Implementing the Grover algorithm in homomorphic encryption schemes

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, Boadilla del Monte, 28660 Madrid, Spain*

(Received 12 March 2024; accepted 21 September 2024; published 6 November 2024)

We apply quantum homomorphic encryption (QHE) schemes suitable for circuits with a polynomial number of $T+T^\dagger$ gates to Grover's algorithm, performing a simulation in Qiskit of a Grover circuit that contains three qubits. The $T+T^\dagger$-gate complexity of Grover's algorithm is also analyzed in order to show that any Grover circuit can be evaluated homomorphically in an efficient manner. We discuss how to apply these QHE schemes to allow for the efficient homomorphic evaluation of any Grover circuit composed of n qubits using $n-2$ extra ancilla qubits. We also show how the homomorphic evaluation of the special case where there is only one marked item can be implemented using an algorithm that makes the decryption process more efficient compared with the standard Grover algorithm.

DOI: 10.1103/PhysRevResearch.6.043109

I. INTRODUCTION

In recent decades, the interest in the field of quantum computation has increased greatly due to the possible benefits it could offer compared with classical computation, since the classical solution to many problems cannot be improved past a certain limit. Making use of different quantum phenomena such as superposition and entanglement, quantum algorithms, including hybrid ones, have shown significant speed-ups to many of these problems beyond what was predicted by classical computation. Some of the first examples of such algorithms are Bernstein and Vazirani's algorithm [1] or Shor's factorization algorithm [2]. The search for more quantum algorithms and protocols continues along the developments in current quantum technologies.

Historically, the first indication of the possible advantage a quantum computer could have over a classical one came from the algorithms proposed by Deutsch [3] and Deutsch-Jozsa [4]. In the latter work, a certain problem could be solved in one query using a quantum computer deterministically. On the other hand, this problem would require an exponential number of queries on a classical computer that used an exact model where no probability of failure was allowed. Nevertheless, if a bounded error was permitted in the algorithm, this advantage disappears [5].

After these algorithms, Bernstein and Vazirani created a quantum algorithm that solved a problem with a lower query complexity than what any classical computer could perform. The Bernstein-Vazirani (BV) [1] algorithm constitutes one of the first examples that showed the possible benefits a quantum computer could posses over a classical one. The most famous version of the problem is its nonrecursive version, which provided a polynomial speed-up regarding the query complexity. This query complexity was improved from the best classical algorithm that solved the problem in $O(n)$ queries to the quantum algorithm that had $O(1)$ for its query complexity. Using this algorithm as the basis they managed to construct a recursive version of the problem known as the recursive Bernstein-Varizani algorithm. This algorithm has a superpolynomial speed-up compared with the best classical algorithms.

One of the most successful algorithms in quantum computing is Grover's search algorithm [6]. This algorithm showed the advantage quantum computers have over any classical ones in the unstructured database search problem. For a database with N elements the best classical algorithm has complexity $O(N)$ whereas Grover's has complexity $O(\sqrt{N})$. This constitutes a quadratic speedup. Grover's algorithm has been studied extensively. In Ref. [7] a family of quantum search algorithms is investigated, showing that Grover's algorithm holds a distinguished place in this family. The algorithm has also been used as the basis for different quantum algorithms such as the collision problem [8] and quantum counting [9]. Surprisingly it also shares an isomorphism with classical kinematics collisions [10]. This algorithm is one of the two building blocks of this article. The other is quantum homomorphic encryption.

In the current internet era, plenty of users take advantage of the services offered by the cloud, uploading enormous quantities of private data for many different services such as storage. Thus, ensuring the security of any computation performed on the cloud is a task of great importance. Similarly to how modern cryptography is used in order to protect the security of communications, any computations performed on the cloud should be protected using cryptographic technology that preserves the data's security. In this regard, a key

*Contact author: pabfer23@ucm.es
†Contact author: mardel@ucm.es

technology that has been developed in recent years is homomorphic encryption. It allows a client to encrypt data and then send it to a server that performs operations on the encrypted data. Once the computations are finished, the server returns the data back so it can be decrypted. This way the client receives the evaluated data and the server never obtains any information about the actual data it operated with. The first classical fully homomorphic encryption scheme (FHE) was created by Gentry [11] in 2009. Since then, more classical schemes have been proposed and perfected. These schemes are computationally secure, which means that their security is based on the difficulty of solving mathematical problems, like ideal lattices [11] or the learning with errors problem (LWE) [12].

Regarding quantum computing, different types of technologies have been proposed to construct quantum computers, such as superconducting qubits [13–19]. Currently, real quantum computers have already been developed and some of them are available in the cloud, such as the IBM Q quantum computers. Researchers have performed plenty of experiments and different types of protocols on them, like obtaining a measurement of the topological Uhlmann phase for a topological insulator [20]. It is expected that most quantum computers will be accessible through the cloud in the near future. To preserve their security, different quantum homomorphic encryption (QHE) schemes have been developed. QHE allows a remote server to apply a quantum circuit, denoted by QC, on the encrypted quantum data $Enc(\rho)$ that a client has provided. Next, the client decrypts the server's output and obtains the result of the quantum circuit, $QC(\rho)$. Contrary to the classical protocols, the security of QHE schemes is based on the fundamental properties of quantum mechanics instead of the difficulty of solving complex mathematical problems.

QHE schemes can be classified as either interactive or noninteractive. If the client and server communicate during the execution of the circuit, then the scheme is said to be interactive. If interactions are allowed the efficiency of the scheme decreases since the server must wait until the client finishes some operations before it can continue executing the scheme. For this reason interactive schemes are easier to construct than noninteractive ones. As an example of these kinds of interactive protocols, a QHE scheme where the number of interactions between client and server is the same as the number of T gates contained in the evaluated circuit was proposed by Liang [21].

Nevertheless, we are more concerned with studying noninteractive schemes. Some definitions about different QHE schemes properties are given below. If a QHE scheme is $\mathcal{F}$ homomorphic then it can evaluate any quantum circuit homomorphically. A QHE scheme is said to be compact if the complexity of its decryption procedure does not depend on the evaluated circuit. Also, for a scheme to be considered a quantum fully homomorphic encryption (QFHE) scheme, it must be $\mathcal{F}$ homomorphic and compact.

The first proposals of QHE schemes began in 2012. A limited symmetric-key QHE scheme using the boson sampling and multiwalker quantum walks was proposed by Rohde [22]. This scheme only allows the homomorphic evaluation of certain circuits because it is not $\mathcal{F}$ homomorphic. It was later implemented experimentally by Zeuner *et al.* in Ref. [23]. Liang [24] defined QFHE and proposed a weak scheme that allows local quantum computations to be performed on encrypted quantum data. Nevertheless, since its evaluation algorithm depends on the secret key that the client has, its application is very limited. A QHE scheme that enables a large class of quantum computations on encrypted data was proposed by Tan [25]. However it cannot provide security in a cryptographic sense because it can only hide a constant fraction of the encrypted information.

Some researchers have also explored questions regarding theoretical limits on QHE schemes. A no-go result stating that any QFHE with perfect security must produce exponential storage overhead was proved by Yu *et al.* in Ref. [26]. Constructing an efficient QFHE scheme with perfect security is impossible for this reason. An enhanced no-go result stating that constructing a QFHE scheme that is both information theoretically secure (ITS) and noninteractive is impossible was proved by Lai and Chung [27]. Thus, the best security a noninteractive QFHE scheme can achieve is computational security. Besides this no-go result, Lai and Chung constructed a compact, noninteractive ITS QHE scheme. It is not $\mathcal{F}$ homomorphic due to their no-go result so it is just a quantum somewhat homomorphic encryption (QSHE) scheme.

Due to the no-go results mentioned no QHE scheme can have noninteraction, compactness, $\mathcal{F}$ homomorphism and perfect security simultaneously. Taking this into account different schemes have been proposed with looser conditions. For example, perfect security can be obtained if the QHE scheme is interactive, like in the one proposed by Liang [21] mentioned previously. Another possibility is downgrading the security level from perfect security to computational security, as done by Broadbent and Jeffery [28] and Dulek *et al.* [29]. Combining quantum one-time pad (QOTP) and classical FHE, two noninteractive quantum homomorphic schemes were constructed by Broadbent and Jeffery. Their security and efficiency are limited by classical homomorphic encryption schemes. Similarly, the scheme proposed by Dulek *et al.* is also noninteractive and depends on the construction of an ancillary gadget which is based on classical FHE. Due to this, the scheme's security is also limited to computational security.

Liang [30] constructed two perfectly secure and noninteractive QHE schemes that are $\mathcal{F}$ homomorphic but not compact. These schemes are quasicompact which means that the complexity of the decryption procedure scales sublinearly in the size of evaluated circuit as defined by Broadbent and Jeffery in Ref. [28]. The first scheme constructed in Ref. [28] is known as EPR (Einstein, Podolsky, and Rosen [31]). Its properties are quasicompactness, $\mathcal{F}$ homomorphism, noninteraction and computational security. EPR was proved to be M^2 quasicompact where M is the number of T gates in the evaluated circuit, which means the complexity of its decryption procedure scales with the square of the number of T gates contained in the evaluated circuit. Both Broadbent's and Liang's schemes make use of Bell states and quantum measurements. However, Liang's schemes [30] are superior regarding security since they are perfectly secure whereas EPR is only computationally secure. Furthermore one of Liang's schemes, named VGT, is M quasicompact, so it is also superior in that aspect. Since these schemes are quasicompact, they do not contradict the no go-result mentioned previously.

The main feature of Liang's schemes [30] is that despite being quasicompact, they allow the efficient homomorphic evaluation of any quantum circuit with low $T+T^\dagger$ gate complexity with perfect security. For this reason they are suitable for circuits with a polynomial number of $T+T^\dagger$ gates. On the other hand, the decryption procedure would be inefficient for circuits that contain an exponential number of $T+T^\dagger$ gates. Gong *et al.* [32] have used Liang's schemes to implement a ciphertext retrieval scheme based on the Grover algorithm. They performed experiments using Qiskit and IBM's quantum computers as an example of the scheme. A quantum ciphertext dimension reduction scheme for homomorphic encrypted data based on Liang's schemes was also constructed by Gong *et al.* [33]. In Ref. [34], Liang's schemes are applied to the recursive Bernstein-Vazirani algorithm, showing different cases in which the QHE schemes are efficient.

In this paper we propose a homomorphic implementation of Grover's algorithm using Liang's [30] schemes. Our proposal allows the homomorphic evaluation of any Grover circuit efficiently, provided we have access to $n-2$ ancilla qubits besides the usual n qubits used in a Grover search. As an example a homomorphic Grover circuit for three qubits has been simulated using IBM's quantum computers. Compared with Gong *et al.* [32], the Grover search simulations they performed were made on a two-qubit circuit that only needed Clifford gates. Our simulation of the homomorphic evaluation of a Grover circuit is more complex because it contains T and $T^\dagger$ gates. We also discuss the $T+T^\dagger$-gate complexity of the algorithm unlike previous works [32]. The correct states can be decrypted after the evaluation. We also show that the quantum search algorithms for a database with only one marked item proposed by Arunachalam and de Wolf [35] and Briański *et al.* [36], which reduce the number of quantum gates needed to solve the problem, can be homomorphically evaluated in a more efficient manner than Grover's algorithm. Finally, we study the efficiency of the homomorphic evaluation of more practical quantum search algorithms used in quantum key attacks for symmetric encryption schemes.

