Journal of Physics A:
Mathematical and
Theoretical

PAPER • **OPEN ACCESS**

Homomorphic encryption of the k=2 Bernstein–Vazirani algorithm

To cite this article: Pablo Fernández and Miguel A Martin-Delgado 2024 *J. Phys. A: Math. Theor.* **57** 365301

OPEN ACCESS
IOP Publishing
Journal of Physics A: Mathematical and Theoretical
J. Phys. A: Math. Theor. **57** (2024) 365301 (27pp)
https://doi.org/10.1088/1751-8121/ad6c04

Homomorphic encryption of the *k*=2 Bernstein–Vazirani algorithm

Pablo Fernández[1,*] **and Miguel A Martin-Delgado**[1,2]

[1] Departamento de Física Teórica, Universidad Complutense, 28040 Madrid, Spain
[2] CCS-Center for Computational Simulation, Campus de Montegancedo UPM, 28660 Boadilla del Monte, Madrid, Spain

E-mail: pabfer23@ucm.es and mardel@ucm.es

Received 24 January 2024; revised 1 August 2024
Accepted for publication 6 August 2024
Published 22 August 2024

Abstract

We introduce a class of circuits that solve a particular case of the Bernstein–Vazirani recursive problem for second-level recursion. This class of circuits allows for the implementation of the oracle using a number of T-gates that grows linearly with the number of qubits in the problem. We find an application of this scheme to quantum homomorphic encryption (QHE), which is an important cryptographic technology useful for delegated quantum computing, allowing a remote server to perform quantum computations on encrypted quantum data, so that the server cannot know anything about the client's data. Liang's QHE schemes are suitable for circuits with a polynomial number of gates $T/T^{\dagger}$. Thus, the simplified circuits we have constructed can be evaluated homomorphically in an efficient manner.

Keywords: quantum information, quantum computation, quantum communication, homomorphic encryption

1. Introduction

Quantum computing has gained great interest in recent decades due to the potential advantages it could offer compared to classical computing. Using quantum phenomena such as superposition and entanglement, quantum algorithms, including hybrid ones, are being sought that are faster than their corresponding better known classical version. Initial examples of such

* Author to whom any correspondence should be addressed.

algorithms were Shor's factorization algorithm [1] or Grover's search algorithm [2]. The search continues as part of developments in current quantum technologies.

The first hint of the possible advantage of quantum computers came from Deutsch [3] and Deutsch–Jozsa [4]. Soon after, Bernstein and Vazirani introduced a problem in which the query complexity of the quantum solution was better than what was possible by any classical computer. The Bernstein–Vazirani (BV) [5] algorithm is the first quantum algorithm that showed the advantages that a quantum computer could have over a classical one. The nonrecursive version of the problem improved the query complexity from $O(n)$, which was the best a classical computer could achieve, to $O(1)$ for the quantum computer. This constitutes a polynomial speed-up. This algorithm has been used as the basis of different applications. As an example of an application in cryptography, the BV algorithm has been used to find the linear structures of a function in order to attack block ciphers [6]. A variant of the algorithm was studied by Cross *et al* [7] for quantum learning robust against noise.

The nonrecursive BV algorithm is classically tractable because the hardness of the classical problem scales only polynomial in the problem size. Bernstein and Vazirani managed to find a version of the problem which is classically intractable, but can be solved on a quantum computer using polynomial resources. This problem is known as the recursive BV problem. This problem has a superpolynomial query complexity in the classical realm and only a polynomial one in the corresponding quantum version. Therefore the recursive BV algorithm has a super polynomial speed-up compared to the best classical algorithms [8]. This problem is one of the two building blocks of this paper. The other is quantum homomorphic encryption (QHE).

Homomorphic encryption allows a client to encrypt data, send it to a server that will operate on this encrypted data and then returns the data once all the operations have finished so the decryption of data can be done. This way the server can never learn anything about the actual data it is operating with. Since the first classical fully homomorphic encryption scheme (FHE) created by Gentry [9], many other classical schemes have been proposed and perfected. The security of these schemes is based on the difficulty of certain mathematical problems, such as ideal lattices [9] or the learning with errors problem [10].

In terms of quantum computing, we have now seen the development of real, incipient quantum computers. Since most quantum computers in the foreseeable future will be accessible through the cloud, QHE schemes have been developed to ensure their security. QHE allows a remote server to perform some quantum circuit QC on encrypted quantum data $Enc(\rho)$ provided by a client, and then the client can decrypt the server's output and obtain the result QC(ρ). The security of these schemes depend on the fundamental properties of quantum mechanics instead of the difficulty of mathematical problems.

These schemes can be constructed with interaction between client and server or without it (interaction in this context means that the client and server communicate during the execution of the circuit). Some quantum FHE (QFHE) schemes where interaction is allowed have been constructed, like Liang's [11] scheme, where the total amount of interactions is the same as the number of T gates in the circuit. However, we are more interested in schemes without interaction between client and server.

The research of QHE schemes began in 2012. Rohde *et al* [12] proposed a symmetric-key QHE scheme, which allows quantum random walk on encrypted quantum data. This protocol has been realized recently by Zeuner *et al* in [13]. This scheme is not $\mathcal{F}$-homomorphic, which means it can not perform universal quantum computation on homomorphic encrypted data (it only allows the homomorphic evaluation of certain circuits).

Tan *et al* [14] proposed a QHE scheme which can perform extensive quantum computation on encrypted data, but it can not provide security in a cryptographic sense. Yu *et al* proved a no-go result in [15], which states that any QFHE with perfect security must produce exponential storage overhead.

Lai and Chung [16] proved an enhanced no-go result, which states that it is impossible to construct a non-interactive and information theoretically secure (ITS) QFHE scheme. This means that if non-interaction is required, the best security a QFHE scheme could hope to achieve is computational security (security based on hard mathematical problems just like classical cryptography). Lai and Chung proceeded to construct a non interactive ITS QHE scheme with compactness, which means that the complexity of the decryption procedure does not depend on the operations of the evaluated quantum circuit. However, due to the no-go result, it is not $\mathcal{F}$-homomorphic, so it is just partially homomorphic.

Attempts have been made to study possible schemes with less stringent conditions. The relevant properties of QHE are non-interaction, $\mathcal{F}$-homomorphism, compactness and perfect security. Due to the no-go result explained above, it is impossible to have all properties at the same time. If interaction is allowed, it is possible to construct a QHE with perfect security, such as the one already mentioned by Liang [11]. Broadbent and Jeffery [17] constructed two non-interactive quantum homomorphic schemes, combining quantum one-time pad (QOTP) and classical FHE, so they downgrade from perfect security to security bounded by classical homomorphic encryption schemes.

Liang [18] constructed two non-interactive and perfectly secure QHE schemes, which are $\mathcal{F}$-homomorphic but not compact. In fact, these schemes are quasi-compact (the decryption procedure has complexity that scales sublinearly in the size of evaluated circuit) as defined by Broadbent and Jeffery in [17]. The first scheme in [17] is known as EPR, with quasi-compactness, $\mathcal{F}$-homomorphism, non-interaction and computational security. EPR was proved to be M^2-quasi-compact, where M is the number of T-gates in an evaluated circuit. Both Broadbent's and Liang's schemes make use of Bell states and quantum measurements. The schemes are not comparable regarding the overall security, since the scheme in [17] is constructed with circuit privacy as a partial goal, while Liang's [18] does not address the issue of circuit privacy. Also one of Liang's schemes, VGT, is M-quasi-compact (the complexity of the decryption procedure scales with the number of T-gates), so it is better in that regard. As these schemes are quasi-compact, they do not contradict the no go-result.

The importance of Liang's schemes [18] is that even though they are quasi-compact, they allow the homomorphic evaluation of any quantum circuit with low $T/T^\dagger$ gate complexity with perfect data security. The decryption procedure would be inefficient for circuits containing an exponential number of $T/T^\dagger$ gates, but it is suitable for circuits with polynomial number of $T/T^\dagger$gates. Liang's schemes have been used to implement a cyphertext retrieval scheme based on the Grover algorithm in Gong *et al* [19].

In the following list we briefly summarize some of our main results:

1. Reduction of the number of T-gates needed: in the recursive BV algorithm the number of T-gates needed to solve the problem is not necessarily a polynomial in general. The circuits we have introduced, named T-linear circuits with reduced circuit complexity (RCC), simplify the number of T-gates needed to construct the oracle by making it grow linearly with the number of qubits in the problem. In this work we have characterized these circuits in great detail, studying all the possible cases in which they can be applied and their corresponding constraints.
2. Efficient homomorphic encryption: the number of T gates in T-linear circuits grows linearly. Therefore, we have shown that they constitute a perfect type of quantum circuits that can be

homomorphically evaluated with perfect security and non-interaction in an efficient manner using Liang's schemes.
3. Extension of the results for more quantum algorithms and an arbitrary recursion: we have also shown a possible extension of the above results in a particular case that works for any arbitrary level of the recursion. These circuits can also be homomorphically evaluated efficiently, although not as efficiently as T-linear circuits. We believe that our work can also lead to the study of homomorphic implementation of more quantum algorithms, using Liang's schemes. There may be cases similar to T-linear circuits where the T-gate complexity is low for more quantum algorithms.

The paper is organized as follows. In section 2 T-linear circuits with RCC will be defined and characterized. The homomorphic implementation of the recursive BV problem is also discussed in this section, along some extended results. Finally, the conclusions are summarized in section 3. In appendix A Liang's QHE scheme is reviewed. In appendix B the nonrecursive and the recursive version of the BV algorithm are reviewed. From this point onwards we assume the reader is familiar with the recursive BV algorithm.

2. T-linear circuits with RCC

The circuit that solves the recursive BV problem for $k=2$ is shown in appendix B, figure 14. The oracle U_s for the recursive BV problem for $k=2$ in the most general case is made of multi-controlled CNOT gates where n control qubits are used for the first register and one for the second. As an example, consider the recursive BV problem for $k=2$ where the secret bit string is $s=11$ and the values of $g(s_{\text{in}})$ are chosen as: $g(00)=0$, $g(01)=g(10)=g(11)=1$. Then the choice of values $s_{00}=s_{11}=00$, $s_{01}=01$, $s_{10}=10$ is compatible with the fact that $g(s_{x_1})=s\cdot x_1$. The circuit that solves this problem is represented graphically in figure 1, where the oracle U_s is implemented using multi-controlled CNOT gates.

However in certain situations U_s can be implemented just using Toffoli gates. We will define the circuits where this is possible in the following section.

2.1. Definition and characterization

We will begin this section defining T-linear circuits with RCC.

Definition 1. T-linear circuits with RCC are quantum circuits that solve the BV recursive problem when the oracle U_s can be constructed just with n Toffoli gates at most and the oracle G just with CNOT gates.

Here n is the same number as the qubits contained in one of the registers of the algorithm and also the number of bits needed to represent s. We want to remark that the circuits allowed here are those composed of H gates in some layers, a layer of Z gates after the first layer of H gates, Toffoli and X gates for the oracle U_s and CNOT gates for G. Recall that the Toffoli gate performs the mapping $\text{TOFF}|a,b,c\rangle=|a,b,c\oplus(a\wedge b)\rangle$. We will now discuss for which values of $g(s_{\text{in}})$ and s_{x_1} a T-linear circuit with RCC can be constructed.

Ultimately, the question we want to answer is: given a mapping from $x_1\in\{0,1\}^n$ to $s_{x_1}\in\{0,1\}^n$ and a function $g:\{0,1\}^n\to\{0,1\}$ such that $g(s_{x_1})=s\cdot x_1$ for some fixed string $s\in\{0,1\}^n$. Can the mapping

$$|x_1\rangle_1|x_2\rangle_2|y\rangle\mapsto|x_1\rangle_1|x_2\rangle_2|y\oplus(s_{x_1}\cdot x_2)\rangle\forall x_1,x_2\in\{0,1\}^n,y\in\{0,1\}, \tag{1}$$

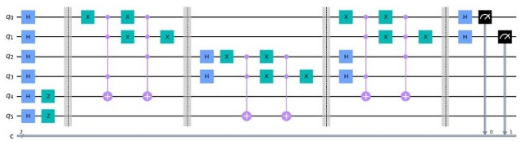

Figure 1. Recursive BV circuit for two recursions for $s = 11$, represented in Qiskit.

be performed with at most n Toffoli gates?

We will start with $g(s_{\text{in}})$. In the BV problem any value for $g(s_{\text{in}})$ can be chosen as long as it equals 0 and 1 for a least one of its inputs. Let g be of the form $g(s_{\text{in}}) = s \cdot s_{\text{in}}$ and denote the Hamming weight of a string by $|s|$. The oracle G implementing $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus g(x)\rangle = |x\rangle|y \oplus (\sum_{i=1}^{n} s_i x_i)\rangle$ in equation (B11) can be performed by a sequence of $|s|$ CNOTs, each one controlled on the qubit $|x_j\rangle, j \in \{\{1,\ldots,n\} : s_j = 1\}$, and targeting register $|y\rangle$. If the target $|y\rangle$ is the state $|-\rangle$, we obtain the desired phase kickback. Therefore if the values of $g(s_{\text{in}})$ fulfil $g(s_{\text{in}}) = s \cdot s_{\text{in}}$ the oracle G can be implemented only using CNOTs. All T-linear circuit with RCC fulfil this condition. This property of G is also valid for any recursion k because G only acts on one register at a time.

