ISSN 1063-7761, Journal of Experimental and Theoretical Physics, 2019, Vol. 128, No. 1, pp. 25–44. © Pleiades Publishing, Inc., 2019.
Russian Text © D.V. Denisenko, M.V. Nikitenkova, 2019, published in Zhurnal Eksperimental'noi i Teoreticheskoi Fiziki, 2019, Vol. 155, No. 1, pp. 32–53.

Application of Grover's Quantum Algorithm for SDES Key Searching

D. V. Denisenko[a],* and M. V. Nikitenkova[a]

[a]Bauman Moscow State Technical University, Moscow, 105005 Russia

*e-mail: DenisenkoDV@bmstu.ru

Received July 6, 2018; revised July 6, 2018; accepted July 10, 2018

Abstract—The problem of finding the key of Simplified-DES (SDES)—a model of a block cipher DES—by Grover's quantum algorithm is considered. Examples of application of Grover's algorithm are presented. A quantum system with the minimum number of qubits is constructed that implements SDES key searching by a single pair of plaintext and ciphertext and requires only 19 qubits. This quantum circuit is simulated by using a *Quipper* quantum simulator.

DOI: 10.1134/S1063776118120142

1. INTRODUCTION

Presently, many research are focused on the development of quantum simulators and quantum processors: in 2017, at the ICQT 2017 Conference, a group of physicists led by M. Lukin, a co-founder of the Russian Quantum Center and professor at Harvard University, reported the development of a programmable 51-qubit quantum simulator [1]. At about the same time, a group of scientists from the University of Maryland developed a 53-qubit simulator based on ions in optical traps [2]. IBM successfully tested a prototype 50-qubit quantum processor [3], and, in December 2017, article [4] was published on the project of a scalable silicon-based quantum processor representing a $24 \times 20 = 480$ array of qubits. In January 2018, at the CES-2018, Intel announced a 49-qubit superconducting quantum chip called Tangle Lake. Intel is developing quantum computers in two directions: the design of superconducting devices and silicon chips with spin qubits. In March 2018, Google announced a 72-qubit quantum processor Bristlecone. The company hopes that Bristlecone will demonstrate a quantum advantage [5, 6].

Quantum computations have a direct effect on the information crypto security by modern cryptographic algorithms and protocols. For example, the application of Shor's quantum algorithm [7] makes the RSA cryptographic system unsafe, and Simon's quantum algorithm [8] makes unsafe the use of block ciphers in CBC-MAC, PMAC, GMAC, and OCB modes [9]. Grover's quantum algorithm (see [10–12]) in the quantum computation model is an analog of exhaustive key search in classical cryptography. In [13], the authors considered the problem of finding the key of Simplified-DES (SDES)—a model of a block cipher DES—by Grover's quantum algorithm and presented a description of the corresponding quantum circuit that implements a SDES key searching by a known pair of blocks of a plaintext and a ciphertext using 61 qubits. The authors of [13] applied a libquantum quantum simulator (see [14]); however, the original program codes were not published.

In the present study, we demonstrate a quantum circuit that implements SDES key searching by a plaintext–ciphertext pair on 19 qubits verified by a Quipper quantum simulator (see [15–17]), which, in contrast to libquantum, allows us to print quantum circuits in PDF files automatically. We show that 19 qubits is the minimum number of logical qubits for a quantum circuit that implements SDES key searching by a single plaintext–ciphertext pair.

The main goals of the present study are to demonstrate the application of Grover's quantum algorithm to the key search problem in the case of an educational block cipher SDES and to estimate the minimum number of logical qubits necessary to implement such a search.

In Appendix A we present a test example of SDES. A program implementation of Grover's algorithm in problems of searching for one of two target values in Wolfram Mathematica is presented in Appendices B and C. A program implementation of the SDES key searching by Grover's quantum algorithm in Quipper is presented in Appendix D.