The paper is organized as follows: In Sec. II, the simulation performed in Qiskit is shown. Then in Sec. III our proposal for the general homomorphic evaluation of Grover's algorithm is discussed. Finally, the conclusions are summarized in Sec. IV. In Appendix A Liang's quantum homomorphic encryption schemes are reviewed. In Appendix B the Grover algorithm is reviewed.

II. HOMOMORPHIC GROVER SIMULATION IN QISKIT

A. Qiskit setup

In this section a simulation of a homomorphic evaluation of a Grover circuit is shown. The simulation has been performed in IBM Qiskit [37].

The client wants to solve the unstructured search problem using Grover's algorithm. It wants to do it without the server learning anything about the data so it decides to use quantum homomorphic encryption to preserve its security.

The circuit that the client wants to evaluate is shown in Fig. 1. The qubits are denoted by $q_0, q_1, \ldots, q_{n-1}$. Notice that in Qiskit the states are represented as $|q_{n-1}\ldots q_1 q_0\rangle$ so the

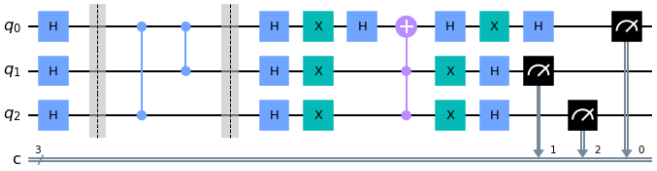

FIG. 1. The quantum circuit that the client wants to evaluate. It implements a Grover search for $n = 3$ qubits, $N = 8$ elements, and $m = 2$ marked items in Qiskit.

circuit in Fig. 1 should be read starting from q_2 and finishing at q_0. The length of the list is $N = 8$ and so $n = \log_2 N = 3$ qubits are needed. This circuit finds two elements in the list: the states $|011\rangle$ and $|101\rangle$ (recall that Qiskit notation is being used here). Then Grover's algorithm for two marked elements is applied. The reason why this circuit was selected is that for $N = 8$ and $m = 2$ only one iteration is needed and the circuit still requires the use of T and $T^\dagger$ gates. This makes the homomorphic evaluation not trivial but still simple enough to be simulated.

All the steps of the circuit are separated by a gray barrier in the figure. The first step applies $3H$ gates to the qubits. The next step is the application of the oracle U_w, so the states $|011\rangle$ and $|101\rangle$ are marked using two controlled-Z gates. The third step is the application of the diffusion operator U_s using a CCZ gate (a controlled Z gate that has two control qubits and one target qubit) surrounded by H and X gates. In Fig. 1, the CCZ gate has already been decomposed into a Toffoli gate which has its target qubit surrounded by two H gates. Since $Z = HXH$, any Z gate can be implemented using H, X, and H. The reason for this decomposition is that in order to implement this circuit homomorphically, it has to be decomposed into gates of the set $\mathcal{G} = \{X, Z, H, S, \text{CNOT}, T, T^\dagger\}$. This means that each controlled-Z gate from U_w has to be substituted by a CNOT gate surrounded by two H gates just like the CCZ gate was substituted by a Toffoli gate surrounded by H gates. The last gate that has to be transformed is the Toffoli gate which can be decomposed into seven $T+T^\dagger$ gates as seen in Fig. 2.

After decomposing the two controlled-Z gate, the CCZ gate and the Toffoli gate into gates of the set $\mathcal{G}$, a new circuit represented in Fig. 3 is obtained. This is the circuit that has to be evaluated homomorphically using the QHE scheme described in Appendix A. The total number of gates of the circuit is $l = 35$. The sequence of gates is $G[1] = H_0$, $G[2] = \text{CNOT}_{2,0}, \ldots$ and so on until the last three gates which are $G[33] = H_0$, $G[34] = H_1$, $G[35] = H_2$.

Following the QHE scheme explained the client starts by generating the random bits a_0, $b_0 \in \{0,1\}^3$ to obtain his secret key $sk = (a_0, b_0)$. In the simulation $n = 3$. We chose $a_0 = a_0(0)a_0(1)a_0(2) = 111$ and

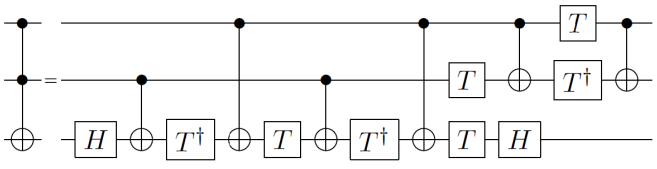

FIG. 2. A Toffoli gate can be decomposed into seven $T+T^\dagger$ gates.

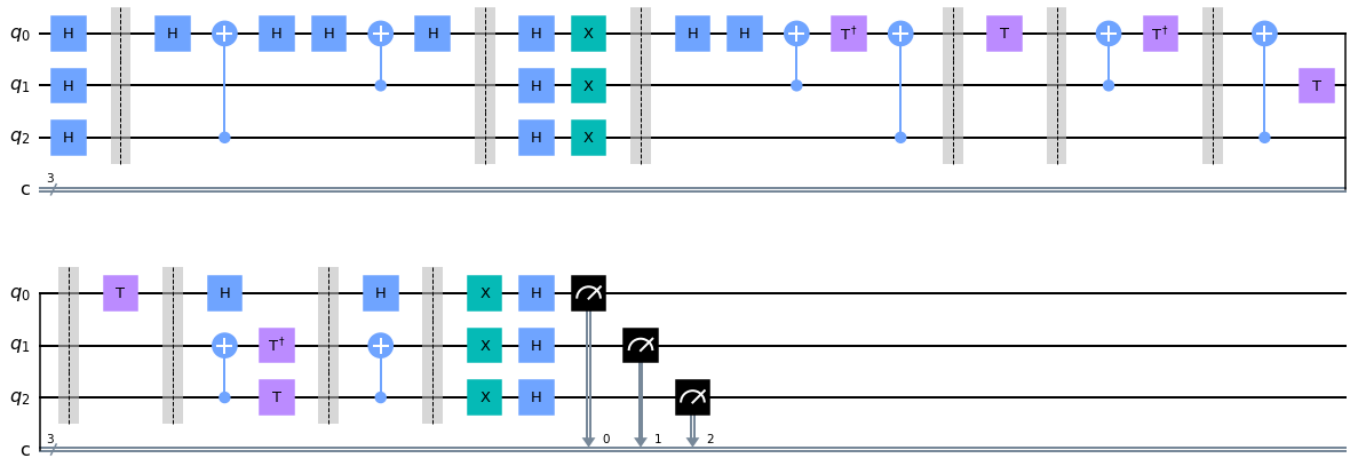

FIG. 3. Compilation of the circuit that the client wants to evaluate decomposed using gates from the set $\mathcal{G}$.

$b_0 = b_0(0)b_0(1)b_0(2) = 111$ where $a_0(n-1)$ and $b_0(n-1)$ refer to the values of the key for q_{n-1}. This way we have that for q_0 the keys are $(a_0(0), b_0(0)) = (1, 1)$. For q_1 and q_2 the keys are $(a_0(1), b_0(1)) = (1, 1)$ and $(a_0(2), b_0(2)) = (1, 1)$, respectively. Next, the first step of Grover's algorithm (state preparation) is performed using $3H$ gates to obtain the superposition state $|s\rangle$. Then $|s\rangle$ is encrypted using the secret key sk. This means that all the qubits are encrypted using X^1Z^1 since the keys for each qubit are (1,1). Now that the data has been encrypted by the client, it is sent to the server.

The server starts by generating as many EPR pairs as $T+T^\dagger$ are in the circuit. Since $M = 7$ then seven entangled pairs have to be generated. An EPR pair can be easily constructed just using a H gate on the first qubit and a CNOT gate in which the target is the second qubit. Fourteen additional qubits are then used in the simulation.

The simulation we performed contains 17 qubits in total. An image containing such a number of qubits would be too large to show properly so we show an equivalent circuit to the one that we used in our Qiskit simulation in Fig. 4. Besides the Qiskit simulation we also calculated the value of each key before any S^a-rotated Bell measurement is performed. There are some details that need further explanation. As we mentioned, 14 extra qubits are needed to evaluate the seven $T+T^\dagger$ gates. However in Fig. 4 there are only two extra qubits denoted by q_3 and q_4. Instead of using seven EPR pairs and performing seven S^a-rotated Bell measurements, we make use of Qiskit reset operation which is denoted by the gate $|0\rangle$ in the circuit. This gate simply turns the state back to $|0\rangle$, so we can perform a S^a-rotated Bell measurement, use the reset operation on the qubits that were measured and entangle them again using a H gate and a CNOT gate. The reason this is done is simply to obtain a more compact image, because instead of using 17 qubits only five are needed. The number of S^a-rotated Bell measurements remains the same: $M = 7$.

Notice the classically controlled S gates used in the circuit. These gates are denoted by an S gate on the target qubit and a control located in the register denoted by c in the circuit, below q_4. This c register contains the results of all the quantum measurements so it is made of classical bits. An example of these classically controlled S gates are the two gates that act on q_3 after the first T (second $T+T^\dagger$) gate of the circuit. In Appendix A 3 it is explained that in the decryption process the client needs to obtain the corresponding value of a_j before performing the correct S^a-rotated Bell measurement. This means that to perform the correct simulation one cannot simply apply S^1-rotated Bell measurements in all cases since there would be cases where no S gate should be applied in the measurement. Instead each possible measurement result from all the previous S^a-rotated Bell measurements has to be taken into account and used to calculate the corresponding a for the next measurement so the simulation returns the correct values for all possible scenarios. Using the key updating rules explained previously, the a_j relative to a measurement is calculated, one for each possible result of all the previous measurements. Then for each $a_j = 1$ a controlled S gate is used, where the control is a classical bit string sequence stored in c that contains one result of all the previous measurements. As an example notice that for the second $T+T^\dagger$ in Fig. 4 there are two control-S gates. This is because in the previous $T+T^\dagger$ gate four measurement results are obtained. After key updating two of them leads to $a_{17} = 1$ for this second $T+T^\dagger$ gate (which is the 18th gate of the circuit), so two classical bit strings stored in c have to be used as the control for two control-S gates. The third $T+T^\dagger$ in Fig. 4 contains eight control-S gates for the same reasons, only now there are 16 possible measurement results because there are two previous measurements. The second reason why the circuit shown in Fig. 4 is not exactly the circuit that was simulated is the number of these control-S gates that were used. For the last $T+T^\dagger$ of the circuit 2048 control-S gates were used, so the resulting image would be too large to show here. The only two differences between the circuit in Fig. 4 and the simulated circuit are that the latter used 17 qubits with no reset operation and that it contained all the necessary control-S gates in the measurements. Nevertheless the circuit in Fig. 4 is still useful to be presented here because it illustrates all the relevant elements of the simulation: the encryption using X and Z gates, the seven S^a-rotated Bell measurements and some classically control-S gates that are needed in these measurements.