Choosing $g(s_{\text{in}}) = s \cdot s_{\text{in}}$ restricts the circuits that can be studied. However there is a good reason to restrict the function this way besides simplifying the problem, which is the fact that the homomorphic evaluation of CNOT gates does not increase the complexity of the decryption procedure. Using any other function g that is not balanced would make implementing G using just CNOT gates impossible. This means that G would be implemented using multi-controlled CNOT gates with n control qubits and since these gates contain a higher number of $T/T^{\dagger}$ gates than the usual Toffoli gates, the decryption process of the QHE scheme would be less efficient. Our main focus with T-linear circuits with RCC is reducing the complexity of this decryption procedure as much as possible.

Then by choosing this g the problem narrows down to the mapping $x_1 \mapsto s_{x_1}$ with the restriction $s \cdot x_1 = s \cdot s_{x_1}$ for some string $s \in \{0,1\}^n$. Let $f\colon \{0,1\}^n \to \{0,1\}^n$ be such mapping so we have $s \cdot x_1 = s \cdot f(x_1)$. The mapping from equation (1) can be generically performed as

$$|x_1\rangle_1|x_2\rangle_2|y\rangle_{\mathtt{T}} \overset{1}{\mapsto} |f(x_1)\rangle_1|x_2\rangle_2|y\rangle_{\mathtt{T}} \overset{2}{\mapsto} |f(x_1)\rangle_1|x_2\rangle_2|y \oplus (f(x_1)\cdot x_2)\rangle_{\mathtt{T}} \overset{3}{\mapsto} |x_1\rangle_1|x_2\rangle_2|y \oplus (f(x_1)\cdot x_2)\rangle_{\mathtt{T}}. \tag{2}$$

Step 2 is simply the application of n Toffoli gates. The j-th Toffoli gate is controlled using the j-th qubit of registers 1 and 2 and it targets register $\mathtt{T}$, for $j \in \{1,\ldots,n\}$. The issue is regarding Steps 1 and 3. Since each step must be unitary, f must be a bijection. The question is then implementing this map f using just single-qubit gates. The reason for this restriction is that we are looking to implement U_s only with Toffoli gates that activate with either 0 or 1. Then the only single-qubit gates that make sense in the context of the function f are X gates.

The different mappings f that can be constructed are all valid T-linear circuits with RCC. The simplest mapping f that satisfies $s \cdot x_1 = s \cdot f(x_1)$ is $f(x_1) = x_1$, which is equivalent to simply applying n Toffoli gates. We summarize this result in lemma 1:

Lemma 1. *A T-linear circuit with RCC can be constructed for $s_{x_1} = x_1$.*

Proof. As it has been explained, finding a T-linear circuit amounts to choosing a f that fulfils the condition $s \cdot x_1 = s \cdot f(x_1)$. In this case $f(x_1) = x_1$ fulfils the condition simply by substitution. □

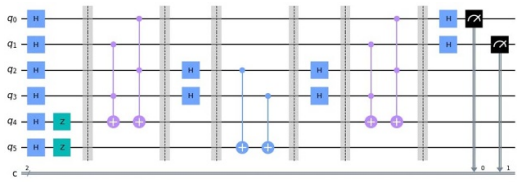

Figure 2. T-linear circuit with RCC for $s = 11$, $s_{00} = 00$, $s_{01} = 01$, $s_{10} = 10$ and $s_{11} = 11$, $g(00) = g(11) = 0$ and $g(01) = g(10) = 1$. This circuit is obtained from lemma 1.

As an example of this type of circuits, we solve a 2 qubits circuit, where $s = 11$. Since $g(s_{\text{in}})$ fulfils $g(s_{\text{in}}) = s \cdot s_{\text{in}}$, we have $g(00) = g(11) = 0$ and $g(01) = g(10) = 1$. For s_{x_1}, we have $s_{x_1} = x_1$ so: $s_{00} = 00$, $s_{01} = 01$, $s_{10} = 10$, $s_{11} = 11$. Since these values were obtained from lemma 1 a T-linear circuit with RCC can be constructed, so this circuit can be implemented using just $n = 2$ Toffoli gates for the oracle U_s and CNOTs for the oracle G. The circuit that completes this process represented in qiskit is shown in figure 2. The qubits are denoted by q_0, q_1,..., q_{n-1}, where q_0 refers to the first one, q_1 to the second one and q_{n-1} to the nth one. In this example, the first register has two qubits, q_0 and q_1, and the second register has qubits q_2 and q_3.

We start by applying U_s to the superposition of states of x_1 and x_2 in equation (B8). Since $s_{01} = 01$, the second qubit of each register, q_1 and q_3, are used as the control qubits for a Toffoli gate whose target is the first extra qubit in the $|-\rangle$ state. This first Toffoli is denoted as $\text{TOFF}_{q_1,q_3,q_4}$ where q_1 and q_3 are control qubits and q_4 is the target qubit. A phase kickback will occur when both control qubits are in the $|1\rangle$ state and the target qubit is in the $|-\rangle$ state: $\text{TOFF}|1,1,-\rangle = |1,1\rangle \left(\frac{|0\oplus(1\wedge 1)\rangle - |1\oplus(1\wedge 1)\rangle}{\sqrt{2}} \right) = -|1,1,-\rangle$. Applying this Toffoli gives us the state (see figure 2):

$$
\begin{aligned}
&\text{TOFF}_{q_1,q_3,q_4}\Bigg[\frac{1}{4}\Big[|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\
&\quad + |01\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\
&\quad + |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\
&\quad + |11\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2)\Big] \otimes |-\rangle\Bigg] \\
&= \frac{1}{4}\Big[|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\
&\quad + |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2) \\
&\quad + |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\
&\quad + |11\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2)\Big] \otimes |-\rangle.
\end{aligned}
$$

The next s_{x_1} is $s_{10} = 10$, so the first qubit of each register is used as the control qubit for a Toffoli gate. This means the control qubits are q_0 and q_2. The second Toffoli transforms the state into:

$$
\begin{aligned}
&\text{TOFF}_{q_0,q_2,q_4}\Bigg[\frac{1}{4}\Big[|00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\
&\quad + |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2)
\end{aligned}
$$

$$
\begin{aligned}
&+|10\rangle_1\otimes(|00\rangle_2+|01\rangle_2+|10\rangle_2+|11\rangle_2)\\
&+|11\rangle_1\otimes(|00\rangle_2-|01\rangle_2+|10\rangle_2-|11\rangle_2)\Big]\otimes|-\rangle\Big]\\
=\frac{1}{4}\Big[&|00\rangle_1\otimes(|00\rangle_2+|01\rangle_2+|10\rangle_2+|11\rangle_2)\\
&+|01\rangle_1\otimes(|00\rangle_2-|01\rangle_2+|10\rangle_2-|11\rangle_2)\\
&+|10\rangle_1\otimes(|00\rangle_2+|01\rangle_2-|10\rangle_2-|11\rangle_2)\\
&+|11\rangle_1\otimes(|00\rangle_2-|01\rangle_2-|10\rangle_2+|11\rangle_2)\Big]\otimes|-\rangle.
\end{aligned}
$$

The combination of both Toffolis also implements $s_{11}=11$, because s_{11} has two 1 s. Applying two H gates to the second register transforms the state into:

$$
\begin{aligned}
&I\otimes H^2\Big[\frac{1}{4}\Big[|00\rangle_1\otimes(|00\rangle_2+|01\rangle_2+|10\rangle_2+|11\rangle_2)\\
&+|01\rangle_1\otimes(|00\rangle_2-|01\rangle_2+|10\rangle_2-|11\rangle_2)\\
&+|10\rangle_1\otimes(|00\rangle_2+|01\rangle_2-|10\rangle_2-|11\rangle_2)\\
&+|11\rangle_1\otimes(|00\rangle_2-|01\rangle_2-|10\rangle_2+|11\rangle_2)\Big]\otimes|-\rangle\Big]\\
=\frac{1}{2}\Big[&|00\rangle_1\otimes|00\rangle_2+|01\rangle_1\otimes|01\rangle_2\\
&+|10\rangle_1\otimes|10\rangle_2+|11\rangle_1\otimes|11\rangle_2\Big]\otimes|-\rangle.
\end{aligned}
$$

Then, we apply G using two CNOTs since $s=11$, which gives us the state (see figure 2):

$$
\begin{aligned}
&I\otimes G\Big[\frac{1}{2}\Big[|00\rangle_1\otimes|00\rangle_2+|01\rangle_1\otimes|01\rangle_2\\
&+|10\rangle_1\otimes|10\rangle_2+|11\rangle_1\otimes|11\rangle_2\Big]\otimes|-\rangle\otimes|-\rangle\Big]\\
=\frac{1}{2}\Big[&|00\rangle_1\otimes|00\rangle_2-|01\rangle_1\otimes|01\rangle_2\\
&-|10\rangle_1\otimes|10\rangle_2+|11\rangle_1\otimes|11\rangle_2\Big]\otimes|-\rangle\otimes|-\rangle.
\end{aligned}
$$

Since $g(01)=g(10)=1$ a phase kickback is applied to the states $|01\rangle_2$ and $|10\rangle_2$. Now we just need to apply H^2 and U_s to the second register again. After two H gates:

$$
\begin{aligned}
&I\otimes H^2\Big[\frac{1}{2}\Big[|00\rangle_1\otimes|00\rangle_2-|01\rangle_1\otimes|01\rangle_2\\
&-|10\rangle_1\otimes|10\rangle_2+|11\rangle_1\otimes|11\rangle_2\Big]\otimes|-\rangle\otimes|-\rangle\Big]\\
=\frac{1}{4}\Big[&|00\rangle_1\otimes(|00\rangle_2+|01\rangle_2+|10\rangle_2+|11\rangle_2)\\
&-|01\rangle_1\otimes(|00\rangle_2-|01\rangle_2+|10\rangle_2-|11\rangle_2)\\
&-|10\rangle_1\otimes(|00\rangle_2+|01\rangle_2-|10\rangle_2-|11\rangle_2)\\
&+|11\rangle_1\otimes(|00\rangle_2-|01\rangle_2-|10\rangle_2+|11\rangle_2)\Big]\otimes|-\rangle\otimes|-\rangle.
\end{aligned}
$$

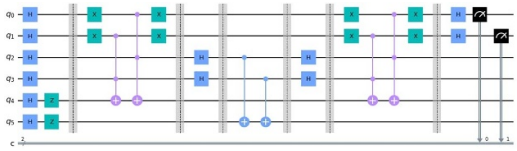

Figure 3. T-linear circuit with RCC for $s = 11$, $s_{00} = 11$, $s_{01} = 10$, $s_{10} = 01$, $s_{11} = 00$ and $g(00) = g(11) = 0$, $g(01) = g(10) = 1$. This circuit is obtained from lemma 2.

After U_s (the same two Toffolis as applied before):

$$
\begin{aligned}
U_s \Bigg[\frac{1}{4} \Big[& |00\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \\
& - |01\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 + |10\rangle_2 - |11\rangle_2) \\
& - |10\rangle_1 \otimes (|00\rangle_2 + |01\rangle_2 - |10\rangle_2 - |11\rangle_2) \\
& + |11\rangle_1 \otimes (|00\rangle_2 - |01\rangle_2 - |10\rangle_2 + |11\rangle_2) \Big] \otimes |-\rangle \otimes |-\rangle \Bigg] \\
= \frac{1}{4} \Big[& (|00\rangle_1 - |01\rangle_1 - |10\rangle_1 + |11\rangle_1) \\
& \otimes (|00\rangle_2 + |01\rangle_2 + |10\rangle_2 + |11\rangle_2) \Big] \otimes |-\rangle \otimes |-\rangle .
\end{aligned}
$$

As always, applying H gates to the first register and measuring it will give us the state $|11\rangle_1$ which is the value of s.

Another mapping f that satisfies $s \cdot x_1 = s \cdot f(x_1)$ is $f(x_1) = \overline{x_1}$ with the restriction that $s \cdot x_1 = s \cdot \overline{x_1} \implies |s|$ is even, where $\overline{x_1}$ is the conjugate of x_1 . This is equivalent to flipping all the bits using X gates that surround n Toffoli gates or simply using n Toffoli gates that activate with 0 instead of 1. We will refer to this type of Toffoli gate by negated Toffoli gate.

