n}{r}$ possible column permutations selecting an $r \times r$. This in turn leads to a number of iterations equal to $\frac{\pi}{4}\sqrt{\binom{n}{r}/0.2887\binom{n-t}{k}}$, considering the probability that the selected submatrix is non-singular.

We note that the asymptotic analysis in [19] does not consider the gate cost of the input preparation and diffusion stages in the gate counts. However, the preparation of the Dicke state, which is required to minimize the number of basis states fed to Grover's algorithm, and it is used also in the diffusion stage, is significantly more complex than the depth-1 stage of Hadamard gates required in the traditional version of Grover's algorithm. Also, the multi-controlled Z gate involved in the diffusion stage and its linear-depth decomposition in basic gates, requires a non-negligible amount of additional resources. The results in Table II allow us to state that the asymptotic estimates in [19] capture quite well the cost of computing concretely Prange's ISD on cryptographic grade parameters. However, we note that, despite the fact that [19] aims at providing a conservative estimate of the cost, not counting some required gates, our proposed realization is still smaller than the asymptotic estimates by a factor of $\approx 2^4$.

B. Comparing computational efforts with breaking AES

Since the bar chosen by NIST to evaluate the cryptographic strength of post-quantum code based cryptosystems is the computational effort required to break AES finding its key via Grover-based key search, we now evaluate our solution, comparing it with the current state-of-the-art solutions.

TABLE III
SUMMARY OF THE COST OF TRANSLATING THE GATE SET EMPLOYED IN DESCRIBING OUR QUANTUM ISD IN CLIFFORD+T GATES

Gate to translate	**Type & no. of gates for translation** H	CNOT	T	**Equivalent T-depth**
X	2	0	4	4
CCNOT	2	7	7	3
CSWAP	2	8	7	4
CZ	2	1	0	0

To this end, we need to express our solution in terms of the widely used Clifford+T gate set, defined by $\{\mathrm{H}, \mathrm{CNOT}, \mathrm{T} \equiv \sqrt[4]{\mathrm{Z}}\}$. The choice of this set of gates is justified by the fact that it can be efficiently implemented in a fault-tolerant manner [38], [40], and was thus adopted in the current state-of-the-art quantum AES implementation [9]. Among the gates in the Clifford+T set, T gates require extensively more resources to be implemented in a fault-tolerant fashion [41]. Common metrics for quantum circuit costs are the number of employed T gates and the so called T-depth, i.e., number of sequential stages in the circuit involving T gates. An additional measure used proposed in [25] is the product of the number of qubits times the depth of the circuit. This measure is intended to capture the computational effort if identity operators have to be applied when no gate acts on a qubit in a given instant.

We thus translate all the gates required in our solution as per Table I into combinations of Clifford+T gates. Table III reports the costs and T-depth of each one of such translations, for elementary gates. The only gate among the one employed by us for which there is no straightforward translation into a Clifford+T gate combination is the $\mathrm{R_y}$ gate, as it acts on a continuum. To this end, we consider the fact that an $\mathrm{R_y}$ gate can be expressed as a combination of 2 H gates, 4 T gates and a $\mathrm{R_z}$ gate [42], namely $\mathrm{R_y}(\theta) = \mathrm{TTHR_z}(\theta)\mathrm{HT^\dagger T^\dagger}$.

Different algorithms were proposed to synthesize $\mathrm{R_z}$ gate up to an arbitrary precision ϵ, trying to optimize the number of T, based on the consideration that T gate requires extensively more resources than any of the Clifford gates. In [42], the authors proposed an exact synthesis algorithm and showed how, 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 that, added to the previously shown decomposition, gives us a T-count of ≈ 153 for a single $\mathrm{R_y}$.

Table IV reports a comparison of the results between our approach, after translating all gates in the Clifford+T set, and a Grover based AES key search employing the state-of-the-art design in [9]. Since [9] only proposes a design for a quantum circuit implementing AES, we computed the cost of an entire Grover based AES key search circuit following the state-of-the-art design in [10]. To do so, we considered that the number of AES circuits required for the Grover oracle finding its key is 2, 2 and 3 for AES-128, -192, -256 respectively, and each one of them needs to be uncomputed, effectively doubling its gate count. Finally, the Grover oracle also needs a multi-controlled X gate, with the number of controls depending on the number of outputs of each AES box. For this reason, we need a $\mathrm{C}^{128\times 2}(\mathrm{X})$ gate, a $\mathrm{C}^{192\times 2}(\mathrm{X})$ gate and a $\mathrm{C}^{256\times 3}(\mathrm{X})$ gate for AES-128, -192, -256, respectively. By using the same decomposition shown in IV-A, we need an additional number of qubits equal to 256, 384 and 768 for the three AES key size. We report that [9] use a different definition of T-depth: they consider the T-depth to be the depth of the CCNOT gates. We adapted their definition to the T-depth definition given before, hence simply multiplying their depth by 3.