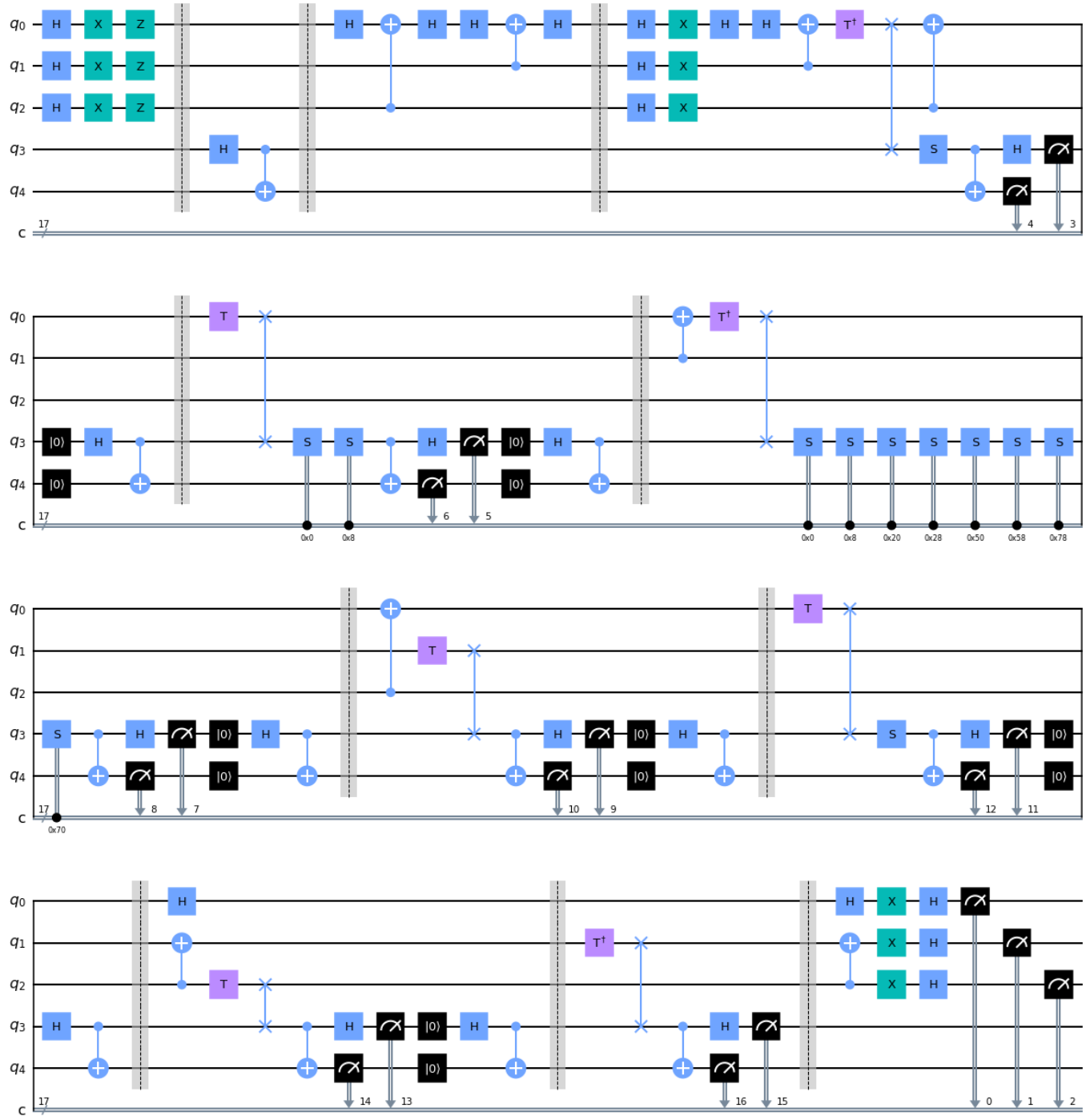

FIG. 4. Simulated circuit in Qiskit. This circuit is a simplified version of the simulation performed in order to make the image more compact.

We want to remark that the main issue about evaluating the T and $T^\dagger$ gates is related to the particular homomorphic encryption scheme used, not to the implementation of the gates on IBMQ. We are already assuming that the T and $T^\dagger$ gates can be implemented with zero error in the QHE scheme. However, all the S^a-rotated Bell measurements must be performed in order by the client and thus the complexity of the decryption process depends necessarily on the number of $T+T^\dagger$ gates contained in the evaluated circuit, even if the T and $T^\dagger$ gates are implemented with zero error.

Now that these details have been properly discussed we can proceed with the remaining steps. After evaluating all the quantum gates in the circuit using the key updating rules explained, the server would send the encrypted qubits back to the client, the key-updating functions and all the entangled qubits. As a remark recall that the seven S^a-rotated Bell measurements in the protocol are performed by the client once he receives all the data from the server and updates its keys accordingly. The last step is measuring q_0, q_1, and q_2 and using the final key dk to decrypt the results.

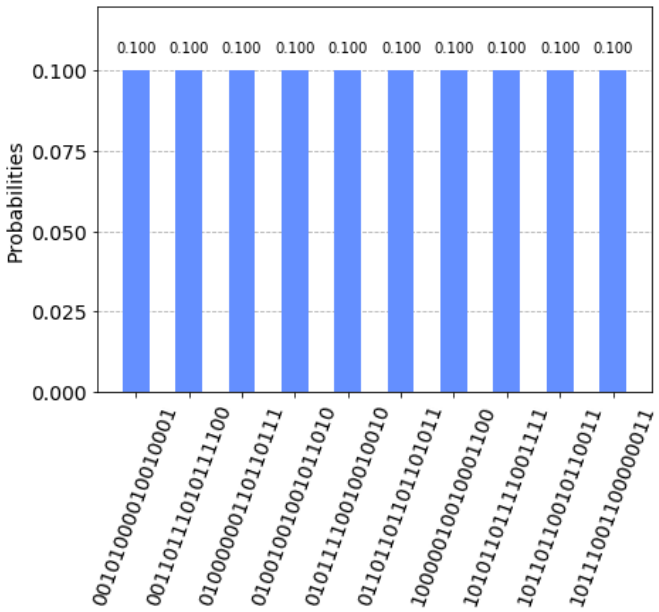

FIG. 5. Results of the three qubits Grover circuit simulation. This histogram shows 10 different measurement results. Each bar contains the results of the seven S^a-rotated Bell measurements and the final encrypted states of qubits q_2, q_1, and q_0.

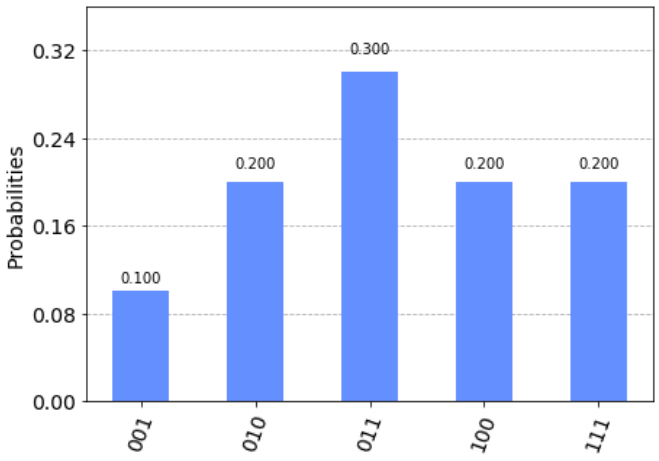

FIG. 6. Results of the three qubits Grover circuit simulation. This histogram shows 10 different measurement results. Only the final encrypted states of qubits q_2, q_1, and q_0 are included.

B. Simulation results

The results from the simulation are shown in Fig. 5. The horizontal axis represents the measurement result obtained and vertical axis shows the probability of obtaining each result. The circuit was executed ten times in order to obtain a compact image. Every state in Fig. 5 is obtained with probability 0.1. Each measurement result contains the results obtained from the seven S^a-rotated Bell measurements and the measurement values of q_0, q_1, and q_2. The first 14 bits in each result represent the results of the S^a-rotated Bell measurements. If we start reading from the left, the first two bits refer to the last measurement and the last two refer to the first. This is because the first Bell measurement is performed on qubits q_4 and q_3. The next S^a-rotated Bell measurement is performed on qubits q_6 and q_5 and so on. Then the last three remaining bits represent q_2, q_1, and q_0. The probability of each state of Fig. 5 is not the main focus here. Instead what we are interested in is obtaining the correct result after decryption given a measurement result of q_0, q_1, and q_2. To show the results obtained for the encrypted qubits more clearly, we also present Fig. 6. The encrypted state 001 was obtained once, the states 010, 100 and 111 were obtained two times and finally the state 011 was obtained three times. These are the measurement results that have to be decrypted.

As an example, if we take the first result from Fig. 5 which is 00101000010010001, the last three bits 001 represent the encrypted value of Grover's algorithm result. The state 001 is not one of the solutions. On the other hand 00101000010010 are the results of the Bell measurements. Using these values and our key updating script based on the rules from Appendix A 3, the final key can be obtained. In this case the final key is $(a_{\text{final}}(0), b_{\text{final}}(0)) = (0, 1)$ for q_0, $(a_{\text{final}}(1), b_{\text{final}}(1)) = (1, 1)$ for q_1, and $(a_{\text{final}}(2), b_{\text{final}}(2)) = (0, 1)$ for q_2 so $dk = (a_{\text{final}}, b_{\text{final}}) = (a_{\text{final}}(0)a_{\text{final}}(1)a_{\text{final}}(2), b_{\text{final}}(0)b_{\text{final}}(1)b_{\text{final}}(2))=(010, 111)$. Finally the result of the algorithm can be unencrypted. Since the measurement of q_0, q_1, and q_2 was performed before decrypting the qubits using the final key, to decrypt a classical result the operation is simply $a_{\text{final}} \oplus r_c$, the bitwise modulo two sum where r_c is the classical bit string obtained from the quantum measurement. Therefore for this example where the encrypted result of Grover's algorithm is $r_c = 001$ we have $010 \oplus 001 = 011$. Recall that for the evaluated Grover circuit the correct results are either 011 or 101, so the result that was obtained is indeed correct.

For the remaining nine results obtained from Fig. 5, the same process was performed to obtain the decrypted result. Table I shows the final unencrypted results along the final key of every qubit for each result obtained from the Qiskit simulation. As it can be seen every simulation result is correctly decrypted into 011 or 101, the desired elements from the Grover search problem. The bit string 011 was obtained seven times and 101 was obtained the remaining three. If the simulation is performed more times, the results get closer to the expected 50% chance of obtaining each state. Therefore the simulation obtains the correct results of the algorithm after decryption for all possible cases, demonstrating the correctness of the QHE scheme.

We want to remark that the physical implementation of the IBMQ gates does not significantly impact the efficiency of the QHE scheme. The physical gates that the IBM quantum computers can implement are limited to some single-qubit gates such as sqrt(X) and one two-qubit gate. On the other hand, the QHE scheme uses the set of gates $\mathcal{G} = \{X, Z, H, S, \text{CNOT}, T, T^\dagger\}$ that can perform universal quantum computation and the scheme assumes that all these gates can be performed in the particular quantum platform that the server is using. The IBMQ platform was used as an example to implement the QHE scheme but the scheme is not restricted to the IBMQ particular platform hardware. In any case even though the IBM quantum computers can only implement a limited number of physical gates, these physical gates

TABLE I. Results from Fig. 5. For each result obtained from the simulation the following information is given: its seven Bell measurements results, the encrypted Grover search result, the corresponding final key for each qubit q_2, q_1, and q_0 and the final Grover search result.

Simulation result	S^a-rotated Bell measurements	Encrypted result	Final q_2 key	Final q_1 key	Final q_0 key	Decrypted result
00101000010010001	00101000010010	001	(0,1)	(1,1)	(0,1)	011
00110111010111100	00110111010111	100	(0,1)	(0,0)	(1,0)	101
01000000110110111	01000000110110	111	(0,0)	(1,0)	(0,1)	101
01001001001011010	01001001001011	010	(0,0)	(0,0)	(1,0)	011
01011110010010010	01011110010010	010	(0,0)	(0,1)	(1,1)	011
01101101101101011	01101101101101	011	(0,1)	(0,1)	(0,0)	011
10000010010001100	10000010010001	100	(1,0)	(1,0)	(1,1)	011
10101101111001111	10101101111001	111	(1,1)	(0,0)	(0,0)	011
10110110010110011	10110110010110	011	(0,1)	(0,1)	(0,0)	011
10111001100000011	10111001100000	011	(1,1)	(1,0)	(0,1)	101

implement all the logical gates contained in the evaluated circuit. Then the number of logical $T+T^\dagger$ gates remains the same and so since the amount of S^a-rotated Bell measurements would be the same, the efficiency of the QHE protocol would not change.