We summarize this result in lemma 2:

Lemma 2. *A T-linear circuit with RCC can be constructed for $s_{x_1} = \overline{x_1}$ provided $|s|$ is even.*

Proof. For $f(x_1) = \overline{x_1}$ we have $s \cdot \overline{x_1} = (\overline{x_1})_p \oplus \cdots \oplus (\overline{x_1})_q$ where each term of the sum has a corresponding value of $s_j = 1$ and $(\overline{x_1})_p$ is just the p-th bit of $\overline{x_1}$. If $|s|$ is even, the previous sum contains an even number of terms. The equation $\overline{a} \oplus \overline{b} = a \oplus b$ holds for arbitrary values of single bits a and b. Applying $\overline{a} \oplus \overline{b} = a \oplus b$ for every pair of bits in the previous sum and taking into account that no bit is left unpaired as the sum contains an even number of terms we have $(\overline{x_1})_p \oplus \cdots \oplus (\overline{x_1})_q = (x_1)_p \oplus \cdots \oplus (x_1)_q$. Reintroducing the same s in the previous equation we arrive at $s \cdot \overline{x_1} = (\overline{x_1})_p \oplus \cdots \oplus (\overline{x_1})_q = (x_1)_p \oplus \cdots \oplus (x_1)_q = s \cdot x_1$ and so the equation $s \cdot \overline{x_1} = s \cdot x_1$ holds. Therefore $s_{x_1} = f(x_1) = \overline{x_1}$ fulfils $s \cdot x_1 = s \cdot f(x_1)$ if $|s|$ is even. □

As an example, if $s = 11$, then $s_{00} = 11$, $s_{01} = 10$, $s_{10} = 01$, $s_{11} = 00$ and the values of $g(s_{\text{in}})$ would be $g(00) = g(11) = 0$, $g(01) = g(10) = 1$. The circuit that solves the BV problem for these values is represented in figure 3. As it was explained, the control qubits of the first register of the Toffolis are surrounded by X gates, so they activate when their state is a 0 instead of a 1.

More generally, we can have $f(x_1) = x_1 \oplus z$ for any string $z \in \{0,1\}^n$. In this case $s \cdot x_1 = s \cdot (x_1 \oplus z) \implies |s \wedge z|$ is even, where $s \wedge z$ is the bitwise AND operation between s and z. This is equivalent to substituting some of the n Toffoli gates by the same amount of negated Toffoli gates, leaving the control qubits as they were before. Specifically the j-th Toffoli gate

is substituted by the negated Toffoli if $z_j = 1$. Naturally if $|z| = 0$ we recover lemma 1 and if $|z| = n$ we have lemma 2.

We summarize this result in lemma 3:

Lemma 3. *A T-linear circuit with RCC can be constructed for $s_{x_1} = x_1 \oplus z$ for any string $z \in \{0,1\}^n$ provided $|s \wedge z|$ is even.*

Proof. For $f(x_1) = x_1 \oplus z$, we can expand the expression $s \cdot (x_1 \oplus z) = s_1((x_1)_1 \oplus z_1) \oplus \cdots \oplus s_n((x_1)_n \oplus z_n) \overset{\star}{=} s_1(x_1)_1 \oplus s_1 z_1 \oplus \cdots \oplus s_n(x_1)_n \oplus s_n z_n = s \cdot x_1 \oplus s \cdot z$ using the distributive property $a(b \oplus c) = ab \oplus ac$ of single bits a, b and c on Step $\star$. If $|s \wedge z|$ is even, then $s \cdot z = \underbrace{1 \oplus \cdots \oplus 1}_{\text{even}} = 0$ and so if this is applied in the previous equation we have $s \cdot (x_1 \oplus z) = s \cdot x_1 \oplus s \cdot z = s \cdot x_1 \oplus 0 = s \cdot x_1$. Therefore $s_{x_1} = f(x_1) = x_1 \oplus z$ fulfils $s \cdot x_1 = s \cdot f(x_1)$ if $|s \wedge z|$ is even. □

Another class of functions f that can be considered is moving part of the mapping f onto the Toffolis from Step 2 in equation (2). It is possible to permute the controls of the Toffolis and delete some of the Toffolis all together. The possible permutations of the control qubits of the second register of the Toffolis also result in T-linear circuits with RCC. A valid permutation of the bits of $f(x_1)$, $\sigma(f(x_1))$, must satisfy $s \cdot x_1 = s \cdot \sigma(f(x_1))$ with the restriction that for each pair of bits permuted, $p \in \{1,\ldots,n\}$ and $q \in \{1,\ldots,n\}$, $s_p = s_q$ in each permutation. A permutation of bits p and q of $f(x_1)$ can be written explicitly as $\sigma(f(x_1)) = \sigma(f(x_1)_1,\ldots,f(x_1)_p,\ldots,f(x_1)_q,\ldots,f(x_1)_n) = (f(x_1)_1,\ldots,f(x_1)_q,\ldots,f(x_1)_p,\ldots,f(x_1)_n)$. This is equivalent to permuting the control qubits located in the second register of the pair of Toffolis p and q, leaving the control qubits of register 1 unchanged. For the mapping f any of the previous mappings is suitable to be permuted, provided each of their respective restrictions are taken into account. For example if $f(x_1) = \overline{x_1}$, besides $s_p = s_q$ for the permuted bits p and q, $|s|$ must be even. The maximum number of valid permutations for a given s is $n! - 1$ (all the possible permutations except the starting position) and occurs if $|s| = n$.

As an example, if $s = 111$ and $s_{x_1} = f(x_1) = x_1$, then there are $3! - 1$ possible permutations available for the values of s_{x_1} obtained from lemma 1. The starting values of $s_{x_1} = x_1$ in this example are: $s_{000} = 000$, $s_{001} = 001$, $s_{010} = 010$, $s_{011} = 011$, $s_{100} = 100$, $s_{101} = 101$, $s_{110} = 110$, $s_{111} = 111$. The first permutation could be switching the position of bits 2 and 3 so $\sigma(f(x_1)) = \sigma(f(x_1)_1, f(x_1)_2, f(x_1)_3) = (f(x_1)_1, f(x_1)_3, f(x_1)_2)$: $s_{000} = 000$, $s_{001} = 010$, $s_{010} = 001$, $s_{011} = 011$, $s_{100} = 100$, $s_{101} = 110$, $s_{110} = 101$, $s_{111} = 111$. Here the pair of bits permuted, 2 and 3, fulfil $s_2 = s_3$. This means that the control qubits 2 and 3 of the second register of the Toffolis will be permuted, as can be seen in figure 4 which shows the circuit that solves this example. The rest of the permuted circuits can be obtained using this procedure.

Regarding the erasure of Toffolis, this can be accomplished by setting some bits of $f(x_1)$ to zero, i.e. by using the mapping $f(x_1) \wedge u$ for some string $u \in \{0,1\}^n$ instead of $f(x_1)$. In this case f is still a bijection that satisfies $s \cdot x_1 = s \cdot f(x_1)$. This is equivalent to simply erasing the j-th Toffoli gate if $s_j = 0$.

We summarize this result in lemma 4:

Lemma 4. *A T-linear circuit with RCC can be constructed for $s_{x_1} = f(x_1) \wedge u$ for some string $u \in \{0,1\}^n$ provided $u \wedge s = s$.*

Proof. For $f(x_1) \wedge u$ we have that $s \cdot (f(x_1) \wedge u) = \sum_{j=1}^n s_j f(x_1)_j u_j \overset{\star}{=} \sum_{j=1}^n s_j^2 f(x_1)_j = \sum_{j=1}^n s_j f(x_1)_j = s \cdot f(x_1) = s \cdot x_1$ if $u \wedge s = s$ (used on Step $\star$), i.e. if the bits of s whose value is 1 are also bits of u whose value is 1. Therefore $s_{x_1} = f(x_1) \wedge u$ if $u \wedge s = s$ and f a bijection are also valid mappings. □

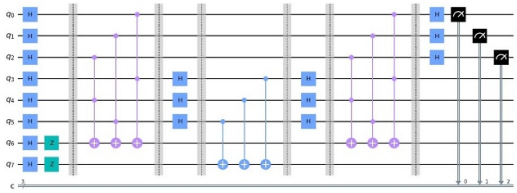

Figure 4. T-linear circuit with RCC for $s=111$, $s_{000}=000$, $s_{001}=010$, $s_{010}=001$, $s_{011}=011$, $s_{100}=100$, $s_{101}=110$, $s_{110}=101$, $s_{111}=111$ and $g(000)=g(011)=g(101)=g(110)=0$, $g(001)=g(010)=g(100)=g(111)=1$. This circuit is obtained from lemma 1 after permuting the second and third control qubits of the second register.

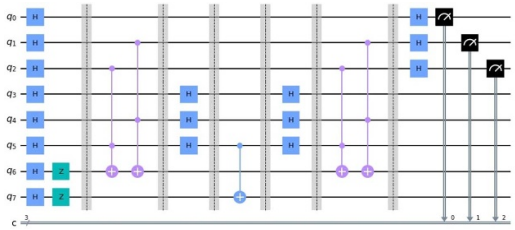

Figure 5. T-linear circuit with RCC for $s=001$, $s_{000}=000$, $s_{001}=001$, $s_{010}=010$, $s_{011}=011$, $s_{100}=000$, $s_{101}=001$, $s_{110}=010$, $s_{111}=011$ and $g(000)=g(010)=g(100)=g(110)=0$, $g(001)=g(011)=g(101)=g(111)=1$. This circuit is obtained from lemma 4.

Naturally, the valid mappings for f in lemma 4 are all the mapping from the previous lemmas such as $f(x_1)=x_1\oplus z$ for any string $z\in\{0,1\}^n$ if $|s\wedge z|$ is even.

As an example, if $s=001$, then the available values for $u=\{001,011,101,111\}$ due to the condition $u\wedge s=s$. Taking $f(x_1)=x_1$ and $u=011$ for this particular case the values of s_{x_1} according to lemma 4 are $s_{x_1}=x_1\wedge u$ so: $s_{000}=000$, $s_{001}=001$, $s_{010}=010$, $s_{011}=011$, $s_{100}=000$, $s_{101}=001$, $s_{110}=010$, $s_{111}=011$. Since $g(s_{\text{in}})$ fulfils $g(s_{\text{in}})=s\cdot s_{\text{in}}$ the values of $g(s_{\text{in}})$ are $g(000)=g(010)=g(100)=g(110)=0$, $g(001)=g(011)=g(101)=g(111)=1$. For these values a T-linear circuit with RCC can be constructed. The circuit that solves this example is represented in figure 5. Notice that U_s is made using 2 Toffolis instead of 3.

Of course, there are other ways of setting $s\cdot(f(x_1)\wedge u)=s\cdot x_1$, e.g. divide the strings $x_1\in\{0,1\}^n$ into the two sets $\mathcal{X}_0:=\{x_1\in\{0,1\}^n : s\cdot x_1=0\}$ and $\mathcal{X}_1:=\{x_1\in\{0,1\}^n : s\cdot x_1=1\}$ and map $\mathcal{X}_0\mapsto\{0^n\}$ and $\mathcal{X}_1\mapsto\{0^{l-1}10^{n-l}\}$, where $l\in\{1,\ldots,n\}$ is any entry such that $s_l=1$. Such mapping can be accomplished by $u=0^{l-1}10^{n-l}$ and $f(x_1)=x_1\oplus\sum_{m\neq l}^{n}0^{l-1}(x_1)_m0^{n-l}$ where $m\neq l$ is any entry such that $s_m=1$, the sum $\sum_{m\neq l}^{n}0^{l-1}(x_1)_m0^{n-l}$ refers to the bitwise sum of vectors (strings containing n bits each) and $(x_1)_m$ is just the m-th bit of x_1. If such an entry does not exist, i.e. $|s|=1$, then $f(x_1)=x_1\oplus\sum_{m\neq l}^{n}0^{l-1}(x_1)_m0^{n-l}=x_1\oplus 0^n=x_1$. This is equivalent to having every Toffoli gate use the same control qubit l in the second register while leaving the control qubits of the first register unchanged.

We summarize this result in lemma 5:

Lemma 5. *A T-linear circuit with RCC can be constructed, given a string $u=0^{l-1}10^{n-l}$, for $s_{x_1}=\left[x_1\oplus\sum_{m\neq l}^{n}0^{l-1}(x_1)_m0^{n-l}\right]\wedge u$, provided $s_l=1$ and $s_m=1$ for any entry where $m\neq l$ in the sum.*

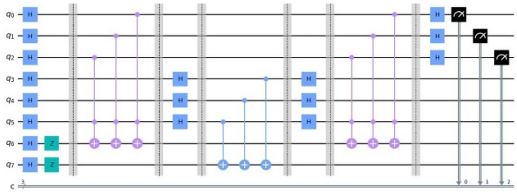

Figure 6. T-linear circuit with RCC for $s=111$, $s_{000}=000$, $s_{001}=001$, $s_{010}=001$, $s_{011}=000$, $s_{100}=001$, $s_{101}=000$, $s_{110}=000$, $s_{111}=001$ and $g(000)=g(011)=g(101)=g(110)=0$, $g(001)=g(010)=g(100)=g(111)=1$. This circuit is obtained from lemma 5.