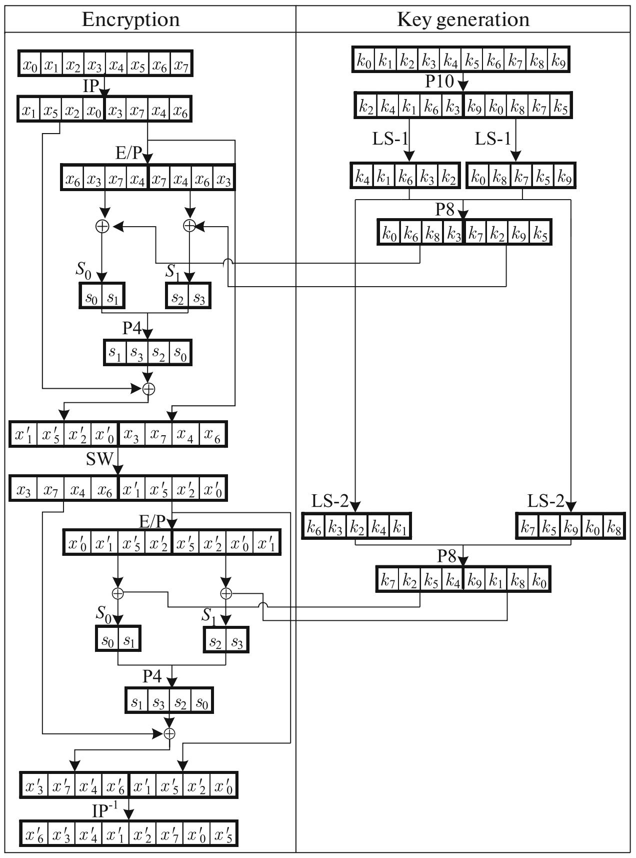

Fig. 1. SDES scheme.

2. DESCRIPTION OF THE SDES ALGORITHM

The SDES block cipher is a two-round Feistel network E_{SDES}: $V_{10} \times V_8 \to V_8$ in which a key $K \in V_{10}$ and the blocks of a plaintext and a ciphertext are eight-bit binary vectors (see Fig. 1).

The procedure of encryption by the SDES algorithm represents a composition of mappings:

$$\mathrm{IP}^{-1} \circ f_{k_2} \circ \mathrm{SW} \circ f_{k_1} \circ \mathrm{IP}.$$

Two round keys $k_1, k_2 \in V_8$ are formed from the main key $K \in V_{10}$. First, a bit permutation P10 is applied to the key K:

P10 – permutation	3 5 2 7 4 10 1 9 8 6

Then, a 1-bit left cyclic shift (LS-1) is applied to each half of P10(K), after which a ten-bit key is again formed from the two halves. Next, P8—an eight-bit sample with appropriate numbers—is applied, and a round key k_1 is obtained:

P8 – sample of bits with numbers	6 3 7 4 8 5 10 9

To form k_2, a two-bit left cyclic shift (LS-2) is applied after LS-1, and then P8 is applied.

Consider the first round of encrypting a plaintext block $P \in V_8$ by the SDES algorithm.

A bit permutation IP is applied to the plaintext P:

IP	2 6 3 1 4 8 5 7

The transformation f_k is defined as follows. Let $P = L \| R$, $L, R \in V_4$. Then $f_k(L, R) = (L \oplus F(R, k), R)$.

The mapping $F(R, k)$: $V_4 \times V_8 \to V_4$ consists of the following successively applied transformations.

1. The procedure of expanding E/P: $V_4 \to V_8$, a sample of bits with appropriate numbers:

E/P	4 1 2 3 2 3 4 1

2. XORing the E/P result with argument k.

3. Application of S-boxes. To each S-box, a 4-bit vector is fed the first and fourth bits of which form the numbers of rows and the second and third bits form the numbers of columns of the substitution tables:

$$S_0 = \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix}, \quad S_1 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix}.$$

The numbering of rows and columns starts from zero. Example: $S_0(1010) = 10$, because the row with number 10 (the third row) and the column with number 01 (the second column) in the substitution table S_0 contain number 2, which is expressed as 10 in binary notation.

The coordinate functions S_0 have the form

$$y_0(x_0, x_1, x_2, x_3) = x_3 \oplus x_1\bar{x}_0 \oplus x_0x_2 \oplus x_0x_1x_2x_3,$$

$$y_1(x_0, x_1, x_2, x_3) = \bar{x}_0\bar{x}_2 \oplus x_0\bar{x}_3 \oplus x_0\bar{x}_1 \oplus x_0x_1\bar{x}_2x_3.$$

The coordinate functions S_1 have the form

$$y_0(x_0, x_1, x_2, x_3) = x_1 \oplus \bar{x}_2x_3 \oplus x_0\bar{x}_3 \oplus x_0\bar{x}_1x_2\bar{x}_3,$$

$$y_1(x_0, x_1, x_2, x_3) = x_2\bar{x}_3 \oplus x_0\bar{x}_3 \oplus \bar{x}_0x_1x_3 \oplus x_0x_2x_3.$$

4. To a 4-bit output from S-boxes, a bit permutation P4 is applied:

P4	2 4 3 1

5. Bit-by-bit XORing with left half IP(P).

6. Permutation of half-bytes SW: $V_8 \to V_8$, SW(L, R) = (R, L).

The second round is performed similar to the first, but with the key k_2. After performing the second round, IP^{-1} is applied:

IP^{-1}	4 1 3 5 7 2 8 6

As a result, we obtain a ciphertext block $C = E_{SDES}(K, P)$, $C \in V_8$.