To obtain the number of physical T and $T^\dagger$ gates contained in the whole algorithm, we only need to calculate how many physical gates are needed to implement one T gate and multiply that number by the number of T gates needed for the whole algorithm, so we have $T_{\text{physical}}M$, where T_{physical} is the number of physical IBMQ gates required to construct one T gate and M is the number of T gates contained in the algorithm. In the next section, this value of M for the whole Grover algorithm is obtained.

III. GENERAL HOMOMORPHIC GROVER EVALUATION

A. Grover search

In this section the $T+T^\dagger$-gate complexity of Grover's algorithm is discussed, along with a proposal to evaluate any Grover circuit homomorphically. For this purpose, the oracle and the diffusion operator have to be decomposed into gates from the set $\mathcal{G}=\{X, Z, H, S, \text{CNOT}, T, T^\dagger\}$. We consider the case where there is only one marked element in the list and then generalize for the case with m marked elements.

Starting with the oracle U_w, a quantum state is marked which is the solution of the problem. It can be constructed using a multicontrolled Z gate with $n-1$ control qubits that will add a negative phase only to the desired quantum state. These control qubits should activate only for the desired state. For each qubit in which $|w\rangle$ contains a 0 the corresponding control qubit has to be surrounded by X gates. This is also true for the target qubit. As we have seen, a controlled Z gate can easily be decomposed into a CNOT gate by surrounding the target qubit with $2H$ gates, so the oracle can be implemented with a multicontrolled CNOT gate with $n-1$ control qubits.

The diffusion operator on the other hand, is independent of the searched element and is the same for every Grover circuit. Since $U_s=2|s\rangle\langle s|-I$ and $|s\rangle=H^{\otimes n}|0^{\otimes n}\rangle$ we have

$$U_s=2|s\rangle\langle s|-I=H^{\otimes n}(2|0^{\otimes n}\rangle\langle 0^{\otimes n}|-I)H^{\otimes n}. \quad (1)$$

This means that the operator $2|0^{\otimes n}\rangle\langle 0^{\otimes n}|-I$ has to be surrounded by $2nH$ gates. If a global (-1) phase is applied we have $I-2|0^{\otimes n}\rangle\langle 0^{\otimes n}|$. This operator is basically another oracle in which the desired state is the $|0^{\otimes n}\rangle$ state since $[I-2|0^{\otimes n}\rangle\langle 0^{\otimes n}|]|0^{\otimes n}\rangle=-|0^{\otimes n}\rangle$ and $[I-2|0^{\otimes n}\rangle\langle 0^{\otimes n}|]|x\rangle=|x\rangle$ for any $|x\rangle\neq|0^{\otimes}\rangle$. Then it can be implemented using a multicontrolled Z gate with $n-1$ control qubits surrounded by $2nX$ gates. Therefore the diffusion operator can be implemented using a multicontrol CNOT gate with $n-1$ control qubits and a target qubit surrounded by $2H$ gates, where all the n qubits are surrounded by $2nX$ gates and $2nH$ gates.

The whole algorithm is shown in Fig. 7 for the state $|1001\rangle$ as an example, to make clear how the oracle and diffusion operator are implemented using gates from the set $\mathcal{G}=\{X, Z, H, S, \text{CNOT}, T, T^\dagger\}$. The only gates that are not from this set are the multicontrolled CNOT gates with three control qubits.

The number of $T+T^\dagger$ gates is the relevant parameter that determines if a certain algorithm can be efficiently encrypted using Liang's QHE scheme. Then the only gates that have to be taken into account from the oracle and the diffusion operator are the multicontrolled CNOT gates, since these gates are the only ones that contain T and $T^\dagger$ gates. Recall that the Grover algorithm needs n qubits to search $N=2^n$ elements in a list. For each iteration of the Grover algorithm the oracle and the diffusion operator are applied once. Then in the case there is only one marked state, one iteration of Grover's algorithm requires two multicontrolled CNOT gates (with $n-1$ control qubits), one for the oracle and another for the diffusion operator. The next step is expressing these gates in terms of $T+T^\dagger$ gates.

A multicontrolled CNOT gate with $n-1$ control qubits can be decomposed into $2(n-2)$ Toffoli gates using $(n-2)$

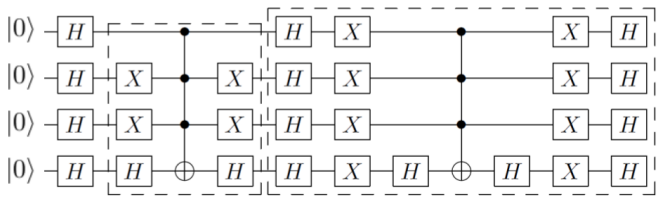

FIG. 7. Grover oracle and diffusion operators for $w=1001$ using gates from the set $\mathcal{G}$ and two multicontrolled CNOT gates with three control qubits. Both operators are surrounded by a dotted box. The operators are used $O(\sqrt{N})$ times and then each qubit is measured.

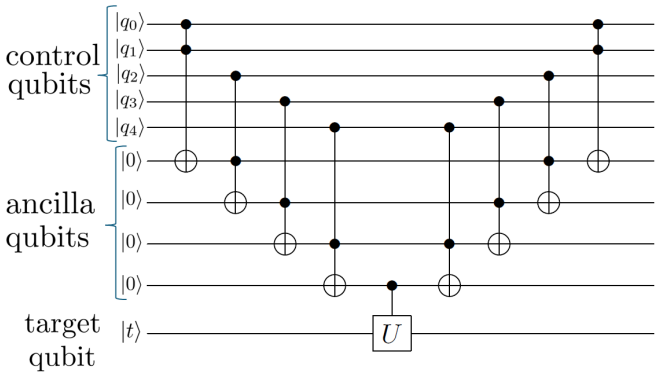

FIG. 8. A $n = 6$ multicontrolled single qubit gate with five control qubits, four ancilla qubits and eight Toffoli gates.

extra ancilla qubits. An example of this decomposition [38] is shown in Fig. 8 for any single qubit gate U. If $U = X$ this corresponds to the decomposition of the multicontrolled CNOT gate.

If two multicontrolled CNOT gates are used to run one query of the Grover algorithm for a list with just one marked item, then the number of Toffoli gates are given by

$$\begin{aligned} 2\ \text{MCNOT} &= 2[2(n-2)]\ \text{Toffolis} = 4(n-2) \\ \text{Toffolis} &= 4[\log_2(N) - 2]\ \text{Toffolis}, \end{aligned} \tag{2}$$

where $n = \log_2(N)$ was used. Since the oracle and the diffusion operator are repeated $O(\sqrt{N})$ times, the number of Toffoli gates needed to run the whole algorithm is simply given by

$$4[\log_2(N) - 2)]\sqrt{N}\ \text{Toffolis}. \tag{3}$$

Since a Toffoli gate can be decomposed in seven $T+T^\dagger$ gates (Fig. 2) the number M of $T+T^\dagger$ gates is finally obtained:

$$M = 28[\log_2(N) - 2)]\sqrt{N} \to O(\log_2(N)\sqrt{N}). \tag{4}$$

M grows slower than a linear function. Since the QHE scheme is only suitable for circuits with a polynomial number of $T+T^\dagger$ gates, the fact that $M = O(\log_2(N)\sqrt{N})$ makes the Grover algorithm a protocol that can be implemented efficiently using the QHE scheme discussed. If there are more desired elements in the N items list instead of just one, this result is still true. The reason is that for each new marked element, the diffusion operator stays the same and only the oracle changes. Instead of just using one multicontrolled CNOT gate, the oracle would be implemented using m multicontrolled CNOT gates. Then the number of $T+T^\dagger$ gates is still $O(\log_2(N)\sqrt{N})$ and so the decryption process of the algorithm remains efficient. Therefore the Grover algorithm can be homomorphically implemented using n qubits and $n-2$ extra ancilla qubits in an efficient manner. For a $N = 2^n$ list, the client should then prepare $2n-2$ qubits in total, encrypt them, send them to the server and then perform the corresponding M measurements to correctly decrypt the results.

We want to mention that more optimizations can be applied, such as decomposing the multicontrolled CNOT gate using the relative phase Toffoli gate from Ref. [39] instead of the usual Toffoli gate that was used in Fig. 8. This results in a reduction of $T+T^\dagger$ gates. In particular, instead of the $14(n-2)$ $T+T^\dagger$ gates that were needed using our decomposition for the multicontrolled CNOT gate with n qubits, only $8n - 17$ $T+T^\dagger$ gates are needed. The number of ancillas also decreases from $n-2$ in our case to $\lceil (n-3)/2 \rceil$. Using this relative phase Toffoli gate decomposition, we have that $M = 2[8\log_2(N) - 17]\sqrt{N} = [16\log_2(N) - 34]\sqrt{N}$ and so M is still bounded by $O(\log_2(N)\sqrt{N})$. Another example of a further optimization that could be used is replacing the diffusion operator with all single qubit gates, as proposed in Ref. [40]. Inspired by quantum partial search algorithms such as Refs. [41] and [42], Zhang and Korepin [43] proposed a version of Grover's algorithm that improves its depth. In Ref. [44], quantum search algorithms with reduced depth compared with the usual Grover algorithm are implemented on the IBM quantum computers. These algorithms with reduced depth constitute examples of other optimizations that can be applied. However, we want to clarify that the purpose of this article is not to find the most optimal homomorphic implementation of Grover's algorithm. Instead, it is to study whether the homomorphic encryption of Grover's algorithm can be implemented in an efficiently enough manner. Using our decomposition, we have found an upper bound that grows slower than any linear function and so we have shown that our homomorphic implementation of the algorithm can be performed in an efficiently enough manner. More efficient implementations can be used (such as the relative phase Toffoli gates) but studying them in detail is out of the scope of this article.

We want to mention some results regarding the special case in which there is only one desired element in the database. Grover [45] proposed an algorithm that could find this unique item using only $O(\sqrt{N}\log_2\log_2 N)$ elementary gates without increasing the number of queries needed significantly. This algorithm is no longer made of $O(\sqrt{N})$ identical iterations and it is more complicated than the usual Grover algorithm. Then in 2017 Arunachalam and de Wolf [35] proposed a more efficient Grover search algorithm regarding its gate complexity. For a sufficiently large database of size N that contains just one desired element and for any constant r this algorithm finds the only solution using $O(\sqrt{N})$ queries and $O(\sqrt{N}\log_2^{(r)} N)$ elementary gates. The iterated binary logarithm is defined as $\log_2^{(s+1)} = \log_2 \circ \log_2^{(s)}$, where $\log_2^{(0)}$ is the identity function. The elementary gates allowed are the Toffoli gate and any unitary single qubit gate such as the H and X gates. If we have a very large N items list that is also a power of two, r can be chosen to be $r = \log_2^\star N$ so the algorithm finds the searched element using only $O(\sqrt{N}\log_2(\log_2^\star N))$ elementary gates in the optimal $\frac{\pi}{4}\sqrt{N}$ queries. Here the function $\log_2^\star N$ represents the number of times the binary logarithm must be iteratively applied to N to obtain a number that is at most one: $\log_2^\star N = \min\{r \geqslant 0 : \log_2^{(r)} N \leqslant 1\}$. Then in 2021, Briański *et al.* [36] proposed an even more efficient algorithm regarding the gate complexity. Fix any $\varepsilon \in (0,1)$ and any $N \in \mathbb{N}$ of the form $N = 2^n$. Suppose a quantum oracle O is given that operates on n qubits marking exactly one element. Then there exists a quantum circuit $\mathcal{A}$ that uses the oracle O at most $(1+\varepsilon)\frac{\pi}{4}\sqrt{N}$ times and uses at most $O(\log_2(1/\varepsilon)\sqrt{N})$ nonoracle basic gates, which finds the element marked by O with certainty. Since these algorithms have a reduced number of $T+T^\dagger$ gates compared with the usual Grover algorithm,

they can also be homomorphically evaluated using the QHE scheme discussed. Furthermore, their homomorphic evaluation is even more efficient than the usual Grover algorithm due to their reduced $T + T^\dagger$-gate complexity.