Proof. For this mapping we have $s_{x_1}=f(x_1)\wedge u=\left[x_1\oplus\sum_{m\neq l}^{n}0^{l-1}(x_1)_m0^{n-l}\right]\wedge u$ and so $s\cdot(f(x_1)\wedge u)=\sum_{j=1}^{n}s_jf(x_1)_ju_j\overset{1}{=}s_lf(x_1)_l$ if $u=0^{l-1}10^{n-l}$ (used on Step 1). Then substituting $f(x_1)$ we have $s_lf(x_1)_l=s_l[(x_1)_l\oplus\sum_{m\neq l}^{n}(x_1)_m]\overset{2}{=}(x_1)_l\oplus\sum_{m\neq l}^{n}(x_1)_m$ if $s_l=1$ (used on Step 2). Then the previous equation is equal to $s\cdot x_1=\sum_{j=1}^{n}s_j(x_1)_j=s_l(x_1)_l\oplus\sum_{m\neq l}^{n}s_m(x_1)_m\overset{3}{=}(x_1)_l\oplus\sum_{m\neq l}^{n}(x_1)_m$ if $s_l=1$ and $s_m=1$ for each term that appears in the sum in m (used on Step 3). Therefore $s_{x_1}=\left[x_1\oplus\sum_{m\neq l}^{n}0^{l-1}(x_1)_m0^{n-l}\right]\wedge u$ is a valid mapping if $u=0^{l-1}10^{n-l}$, $s_l=1$ and $s_m=1$ for any entry where $m\neq l$ in the sum. □

Notice that if negated Toffoli gates were used, the results remain the same as long as the corresponding restrictions are taken into consideration. If instead of $f(x_1)=x_1\oplus\sum_{m\neq l}^{n}0^{l-1}(x_1)_m0^{n-l}$ a different mapping is used such as $f(x_1)=\overline{x_1}\oplus\sum_{m\neq l}^{n}0^{l-1}(\overline{x_1})_m0^{n-l}$ with $|s|$ even or $f(x_1)=x_1\oplus z\oplus\sum_{m\neq l}^{n}0^{l-1}(x_1\oplus z)_m0^{n-l}$ with $|s\wedge z|$ even and $z\in\{0,1\}^n$, the same results are obtained.

As an example, if $s=111$ and taking $l=3$ since $s_3=1$, we have $u=0^{3-1}10^{3-3}=001$. Then $m=1,2$ since $s_1=s_2=1$. This means that $s_{x_1}=[x_1\oplus\sum_{m\neq l}^{n}0^{l-1}(x_1)_m0^{n-l}]\wedge u=[x_1\oplus 00(x_1)_1\oplus 00(x_1)_2]\wedge 001$ so: $s_{000}=000$, $s_{001}=001$, $s_{010}=001$, $s_{011}=000$, $s_{100}=001$, $s_{101}=000$, $s_{110}=000$, $s_{111}=001$. The resulting circuit is represented in figure 6.The control qubit of the second register for all Toffoli gates is the third qubit.

In lemma 5 all Toffoli gates use the same control qubit l but of course this lemma can be generalized further if only some Toffoli gates use the same control qubit. In this case, a valid transformation of $f(x_1)$, denoted by $\tau(f(x_1))$, applied on its bits $p\in\{1,\ldots,n\}$ and $q\in\{1,\ldots,n\}$ can be written as: $\tau(f(x_1))=\tau(f(x_1)_1,\ldots,f(x_1)_p,f(x_1)_q,\ldots,f(x_1)_n)=(f(x_1)_1,\ldots,0_p,f(x_1)_p\oplus f(x_1)_q,\ldots,f(x_1)_n)$. The transformation must satisfy $s\cdot x_1=s\cdot\tau(f(x_1))$ with the restriction that $s_p=s_q=1$ as in lemma 5. This transformation can be performed for as many bits as desired. If it is applied $n-1$ times, choosing all possible values of p except one that acts as q each time, lemma 5 is recovered assuming that τ is applied to $f(x_1)=x_1$.

Next, similarly to lemma 5 we can divide the strings $x_1\in\{0,1\}^n$ into the two sets $\mathcal{X}_0:=\{x_1\in\{0,1\}^n:s\cdot x_1=0\}$ and $\mathcal{X}_1:=\{x_1\in\{0,1\}^n:s\cdot x_1=1\}$ and map $\mathcal{X}_0\mapsto\{0^n\}$ and $\mathcal{X}_1\mapsto\{1^n\}$ where again $l\in\{1,\ldots,n\}$ is any entry such that $s_l=1$. Such mapping can be accomplished by $f(x_1)=(x_1\wedge u)\oplus\sum_{m\neq l}^{n}0^{m-1}(x_1)_l0^{n-m}=\sum_{m=1}^{n}0^{m-1}(x_1)_l0^{n-m}$ if $u=0^{l-1}10^{n-l}$, where $m\in\{1,\ldots,n\}$, and $(x_1)_l$ is just the lth bit of x_1. This is equivalent to having every Toffoli gate use the same control qubit l in the first register while leaving the control qubits of the second register unchanged.

We summarize this result in lemma 6:

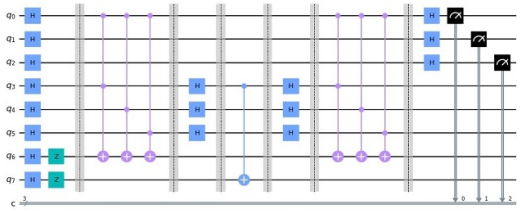

Figure 7. T-linear circuit with RCC for $s = 100$, $s_{000} = 000$, $s_{001} = 000$, $s_{010} = 000$, $s_{011} = 000$, $s_{100} = 111$, $s_{101} = 111$, $s_{110} = 111$, $s_{111} = 111$, and $g(000) = g(001) = g(010) = g(011) = 0$, $g(100) = g(101) = g(110) = g(111) = 1$. This circuit is obtained from lemma 6.

Lemma 6. *A T-linear circuit with RCC can be constructed for* $s_{x_1} = (x_1 \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1)_l 0^{n-m} = \sum_{m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m}$, *provided* $u = 0^{l-1}10^{n-l}$, $s_l = 1$ *and also* $|s| = 1$.

Proof. For this mapping we have $s \cdot f(x_1) = s \cdot [(x_1 \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1)_l 0^{n-m}] \overset{1}{=} s \cdot [\sum_{m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m}] = \sum_{j=1}^{n} s_j (x_1)_l$ if $u = 0^{l-1}10^{n-l}$ (used on Step 1). If $s_l = 1$ and $|s| = 1$, then all $s_j = 0$ except s_l and the previous equation can be expanded as $\sum_{j=1}^{n} s_j (x_1)_l = s_l (x_1)_l = (x_1)_l$. Then the previous equation is equal to $s \cdot x_1 = \sum_{j=1}^{n} s_j (x_1)_j \overset{2}{=} (x_1)_l$ if $s_l = 1$ and $|s| = 1$ (used on Step 2). Both sides of the equation $s \cdot f(x_1) = s \cdot x_1$ are equal and therefore $s_{x_1} = (x_1 \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1)_l 0^{n-m}$ is a valid mapping if $u = 0^{l-1}10^{n-l}$, $s_l = 1$ and $|s| = 1$. □

Contrary to lemmas 4 and 5, not all the Toffoli gates can be substituted by negated Toffoli gates. In this case the mapping $f(x_1) = (\overline{x_1} \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(\overline{x_1})_l 0^{n-m} = \sum_{m=1}^{n} 0^{m-1}(\overline{x_1})_l 0^{n-m}$ can not be applied due to the incompatibility between the restrictions $|s| = 1$ and $|s|$ even. The mapping $f(x_1) = ((x_1 \oplus z) \wedge u) \oplus \sum_{m\neq l}^{n} 0^{m-1}(x_1 \oplus z)_l 0^{n-m} = \sum_{m=1}^{n} 0^{m-1}(x_1 \oplus z)_l 0^{n-m}$ can be applied as long as $|s \wedge z|$ is even which implies $z_l = 0$ due to $|s| = 1$. Therefore, the mapping $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{1^n\}$ can only be done with regular Toffolis as in lemma 6.

As an example if $s = 100$, then $l = 1$ since $s_1 = 1$. We have $u = 100$ which means $s_{x_1} = \sum_{m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m} = (x_1)_1 00 \oplus 0(x_1)_1 0 \oplus 00(x_1)_1$ so: $s_{000} = 000$, $s_{001} = 000$, $s_{010} = 000$, $s_{011} = 000$, $s_{100} = 111$, $s_{101} = 111$, $s_{110} = 111$, $s_{111} = 111$. The resulting circuit is represented in figure 7.The control qubit of the first register for all Toffoli gates is the first qubit.

In lemma 6 all Toffoli gates use the same control qubit of the first register l but as in lemma 5 this can be generalized further if only some Toffoli gates use the same control qubit. In this case, a valid transformation of $f(x_1)$, denoted by $\beta(f(x_1))$, applied on its bits $p \in \{1,\ldots,n\}$ and $q \in \{1,\ldots,n\}$ can be written as: $\beta(f(x_1)) = \beta(f(x_1)_1,\ldots,f(x_1)_p,f(x_1)_q,\ldots,f(x_1)_n) = (f(x_1)_1,\ldots,f(x_1)_p,f(x_1)_p,\ldots,f(x_1)_n)$. The transformation must satisfy $s \cdot x_1 = s \cdot \beta(f(x_1))$ with the restriction that $s_p = 1$ and $s_q = 0$. This transformation can be performed for as many bits as desired. Once again, if it is applied $n-1$ times, choosing all possible values of q except one that acts as p each time, lemma 6 is recovered assuming that β is applied to $f(x_1) = x_1$.

By combining all the previous lemmas and transformations, the remaining valid T-linear circuits with RCC can be constructed, i.e. a circuit that contains negated and regular Toffolis, where some of the control qubits are permuted, some others act on the same qubit of the first register, others on the same qubit of the second register and some Toffoli gates are deleted

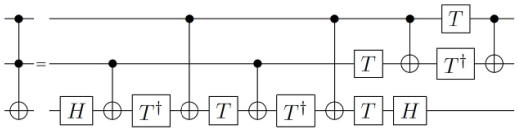

Figure 8. A Toffoli gate requires 7 $T/T^\dagger$ gates to be implemented.

altogether is also a T-linear circuit with RCC, provided the corresponding restrictions are taken into account for the qubits involved.

For the function $g(s_{\text{in}})$ chosen, the degrees of freedom of the problem are: which type of Toffoli is applied regarding if it activates with a 1 or a 0, the number of Toffolis used, and the control qubits used. Then all possible cases are: using n Toffoli gates, negated Toffolis or a combination of both types of gates (lemmas 1–3 respectively), eliminating some of these n Toffoli gates (lemma 4), all the possible permutations of the control qubits of the second register for all the previous lemmas (given by $\sigma(f(x_1))$), using the same control qubit of the second register for all the Toffoli gates (lemma 5) or just for some of them (given by $\tau(f(x_1))$) and finally using the same control qubit of the first register for all the Toffoli gates (lemma 6) or just for some of them (given by $\beta(f(x_1))$). Therefore the circuits obtained from all the possible combinations of all these cases constitute the complete set of T-linear circuits with RCC for $k=2$. Recall that the circuits allowed here are those composed of H gates in some layers, a layer of Z gates after the first layer of H gates, Toffoli and X gates for the oracle U_s and CNOT gates for G. Regarding the number of Toffoli gates that T-linear circuits with RCC require, at worst n gates are needed and the best case scenario occurs for a circuit obtained from lemma 4 that erases all Toffolis but one.

2.2. *Homomorphic implementation and extended results*

The main feature of these circuits is the fact that the oracle U_s was simplified from multi-controlled CNOT gates to just n Toffoli gates at most and the oracle G was also simplified so it could be implemented using CNOT gates instead of multi-controlled CNOT gates. The importance of this idea is related to the number of T gates required to implement each operation.

For T-linear circuits with RCC, like in figure 2, the number of T gates grows linearly with the number of qubits of each register (the same number as bits needed to represent s), n, required to solve the problem. This is because as it has already been discussed, in the worst case scenario n Toffolis are needed to implement U_s. Since a Toffoli can be decomposed in 7 T gates as seen in figure 8 ($T^\dagger$ gates are counted as T gates) and considering that U_s has to be applied twice to solve the recursive BV problem for $k=2$, the number of T gates needed is $14\cdot n$, which means this number grows linearly with n, so the number of T gates grows as $O(n)$.

In appendix A we can see that Liang's QHE schemes are only suitable for circuits with a polynomial number of $T/T^\dagger$ gates. This means that T-linear circuits are a perfect example of an algorithm that can be evaluated efficiently using Liang's QHE schemes. Summarizing in corolary 1:

Corollary 1. *The number of $T/T^\dagger$ gates needed to implement T-linear circuits with RCC grows linearly with the number of qubits of each register needed to solve the problem: $O(n)$. Therefore T-linear circuits with RCC can be implemented efficiently using Liang's QHE schemes.*

On the other hand, for the $k = 2$ case, if U_s has to be implemented using just multi-controlled CNOTs, the number of T gates grows differently. Since $g(s_{x_1}) = s \cdot x_1$, half of the values of $g(s_{x_1})$ will be 1, so half of all s_{x_1} strings will have $|s_{x_1}| \neq 0$ (they will not be equal to 0^n). This means that U_s will require at least a multi-controlled CNOT for half of the s_{x_1} strings, because unlike Toffoli gates, a multi-controlled CNOT is restricted to just one value of x_1 each time it is applied. Then, the number of multi-controlled CNOTs will be 2^{n-1} at minimum, since there are 2^n s_{x_1} strings and we need to take half of that number.