3. GROVER'S ALGORITHM

Suppose given a numbered set of $N = 2^n$ elements and one should find at least one element of this set that satisfies a certain search criterion; the set of elements satisfying the chosen criterion is not empty and consists of M elements, $M \le N/2$ (see [12, p. 318]).

We can assume that a certain Boolean function f: $V_n \to V_1$ is defined such that $f(x) = 1$ if and only if the element of the set with number x satisfies the search criterion. Here it is assumed that the function f can be efficiently implemented as a quantum circuit.

When solving a search problem on a classical computer, one should generally search through all elements of the set, which ultimately gives complexity on the order of $O(N/M)$, whereas Grover's quantum algorithm (see [10–12]) has complexity $O(\sqrt{N/M})$ when solving this problem on a quantum computer.

For an arbitrary block cipher, the key search problem by Grover's quantum algorithm is formulated as follows. Consider a block cipher with an n-bit-long key and an m-bit-long block E: $V_n \times V_m \to V_m$. There is a certain number of plaintext–ciphertext pairs obtained on the same unknown key $K \in V_n$, $C_i = E(K, P_i)$, $i \in \overline{1,t}$, and we solve the standard key recovery problem. An unambiguous recovery of the key on the basis of the uniqueness distance of the cipher [18] requires that the number of pairs of texts should be not less than $t = \lceil n/m \rceil$.

In this case, the key is unique with high probability, and the corresponding Boolean function f: $V_n \to V_1$ is defined as follows:

$$f(x) = \bigwedge_{i=1}^{t} z(E(x, P_i) \oplus C_i),$$

where z: $V_m \to V_1$; here $z(x) = 1$ if $x = 0^m$ and $z(x) = 0$ otherwise.

Note that, in [13], a key search is performed by a single pair of a plaintext and a ciphertext (P, C), and we consider two cases: in the first case, the pair (P, C) has no equivalent keys (i.e., there exists exactly one key on which the block of plaintext P turns into the block of ciphertext C), and, in the second case, the pair (P, C) has two equivalent keys. In the present study, following [13], we organize SDES key searching by a single pair of a plaintext and a ciphertext and consider the same examples as in [13].

In Appendices B and C, we present program implementations that demonstrate an evolution of amplitudes of quantum states during the execution of Grover's algorithm in search problems for one and two target values, respectively; i.e., we consider the cases of $M = 1$ and $M = 2$.

Algorithm 1. Grover's algorithm.

Input. A set $\{a_1, a_2, ..., a_N\}$ of $N = 2^n$ elements, f: $V_n \to V_1$, $f(x) = 1$ if and only if a_x satisfies a certain search criterion, where x is the number of the element a_x in binary notation, i.e., $x \in V_n$.

Output. With probability $p > 1/2$, an arbitrary $a_{x'}$: $f(x') = 1$.

1. Initialization of $n + 1$ qubits to a state $|\psi_0\rangle = |0\rangle^{\otimes n}|1\rangle$; additional working qubits are initialized depending on the function f.

2. Application of Hadamard gates H. We obtain

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}.$$

3. $(\pi/4)\sqrt{N/M}$-Times application of Grover iteration:

(a) change of sign of the amplitude of the target state for any $i \in \overline{0, N-1}$ (in [12], the application of oracle O):

$$|i\rangle \xrightarrow{O} (-1)^{f(i)}|i\rangle;$$

(b) inversion about the mean (increasing the probability to obtain one of the target values; in [12], application of the operator $2|\psi\rangle\langle\psi| - I$, where I is the $2^n \times 2^n$ identity matrix):

–apply the operator $H^{\otimes n}$;

–apply the operator $2|0\rangle\langle 0| - I$;

–apply the operator $H^{\otimes n}$.

4. Measurement of qubits. With probability $p > 1/2$, we obtain an arbitrary $a_{x'}$: $f(x') = 1$.

4. SDES QUANTUM CIRCUIT AND GROVER'S ALGORITHM

In [13], the authors presented the implementation of SDES as a quantum circuit on 60 qubits and a quantum circuit of Grover's algorithm on 61 qubits (Figs. 2 and 3):

(1) 10 qubits for writing a key;

(2) 8 qubits for writing a plaintext block;

(3) 8 qubits for writing a ciphertext block;

(4) 34 qubits—a working space for writing the results of intermediate computations.

Figures 4–7 demonstrate a more efficient quantum circuit of Grover iteration (compared with the quantum circuit described in [13]), which implements SDES key searching by a single pair of a plaintext and a ciphertext and a final measurement of qubits. A program implementation of an SDES key search in a Quipper quantum simulator is presented in Appendix D.