Arunachalam and de Wolf [35] mentioned that most applications of Grover's algorithm study databases with an unknown number of desired elements and only focus on the number of queries. They ended asking whether there are any applications where the reduction in the number of quantum gates for the special case of just one marked element is both applicable and significant. We want to answer this question positively. Quantum homomorphic encryption is a perfect application for this algorithm, since the advantage it offers regarding the gate complexity is significant in the context of improving the efficiency of the decryption process.

B. Homomorphic quantum key attack

To conclude this section, we want to comment on the complexity of the oracle of Grover's algorithm. The oracle we have studied is the simplest one that can be chosen. For more practical quantum search problems, such as quantum key search for block ciphers like the advanced encryption standard (AES), more complex oracles are used. These oracles require more resources, so it is interesting to check if the homomorphic evaluation of these oracles is still efficient. In Ref. [46], quantum circuits that perform quantum key search for the advanced encryption standard (AES) are proposed. In particular, the number of T gates for the oracle and the whole Grover key search are given for these circuits. We are interested in the number of T gates of these circuits, since they determine the complexity of the decryption process of the QHE scheme. For a key size of $k = 128$ bits, the oracle is composed of a 384-fold controlled NOT gate and six AES layers that can be decomposed into 12204 and $6 \times 1060864 = 6365184$ T gates, respectively. Thus, the oracle for this problem requires 6377388 T gates. As a crude estimate, if we assume that one Bell measurement and the subsequent key-updating needed to obtain the key of the next Bell measurement can be performed in 10^{-3} s at best and 1 s at worst, between 1.771 and 1771 hours would be needed to decrypt the whole oracle. Considering this estimation and the number of T gates we believe implementing this oracle homomorphically is quite complex but still feasible. To implement the whole Grover algorithm, a 128-fold controlled NOT implements the diffusion operator and $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ iterations are needed. The estimated number of T gates for the whole Grover algorithm on AES-128 is $(6377388 + 1000)\lfloor \frac{\pi}{4} 2^{64} \rfloor = 9.24 \times 10^{25} = 1.19 \times 2^{86}$. Considering this large number of T gates, we believe that the homomorphic implementation of the whole Grover algorithm on AES-128 cannot be implemented efficiently since the decryption procedure would take too much time to be completed. For AES-192 and AES-256 the whole algorithm contains 1.81×2^{118} and 1.41×2^{151} T gates. More gates are needed compared with AES-128 and so the homomorphic implementation of AES-192 and AES-256 is even more inefficient.

We have studied more articles that study practical Grover search problems. In Ref. [47], the cost analysis of implementing Grover's key search algorithm on the family of KATAN block ciphers is studied. For the oracle, 64630, 85680, and 128464 T gates are needed for KATAN32, KATAN48, and KATAN64, respectively, using Toffoli gates in the decomposition. If the oracle is constructed using the AND gate presented in the paper, 37248, 49104, and 73600 T gates are needed for KATAN32, KATAN48, and KATAN64, respectively. These T counts are lower than those of AES and so these oracles can be implemented more efficiently. Using the same crude estimation as in AES, the worst KATAN64 oracle that contains 128464 T gates would be implemented in 35 hours at worst and 0.035 hours at best. To implement the whole algorithm $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ iterations are needed with no parallelization and so the T-gate count is $2^{55.63}$, $2^{56.04}$, and $2^{56.62}$ for KATAN32, KATAN48, and KATAN64, respectively, if the Toffoli gate is used. If the AND gate is used for the whole algorithm, then the T gate count is $2^{54.84}$, $2^{55.23}$, and $2^{55.82}$ for KATAN32, KATAN48, and KATAN64. These numbers are lower compared with AES but they are still too large for efficient homomorphic implementation, since taking the lowest value of $2^{54.84}$ and using the previous time estimation, at best 8.95×10^9 hours would be needed to decrypt the whole algorithm.

In Ref. [48], Grover's search algorithm is applied on all the variants of a lightweight cipher known as SIMON. The quantum resources needed for this attack are also estimated. For the oracle, the lowest number of T gates found is 24492 for SIMON 32/64 and the highest is 205740 for SIMON 128/256. Once again the oracles present a lower T-gate count compared with AES. For the whole algorithm, the lowest number of T gates is 1.27×2^{46} for SIMON 32/64 and the highest is $1.11 \times 2^{145.2}$ for SIMON 128/256.

In Ref. [49], Grover's algorithm is applied to an optimized implementation of Simplified AES and the resources for the attack are also given. To construct the oracle, 192 Toffoli gates and 32 multicontrol CNOT gates with three control qubits are needed. Using the decompositions of these gates from our paper for simplicity, the upper bound for the number of T gates is $192 \times 7 + 32 \times 4 \times 7 = 2240$. For the whole algorithm, the number of T gates is $2240\lfloor \frac{\pi}{4} 2^{16/2} \rfloor = 450240$. This number of T gates is significantly lower compared with the previous cases and so the efficient homomorphic encryption of Grover's algorithm applied to simplified AES is feasible, since using the same crude estimation as in AES, the decryption process would take between 125 hours at worst and 0.125 hours at best. On the other hand, simplified AES is a teaching tool designed to help students understand the basic structures of AES and is not a block cipher that is actually used in practical cryptography.

Regarding AES, there are articles that improve the results presented in Ref. [46]. In Ref. [50], an explicit quantum design of AES-128 is given. The number of T gates for one invocation of AES-128 is 1053696. Besides this improvement, the number of qubits used is also reduced. In Ref. [51], instead of focusing on minimizing the number of qubits, the authors study AES-128, AES-192, and AES-256 focusing on minimizing the oracles depth. In case of the oracle, for AES-128 they report 54908 T gates for $r = 1$ and 109820 T gates for $r = 2$, where r is the number of known plaintext-ciphertext pairs that are required for a successful key-recovery attack. These oracles are a considerable improvement compared with

TABLE II. Number of T gates required to implement the oracle and the whole Grover key search for different schemes. Using the mentioned crude time estimation, each number also represents the time it would take to complete the decryption process of the oracle and the whole algorithm in the worst case expressed in seconds. The time needed in the best case is obtained by simply multiplying each worst case value by 10^{-3}. In Ref. [51], r is the number of known plaintext-ciphertext pairs that are required for a successful key-recovery attack.

Reference	Variant	Number of T gates: oracle	Number of T gates: Grover key search
[46]	AES-128	6377388	1.19×2^{86}
	AES-192	9650092	1.81×2^{118}
	AES-256	15073196	1.41×2^{151}
[47]	KATAN32-Toffoli gate	64630	$2^{55.63}$
	KATAN48-Toffoli gate	85680	$2^{56.04}$
	KATAN64-Toffoli gate	128464	$2^{56.62}$
	KATAN32-AND gate	37248	$2^{54.84}$
	KATAN48-AND gate	49104	$2^{55.23}$
	KATAN64-AND gate	73600	$2^{55.82}$
[48]	SIMON 32/64	24492	1.27×2^{46}
	SIMON 128/256	205740	$1.11 \times 2^{145.2}$
[49]	Simplified AES	2240	450240
[51]	AES-128 with $r = 1$	54908	1.32×2^{79}
	AES-128 with $r = 2$	109820	1.32×2^{80}

the previous AES oracles mentioned and their homomorphic implementation seems feasible. For the whole algorithm, this work reports 1.32×2^{79} and 1.32×2^{80} T gates for AES-128 with $r = 1$ and $r = 2$, respectively. Even though the number of T gates is significantly reduced, we believe these numbers are still too large for efficient homomorphic encryption, since using our usual estimation the decryption process would take too much time to be completed. Table II shows the number of T gates needed to implement the corresponding oracle and the whole Grover key search for the different schemes that have been presented in this section. Using the usual crude time estimation, each number also represents the time it would take to complete the decryption process of the oracle and the whole algorithm in the worst case expressed in seconds. To obtain the corresponding time needed in the best case, we simply have to multiply each worst case value by 10^{-3}.

In conclusion we have studied the encryption resources of different practical Grover's algorithms. We have found that the number of T gates of quantum key attacks for symmetric encryption schemes is too large to be implemented homomorphically in an efficient manner except in toy models such as the simplified AES. Thus, applying homomorphic encryption to these algorithms seems too inefficient. Nevertheless, other practical problems where Grover's algorithm can be applied, such as the Boolean satisfiability problem, may have simpler oracles that can lead to an efficient homomorphic encryption of the whole algorithm. The study of more practical versions of Grover's algorithm that have an efficient homomorphic implementation could serve as future work. Still, we believe that the work that has been done in this article can serve as a good starting point in this endeavor.

IV. CONCLUSIONS

In this article Liang's [30] quasicompact quantum homomorphic encryption scheme and Grover's algorithm have been combined. This QHE scheme has perfect security, $\mathcal{F}$ homomorphism, no interaction between client and server, and quasicompactness. The scheme is not compact so it does not contradict the no-go result given by Yu [26]. The decryption procedure is independent of the size of the evaluated circuit and depends only on the number of $T+T^\dagger$ gates contained in the circuit.

Making use of this QHE scheme, a quantum circuit that implements the Grover algorithm homomorphically for a particular case has been simulated using Qiskit. We showed how to apply controlled-S gates in the Qiskit simulation in order to account for every possible result. The results obtained from this simulation can always be correctly decrypted after obtaining the final key for each qubit involved.

Regarding the complexity of the decryption procedure of the QHE scheme, it is only efficient for circuits with a polynomial number of $T+T^\dagger$ gates. As it has been seen, the number of $T+T^\dagger$ gates of any Grover circuit is $O(\log_2(N)\sqrt{N})$ which means it grows slower than any linear function. Therefore, Grover's algorithm is a perfect example of a quantum algorithm that can be evaluated homomorphically with perfect security and noninteraction in an efficient manner. In the particular case of a database with just one solution, more efficient algorithms proposed by Arunachalam and Briański can be used. Their gate complexity is $O(\sqrt{N}\log_2(\log_2^\star N))$ and $O(\log_2(1/\varepsilon)\sqrt{N})$, respectively, so the decryption process of their homomorphic evaluation is more efficient than the usual Grover algorithm. The efficiency of the homomorphic evaluation of more practical quantum search algorithms (quantum key attacks for symmetric encryption schemes) has also been studied. Applying homomorphic encryption to these algorithms seems too inefficient in most cases due to the large amounts of T and $T^\dagger$ gates they contain. Nevertheless, the work done here can serve as a starting point in the task of finding more variants of practical quantum search algorithms that have an efficient homomorphic encryption.