TABLE IV
COMPUTATIONAL EFFORT REQUIRED TO BREAK AES VIA GROVER-BASED KEY SEARCH USING THE AES IMPLEMENTATION IN [9] COMPARED TO THE COMPUTATIONAL EFFORT TO SOLVE THE ISD PROBLEM ON CLASSIC MCELIECE AND BIKE WITH OUR ISD IMPLEMENTATION.

Sec. Level	**Primitive**	**Qubits**	**T-Count**	**T-Depth**	**Depth·Qubits**
AES-128	AES [9]	2^{10}	2^{82}	2^{77}	2^{88}
	BIKE	2^{28}	2^{182}	2^{110}	2^{138}
	McEliece	2^{21}	2^{180}	2^{103}	2^{124}
AES-192	AES [9]	2^{11}	2^{114}	2^{109}	2^{120}
	BIKE	2^{30}	2^{251}	2^{145}	2^{176}
	McEliece	2^{23}	2^{224}	2^{126}	2^{149}
AES-256	AES [9]	2^{12}	2^{147}	2^{141}	2^{153}
	BIKE	2^{32}	2^{318}	2^{180}	2^{212}
	McEliece	2^{24}	2^{341}	2^{185}	2^{209}

We observe from our results that both Classic McEliece and BIKE 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 and BIKE are strongly conservative in terms of security.

V. CONCLUDING REMARKS

We presented a fully-detailed quantum circuit to speed up the execution of Prange's variant of ISD, together with detailed figures for the width, depth and gate size. We translated our generic gate set into the widely-employed Clifford+T gate set, the most promising choice for fault-tolerant computation, providing figures for the T-count and T-depth of our implementation. Our proposal shows how the NIST finalist cryptoscheme parameters are overdesigned with respect to the security margins required by NIST. In implementing our circuit, we proposed quantum circuits to compute matrix column permutations, the Hamming weight of a qubit string, and a Gaussian elimination of a binary matrix which may be of independent interest.

An interesting future research direction is to investigate the quantum efficiency of algorithmic improvements to Prange's ISD, evaluating approaches such as the one by Lee and Brickell, for which a partial accelerator is proposed in [43]. In particular, we note that our approach to Gaussian elimination is readily adapted to any prime field replacing the additions over $\mathbb{F}_2$ with additions over $\mathbb{F}_p$ such as the one described in [17].

To conclude, we highlight that our algorithm solves a general combinatorial problem, and it can be seen as a binary constraint satisfaction problem.

ACKNOWLEDGMENT

The research activity was partially funded by Atos Italia S.p.A. with a research grant on quantum computing.

References

[1] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, 1999. [Online]. Available: https://doi.org/10.1137/S0036144598347011

[2] National Institute of Standards and Technology, "Post-Quantum Cryptography Standardization process," https://nist.gov/pqcrypto, 2017.

[3] European Telecommunications Standards Institute (ETSI), "Quantum-Safe Cryptography," https://www.etsi.org/technologies/quantum-safe-cryptography, 2020.

[4] T. Kleinjung, C. Diem, A. K. Lenstra, C. Priplata, and C. Stahlke, "Computation of a 768-Bit Prime Field Discrete Logarithm," in *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, ser. Lecture Notes in Computer Science, J. Coron and J. B. Nielsen, Eds., vol. 10210, 2017, pp. 185–201. [Online]. Available: https://doi.org/10.1007/978-3-319-56620-7_7

[5] M. Delcourt, T. Kleinjung, A. K. Lenstra, S. Nath, D. Page, and N. P. Smart, "Using the Cloud to Determine Key Strengths - Triennial Update," *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 1221, 2018. [Online]. Available: https://eprint.iacr.org/2018/1221

[6] T. Kleinjung, J. W. Bos, A. K. Lenstra, D. A. Osvik, K. Aoki, S. Contini, J. Franke, E. Thomé, P. Jermini, M. Thiémard, P. C. Leyland, P. L. Montgomery, A. Timofeev, and H. Stockinger, "A heterogeneous computing environment to solve the 768-bit RSA challenge," *Clust. Comput.*, vol. 15, no. 1, pp. 53–68, 2012. [Online]. Available: https://doi.org/10.1007/s10586-010-0149-0

[7] J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery, "Solving a 112-bit prime elliptic curve discrete logarithm problem on game consoles using sloppy reduction," *Int. J. Appl. Cryptogr.*, vol. 2, no. 3, pp. 212–228, 2012. [Online]. Available: https://doi.org/10.1504/IJACT.2012.045590