Considering that U_s is applied twice, the number of multi-controlled CNOTs would be 2^n. Even if for simplicity we assume that each multi-controlled CNOT could be implemented using 7 T gates like the Toffoli gates, the number of T gates grows exponentially with the number of qubits of each register n, so the number of T gates grows as $\Omega(2^n)$. Therefore this implementation of U_s would not be suitable to be evaluated using the QHE schemes that have been presented.

Assuming that the multi-controlled CNOTs have the same number of T gates as the Toffoli gates is an extreme simplification, but since the number of T gates is exponential even in this case, it is enough for our purpose: checking if the number of T gates in the circuit is exponential or polynomial.

We want to mention some cases where U_s has to be implemented with multi-controlled CNOTs because Toffolis are not enough. A group of circuits where this happens are those where $2^{n-1} + 1$ of the s_{x_1} strings are mapped using any of the previous lemmas but the remaining strings are mapped to different values that are still valid. For instance it is possible to construct a BV problem in which one half of the s_{x_1} strings are mapped to x_1 and the other half of the s_{x_1} strings are mapped to $s_{\max} = 1^n$ since $g(s_{\max}) = s \cdot s_{\max} = 0 \; \forall s$ if $|s|$ is even and $g(s_{\max}) = s \cdot s_{\max} = 1 \; \forall s$ if $|s|$ is odd (assuming $g(s_{\text{in}})$ fulfils $g(s_{\text{in}}) = s \cdot s_{\text{in}}$).

As an example if $s = 111$, then $s_{000} = 000$, $s_{001} = 111$, $s_{010} = 111$, $s_{011} = 011$, $s_{100} = 111$, $s_{101} = 101$, $s_{110} = 110$, $s_{111} = 111$ and $g(000) = g(011) = g(101) = g(110) = 0$, $g(001) = g(010) = g(100) = g(111) = 1$. Since $|s|$ is odd, $g(s_{\max}) = s \cdot s_{\max} = s \cdot 111 = 1$ so $s_{\max}$ is a valid value for all s_{x_1} strings where $g(s_{x_1}) = s \cdot x_1 = 1$. This example is basically what would be obtained from lemma 1 for $s = 111$ but substituting half of the values of s_{x_1} for $s_{\max}$. However, it is impossible to implement this circuit just using n Toffoli gates, because $s_{001} = s_{010} = s_{100} = 111$, but according to lemma 6 $\mathcal{X}_1 \mapsto \{1^n\}$ only if $|s| = 1$. This means that if s_{001} is mapped to $s_{\max}$, then $s_{010} = s_{100} \neq s_{\max}$ so this particular case can not be solved with Toffoli gates and needs multi-controlled CNOTs. In this particular example, using $s_{x_1} = x_1$ as in lemma 1 to apply 3 Toffolis would implement U_s for all the s_{x_1} strings where $g(s_{x_1}) = 0$ and would also implement $s_{111} = 111$. This is done to minimize the number of multi-controlled CNOTs as much as possible. However for the remaining values (s_{001}, s_{010}, s_{100}) two multi-controlled CNOTs would be needed for each s_{x_1} so U_s implements $s_{001} = s_{010} = s_{100} = s_{\max}$.

In the general case for an arbitrary n where $\mathcal{X}_1 \mapsto \{1^n\}$, $\mathcal{X}_0 \mapsto \{x_1\}$ and assuming that n Toffolis are used to map $\mathcal{X}_0 \mapsto \{x_1\}$ so the maximum amount of phase kickbacks are applied, minimizing the number of multi-controlled CNOT gates used in these circuits, $2^{n-1} - 1$ s_{x_1} strings still need to be mapped to $s_{\max}$ after the Toffolis are applied. Then each one of these strings would need at least a multi-controlled CNOT (most would need more) in order to obtain the desired mapping, so at least $2^{n-1} - 1$ multi-controlled CNOTs would be needed to implement U_s. Then, as it has been discussed, the best possible number of T gates for these types of circuits grows as $O(2^n)$. If no Toffolis were used then each of the 2^{n-1} s_{x_1} strings that have to be mapped to $s_{\max}$ would need n multi-controlled CNOT gates, making the number

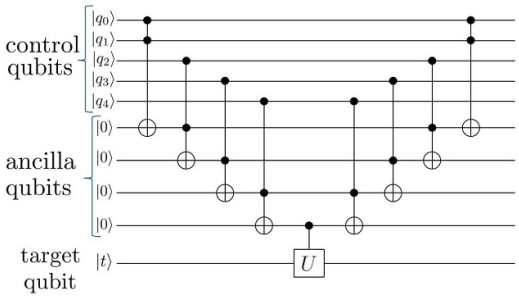

Figure 9. A multi-controlled single qubit gate U with $n=5$ control qubits, decomposed into 8 Toffoli gates using 4 ancilla qubits in the state $|0\rangle$. If $U=X$ we have a multi-controlled CNOT gate.

of T gates in the circuit grow as an exponential once more (worse than the former n Toffolis case).

Finally we would like to briefly discuss how the results obtained for T-linear circuits with RCC could be extended for different functions $g(s_{\text{in}})$ than the one used and for $k>2$.

A simple function g could be $g(s_{\text{in}})=s\cdot\overline{s_{\text{in}}}$. This g is simply the application of $|s|$ negated CNOTs. Then we have that f has to obey $s\cdot x_1=s\cdot\overline{f(x_1)}$ and all the previous results discussed for T-linear circuits with RCC still apply here, taking into account that the restrictions that negated Toffolis had are now the restrictions of the usual Toffolis. For example if $f(x_1)=x_1 \implies s\cdot x_1=s\cdot\overline{f(x_1)}=s\cdot\overline{x_1}$, so by using the mapping f of lemma 1 we recover the condition of lemma 2. More generally any balanced function $g(s_{\text{in}})=s\cdot(s_{\text{in}}\oplus z)$ for any string $z\in\{0,1\}^n$ can be chosen, taking into consideration the corresponding restrictions when selecting a specific $f(x_1)$. All these functions can still be implemented just with $|s|$ CNOTs, mixing negated and usual CNOT gates.

Next we can look into functions g that are not balanced. As we discussed previously, these functions have to be implemented with multi-controlled CNOT gates with n control qubits that activate with 0 or 1 as needed and target the $|-\rangle$ state in order to obtain phase kickbacks. A multi-controlled CNOT gate with n control qubits can be decomposed into $2(n-1)$ Toffoli gates using $(n-1)$ extra ancilla qubits. An example of this decomposition [20] is shown in figure 9 for any multi-controlled U gate.

We are still interested in implementing U_s with n Toffoli gates at most for these kind of functions g. As an example the function g defined as $g(s_{\text{in}})=1$ if $s_{\text{in}}=0^{n-1}1$ and $g(s_{\text{in}})=0$ for the remaining inputs can be implemented using just one multi-controlled CNOT that activates for the state $|0^{n-1}1\rangle$ and targets the state $|-\rangle$. Using lemma 5 we can map $\mathcal{X}_0\mapsto\{0^n\}$ and $\mathcal{X}_1\mapsto\{0^{n-1}1\}$ taking $l=n$. Notice that for this type of function g, the restriction $s_l=1$ from lemma 5 does not longer apply and instead we have $l\in\{\{1,\ldots,n\}:g(0^{l-1}10^{n-l})=1\}$, so for this example $g(0^{l-1}10^{n-l})=g(0^{n-1}1)=1$ and $l=n$. The mapping for f is now $f(x_1)\wedge u=\left[\sum_m^n 0^{l-1}(x_1)_m 0^{n-l}\right]\wedge u$ for $u=0^{l-1}10^{n-l}$ and any entry m such that $s_m=1$, then for the example chosen $l=n$ and we have $f(x_1)\wedge u=\left[\sum_m^n 0^{n-1}(x_1)_m\right]\wedge 0^{n-1}1=\sum_m^n 0^{n-1}(x_1)_m$. From this a valid implementation of U_s is obtained for this function g that still uses only n Toffoli gates at most. However after decomposing the multi-controlled CNOT into $2(n-1)$ Toffoli gates, this circuit now requires $n-1$ ancilla qubits and contains $2n+2(n-1)$ Toffoli gates at most. The number of T gates then still grows as $O(n)$. Then the decryption process of

this circuit is still efficient provided $n-1$ extra ancilla qubits are used in the QHE scheme. Of course this result also works for a function g defined as $g(s_{\text{in}})=1$ if $s_{\text{in}}=0^{l-1}10^{n-l}$, $g(s_{\text{in}})=0$ for the remaining inputs and any value of l.

Using the previous example we can easily generalize the result for every function g that obeys:

$$g\left(s_{\text{in}}\right)=\begin{cases}1 & \text{if } |s_{\text{in}}|=1\\ 0 & \text{if } |s_{\text{in}}|\neq 1.\end{cases} \tag{3}$$

This function g is implemented using n multi-controlled CNOT gates, the l-th gate activating for the state $|0^{l-1}10^{n-l}\rangle$, where $l\in\{1,\ldots,n\}$. The target is the $|-\rangle$ state as usual. For this type of g we can find valid mappings for U_s if we use lemma 5 again, mapping $\mathcal{X}_0\mapsto\{0^n\}$ and $\mathcal{X}_1\mapsto\{0^{l-1}10^{n-l}\}$. The restriction $s_l=1$ is no longer required and instead any value of l can be chosen, since any l fulfils $g(0^{l-1}10^{n-l})=1$. The mapping f is then $f(x_1)\wedge u=\left[\sum_m^n 0^{l-1}(x_1)_m0^{n-l}\right]\wedge 0^{l-1}10^{n-l}=\sum_m^n 0^{l-1}(x_1)_m0^{n-l}$ for any entry m such that $s_m=1$. These circuits now require $n-1$ ancilla qubits and contain $2n+n\cdot 2(n-1)$ Toffoli gates at most. The number of T gates then grows as $O(n^2)$. The decryption process of these circuits then still has polynomial complexity but it is now quadratic. Therefore their homomorphic evaluation, while less efficient compared to T-linear circuits with RCC, is still feasible.

In general, lemma 5 can be used to construct a valid U_s for these types of functions g that are implemented with multi-controlled CNOT gates. As long as $g(s_{\text{in}})=1$ if $s_{\text{in}}=0^{l-1}10^{n-l}$ for a given l and $g(s_{\text{in}})=0$ for $s_{\text{in}}=0^n$, the remaining values of $g(s_{\text{in}})$ are not relevant and the mapping obtained from lemma 5 works, taking into account that the restriction $s_l=1$ is no longer required. The reason this lemma can be applied is simply that $g(0^n)=0\implies\mathcal{X}_0\mapsto\{0^n\}$ and $g(0^{l-1}10^{n-l})=1\implies\mathcal{X}_1\mapsto\{0^{l-1}10^{n-l}\}$.

The next type of function g that can be considered to expand the previous results further is:

$$g\left(s_{\text{in}}\right)=\begin{cases}1 & \text{if } |s_{\text{in}}|=n\\ 0 & \text{if } |s_{\text{in}}|\neq n.\end{cases} \tag{4}$$

This function g is implemented using one multi-controlled CNOT gate that activates for the state $|1^n\rangle$.The target is the $|-\rangle$ state as usual. If lemma 6 is used to map $\mathcal{X}_0\mapsto\{0^n\}$ and $\mathcal{X}_1\mapsto\{1^n\}$ a valid implementation for U_s and this g is obtained that works for the special case where $|s|=1$. These circuits contain $2n+2(n-1)$ Toffoli gates and $n-1$ ancilla qubits again, so their number of T gates grows as $O(n)$. However, this implementation of U_s given by lemma 6 is limited by the restriction $|s|=1$. In order to go beyond this condition, another implementation of U_s that can be considered is using n^2 Toffoli gates. Each of these Toffoli gates is controlled using the l-th qubit of the first register and the m-th qubit of the second register, for $l,m\in\{1,\ldots,n\}$. This way each qubit in the first register acts as the control qubit for n Toffoli gates and each of these gates uses a different control qubit from the second register. Using this implementation of U_s we obtain the mapping $\mathcal{X}_0\mapsto\{0^n\}$ and $\mathcal{X}_1\mapsto\{1^n\}$ with the condition that $|s|=n$. The mapping f is then $f(x_1)=\sum_{l=1,m=1}^{n}0^{m-1}(x_1)_l0^{n-m}$, where the sum is performed over all the values of l and m. These circuits require $n-1$ ancilla qubits and contain $2n^2+2(n-1)$ Toffoli gates. The number of T gates then grows as $O(n^2)$ which means that the decryption process of these circuits still has polynomial complexity. Therefore their homomorphic evaluation is still feasible even though it is less efficient than the evaluation of T-linear circuits with RCC. For an arbitrary s and this g, the implementation of U_s is constructed using $n\cdot|s|$ Toffoli gates. In this case, each of these Toffoli gates is controlled using the l-th qubit of the first register and the m-th qubit of the second register, for any entry $l\in\{\{1,\ldots,n\}:s_l=1\}$,

all the available values of $m \in \{1,\dots,n\}$ and the mapping $f(x_1) = \sum_{l,m=1}^{n} 0^{m-1}(x_1)_l 0^{n-m}$. After the usual gate decomposition the number of T gates grows as a polynomial, since at worst for $|s| = n$ it grows as $O(n^2)$ as explained.