Future works in this area of research could be studying the $T+T^\dagger$-gate complexity of more quantum algorithms, in order

to analyze their homomorphic implementation using Liang's QHE scheme. In the case it is not possible to implement them homomorphically in the most general scenario, there may be particular cases, similar to Grover's search on a database with just one marked element, where the $T + T^\dagger$-gate complexity is low enough so the homomorphic implementation is still efficient. These restricted algorithms would then still be useful as an application of quantum homomorphic encryption.

ACKNOWLEDGMENTS

We acknowledge support from grants MINECO/FEDER Projects, PID2021-122547NB-I00 FIS2021, the "MADQuantumCM" project funded by Comunidad de Madrid. M.A.M.-D. has been financially supported by the Ministry of Economic Affairs and Digital Transformation of the Spanish Government through the QUANTUM ENIA project call-Quantum Spain project, and by the European Union through the Recovery, Transformation and Resilience Plan-NextGenerationEU within the framework of the Digital Spain 2026 Agenda. 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).

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

APPENDIX A: QUANTUM HOMOMORPHIC ENCRYPTION SCHEME

1. Preliminaries

In the circuit model of quantum computation, quantum gates are the basic operations that constitute any circuit that can be implemented [52]. The usual Clifford gates, which are $\{X, Z, H, S, \text{CNOT}\}$, will be used. Gates X and Z are the single-qubit Pauli gates. The Hadamard gate is $H = (X + Z)/\sqrt{2}$. Their matrix representation is given by

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{A1}$$

The phase gate S and CNOT gate matrices are

$$S = \sqrt{Z} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad \text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{A2}$$

To perform universal quantum computation, a non-Clifford gate must be added to the Clifford gates, in this case it is the gate T. Its conjugate is simply $T^\dagger$. Their matrix representation is

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}, \quad T^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{pmatrix}. \tag{A3}$$

Then the complete set of gates is $\mathcal{G} = \{X, Z, H, S, \text{CNOT}, T, T^\dagger\}$. The main issue regarding the homomorphic evaluation of a quantum circuit is evaluating the T and $T^\dagger$ gates because these gates produce an S error:

$$TX^a Z^b |\phi\rangle = \left(S^\dagger\right)^a X^a Z^{a\oplus b} T|\phi\rangle. \tag{A4}$$

This error has to be corrected as efficiently as possible. In Liang's schemes, it is corrected by making use of a generalization of quantum teleportation called gate teleportation.

2. Gate teleportation

An EPR state is an entangled quantum state given by $|\Phi_{00}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The state $|\Phi_{00}\rangle$ can be constructed by using a H gate followed by a CNOT acting on the state $|00\rangle$. From this entangled state, we can express the four well-known Bell states in a compact expression:

$$|\Phi_{ab}\rangle = (Z^b X^a \otimes I)|\Phi_{00}\rangle \; \forall \, a, b \in 0, 1. \tag{A5}$$

Quantum teleportation [53] is a technique that allows the transferring of quantum states between a sender and a receiver using a quantum communication channel. In this procedure, Alice wants to send Bob a qubit in the state $|\psi\rangle$, so they first share an entangled pair of qubits in the state $|\Phi_{00}\rangle$. This way both have a qubit from the EPR pair. Then Alice performs a quantum measurement in the Bell basis using the two qubits she has, $|\psi\rangle$ and one qubit of the pair $|\Phi_{00}\rangle$. Due to entanglement, Bob can now obtain the original state $|\psi\rangle$ if he applies the correct sequence of quantum gates (a combination of X and Z) to the qubit in his possession.

For any single-qubit gate U, the "U-rotated Bell basis" can be defined as [54]: $\Phi(U) = \{|\Phi(U)_{ab}\rangle, a, b \in \{0, 1\}\}$, 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 for quantum teleportation:

$$|\alpha\rangle \otimes |\Phi_{00}\rangle = \sum_{a,b\in\{0,1\}} |\Phi_{ab}\rangle \otimes X^a Z^b |\alpha\rangle. \tag{A6}$$

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

$$|\alpha\rangle \otimes |\Phi_{00}\rangle = \sum_{a,b\in\{0,1\}} |\Phi(U)_{ab}\rangle \otimes X^a Z^b U |\alpha\rangle, \tag{A7}$$

where U is any single-qubit gate. Then Eq. (A7) describes gate teleportation. Just like in the usual quantum teleportation, Alice and Bob first share an entangled EPR pair in the state $|\Phi_{00}\rangle$. Then Alice prepares a qubit in the state $|\alpha\rangle$ and performs a "U-rotated Bell measurement." This is simply a quantum measurement in which the U-rotated Bell basis is selected as its measurement basis. She performs the measurement on the two qubits she possesses, one in the state $|\alpha\rangle$ and one of the EPR pair in the state $|\Phi_{00}\rangle$. From this measurement she obtains the results a and b. After the measurement Bob's qubit transforms into the state $X^a Z^b U|\alpha\rangle$. Next, Alice tells Bob the results of her measurement, so Bob can apply the correct Pauli X and Z operators to finally obtain $U|\alpha\rangle$. The gate teleportation procedure is represented in Fig. 9. Gate teleportation is a clear extension of quantum teleportation since if U were the identity then the U-rotated Bell basis $\Phi(U)$ would reduce to the standard Bell basis, and therefore gate teleportation would implement the standard quantum teleportation protocol.

This gate teleportation protocol is the basis of Liang's QHE schemes, because it corrects the error that results from the homomorphic evaluation of a T gate.

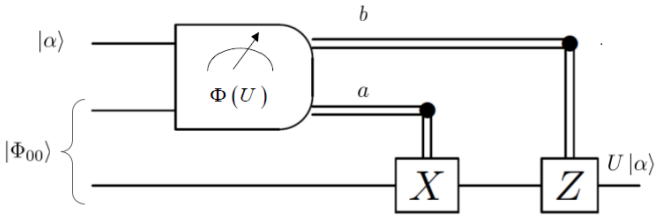

FIG. 9. Quantum circuit for gate teleportation. The box represents the quantum measurement Alice performs on the qubit $|\alpha\rangle$ and one qubit of the pair $|\Phi_{00}\rangle$. The measurement basis chosen is the U-rotated Bell basis $\Phi(U)$. Depending on the measurement results, a and b, Bob applies X^a and Z^b in order to obtain $U|\alpha\rangle$.

3. Encryption, evaluation, and decryption

Suppose that there is a quantum circuit composed of gates from the set $\mathcal{G} = \{X, Z, H, S, \text{CNOT}, T, T^\dagger\}$ that acts on n qubits. There are l gates in the circuit in total. The quantum gates in the circuit are numbered starting from the left to right. This way the first quantum gate is denoted by $G[1]$, the second by $G[2]$ and so on until the last gate $G[l]$. The jth quantum gate is denoted by $G[j]$. The qubit it acts on is denoted by the subscript w. For example the gate $G[j] = X_w$ is a Pauli X gate acting on the wth qubit of the circuit. The CNOT gate acting on the wth control qubit and the w'th target qubit is denoted $\text{CNOT}_{w,w'}$. Among the total number of gates l, the total number of T and $T^\dagger$ gates are denoted by M. Each T and $T^\dagger$ gate has its own number, j_i ($i \leqslant j_i \leqslant l$, $1 \leqslant i \leqslant M$) in the sequence of gates $\{G[j], j = 1, 2, \ldots, l\}$. Then $G[j_i] = T/T^\dagger$ where $j_i < j_{i+1}$. If $G[j_i]$ ($1 \leqslant i \leqslant M$) is applied on the w_ith qubit ($1 \leqslant w_i \leqslant n$) then we can write $G[j_i] = T_{w_i}/T^\dagger_{w_i}$.

The first step in Liang's schemes is encrypting the data that will be sent to the server. This is achieved by applying a symmetric-key encryption scheme, called quantum one time pad (QOTP), to the data. It consists on applying a combination of X^a and Z^b gates to each qubit. The bits a and b are randomly selected from $\{0, 1\}$ and constitute the secret key of the client sk. If the plaintext data contains n qubits, the secret key sk has $2n$ bits $sk = (a_0, b_0)$, $a_0, b_0 \in \{0, 1\}^n$. The wth plaintext qubit is encrypted using the secret bits $(a_0(w), b_0(w))$ such that $|\alpha\rangle_w \to X^{a_0(w)} Z^{b_0(w)} |\alpha\rangle_w = |\rho_0\rangle_w$ where $|\rho_0\rangle_w$ represents the encrypted state of the wth qubit.

QOTP was proposed by Boykin and Roychowdhury [55]. They proved that if the bits a and b are randomly selected from $\{0, 1\}$ and used only once, QOTP has perfect security. This is because if QOTP is applied to any arbitrary quantum state σ, the totally mixed state $I_{2^n}/2^n$ is obtained: $\frac{1}{2^{2n}} \sum_{a,b \in \{0,1\}^n} X^a Z^b \sigma (X^a Z^b)^\dagger = I_{2^n}/2^n$.

Next, the encrypted data will be sent to the server. To complete its designated quantum circuit, the server performs quantum gates from the set $\mathcal{G} = \{X, Z, H, S, \text{CNOT}, T, T^\dagger\}$. As long as the gates are Clifford and not T or $T^\dagger$, the homomorphic evaluation can be performed easily. Each time the server performs one of these gates on a qubit, the key is updated according to the algorithm shown in Fig. 10. After the jth ($1 \leqslant j \leqslant l - 1$) gate is performed, a new key [denoted as (a_j, b_j), $a_j, b_j \in \{0, 1\}^n$] is obtained through key updating. This is the intermediate key. As an example, the new key

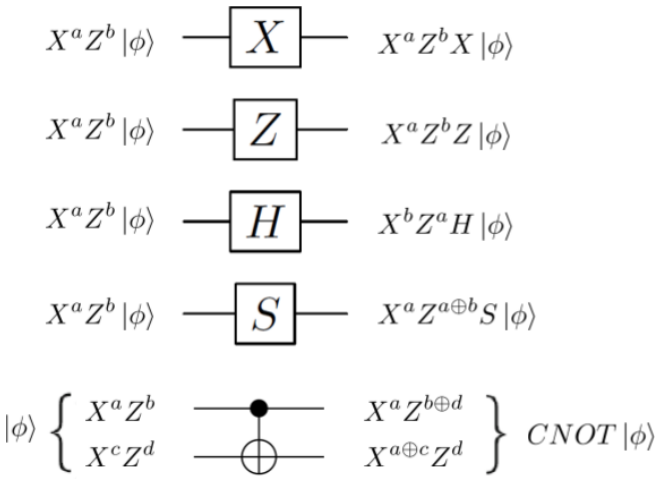

FIG. 10. Key updating rules for the homomorphic evaluation of Clifford gates.

that would be obtained after applying a H gate is $(a_1, b_1) = (b_0, a_0)$ assuming it was the first gate in the circuit so $j = 1$.