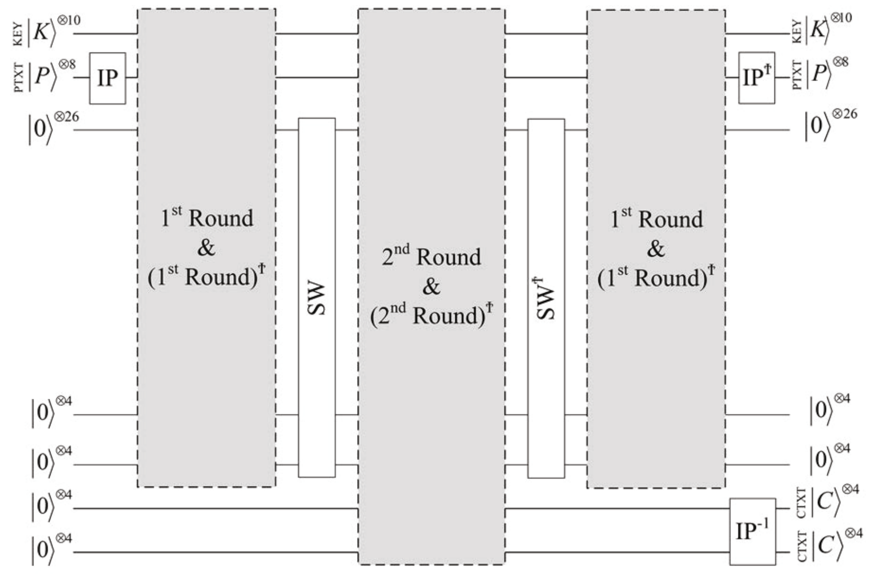

Fig. 2. Implementation of SDES as a quantum circuit.

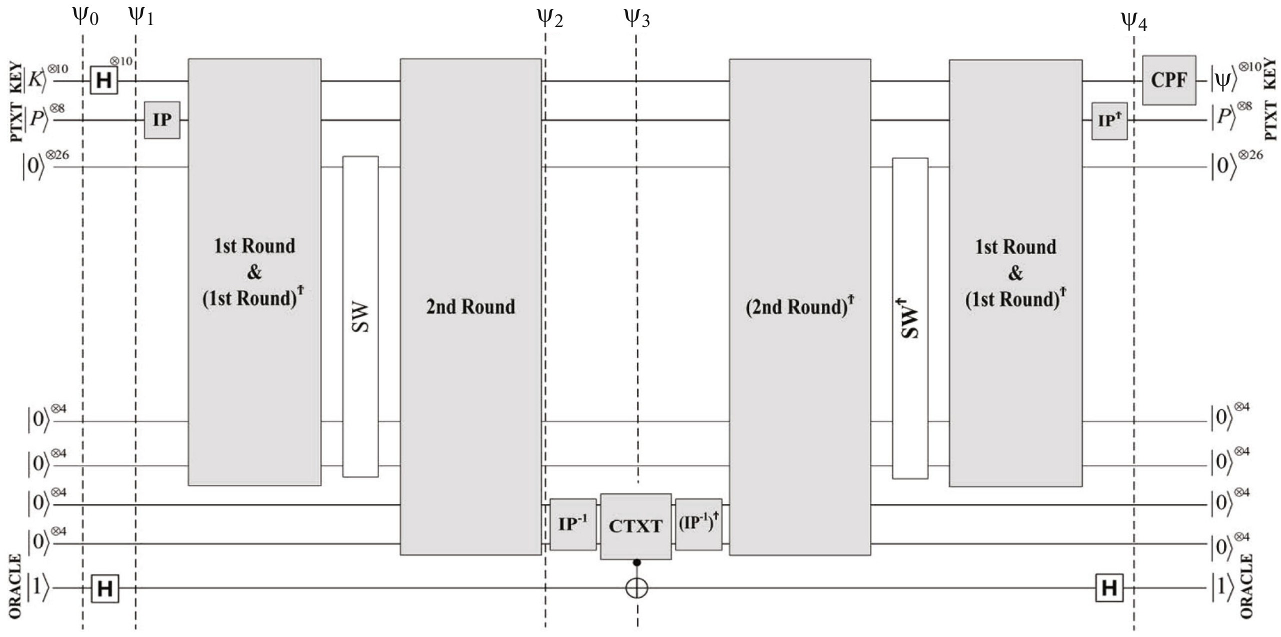

Fig. 3. Quantum circuit of a single Grover iteration.

Consider two cases.

In the first case, we take $P = 00010000$ as a plaintext block and $C = 00110011$ as a ciphertext block. For such a pair (P, C), there exists a single key $K = 1100010011$ on which $E_{SDES}(K, P) = C$.

In the second case, we take $P = 10100101$ and $C = 00110110$. For such a pair (P, C), there exist two keys,

$$K_1 = 0010010111 \quad \text{and} \quad K_2 = 0011011111,$$

on which $E_{SDES}(K_i, P) = C$, $i \in \{1, 2\}$.