[8] C. Costello, P. Longa, M. Naehrig, J. Renes, and F. Virdia, "Improved Classical Cryptanalysis of SIKE in Practice," in *Public-Key Cryptography - PKC 2020 - 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, Eds., vol. 12111. Springer, 2020, pp. 505–534. [Online]. Available: https://doi.org/10.1007/978-3-030-45388-6_18

[9] J. Zou, Z. Wei, S. Sun, X. Liu, and W. Wu, "Quantum Circuit Implementations of AES with Fewer Qubits," in *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, S. Moriai and H. Wang, Eds., vol. 12492. Springer, 2020, pp. 697–726. [Online]. Available: https://doi.org/10.1007/978-3-030-64834-3_24

[10] S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, "Implementing Grover Oracles for Quantum Key Search on AES and LowMC," in *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, A. Canteaut and Y. Ishai, Eds., vol. 12106. Springer, 2020, pp. 280–310. [Online]. Available: https://doi.org/10.1007/978-3-030-45724-2{_}10

[11] R. Anand, A. Maitra, and S. Mukhopadhyay, "Grover on $SIMON$," *Quantum Inf. Process.*, vol. 19, no. 9, p. 340, 2020. [Online]. Available: https://doi.org/10.1007/s11128-020-02844-w

[12] ——, "Evaluation of Quantum Cryptanalysis on SPECK," in *Progress in Cryptology - INDOCRYPT 2020 - 21st International Conference on Cryptology in India, Bangalore, India, December 13-16, 2020, Proceedings*, ser. Lecture Notes in Computer Science, K. Bhargavan, E. Oswald, and M. Prabhakaran, Eds., vol. 12578. Springer, 2020, pp. 395–413. [Online]. Available: https://doi.org/10.1007/978-3-030-65277-7_18

[13] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," *Quantum – the open journal for quantum science*, vol. 5, p. 433, Apr 2021. [Online]. Available: http://dx.doi.org/10.22331/q-2021-04-15-433

[14] N. Kunihiro, "Quantum Factoring Algorithm: Resource Estimation and Survey of Experiments," in *International Symposium on Mathematics, Quantum Theory, and Cryptography*, T. Takagi, M. Wakayama, K. Tanaka, N. Kunihiro, K. Kimoto, and Y. Ikematsu, Eds. Singapore: Springer Singapore, 2021, pp. 39–55.

[15] M. Wroński, "Solving discrete logarithm problem over prime fields using quantum annealing and $\frac{n^3}{2}$ logical qubits," Cryptology ePrint Archive, Report 2021/527, 2021, https://eprint.iacr.org/2021/527.

[16] T. Häner, S. Jaques, M. Naehrig, M. Roetteler, and M. Soeken, "Improved Quantum Circuits for Elliptic Curve Discrete Logarithms," in *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, ser. Lecture Notes in Computer Science, J. Ding and J. Tillich, Eds., vol. 12100. Springer, 2020, pp. 425–444. [Online]. Available: https://doi.org/10.1007/978-3-030-44223-1_23

[17] M. Roetteler, M. Naehrig, K. M. Svore, and K. E. Lauter, "Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms," in *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, ser. Lecture Notes in Computer Science, T. Takagi and T. Peyrin, Eds., vol. 10625. Springer, 2017, pp. 241–270. [Online]. Available: https://doi.org/10.1007/978-3-319-70697-9_9

[18] G. Banegas, D. J. Bernstein, I. van Hoof, and T. Lange, "Concrete quantum cryptanalysis of binary elliptic curves," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2021, no. 1, pp. 451–472, 2021. [Online]. Available: https://doi.org/10.46586/tches.v2021.i1.451-472

[19] D. J. Bernstein, "Grover vs. McEliece," in *Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings*, ser. Lecture Notes in Computer Science, N. Sendrier, Ed., vol. 6061. Springer, 2010, pp. 73–80. [Online]. Available: https://doi.org/10.1007/978-3-642-12929-2_6

[20] G. Kachigar and J. Tillich, "Quantum Information Set Decoding Algorithms," in *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, ser. Lecture Notes in Computer Science, T. Lange and T. Takagi, Eds., vol. 10346. Springer, 2017, pp. 69–89. [Online]. Available: https://doi.org/10.1007/978-3-319-59879-6_5

[21] E. Kirshanova, "Improved Quantum Information Set Decoding," in *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*, ser. Lecture Notes in Computer Science, T. Lange and R. Steinwandt, Eds., vol. 10786. Springer, 2018, pp. 507–527. [Online]. Available: revisedversionat\url{https://crypto-kantiana.com/elena.kirshanova/Papers/quantumISD.pdf}