Finally we can generalize these results for a g that fulfils $g(s_{\text{in}}) = 1$ if $s_{\text{in}} = 0^{n-i}1^i$ for $i \in \{1,\dots,n\}$ and $g(s_{\text{in}}) = 0$ for the remaining values, i.e. $g(s_{\text{in}}) = 1$ only for one string s_{in} with any number of ones. Then g can be implemented using one multi-controlled CNOT that activates for the state $|0^{n-i}1^i\rangle$ and targets the $|-\rangle$ state once more. Here U_s is constructed using $i \cdot |s|$ Toffoli gates. In this case, each of these Toffoli gates is controlled using the l-th qubit of the first register, the m-th qubit of the second register, for any $l \in \{\{1,\dots,n\} : s_l = 1\}$, any $m \in \{1,\dots,n\}$ such that $(s_{\text{in}})_m = 1$ when $g(s_{\text{in}}) = 1$ and the mapping $f(x_1) = \sum_{l,m}^{n} 0^{m-1}(x_1)_l 0^{n-m}$. Naturally the mapping $\mathcal{X}_0 \mapsto \{0^n\}$ and $\mathcal{X}_1 \mapsto \{0^{n-i}1^i\}$ is obtained from this U_s. As in the previous case, the homomorphic evaluation of the circuits resulting from this U_s and g is feasible because the number of T gates grows as a polynomial. This is because once the Toffoli gates and the multi-controlled CNOT gate are decomposed, the number of T gates grows as $O(n^2)$ at worst. This corresponds with the $|s| = i = n$ case described by the function 4 and implemented using n^2 Toffoli gates for U_s. Of course the results obtained here also apply to a g where $g(s_{\text{in}}) = 1$ only for one string s_{in} whose ones are located in any position, not necessarily restricted as in $s_{\text{in}} = 0^{n-i}1^i$.

Regarding the extension of these results for $k > 2$, the mapping that U_s implements for the general k case is

$$|x_1\rangle_1 \dots |x_k\rangle_k |y\rangle \mapsto |x_1\rangle_1 \dots |x_k\rangle_k |y \oplus \left(s_{x_1,\dots,x_{k-1}} \cdot x_k\right)\rangle \quad \forall x_1,\dots,x_k \in \{0,1\}^n, y \in \{0,1\}, \tag{5}$$

We want to implement this mapping using n multi-controlled CNOT gates with k control qubits. We will just consider the function $g(s_{\text{in}}) = s \cdot s_{\text{in}}$, so each application of G can be implemented with $|s|$ CNOT gates at most just like in T-linear circuits with RCC. Then by choosing this g the problem narrows down to the mapping $x_1,\dots,x_{k-1} \mapsto s_{x_1,\dots,x_{k-1}}$ with the restriction $s_{x_1,\dots,x_{k-2}} \cdot x_{k-1} = s \cdot s_{x_1,\dots,x_{k-1}}$ for some string $s \in \{0,1\}^n$. Let $f\colon \{0,1\}^n \to \{0,1\}^n$, $f(x_1,\dots,x_{k-1})$, be this mapping. Then we have $s_{x_1,\dots,x_{k-2}} \cdot x_{k-1} = s \cdot f(x_1,\dots,x_{k-1})$. As in the $k = 2$ case, the mapping from equation (5) can be generically performed as:

$$\begin{aligned} &|x_1\rangle_1 \dots |x_k\rangle_k |y\rangle_{\mathtt{T}} \overset{1}{\mapsto} |f(x_1,\dots,x_{k-1})\rangle_1 \dots |x_k\rangle_k |y\rangle_{\mathtt{T}} \overset{2}{\mapsto} \\ &|f(x_1,\dots,x_{k-1})\rangle_1 \dots |x_k\rangle_k |y \oplus (f(x_1,\dots,x_{k-1}) \cdot x_k)\rangle_{\mathtt{T}} \overset{3}{\mapsto} \\ &|x_1\rangle_1 \dots |x_k\rangle_k |y \oplus (f(x_1,\dots,x_{k-1}) \cdot x_k)\rangle_{\mathtt{T}}. \end{aligned} \tag{6}$$

Step 2 is simply the application of n multi-controlled CNOT gates with k control qubits. The jth multi-controlled CNOT gate is controlled using the jth qubit of registers 1, 2,…, k and it targets register $\mathtt{T}$, for $j \in \{1,\dots,n\}$. The issue is regarding Steps 1 and 3. As in the $k = 2$ case, f must be a bijection since each step must be unitary. Also as before, we want to implement this map f using just single-qubit gates, so the X gates are allowed. Then the multi-controlled CNOT gates that can be used in general can be activated with either 0 or 1.

Studying this mapping in general is out of the scope of this work, so we will just give an example of a possible extension in the general k case. We can simply use the map $f(x_1,x_2,\dots,x_{k-1}) = x_1 \wedge x_2 \wedge \dots \wedge x_{k-1}$. In order to satisfy the condition $s_{x_1,\dots,x_{k-2}} \cdot x_{k-1} = s \cdot f(x_1,\dots,x_{k-1})$ we require $|s| = n$ and we assume that $s_{x_1,\dots,x_{k-2}} = x_1 \wedge \dots \wedge x_{k-2}$, $s_{x_1,\dots,x_{k-3}} = x_1 \wedge \dots \wedge x_{k-3}$, …, $s_{x_1} = x_1$. Then we have $s_{x_1,\dots,x_{k-2}} \cdot x_{k-1} = s \cdot f(x_1,\dots,x_{k-1}) = s \cdot (x_1 \wedge \dots \wedge x_{k-1}) \implies (x_1 \wedge \dots \wedge x_{k-2}) \cdot x_{k-1} = s \cdot (x_1 \wedge \dots \wedge x_{k-1}) \implies |s| = n$. This mapping f is equivalent to simply using n multi-controlled CNOT gates with k control qubits. A single multi-controlled CNOT gate with k control qubits can be decomposed into $2(k-1)$ Toffoli gates.

Decomposing the n multi-controlled CNOT gates we then have $n \cdot 2(k-1)$ Toffoli gates and $k-1$ extra ancilla qubits, which means that the number of T gates grows as $O(n \cdot k)$. Then the decryption procedure of the QHE for these circuits still has polynomial complexity. Thus their homomorphic evaluation is still efficient, just not as much as T-linear circuits with RCC in the $k=2$ case. Therefore even in the general k case, some extension of T-linear circuits with RCC that are relevant in the context of QHE can still be found.

We conclude this section with a possible application of T-linear circuits with RCC. These circuits could be used as a test to asses that a quantum computer is functioning correctly. These circuits can not be efficiently simulated using classical computers since they contain T gates but they require less computational power than circuits with exponential number of T gates. We are currently in the so called 'noisy intermediate-scale quantum' era (NISQ) in which the best quantum computers available have at most a few hundred qubits but still have not reached fault-tolerance and are still not large enough to make use of the full computational power quantum mechanics offer. Since the number of T gates grows as $O(n)$ for T-linear circuits with RCC, these circuits could be implemented in quantum computers with large number of qubits to certify their correct functioning by making sure the correct s is returned and it would be less computationally difficult than other algorithms where the number of T gates is higher than linear.

3. Conclusions

In this article the nonrecursive BV algorithm and its recursive counterpart have been reviewed. The recursive version has historical importance because it showed for the first time a super-polynomial speed-up compared to what classical algorithms could achieve.

In this paper a certain group of quantum circuits, T-linear circuits with RCC, that solve some particular cases of the recursive BV problem for $k=2$ have been defined and characterized. The main feature of these circuits is simplifying the quantum oracle needed to solve the problem from multi-controlled CNOTs to Toffoli gates. This reduces the T gate complexity of the oracle so the number of T gates needed to solve the problem grows linearly with the number of qubits needed to represent the secret key s. This property is important in the realm of QHE.

Liang's [18] quasi-compact QHE schemes were reviewed. These schemes had perfect data security, $\mathcal{F}$-homomorphism, no interaction between client and server and quasi-compactness. Since the schemes are not compact they do not contradict the no-go result given by Yu *et al* [15]. The decryption procedure is independent of the size of the evaluated circuit and depends only on the number of $T/T^\dagger$ gates in the circuit. The decryption procedure is only efficient for circuits with a polynomial number of $T/T^\dagger$ gates. T-linear circuits with RCC number of $T/T^\dagger$ gates grow linearly as $O(n)$. Therefore, they constitute a perfect example of quantum circuits that can be evaluated homomorphically with perfect security and non interaction in an efficient manner.

On the other hand, a circuit that solves the recursive BV problem for $k=2$ that can not be implemented using a polynomial number of $T/T^\dagger$ gates would not be suitable to be implemented using these schemes, unlike T-linear circuits with RCC. This is because as it was shown such a circuit requires an exponential number of $T/T^\dagger$ gates, so the decryption process would be inefficient due to the quasi-compactness of the QHE schemes.

Some other possibilities of implementing the oracle G using different functions g were discussed. In these cases, some implementations of U_s using just Toffoli gates instead of multi-controlled CNOTs can still be found. Their number of $T/T^\dagger$ gates grows polynomially, so their homomorphic evaluation using the QHE schemes explained is still efficient, although not as

much as in T-linear circuits with RCC. Furthermore, these circuits require $n-1$ extra ancilla qubits due to the decomposition of the multi-controlled CNOT gates used to implement G. Finally, an extension of T-linear circuits for the general case k of the recursion was found for a simple case. For these circuits the number of $T/T^\dagger$ gates grows as $O(n\cdot k)$. Therefore their homomorphic implementation is also efficient, provided $k-1$ ancilla qubits are available.

Future works in this area of research could be studying the extension of T-linear circuits with RCC for higher orders of the recursion in more detail and whether or not these circuits can be evaluated using QHE schemes efficiently.

Furthermore, more quantum algorithms could be studied regarding their T gates complexity. This way their homomorphic implementation using Liang's schemes could be analysed. It may not be possible to implement them homomorphically in the most general case but particular cases similar to T-linear circuits with RCC where the T gate complexity is low may exist and be useful for certifying instances of quantum computers in the NISQ era where classical simulations can not do otherwise.

Data availability statement

No new data were created or analysed in this study.

Acknowledgments

We acknowledge the support from the CAM/FEDER Project No. S2018/TCS-4342 (QUITEMAD-CM), Spanish MINECO Grants MINECO/FEDER Projects, PID2021-122547NB-I00 FIS2021, the 'MADQuantum-CM' project funded by Comunidad de Madrid and by the Recovery, Transformation, and Resilience Plan—Funded by the European Union—NextGenerationEU and Ministry of Economic Affairs Quantum ENIA project. M. A. M.-D. has been partially supported by the U.S.Army Research Office through Grant No. W911NF-14-1-0103. P F O acknowledges support from a MICINN Contract PRE2019-090517 (MICINN/AEI/FSE,UE).

Conflict of interest

The authors declare that they have no competing interests that are relevant to the content of this article.

Appendix A. QHE scheme

On the next sections we will briefly explain how the QHE scheme is constructed.

A.1. Preliminaries

In the circuit model of quantum computation, quantum gates are the basis of any algorithm that could be implemented [21]. We will use the usual Clifford gates, which are $\{X, Z, H, S, \mathrm{CNOT}\}$. $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ are the usual Pauli gates. The Hadamard gate is $H = \frac{X+Z}{\sqrt{2}} =$

$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$. S is just $\sqrt{Z}$ and CNOT $= \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}$. To perform universal quantum computation the $T = \begin{pmatrix}1 & 0\\ 0 & \mathrm{e}^{\mathrm{i}\frac{\pi}{4}}\end{pmatrix}$ and $T^\dagger = \begin{pmatrix}1 & 0\\ 0 & \mathrm{e}^{-\mathrm{i}\frac{\pi}{4}}\end{pmatrix}$ need to be added to the Clifford gates. The set of gates then becomes $\mathcal{G} = \{X, Z, H, S, \mathrm{CNOT}, T, T^\dagger\}$. The main difficulty of the homomorphic evaluation of a quantum circuit consists on the evaluation of the T and $T^\dagger$ gates because it causes a S-error: $TX^aZ^b|\phi\rangle = (S^\dagger)^a X^a Z^{a\oplus b} T|\phi\rangle$, so somehow this error has to be corrected. Liang's schemes correct it by making use of gate teleportation, which is a generalization of quantum teleportation.

A.2. Gate teleportation

An EPR state is an entangled quantum state $|\Phi_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. From this entangled state, we can express the usual four Bell states in a compact expression:

$$|\Phi_{ab}\rangle = \left(Z^b X^a \otimes I\right)|\Phi_{00}\rangle, \forall a, b \in 0, 1. \tag{A1}$$

The state $|\Phi_{00}\rangle$ can be constructed by using a H gate followed by a CNOT acting on the state $|00\rangle$. Quantum teleportation is a procedure that transports quantum states between a sender and a receiver using a quantum communication channel. Alice wants to send Bob a state $|\psi\rangle$, so they share an entangled pair $|\Phi_{00}\rangle$ so each one has a qubit from the EPR pair. Alice makes a measurement in the Bell basis using her two qubits, $|\psi\rangle$ and one qubit of $|\Phi_{00}\rangle$. Due to entanglement, if Bob applies the correct quantum gates (a combination of X and Z), he can reconstruct $|\psi\rangle$ [22].