According to the theoretical estimate of the number of Grover iterations, in the first case the optimal number of Grover iterations is

$$R = \frac{\pi}{4}\sqrt{\frac{N}{M}} = \frac{\pi}{4}\sqrt{\frac{1024}{1}} \approx [25.1327] = 25,$$

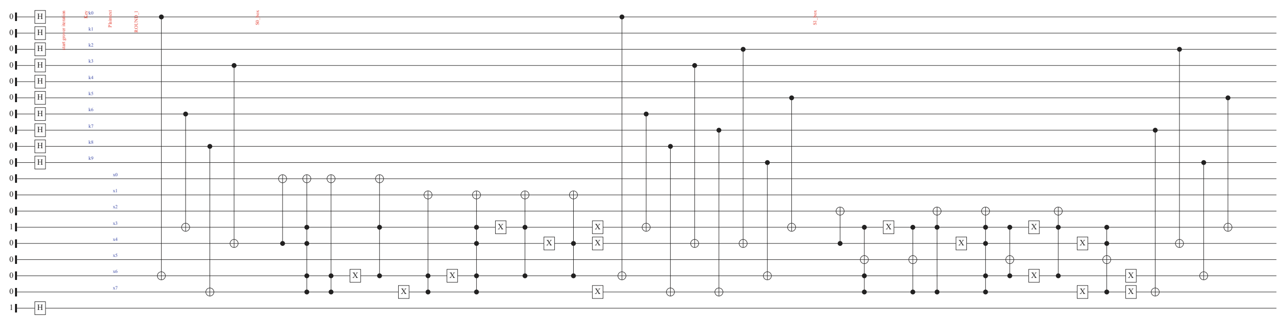

Fig. 4. The first part of the circuit: The first round of SDES.

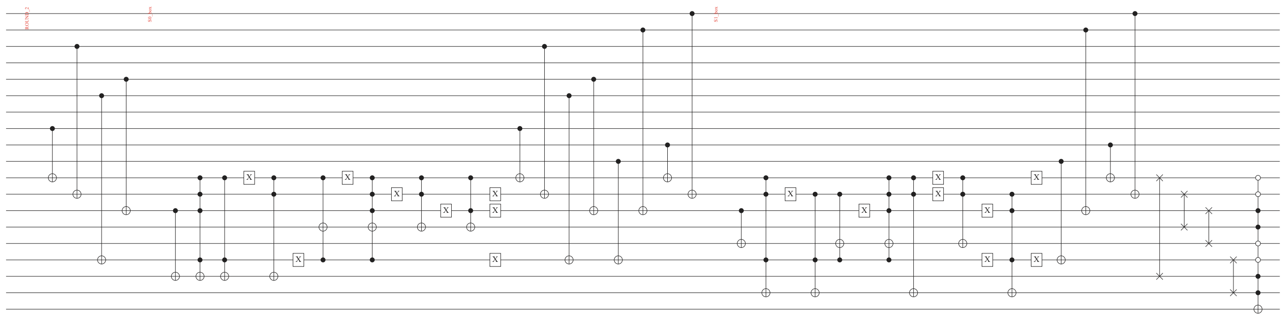

Fig. 5. The second part of the circuit: The second round of SDES, inversion of the lower qubit.

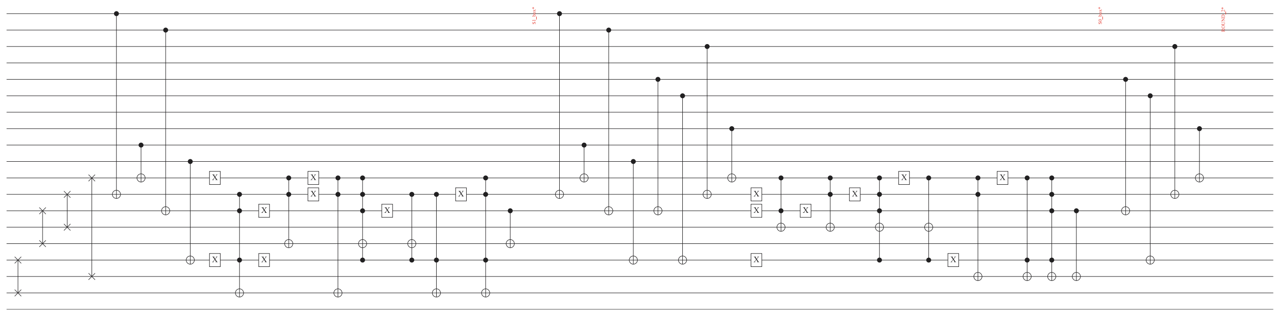

Fig. 6. The third part of the circuit: Inverse transformation of the second round of SDES.

and, in the second case,

$$R = \frac{\pi}{4}\sqrt{\frac{N}{M}} = \frac{\pi}{4}\sqrt{\frac{1024}{2}} \approx [17.7715] = 18.$$

Table 1 presents the probability distribution of keys after 25 Grover iterations for P = 00010000 and $C = 00110011$ (the first case) obtained by a Quipper quantum simulator.