[22] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, "A Finite Regime Analysis of Information Set Decoding Algorithms," *Algorithms*, vol. 12, no. 10, p. 209, 2019. [Online]. Available: https://doi.org/10.3390/a12100209

[23] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems (Corresp.)," *IEEE Trans. Information Theory*, vol. 24, no. 3, pp. 384–386, 1978. [Online]. Available: https://doi.org/10.1109/TIT.1978.1055873

[24] M. R. Albrecht, V. Gheorghiu, E. W. Postlethwaite, and J. M. Schanck, "Estimating Quantum Speedups for Lattice Sieves," in *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, S. Moriai and H. Wang, Eds., vol. 12492. Springer, 2020, pp. 583–613. [Online]. Available: https://doi.org/10.1007/978-3-030-64834-3_20

[25] S. Jaques and J. 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*, ser. Lecture Notes in Computer Science, A. Boldyreva and D. Micciancio, Eds., vol. 11692. Springer, 2019, pp. 32–61. [Online]. Available: https://doi.org/10.1007/978-3-030-26948-7_2

[26] E. Prange, "The use of information sets in decoding cyclic codes," *IRE Trans. Information Theory*, vol. 8, no. 5, pp. 5–9, 1962. [Online]. Available: https://doi.org/10.1109/TIT.1962.1057777

[27] L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in *Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, 1996, pp. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866

[28] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quan-

tum searching," *Fortschritte der Physik: Progress of Physics*, vol. 46, no. 4-5, pp. 493–505, 1998.

[29] D. E. Knuth, *The Art of Computer Programming, , Volume III, 2nd Edition*. Addison-Wesley, 1998. [Online]. Available: https://www.worldcat.org/oclc/312994415

[30] M. Ajtai, J. Komlós, and E. Szemerédi, "An O(n log n) Sorting Network," in *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, and J. I. Seiferas, Eds. ACM, 1983, pp. 1–9. [Online]. Available: https://doi.org/10.1145/800061.808726

[31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[32] C. S. Mukherjee, S. Maitra, V. Gaurav, and D. Roy, "Preparing Dicke States on a Quantum Computer," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–17, 2020.

[33] A. Bärtschi and S. J. Eidenbenz, "Deterministic Preparation of Dicke States," in *Fundamentals of Computation Theory - 22nd International Symposium, FCT 2019, Copenhagen, Denmark, August 12-14, 2019, Proceedings*, ser. Lecture Notes in Computer Science, L. A. Gasieniec, J. Jansson, and C. Levcopoulos, Eds., vol. 11651. Springer, 2019, pp. 126–139.

[34] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton, "A New Quantum Ripple-Carry Addition Circuit," *arXiv preprint quant-ph/0410184*, 2004.

[35] Atos, "Quantum Learning Machine," 2019. [Online]. Available: \url{https://atos.net/en/solutions/quantum-learning-machine}

[36] D. J. Bernstein, T. Chou, T. Lange, I. von Maurich, R. Misoczki, R. Niederhagen, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, and W. Wang, "Classic McEliece: conservative code-based cryptography," https://classic.mceliece.org/nist/mceliece-20201010.pdf, 2020.

[37] N. Aragon, P. S. L. M. Barreto, S. Bettaieb, L. Bidoux, O. Blazy *et al.*, "BIKE: Bit Flipping Key Encapsulation," https://bikesuite.org.

[38] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[39] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary Gates for Quantum Computation," *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, Nov. 1995.

[40] P. O. Boykin, T. Mor, M. Pulver, V. P. Roychowdhury, and F. Vatan, "A New Universal and Fault-Tolerant Quantum Basis," *Inf. Process. Lett.*, vol. 75, no. 3, pp. 101–107, 2000. [Online]. Available: https://doi.org/10.1016/S0020-0190(00)00084-3

[41] M. Amy, D. Maslov, M. Mosca, and M. Roetteler, "A Meet-in-the-Middle Algorithm for Fast Synthesis of Depth-Optimal Quantum Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 6, pp. 818–830, 2013. [Online]. Available: https://doi.org/10.1109/TCAD.2013.2244643

[42] V. Kliuchnikov, D. Maslov, and M. Mosca, "Practical Approximation of Single-Qubit Unitaries by Single-Qubit Quantum Clifford and T Circuits," *IEEE Trans. Computers*, vol. 65, no. 1, pp. 161–172, 2016. [Online]. Available: \url{https://doi.org/10.1109/TC.2015.2409842}

[43] S. Perriello, A. Barenghi, and G. Pelosi, "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 2021, Canterbury, Great Britain (online), September 6-9, 2021. International Workshop on Post-quantum Cryptography for Secure Communications (PQC-SC).*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, Cham, 2021. Springer, 2021.