For any single gate U, the 'U-rotated Bell basis' is defined as: $\Phi(U) = \{|\Phi(U)_{ab}\rangle, a, b \in \{0, 1\}\}$, where $|\Phi(U)_{ab}\rangle = (U^\dagger \otimes I)|\Phi_{ab}\rangle = (U^\dagger Z^b X^a \otimes I)|\Phi_{00}\rangle$.

For a single qubit we have the following expression from [23], which is just an expression for quantum teleportation:

$$|\alpha\rangle \otimes |\Phi_{00}\rangle = \sum_{a,b\in\{0,1\}} |\Phi_{ab}\rangle \otimes X^a Z^b |\alpha\rangle. \tag{A2}$$

It can be extended pretty easily for the 'U-rotated Bell basis':

$$|\alpha\rangle \otimes |\Phi_{00}\rangle = \sum_{a,b\in\{0,1\}} |\Phi\,(U)_{ab}\rangle \otimes X^a Z^b U|\alpha\rangle \tag{A3}$$

where U is any single qubit gate.

Equation (A3) then describes 'gate teleportation'. Alice and Bob first share an entangled EPR pair $|\Phi_{00}\rangle$, then Alice prepares a state $|\alpha\rangle$ and performs a 'U-rotated Bell measurement'. The U-rotated Bell basis is selected as the measurement basis in the quantum measurement on the two qubits she has, $|\alpha\rangle$ and one of the pair $|\Phi_{00}\rangle$. She obtains the results a and b from the measurement, and Bob's qubit transforms into the state $X^a Z^b U|\alpha\rangle$, so once Alice tells Bob the results of her measurement, Bob can apply the Pauli X and Z operators as necessary to obtain $U|\alpha\rangle$. 'Gate teleportation' is represented in figure 10.

Gate teleportation is an extension of quantum teleportation since if U were the identity then $\Phi(U)$ would be the standard Bell basis, and gate teleportation would be the standard quantum teleportation protocol.

This gate teleportation idea is the basis of Liang's QHE schemes, since it allows the correction of the error that results from homomorphically evaluating a T gate.

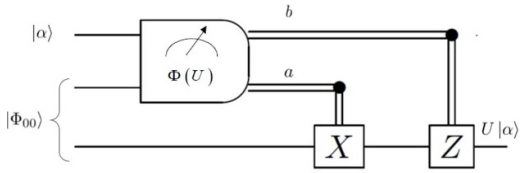

Figure 10. Quantum circuit for 'gate teleportation'. The box represents the quantum measurement Alice performed on $|\alpha\rangle$ and one qubit of the pair $|\Phi_{00}\rangle$. The measurement basis is the U- rotated Bell basis $\Phi(U)$. Depending on the results of the measurement, a and b, Bob applies X^a and Z^b in order to obtain $U|\alpha\rangle$.

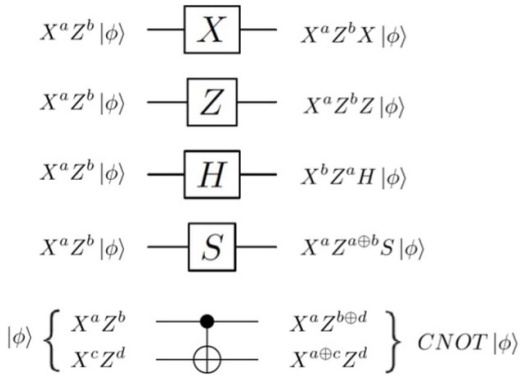

Figure 11. Key updating rules for homomorphic evaluation of Clifford gates.

A.3. Encryption and evaluation

The first step of Liang's schemes is applying QOTP to the data that will be sent to the server. This means applying a combination of X^a and Z^b to each qubit, where a and b are random bits selected from $\{0,1\}$ which constitute the secret key of the client sk. If the plaintext data has n qubits the secret key sk has $2n$ bits $sk=(a_0,b_0),a_0,b_0\in\{0,1\}^n$. The w-th plaintext qubit is encrypted using the secret bits $(a_0(w);b_0(w))$ so: $|\alpha\rangle\rightarrow X^{a_0}Z^{b_0}|\alpha\rangle=|\rho_0\rangle$. QOTP was proposed by Boykin and Roychowdhury in [24], where they proved that as long as a and b are randomly selected from $\{0,1\}$ and used only once, QOTP has perfect security.

Once the data has been encrypted it will be sent to the server. There, the server will perform quantum gates from the set $\mathcal{G}=\{X,Z,H,S,\mathrm{CNOT},T,T^\dagger\}$ to complete its designated quantum circuit. If the gates are not T or $T^\dagger$ (just Clifford), the homomorphic evaluation con be performed easily. When the server performs one of these gates on a qubit, the key is updated according to the algorithm shown in figure 11. After the jth $(1\leqslant j\leqslant N-1)$ gate is performed, a new key (denoted as $(a_j,b_j),a_j,b_j\in\{0,1\}^n$) can be obtained through key updating; this is the intermediate key. As an example applying an H gate would transform the key of the qubit as $(a_1,b_1)=(b_0,a_0)$ assuming it was the first gate in the circuit.

If the gate is a $T/T^\dagger$, its homomorphic evaluation is done using the S^a-rotated Bell measurement in order to correct the S-error that we have already mentioned. At the start of the evaluation of the whole circuit, an EPR source generates M Bell states (as many as $T/T^\dagger$ gates are in the circuit) $\{|\Phi_{00}\rangle_{c_i,s_i},i=1,\ldots,M\}$, where qubits $c_i,i=1,\ldots,M$ and qubits $s_i,i=1,\ldots,M$ are kept by the client and the server respectively. When the server has to evaluate a $T/T^\dagger$ gate,

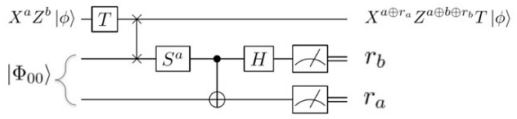

Figure 12. Homomorphic evaluation of T gate. $|\Phi_{00}\rangle$ represents an EPR pair.

it first applies the gate to a qubit, then performs a SWAP gate between the encrypted qubit and one of the entangled qubits the server has in its position, s_i. Then the only step remaining is to apply a S^a-rotated Bell measurement on the qubits s_i and c_i. From this measurement, r_a and r_b will be obtained, so the encryption key can be updated. From $TX^aZ^b|\alpha\rangle$, this whole process returns $X^{a\oplus r_a}Z^{a\oplus b\oplus r_b}T|\alpha\rangle$. If the gate evaluated is a $T^\dagger$ gate the key will be updated from $T^\dagger X^a Z^b|\alpha\rangle$ to $X^{a\oplus r_a}Z^{b\oplus r_b}T^\dagger|\alpha\rangle$. It is graphically represented in figure 12. This process will be performed M times, as many as $T/T^\dagger$ gates are in the circuit.

Due to the principle of deferred measurement, the M measurements can be postponed until the server has finished all the quantum operations. Then, the server sends all the encrypted qubits, the key-updating functions based on the rules of figure 11 and all the ancillary s_i qubits to the client. Then the client will update his keys accordingly and perform a measurement, alternating between updating and measuring, since to perform the correct S^a-rotated Bell measurement the updated a key is needed. Measurements must be performed in the pre-established order, because the updated key depends on the result of the previous measurement. Once all measurements are completed and all the keys updated, the client obtains the final key $\mathrm{d}k = (a_{\text{final}}, b_{\text{final}}), a_{\text{final}}, b_{\text{final}} \in \{0,1\}^n$. Finally the client can decrypt the qubits which are in the state $|\rho_{\text{final}}\rangle$ (the final state that the server outputs) using his final updated keys to obtain the plaintext result $|\alpha_{\text{final}}\rangle$: $X^{a_{\text{final}}}Z^{b_{\text{final}}}|\rho_{\text{final}}\rangle = |\alpha_{\text{final}}\rangle$.

The whole scheme can be described in five steps: **Setup**, **Key Generation**, **Encryption**, **Evaluation**, **Decryption**.

1. **Setup**: an EPR source generates M Bell states (as many as $T/T^\dagger$ gates are in the circuit) $\{|\Phi_{00}\rangle_{c_i,s_i}, i = 1,\ldots,M\}$, where qubits $c_i, i = 1,\ldots,M$ and qubits $s_i, i = 1,\ldots,M$ are kept by the client and the server respectively.
2. **Key Generation**: Generate random bits a_0, $b_0 \in \{0,1\}^n$ and output $sk = (a_0, b_0)$ as the secret key.
3. **Encryption**: for any n qubit data $|\alpha\rangle$, client performs QOTP encryption with the secret key $sk = (a_0, b_0)$: $|\alpha\rangle \to X^{a_0}Z^{b_0}|\alpha\rangle = |\rho_0\rangle$.
4. **Evaluation**: The server applies the quantum gates in order on the n encrypted qubits and updates the key updating function for each qubit according to the key updating algorithm of figure 11. If the gate is a $T/T^\dagger$ acting on the w-th qubit the server applies a SWAP gate on the w-th qubit and one of the entangled qubits s_i. After the last gate is applied the server sends all the encrypted qubits, the key-updating functions based on the rules of figure 11 and all the ancillary s_i qubits to the client.
5. **Decryption**: the client uses the key updating functions to obtain the corresponding a to the first S^a-rotated Bell measurement and measures in the correct basis, obtains r_a and r_b and updates his keys, then repeats the process until there are no more measurements to be made and obtains the final key $\mathrm{d}k = (a_{\text{final}}, b_{\text{final}})$. Finally the client performs QOTP decryption on the encrypted qubits to obtain the final states: $X^{a_{\text{final}}}Z^{b_{\text{final}}}|\rho_{\text{final}}\rangle = |\alpha_{\text{final}}\rangle$.

Therefore, this procedure can implement any quantum circuit homomorphically ($\mathcal{F}$-homomorphic), with perfect security since the server can never learn any information about

the plaintext or the evaluation keys at any point: the data the server receives is encrypted with QOTP so it is perfectly secure, there are no interactions in the evaluation process so the server can not learn anything about the data there either and once it sends the data to the client for the decryption process it is impossible to obtain any more information.

From all this process the importance of the number of T gates in the circuit becomes clear, because unlike the Clifford gates, they make the complexity of the decryption process grow due to the quantum measurements the client has to perform in succession. This is the reason why the discussed schemes are only suitable for circuits with a polynomial number of $T/T^\dagger$ gates. In section 2 it is proved that the simplified circuits we have constructed for the recursive BV problem for $k=2$ are a perfect example of an algorithm that can be evaluated efficiently using Liang's QHE schemes.

Appendix B. Review of the BV algorithm

In the BV problem a n bit function $f\colon \{0,1\}^n \to \{0,1\}$ is given. The function obeys the relation: $f_s(x) = s\cdot x = x_1s_1 + x_2s_2 + \ldots + x_ns_n \mod(2)$ where s is an unknown n bit string and the objective is to find s. The best classical algorithm to solve this problem in the exact query complexity model takes n queries since the function only returns one bit, so the best algorithm is simply using as input a bit string where all bits are a 0 except the ith bit which would be a 1 in order to obtain s_i from the function. If bounded error is allowed, suppose the function could be queried in a probabilistic way so all of the bits of s could be found in less than n queries with some probability of failure that is bounded below 1/2. This bounded error algorithm would still need $\Omega(n)$ queries [8]. This contrasts with the Deutsch–Jozsa algorithm because a classical probabilistic algorithm makes the quantum advantage disappear.

Regarding the quantum algorithm, the main idea is implementing $f_s(x)$ in a reversible way using a quantum oracle:

$$U_s = \sum_{x\in\{0,1\}^n}\sum_{y\in\{0,1\}} |x\rangle\langle x| \otimes |y\oplus(s\cdot x)\rangle\langle y|. \tag{B1}$$

The quantum algorithm uses n qubits all starting in the $|0\rangle$ state. It begins by applying n H gates, one to each qubit. It will result in an equal superposition of all possible n-bit strings:

$$H^{\otimes n}|00\ldots0\rangle = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}|x\rangle. \tag{B2}$$

This will be the input for the oracle. The oracle will be implemented using an extra qubit in the state $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. If a CNOT is applied to the $|-\rangle$ state as the target qubit, a phase kickback will occur where the control qubit will change its sign and the target qubit will remain unchanged. This is because if a CNOT is applied to the $|-\rangle$ state with $|1\rangle$ as the control qubit: $\mathrm{CNOT}[|1\rangle\otimes\frac{|0\rangle-|1\rangle}{\sqrt{2}}] = |1\rangle\otimes\frac{|1\rangle-|0\rangle}{\sqrt{2}} = |1\rangle\otimes(-\frac{|0\rangle-|1\rangle}{\sqrt{2}}) = -|1\rangle\otimes\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. This means the phase of the terms where $s\cdot x = 1$ will be flipped so after applying U_s:

$$|\phi\rangle_s = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{f_s(x)}|x\rangle = \frac{1}{\sqrt{2^n}}\sum_{x\in\{0,1\}^n}(-1)^{s\cdot x}|x\rangle. \tag{B3}$$

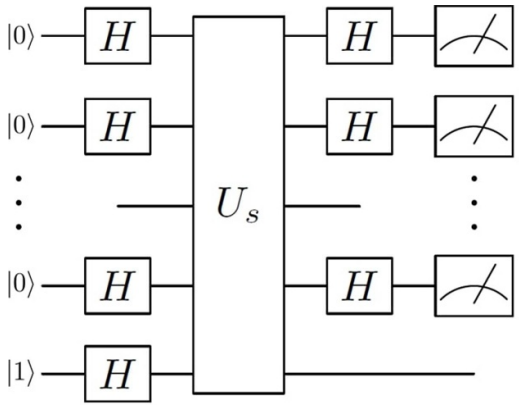

Figure 13. Nonrecursive BV circuit.