If the gate is a $T+T^\dagger$, its homomorphic evaluation is performed using the gate teleportation procedure explained in Appendix A 2. In this case, the S^a-rotated Bell measurement is used in order to correct the S-error described previously. At the start of the circuit's evaluation an EPR source generates M Bell states, as many as $T+T^\dagger$ gates are in the circuit, denoted by $\{|\Phi_{00}\rangle_{c_i,s_i}, i = 1, \ldots, M\}$, where qubits $c_i, i = 1, \ldots, M$ and qubits $s_i, i = 1, \ldots, M$ are kept by the client and server respectively. If the server has to evaluate a $T+T^\dagger$ gate, it first applies the gate to the desired qubit, then performs a SWAP gate between this encrypted qubit and one of the entangled qubits the server possesses, s_i. The next step is applying a S^a-rotated Bell measurement on the qubits s_i and c_i. Using the results from this measurement, r_a and r_b, the encryption key can be updated. From $TX^a Z^b |\alpha\rangle$, this whole process returns $X^{a\oplus r_a} Z^{a\oplus b\oplus r_b} T|\alpha\rangle$. In case the evaluated gate is a $T^\dagger$ gate, the key will be updated from $T^\dagger X^a Z^b |\alpha\rangle$ to $X^{a\oplus r_a} Z^{b\oplus r_b} T^\dagger |\alpha\rangle$. This whole process is shown in Fig. 11. It will be performed M times as many as $T+T^\dagger$ gates are in the circuit.

The M measurements can be postponed until the server has finished all the quantum operations due to the principle of deferred measurement. The server must also generate the key-updating functions according to the key-updating rules of each gate from the set $\mathcal{G}$ that have already been explained. For the scheme VGT, the server must generate $2M + 1$ key-updating functions $\{g_i\}_{i=1}^M$ and $\{f_i\}_{i=1}^{M+1}$. Once the quantum circuit has been completed, the server sends all the encrypted qubits, the key-updating functions based on the key-updating rules

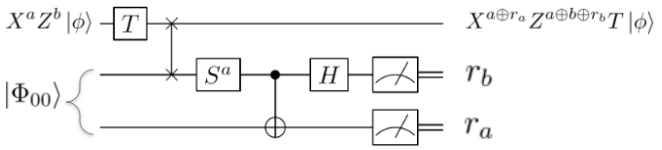

FIG. 11. Homomorphic evaluation of T gate. $|\Phi_{00}\rangle$ represents an EPR pair.

and all the ancillary s_i qubits to the client. Then the client updates its keys according to the key-updating functions and performs a S^a-rotated Bell measurement on qubits c_i and s_i. Since to perform the correct S^a-rotated Bell measurement the updated a key is needed, the client needs to alternate between updating the keys and measuring. These measurements must be performed in the pre-established order, since 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 $dk = (a_{\text{final}}, b_{\text{final}})$, $a_{\text{final}}, b_{\text{final}} \in \{0,1\}^n$. Finally the client decrypts the qubits that are in the state $|\rho_{\text{final}}\rangle$, which is the final state that the server outputs, using the final key to obtain the plaintext results in the state $|\alpha_{\text{final}}\rangle$: $X^{a_{\text{final}}} Z^{b_{\text{final}}} |\rho_{\text{final}}\rangle = |\alpha_{\text{final}}\rangle$.

The whole QHE 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 the secret key $sk = (a_0, b_0)$.

(3) *Encryption.* For any n qubit data $|\alpha\rangle$, client performs QOTP encryption using 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 $G[1]$, $G[2],\ldots,G[l]$ in order on the n encrypted qubits. For each $j \in \{1,\ldots,l\}$ there are two possible cases: if $j \notin \{j_1,\ldots,j_M\}$ then $G[j]$ is a Clifford gate and the server applies it. If $j = j_i (1 \leqslant i \leqslant M)$ then the gate $G[j] = G[j_i] = T_{w_i}$ or $T_{w_i}^\dagger$, the server performs this gate $G[j]$ on the qubit w_i and then the server applies a SWAP gate on this w_ith qubit and one of the entangled qubits s_i. Taking $j_0 = 0$ and using the key-updating rules the server generates the polynomial $\{g_i\}_{i=1}^M$ for one key bit $a_{j_i-1}(w_i) = g_i(a_{j_{i-1}}, b_{j_{i-1}})$, $i = 1,\ldots,M$, with $a_{j_i-1} \in \{0,1\}$. Likewise according to the key updating rules the server generates the polynomial $\{f_i\}_{i=1}^M$ for the intermediate key $(a_{j_i}, b_{j_i}) = f_i(a_{j_{i-1}}, b_{j_{i-1}}, r_a(i), r_b(i))$, $i = 1,\ldots,M$, with $(a_{j_i}, b_{j_i}) \in \{0,1\}^{2n}$. The final polynomial that the server generates is f_{M+1} for the final key $(a_{\text{final}}, b_{\text{final}}) = f_{M+1}(a_{j_M}, b_{j_M})$, with $(a_{\text{final}}, b_{\text{final}}) \in \{0,1\}^{2n}$. After the last gate is applied the server sends all the encrypted qubits, the key-updating functions and all the ancillary s_i qubits to the client.

(5) *Decryption.* The client alternates between key updating and measurements. For each $i = 1,\ldots,M$, the client computes g_i and obtains the corresponding a for the ith S^a-rotated Bell measurement: according to the key $(a_{j_{i-1}}, b_{j_{i-1}})$ and the key-updating function g_i, the client obtains $a = g_i(a_{j_{i-1}}, b_{j_{i-1}})$. If $i = 1$, the secret key $sk = (a_0, b_0)$ is $(a_{j_{i-1}}, b_{j_{i-1}})$. Then using this measurement basis the client performs a S^a-rotated Bell measurement on qubits c_i and s_i and obtains the measurement results $r_a(i)$ and $r_b(i)$. Then the client computes the intermediate key (a_{j_i}, b_{j_i}) according to the key updating function f_i: $(a_{j_i}, b_{j_i}) = f_i(a_{j_{i-1}}, b_{j_{i-1}}, r_a(i), r_b(i))$. After the last round, $i = M$, of this process is performed the client obtains the intermediate key (a_{j_M}, b_{j_M}). Using this key and the last key-updating function f_{M+1} the client obtains the final key: $(a_{\text{final}}, b_{\text{final}})$. Finally the client performs QOTP decryption on the encrypted qubits to obtain the final states: $X^{a_{\text{final}}} Z^{b_{\text{final}}} |\rho_{\text{final}}\rangle = |\alpha_{\text{final}}\rangle$.

A remark about the setup step in which the client and server share M Bell states is in order. It is not really necessary for them to preshare the Bell states. Instead, a Bell state can be generated by the server each time it evaluates a T or $T^\dagger$ gate. Then the server can produce the M Bell states required and send them back to the client along with all the encrypted qubits and the key-updating functions.

Therefore, this scheme can evaluate any quantum circuit homomorphically ($\mathcal{F}$ homomorphic) with perfect security. 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 and the secret key sk is hidden perfectly too. There are no interactions in the evaluation process so the server cannot obtain any information about the plaintext there either. Finally once the data is sent back to the client for the decryption process it is impossible to obtain any information, since the client performs quantum measurements and key-updating locally and does not interact with the server. Now that the whole scheme has been explained, the reason why it is quasicompact becomes clear. Unlike Clifford gates, the number of $T + T^\dagger$ gates contained in the evaluated circuit make the complexity of the decryption process grow due to the quantum measurements the client has to perform in succession. For this reason the discussed scheme is only suitable for circuits with a polynomial number of $T + T^\dagger$ gates.

APPENDIX B: GROVER'S ALGORITHM REVIEW

In the unstructured search problem, a list of N unordered elements is given. The components of this list are labeled from zero to $N-1$ without any loss of generality. There is an element in this list that has to be found called w. In the classical scenario, there is no structure in this list so all the elements have to be checked in order to find the desired element. This takes on average $N/2$ attempts and at worst it takes N tries, making the complexity of the classical problem $O(N)$.

In the quantum algorithm given by Grover, the searched element can be found in just $O(\sqrt{N})$ steps which constitutes a quadratic speedup. For simplicity, the elements of the list can be written as $N = 2^n$ for some integer n. This way the N elements of the list can be represented using n qubits.

The algorithm begins with n qubits in the $|0\rangle$ state. The first step in Grover's algorithm is state preparation. In this step, n Hadamard gates are applied to each qubit in order to obtain a uniform superposition of all possible n bit strings:

$$|s\rangle = H^{\otimes n}|0^{\otimes n}\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle. \tag{B1}$$

In the next step, the oracle denoted by $U_w = I - 2|w\rangle\langle w|$ is applied. Given any state $|x\rangle$ as input, the oracle will output

$$U_w|x\rangle = \begin{cases} |x\rangle & \text{if} \quad x \neq w \\ -|x\rangle & \text{if} \quad x = w. \end{cases} \tag{B2}$$

This oracle changes the amplitude of the desired state while leaving the rest unaffected. The action of the oracle on $|s\rangle$ is

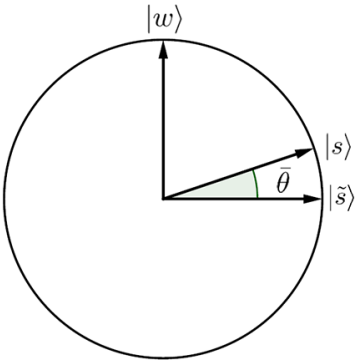

FIG. 12. Circle defined by Eq. (B5). The basis is $|\tilde{s}\rangle$ and $|w\rangle$.

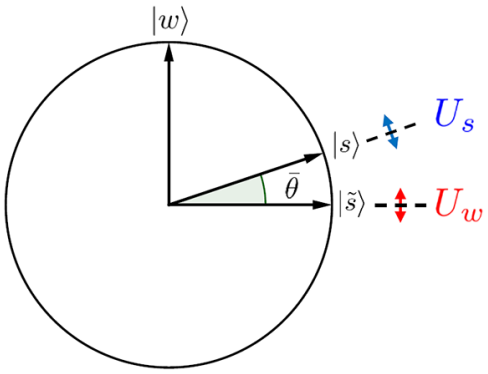

FIG. 13. U_w reflects about $|\tilde{s}\rangle$ and U_s reflects about $|s\rangle$.

given by

$$U_w|s\rangle = \frac{1}{\sqrt{N}}\sum_{\substack{x=0\\x\neq w}}^{N-1}|x\rangle - \frac{1}{\sqrt{N}}|w\rangle. \tag{B3}$$

The next step of the algorithm is applying the diffusion operator given by $U_s = 2|s\rangle\langle s| - I$ to $U_w|s\rangle$. This operator flips the amplitudes around the mean. This makes the amplitude of each state decrease except in the case of desired state because its amplitude increases. This process of amplifying the desired state's amplitude is known as amplitude amplification. Grover's algorithm consists on applying the operators U_w and U_s iteratively.

The algorithm has a well-known geometric interpretation based on two reflections that lead to a rotation around an angle θ in a plane. In this interpretation, an orthonormal coordinate system is used. Since the state $|s\rangle$ is not orthogonal to $|w\rangle$ we can introduce a new state $|\tilde{s}\rangle$ that is perpendicular to $|w\rangle$ defined by

$$|\tilde{s}\rangle = \frac{1}{\sqrt{N-1}}\sum_{\substack{x=0\\x\neq w}}^{N-1}|x\rangle. \tag{B4}$$

$|\tilde{s}\rangle$ is then obtained from $|s\rangle$ by removing the desired state $|w\rangle$ and rescaling so that the state $|\tilde{s}\rangle$ still has its norm equal to one. The states $|\tilde{s}\rangle$ and $|w\rangle$ form a basis and any state can be expressed in terms of an angle θ:

$$|\theta\rangle = \cos(\theta)|\tilde{s}\rangle + \sin(\theta)|w\rangle. \tag{B5}$$

Both U_w and U_s keep the resulting state in the circle defined by Eq. (B5). This circle is represented in Fig. 12. The first step of the algorithm is also represented in Fig. 12 since the state shown is the starting superposition state $|s\rangle$. The angle $\bar{\theta}$ that this state forms with the horizontal axis is given by Eq. (B6):

$$\bar{\theta} = \arcsin\langle s|w\rangle = \arcsin\frac{1}{\sqrt{N}}. \tag{B6}$$

The next step is applying the oracle U_w. Geometrically this is equivalent to a reflection about the $|\tilde{s}\rangle$ axis since the amplitude of $|w\rangle$ changes its sign. The last step to complete an iteration is applying U_s. This operator reflects the state about $|s\rangle$ as seen in Fig. 13.