Table 2 presents the probability distribution of keys after 18 Grover iterations for P = 10100101 and $C = 00110110$ (the second case), which is also obtained by a Quipper quantum simulator. Note that the keys 0010010111_{10} = 151 and 0011011111_{10} = 223; i.e., they correspond to TargetValues ∈ {151, 223} in the problem considered in Appendix C. The sum of probabilities

$$P(K{=}151_2) + P(K{=}223_2) = 2 \cdot 0.4978955 \equiv 0.995791$$

coincides with the probability of success of Grover's algorithm after 18 iterations in Table 7, calculated in Wolfram Mathematica (see Appendix C).

Summary data on the characteristics of the constructed quantum circuits are presented in Table 3.

In the quantum circuit proposed, the so-called generalized CNOT(n) gates—gates with one controlled qubit and n controlling qubits—are used, whose implementation does not require additional working qubits (see [12, p. 236]).

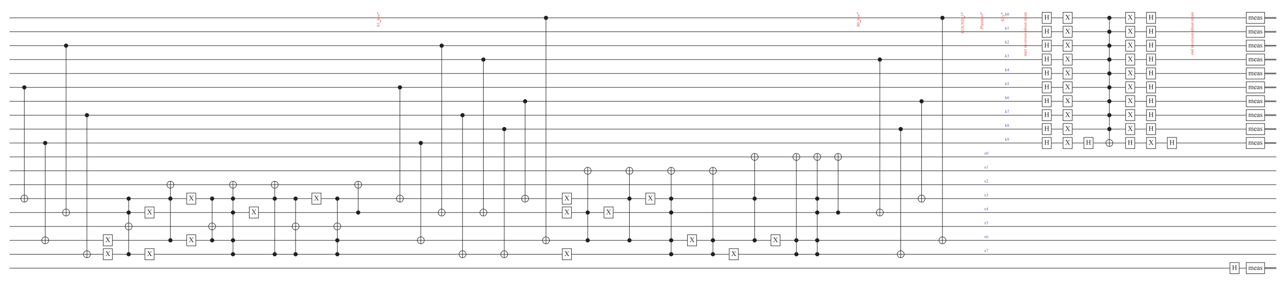

Fig. 7. The fourth part of the circuit: Inverse transformation of the first round of SDES, the Grover diffusion, and measurement of qubits.

We are interested in the minimum estimate for the number of logical qubits to implement Grover's algorithm in the SDES key searching problem. The mapping E_{SDES}: $V_{10} \times V_8 \to V_8$ consists of eight Boolean coordinate functions $f_i(x_1, \ldots, x_{10+8})$, $i \in \overline{1,8}$, that depend on 18 variables. The construction of a quantum circuit performing the sign change of the amplitude of the sought state (see Grover's algorithm) requires at least 18 qubits for implementing SDES and one more flag qubit. Thus, Figs. 4–7 demonstrate a quantum circuit with the minimum number of logical qubits that implements SDES key searching by Grover's algorithm using a single plaintext–ciphertext pair.

Table 1. Probability distribution of keys after 25 Grover iterations when searching for exactly one target value

SDES key	Probability to obtain an appropriate key as a result of measurement of qubits
00000 00000	5.26642×10^{-7}
00000 00001	5.26642×10^{-7}
$\vdots$	$\vdots$
11000 10011	0.99946124
$\vdots$	$\vdots$
11111 11110	5.26642×10^{-7}
11111 11111	5.26642×10^{-7}

Table 2. Probability distribution of keys after 18 Grover iterations when searching for one of two target values

SDES key	Probability to obtain an appropriate key as a result of measurement of qubits
00000 00000	4.118199×10^{-6}
00000 00001	4.118199×10^{-6}
$\vdots$	$\vdots$
00100 10111	0.4978955
$\vdots$	$\vdots$
00110 11111	0.4978955
$\vdots$	$\vdots$
11111 11110	4.118199×10^{-6}
11111 11111	4.118199×10^{-6}

Let us compare the total number of quantum gates in a single Grover iteration. In [13], quantum gates are calculated in the section "Complexity Analysis"; however, the authors do not explicitly present the total number of quantum gates. According to [13], the procedure of development of SDES round keys is integrated into a single step and requires eight CNOT gates; the distribution of gates over other SDES operations is shown in Table 4.