Since the action of the Hadamard gate on a general computational basis state of a n-qubit system is given by:

$$H^{\otimes n}|u\rangle = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{u\cdot x}|x\rangle. \tag{B4}$$

and the H gate is its own inverse:

$$H^{\otimes n}\left[\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{u\cdot x}|x\rangle\right] = |u\rangle. \tag{B5}$$

Then if we now apply again n H gates, according to equation (B5) the state of the qubits will be $|s\rangle$:

$$H^{\otimes n}|\phi\rangle_s = H^{\otimes n}\left[\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{s\cdot x}|x\rangle\right] = |s\rangle. \tag{B6}$$

Finally once the qubits are measured s will be obtained with a 100% probability. The algorithm finds the value of s in just one query. The circuit that solves the BV problem is represented in figure 13.

In the nonrecursive BV problem, we are given a function $f_s(x) = s\cdot x$ and s has to be found. In the recursive BV problem, what has to be found is some function $g:\{0,1\}^n \to \{0,1\}$ on s, $g(s)$. We will refer to all the possible n bits strings that can be used as inputs to the function g by $s_{\text{in}} \in \{0,1\}^n$.

To construct the first level of the recursion, an oracle based on two n bits strings $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ is used. As in the nonrecursive BV algorithm the function queried takes these two bit strings as input and outputs $s_x \cdot y$, so the function can be expressed as $f:\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ given by $f(x,y) = s_x \cdot y$. The s_x are 2^n different bit strings labelled by $x \in \{0,1\}^n$ with one important property: the value of $g(s_x)$ has to coincide with the product of $s\cdot x$ for some unknown s. The two level BV problem then is to identify this s and $g(s)$.

When $f(x,y)$ is queried, $s_x \cdot y$ is obtained. To identify s_x for a fixed x, the original nonrecursive problem would have to be solved. Then once s_x has been obtained for a fixed x, s_x has to be used to calculate $g(s_x)$. However each $g(s_x)$ for different values of x is now part of a nonrecursive BV problem, so this way the problem becomes more complex.

This process of creating the two level BV problem can be done k times to create the k level BV problem. At the kth level the function f takes as input k strings of n bits denoted by x_i and computes $f(x_1, x_2, \ldots, x_k) = x_k \cdot s_{x_1,x_2,\ldots,x_{k-1}}$. The secret bit string $s_{x_1,x_2,\ldots,x_{k-1}} \in \{0,1\}^n$ produces the next lower level problem since $g(s_{x_1,x_2,\ldots,x_{k-1}}) = x_{k-1} \cdot s_{x_1,x_2,\ldots,x_{k-2}}$ where as before $s_{x_1,x_2,\ldots,x_{k-2}} \in \{0,1\}^n$. This process can be continued until the final level is reached where $g(s_{x_1}) = s \cdot x_1$. At this level s can finally be found.

We will be discussing the recursive BV problem for $k = 2$ from this point onwards.

We will first solve the problem for an arbitrary case in $k = 2$. As in the usual BV problem, we are given access to a unitary which computes in a reversible manner $f(x_1, x_2) = s_{x_1} \cdot x_2$. Using the usual BV algorithm the value of a s_{x_1} for a fixed x_1 would be easy to find. The tricky part of the recursive BV problem is that when g is applied to these s_{x_1}'s we get a new BV problem to solve. This implies that we have to use the BV algorithm over all x_1's as well.

The oracle for the two level problem is:

$$U_s = \sum_{x_1,x_2 \in \{0,1\}^n} \sum_{y \in \{0,1\}} |x_1\rangle_1 \langle x_1|_1 \otimes |x_2\rangle_2 \langle x_2|_2 \otimes |y \oplus (s_{x_1} \cdot x_2)\rangle \langle y|. \tag{B7}$$

If we start with two quantum registers, x_1 and x_2 with all their qubits in the $|0\rangle$ state and apply H gates to both of them we will get the usual state containing the superposition over all possible bitstrings on the x_1 and x_2 registers:

$$H^{\otimes 2n}|00\ldots0\rangle = \frac{1}{2^n} \sum_{x_1,x_2 \in \{0,1\}^n} |x_1\rangle_1 \otimes |x_2\rangle_2. \tag{B8}$$

The two registers are labelled by a 1 and 2 subscript respectively. If we use the usual phase kick-back trick using a qubit in the $|-\rangle$ state as the target qubit we can implement the oracle U_s and we will have the following state:

$$\frac{1}{2^n} \sum_{x_1,x_2 \in \{0,1\}^n} (-1)^{s_{x_1} \cdot x_2} |x_1\rangle_1 \otimes |x_2\rangle_2 \otimes |-\rangle. \tag{B9}$$

By performing the n qubit Hadamard gate on the second register, we see that it will transform this state to:

$$\frac{1}{\sqrt{2^n}} \sum_{x_1 \in \{0,1\}^n} |x_1\rangle_1 \otimes |s_{x_1}\rangle_2 \otimes |-\rangle. \tag{B10}$$

We now must apply g to s_{x_1} to solve the next order BV problem, since $g(s_{x_1}) = s \cdot x_1$. The oracle for g is given by:

$$G = \sum_{x \in \{0,1\}^n} \sum_{y \in \{0,1\}} |x\rangle_2 \langle x|_2 \otimes |y \oplus g(x)\rangle \langle y| \tag{B11}$$

If we attach an extra ancilla qubit in the state $|-\rangle$, we can use the phase kick-back trick to use this unitary to turn equation (B10) into:

$$\frac{1}{\sqrt{2^n}} \sum_{x_1 \in \{0,1\}^n} (-1)^{g(s_{x_1})} |x_1\rangle_1 \otimes |s_{x_1}\rangle_2 \otimes |-\rangle \otimes |-\rangle. \tag{B12}$$

This oracle acts only on the second register and the second extra qubit in the $|-\rangle$ state. Naturally, just one qubit in the $|-\rangle$ state is enough to implement both U_s and G, we are using

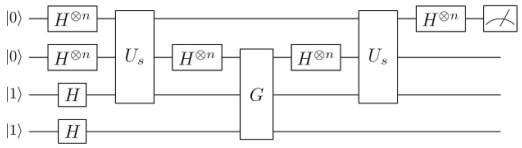

Figure 14. Recursive Bernstein–Vazirani circuit for level-2 recursion.

two only to separate each oracle more clearly. Next, we need to apply the n qubit Hadamard to the second register and then we apply our oracle U_s again. After applying the H gates:

$$\frac{1}{2^n}\sum_{x_1,x_2\in\{0,1\}^n}(-1)^{g\left(s_{x_1}\right)}(-1)^{s_{x_1}\cdot x_2}|x_1\rangle_1\otimes|x_2\rangle_2\otimes|-\rangle\otimes|-\rangle. \tag{B13}$$

After applying U_s we get another phase kickback:

$$\begin{aligned}&\frac{1}{2^n}\sum_{x_1,x_2\in\{0,1\}^n}(-1)^{g\left(s_{x_1}\right)}(-1)^{s_{x_1}\cdot x_2}(-1)^{s_{x_1}\cdot x_2}|x_1\rangle_1\otimes|x_2\rangle_2\otimes|-\rangle\otimes|-\rangle\\&\quad=\frac{1}{2^n}\sum_{x_1,x_2\in\{0,1\}^n}(-1)^{g\left(s_{x_1}\right)}|x_1\rangle_1\otimes|x_2\rangle_2\otimes|-\rangle\otimes|-\rangle.\end{aligned} \tag{B14}$$

Then, we can apply H gates to all the qubits in the first register so we get the following state:

$$\frac{1}{\sqrt{2^n}}\sum_{x_2\in\{0,1\}^n}|s\rangle_1\otimes|x_2\rangle_2\otimes|-\rangle\otimes|-\rangle. \tag{B15}$$

Finally, by measuring the first register we will get s with 100% probability. The recursive BV circuit represented graphically is shown in figure 14.

ORCID iDs

Pablo Fernández https://orcid.org/0009-0008-5113-3248
Miguel A Martin-Delgado https://orcid.org/0000-0003-2746-5062

References

[1] Shor P W 1994 Algorithms for quantum computation: discrete logarithms and factoring *Proc. 35th Annual Symp. on Foundations of Computer Science* (IEEE Computer Society Press) pp 124–34
[2] Grover L K 1996 A fast quantum mechanical algorithm for database search *Proc. 28th Annual ACM Symp. on the Theory of Computing* (*Philadelphia, Pennsylvania, USA*) pp 212–9
[3] Deutsch D 1985 Quantum theory, the Church-Turing principle and the universal quantum computer *Proc. R. Soc. Lond.* A **400** 97–117
[4] Deutsch D and Jozsa R 1992 Rapid solution of problems by quantum computation *Proc. R. Soc.* A **439** 553–8
[5] Bernstein E and Vazirani U 1993 Quantum complexity theory *Proc. 25th Annual ACM Symp. on Theory of Computing* pp 11–20
[6] Xie H and Yang L 2019 Using Bernstein-Vazirani algorithm to attack block ciphers *Des. Codes Cryptogr.* **87** 1161–82
[7] Cross A W, Smith G and Smolin J A 2015 Quantum learning robust against noise *Phys. Rev.* A **92** 012327

[8] Bacon D 2006 *The Recursive and Nonrecursive Bernstein-Vazirani Algorithm (Lecture Notes, CSE 599d—Quantum Computing)* (University of Washington) (available at: https://courses.cs.washington.edu/courses/cse599d/06wi/lecturenotes7.pdf)
[9] Gentry C 2009 A fully homomorphic encryption scheme *PhD Thesis* Stanford University
[10] Brakerski Z, Gentry C and Vaikuntanathan V 2012 (Leveled) fully homomorphic encryption without bootstrapping *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* 309–25
[11] Liang M 2015 Quantum fully homomorphic encryption scheme based on universal quantum circuit *Quantum Inf. Process.* **14** 2749–59
[12] Rohde P P, Fitzsimons J F and Gilchrist A 2012 Quantum walks with encrypted data *Phys. Rev. Lett.* **109** 150501
[13] Zeuner J, Pitsios I, Tan S H, Sharma A N, Fitzsimons J F, Osellame R and Walther P 2021 Experimental quantum homomorphic encryption *npj Quantum Inf.* **7** 25
[14] Tan S H, Kettlewell J A, Ouyang Y, Chen L and Fitzsimons J F 2016 A quantum approach to homomorphic encryption *Sci. Rep.* **6** 33467
[15] Yu L, Pérez-Delgado C A and Fitzsimons J F 2014 Limitations on information-theoretically-secure quantum homomorphic encryption *Phys. Rev.* A **90** 050303
[16] Lai C Y and Chung K M 2018 On statistically-secure quantum homomorphic encryption *Quantum Info. Comput.* **18** 785–94
[17] Broadbent A and Jeffery S 2015 Quantum homomorphic encryption for circuits of low T-gate complexity *Advances in Cryptology – CRYPTO 2015* (*Berlin, Heidelberg*) pp 609–29
[18] Liang M 2020 Teleportation-based quantum homomorphic encryption scheme with quasi-compactness and perfect security *Quantum Inf. Process.* **19** 28
[19] Gong C, Du J, Dong Z, Guo Z, Gani A, Zhao L and Qi H 2020 Grover algorithm-based quantum homomorphic encryption ciphertext retrieval scheme in quantum cloud computing *Quantum Inf. Process.* **19** 105
[20] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press) (https://doi.org/10.1017/CBO9780511976667)
[21] Galindo A and Martín-Delgado M A 2002 Information and computation: Classical and quantum aspects *Rev. Mod. Phys.* **74** 347
[22] Bennett C H, Brassard G, Crépeau C, Jozsa R, Peres A and Wootters W K 1993 Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels *Phys. Rev. Lett.* **70** 1895
[23] Jozsa R 2005 An introduction to measurement based quantum computation (arXiv:quant-ph/0508124v2)
[24] Boykin P O and Roychowdhury V 2003 Optimal encryption of quantum bits *Phys. Rev.* A **67** 042317