Two consecutive reflections constitute a rotation. The angle corresponding to this rotation is $2\bar{\theta}$ as seen in Fig. 14. This way the combined action of both operators on any state is given by Eq. (B7):

$$U_sU_w|\theta\rangle = |\theta + 2\bar{\theta}\rangle. \tag{B7}$$

Applying both operators to the initial state $|s\rangle$ gives $U_sU_w|s\rangle = |\bar{\theta} + 2\bar{\theta}\rangle$. The combination of the operators is then repeated iteratively in order to rotate the initial $|s\rangle$ state closer to the state $|w\rangle$. This means rotating from $\theta = \bar{\theta}$ to $\theta = \pi/2$ in steps of $2\bar{\theta}$. After $O(\sqrt{N})$ iterations, the amplitude of the state $|w\rangle$ reaches its maximum so the probability of obtaining $|w\rangle$ in a quantum measurement approaches 1. The final step of the algorithm is then a quantum measurement of the n qubits that returns the desired element of the list w with high probability.

In case there are m elements to be found in the list instead of just one, the steps of the algorithm remain the same: preparation of the $|s\rangle$ state, application of the operators U_w and U_s iteratively and measurement of the n qubits. It can be shown that the number of steps required to obtain the desired elements is bounded by $O(\sqrt{N/m})$ [56] in this case.

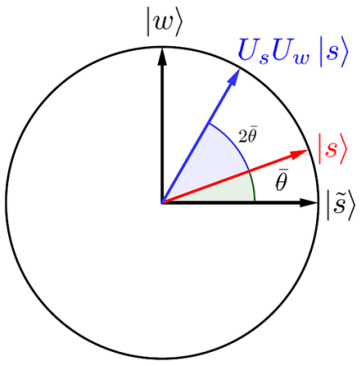

FIG. 14. The combination of U_w and U_s rotate the initial state $2\bar{\theta}$.

[1] E. Bernstein and U. Vazirani, *Quantum complexity theory*, in *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (ACM Press, New York, 1993), pp. 11–20.

[2] P. W. Shor, Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (IEEE, New York, 1994), pp. 124–134.

[3] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer, Proc. R. Soc. London, Ser. A **400**, 97 (1985).

[4] D. Deutsch and R. Jozsa, Rapid solution of problems by quantum computation, Proc. R. Soc. London, Ser. A **439**, 553 (1992).

[5] J. D. Hidary, *Quantum Computing: An Applied Approach* (Springer International Publishing, Cham, 2019).

[6] L. K. Grover, A fast quantum mechanical algorithm for database search, in *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (Philadelphia, Pennsylvania)* (ACM Press, New York, 1996), pp. 212–219.

[7] A. Galindo and M. A. Martin-Delgado, Family of Grover's quantum-searching algorithms, Phys. Rev. A **62**, 062303 (2000).

[8] G. Brassard, P. Høyer, and A. Tapp, Quantum cryptanalysis of hash and claw-free functions, in *LATIN '98: Theoretical Informatics*, Lecture Notes in Computer Science Vol. 1380 (Springer, Berlin, Heidelberg, 1998), pp. 163–169.

[9] G. Brassard, P. Høyer, and A. Tapp, Quantum counting, automata, languages and programming, Lect. Notes Comput. Sci. **1443**, 820 (1998).

[10] A. R. Brown, Playing pool with $|\psi\rangle$: From bouncing billiards to quantum search, Quantum **4**, 357 (2020).

[11] C. Gentry, A fully homomorphic encryption scheme, Ph.D. thesis, Stanford University, 2009.

[12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, (Levelled) fully homomorphic encryption without bootstrapping, in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (ACM Press, New York, 2012), pp. 309–325.

[13] M. H. Devoret, A. Wallraff, and J. M. Martinis, Superconducting qubits: A short review, arXiv:cond-mat/0411174.

[14] D. DiVincenzo, The physical implementation of quantum computation, Fortschr. Phys. **48**, 771 (2000).

[15] M. H. Devoret and R. J. Schoelkopf, Superconducting circuits for quantum information: An outlook, Science **339**, 1169 (2013).

[16] D. Castelvecchi, Quantum computers ready to leap out of the lab in 2017, Nature (London) **541**, 9 (2017).

[17] G. Wendin, Quantum information processing with superconducting circuits: A review, Rep. Prog. Phys. **80**, 106001 (2017).

[18] J. M. Gambetta, J. M. Chow, and M. Steffen, Building logical qubits in a superconducting quantum computing system, npj Quantum Inf. **3**, 2 (2017).

[19] J. J. García-Ripoll, E. Solano, and M. A. Martin-Delgado, Quantum simulation of Anderson and Kondo lattices with superconducting qubits, Phys. Rev. B **77**, 024522 (2008).

[20] O. Viyuela, A. Rivas, S. Gasparinetti, A. Wallraff, S. Filipp, and M. A. Martin-Delgado, Observation of topological Uhlmann phases with superconducting qubits, npj Quantum Inf. **4**, 10 (2018).

[21] M. Liang, Quantum fully homomorphic encryption scheme based on universal quantum circuit, Quantum Inf. Process. **14**, 2749 (2015).

[22] P. P. Rohde, J. F. Fitzsimons, and A. Gilchrist, Quantum walks with encrypted data, Phys. Rev. Lett. **109**, 150501 (2012).

[23] J. Zeuner, I. Pitsios, S. H. Tan, A. N. Sharma, J. F. Fitzsimons, R. Osellame, and P. Walther, Experimental quantum homomorphic encryption, npj Quantum Inf. **7**, 25 (2021).

[24] M. Liang, Symmetric quantum fully homomorphic encryption with perfect security, Quantum Inf. Process. **12**, 3675 (2013).

[25] S. H. Tan, J. A. Kettlewell, Y. Ouyang, L. Chen, and J. F. Fitzsimons, A quantum approach to homomorphic encryption, Sci. Rep. **6**, 33467 (2016).

[26] L. Yu, C. A. Pérez-Delgado, and J. F. Fitzsimons, Limitations on information-theoretically-secure quantum homomorphic encryption, Phys. Rev. A **90**, 050303 (2014).

[27] C. Y. Lai and K. M. Chung, On statistically-secure quantum homomorphic encryption, Quantum Inf. Comput. **18**, 0785 (2018).

[28] A. Broadbent and S. Jeffery, Quantum homomorphic encryption for circuits of low T-gate complexity, in *Advances in Cryptology—CRYPTO 2015* (Springer, Berlin, Heidelberg, 2015), pp. 609–629.

[29] Y. Dulek, C. Schaffner, and F. Speelman, Quantum homomorphic encryption for polynomial-sized circuits, in *Advances in Cryptology—CRYPTO 2016*, Lecture Notes in Computer Science Vol. 9816, edited by M. Robshaw and J. Katz (Springer, Berlin, Heidelberg, 2016).

[30] M. Liang, Teleportation-based quantum homomorphic encryption scheme with quasi-compactness and perfect security, Quantum Inf. Process. **19**, 28 (2020).

[31] A. Einstein, B. Podolsky, and N. Rosen, Can quantum-mechanical description of physical reality be considered complete? Phys. Rev. **47**, 777 (1935).

[32] C. Gong, J. Du, Z. Dong, Z. Guo, A. Gani, L. Zhao, and H. Qi, Grover algorithm-based quantum homomorphic encryption ciphertext retrieval scheme in quantum cloud computing, Quantum Inf. Process. **19**, 105 (2020).

[33] C. Gong, Z. Dong, A. Gani, and H. Qi, Quantum ciphertext dimension reduction scheme for homomorphic encrypted data, in *Proceedings of the 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Shenyang, China* (IEEE, Piscataway, NJ, 2021), pp. 903–910.

[34] P. Fernández and M. A. Martin-Delgado, Homomorphic encryption of the $k = 2$ Bernstein-Vazirani algorithm, J. Phys. A: Math. Theor. **57**, 365301 (2024).

[35] S. Arunachalam and R. de Wolf, Optimizing the number of gates in quantum search, Quantum Inf. Comput. **17**, 0251 (2017).

[36] M. Briański, J. Gwinner, V. Hlembotskyi, W. Jarnicki, S. Pliś, and A. Szady, Introducing structure to expedite quantum searching, Phys. Rev. A **103**, 062425 (2021).

[37] Qiskit contributors, Qiskit: An open-source framework for quantum computing (2023).

[38] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, Cambridge, 2010).

[39] D. Maslov, Advantages of using relative-phase Toffoli gates with an application to multiple control Toffoli optimization, Phys. Rev. A **93**, 022311 (2016).

[40] Z. Jiang, E. G. Rieffel, and Z. Wang, Near-optimal quantum circuit for Grover's unstructured search using a transverse field, Phys. Rev. A **95**, 062317 (2017).

[41] L. K. Grover and J. Radhakrishnan, Is partial quantum search of a database any easier? in *Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures* (ACM Press, New York, 2005), pp. 186–194.

[42] V. E. Korepin and L. K. Grover, Simple algorithm for partial quantum search, Quantum Inf. Process. **5**, 5 (2006).

[43] K. Zhang and V. E. Korepin, Depth optimization of quantum search algorithms beyond Grover's algorithm, Phys. Rev. A **101**, 032346 (2020).

[44] K. Zhang, P. Rao, K. Yu, H. Lim, and V. Korepin, Implementation of efficient quantum search algorithms on NISQ computers, Quantum Inf. Process. **20**, 233 (2021).

[45] L. K. Grover, Trade-offs in the quantum search algorithm, Phys. Rev. A **66**, 052314 (2002).

[46] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, Applying Grover's algorithm to AES: Quantum resource estimates, arXiv:1512.04965.

[47] M. Rahman and G. Paul, Grover on KATAN: Quantum resource estimation, IEEE Trans. Quantum Eng. **3**, 1 (2022).

[48] R. Anand, A. Maitra and S. Mukhopadhyay, Grover on SIMON, Quantum Inf. Process. **19**, 340 (2020).

[49] K.-B. Jang, G.-J. Song, H.-J. Kim, and H.-J. Seo, Grover on simplified AES, in *IEEE International Conference on Consumer Electronics-Asia* (ICCE-Asia, Gangwon, Korea, 2021), pp. 1–4.

[50] M. Almazrooie, A. Samsudin, R. Abdullah, and K. N. Mutter, Quantum reversible circuit of AES-128, Quantum Inf. Process. **17**, 112 (2018).

[51] S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, Implementing Grover oracles for quantum key search on AES and LowMC, in *Proceedings of the Advances in Cryptology - EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Part II* (Springer, Cham, 2020), pp. 280–310.

[52] A. Galindo and M. A. Martín-Delgado, Information and computation: Classical and quantum aspects, Rev. Mod. Phys. **74**, 347 (2002).

[53] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels, Phys. Rev. Lett. **70**, 1895 (1993).

[54] R. Jozsa, An introduction to measurement based quantum computation, arXiv:quant-ph/0508124.

[55] P. O. Boykin and V. Roychowdhury, Optimal encryption of quantum bits, Phys. Rev. A **67**, 042317 (2003).

[56] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, Tight bounds on quantum searching, Fortschr. Phys. **46**, 493 (1998).