According to the SDES encryption algorithm and Table 4, using Figs. 2 and 3, we can calculate the total number of gates in the quantum circuit of a single Grover iteration from [13]. In the calculation, one should take into account the inversion of round transformations (the number of gates taking part in the round transformations is multiplied by two), the inversion of a flag cubit (a single generalized CNOT is required), the Grover diffusion (another generalized CNOT), as well as the fact that the permutation of two qubits (SWAP) is performed by three CNOTs.

In the quantum circuit shown in Figs. 4–7, the operations IP, P10, P8, E/P, LS-1, LS-2, and P4 are implemented without any gates, which is achieved by a simple renumbering of qubits, which can be calculated in advance.

Table 5 illustrates the comparison of the number of gates in a single Grover iteration.

5. CONCLUSIONS

In this paper, we have presented a quantum circuit with the minimum number of qubits (19 qubits) that implements SDES key searching by Grover's quantum algorithm using a single pair of a plaintext and a

Table 3

Number of Grover iterations	Characteristics of quantum circuits and their simulation time in Quipper on an Intel Core i7-4470K 3.50 GHz processor
A single Grover iteration	Quantum exhaustive key search (1 key): 34: "H, arity 1" 17: "Init0" 2: "Init1" 11: "Meas" 84: "X, arity 1" 72: "not, arity 1 controls 1" 36: "not, arity 1 controls 2" 8: "not, arity 1 controls 3" 12: "not, arity 1 controls 4" 1: "not, arity 1 controls 4+4" 1: "not, arity 1 controls 9" 8: "swap, arity 2" Total gates: 286 Inputs: 0 Outputs: 19 Qubits in circuit: 19 Operation time: 290.288319 s
25 Grover iterations (search for a single key)	Quantum exhaustive key search (1 key): 562: "H, arity 1" 17: "Init0" 2: "Init1" 11: "Meas" 2100: "X, arity 1" 1800: "not, arity 1 controls 1" 900: "not, arity 1 controls 2" 200: "not, arity 1 controls 3" 300: "not, arity 1 controls 4" 25: "not, arity 1 controls 4+4" 25: "not, arity 1 controls 9" 200: "swap, arity 2" Total gates: 6142 Inputs: 0 Outputs: 19 Qubits in circuit: 19 Operation time: 7665.869522 s

Table 3 (Contd.)

Number of Grover iterations	Characteristics of quantum circuits and their simulation time in Quipper on an Intel Core i7-4470K 3.50 GHz processor
18 Grover iterations	Quantum exhaustive key search (2 keys):
(search for one of two keys)	408: “H, arity 1”
	14: “Init0”
	5: “Init1”
	11: “Meas”
	1512: “X, arity 1”
	1296: “not, arity 1 controls 1”
	648: “not, arity 1 controls 2”
	144: “not, arity 1 controls 3”
	216: “not, arity 1 controls 4”
	18: “not, arity 1 controls 4+4”
	18: “not, arity 1 controls 9”
	144: “swap, arity 2”
	Total gates: 4434
	Inputs: 0
	Outputs: 19
	Qubits in circuit: 19
	Operation time = 6593.336957 s

Table 4. Quantum gates distribution of SDES operations from [13]

E/P	8 CNOT
XOR semibytes	4 CNOT
P4	4 CNOT
XOR with a round key	8 CNOT
SWAP	12 CNOT
For each S-box	2 × 32 X gates
	2 × 48 Toffoli
	2 × 32 CNOT

Table 5. Comparison of the number of gates in a single Grover iteration

Operation	In [13]	In Figs. 4–7
X	404	84
H	34	34
CNOT	936	72 + 24
Toffoli	576	36
Generalized CNOT	2	22

ciphertext, whereas, in [13], the corresponding quantum circuit is constructed on 61 qubits.

Using a Quipper quantum simulator, we have performed a simulation of the constructed quantum circuits for 18 and 25 Grover iterations, obtained the corresponding success probabilities of Grover’s algorithm in the SDES key searching problem in the cases when, for a known pair of blocks of a plaintext and a ciphertext (P, C), there exists a single key $K \in V_{10}$ on which $E_{SDES}(K, P) = C$ and two keys $K_1, K_2 \in V_{10}$ on which $E_{SDES}(K_1, P) = C$ and $E_{SDES}(K_2, P) = C$.

A test example of an educational block cipher SDES is presented in Appendix A, and program implementations of Grover’s algorithm in problems of searching for one and two target values are demonstrated in Appendices B and C. A program implementation of an SDES key search by Grover’s quantum algorithm in a Quipper quantum simulator is presented in Appendix D.

APPENDICES

APPENDIX A: A Test Example of SDES

Take a block of a plaintext $P = 00101000$ and $K = 1100011110$. Formation of a round key k_1:

Bit numbers	1	2	3	4	5	6	7	8	9	10
K	1	1	0	0	0	1	1	1	1	0
P10(K)	0	0	1	1	0	0	1	1	1	1
LS-1(P10(K))	0	1	1	0	0	1	1	1	1	1
k_1 = P8(LS-1(P10(K)))	1	1	1	0	1	0	0	1		

Formation of a round key k_2:

Bit numbers	1	2	3	4	5	6	7	8	9	10
K	1	1	0	0	0	1	1	1	1	0
P10(K)	0	0	1	1	0	0	1	1	1	1
LS-3(P10(K))	1	0	0	0	1	1	1	0	1	1
k_2 = P8(LS-3(P10(K)))	1	0	1	0	0	1	1	1		

In this example, $P = 00101000$ and IP(P) = 00100010.

1. Take $P = 00101000$ and $K = 1100011110$.

2. IP(P) = 00100010.

3. $f_{k_1}(L, R) = f_{11101001}(00100010) = (0010 \oplus F(0010, 11101001), 0010)$.

4. $F(0010, 11101001) = \mathrm{P4} \circ \mathrm{Sboxes} \circ (11101001 \oplus \mathrm{E/P}(0010))$:

Bit numbers	1	2	3	4	5	6	7	8
R	0	0	1	0				
E/P(R)	0	0	0	1	0	1	0	0
k_1	1	1	1	0	1	0	0	1
E/P(R) $\oplus k_1$	1	1	1	1	1	1	0	1
Sboxes(E/P(R) $\oplus k_1$)	1	0	0	0				
P4(Sboxes(E/P)$\oplus k_1$))	0	0	0	1				

5. We have calculated $F(0010, 11101001) = 0001$ and obtained $f_{k_1}(L, R) = (0011, 0010)$.

6. SW(0011, 0010) = (0010, 0011).

7. $f_{k_2}(L, R) = f_{10100111}(00100011) = (0010 \oplus F(0011, 10100111), 0011)$:

Bit numbers	1	2	3	4	5	6	7	8
R	0	0	1	1				
E/P(R)	1	0	0	1	0	1	1	0
k_2	1	0	1	0	0	1	1	1
E/P(R) $\oplus k_2$	0	0	1	1	0	0	0	1
Sboxes(E/P(R) $\oplus k_2$)	1	0	1	0				
P4(Sboxes(E/P)$\oplus k_2$))	0	0	1	1				

8. We have calculated $F(0011, 10100111) = 0011$ and obtained $f_{k_2}(L, R) = (0001, 0011)$.

9. Apply IP^{-1}:

Bit numbers	1	2	3	4	5	6	7	8
L, R	0	0	0	1	0	0	1	1
IP^{-1}(L, R)	1	0	0	0	1	0	1	0

Examples of SDES encryption with the key $K = 1100011110$:

$$E_{SDES}(K, 00101000) = 10001010,$$
$$E_{SDES}(K, 10001101) = 11010000,$$
$$E_{SDES}(K, 11110010) = 11011010,$$
$$E_{SDES}(K, 01010111) = 01100000.$$

APPENDIX B: Search for a Single Target Value by Grover's Algorithm

Let $N = 2^3$, $M = 1$, and the target value be TargetValue=7. Determine the probability of success in the search for a target value depending on the number of iterations in Grover's algorithm.

Listing 1. Program implementation in Wolfram Mathematica

```
TargetValue=7; (*Define the number of an element that we want to obtain by measuring qubits*)
NumberOfQubits=3; (*determined the number of qubits*)
H= HadamardMatrix[2^NumberOfQubits];
(*Initiated the Hadamard matrix*)
Numb=2^NumberOfQubits; (*introduced an additional variable for a shorter expression*)
matrixD=ConstantArray[ConstantArray[2/Numb,Numb],Numb]-IdentityMatrix[Numb];
(*Initiated the D matrix (the Grover diffusion)*)
```

```
Print[" Hadamard matrix: \n ",MatrixForm[H]];
```

$$H=\begin{pmatrix}
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\
\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}}
\end{pmatrix}$$

```
Print["Matrix D (Grover diffusion): \n ",MatrixForm[matrixD]];
```

$$D=\begin{pmatrix}
-\frac{3}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & -\frac{3}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & -\frac{3}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{3}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{3}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{3}{4} & \frac{1}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{3}{4} & \frac{1}{4} \\
\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{3}{4}
\end{pmatrix}$$

```
FirstState=ConstantArray[0,2^NumberOfQubits];
FirstState[[1]]=1;
(*Initiated the initial state*)
Print["Initiated the initial state:",FirstState];
```

$$\text{FirstState} = \{1,0,0,0,0,0,0,0\}$$

```
State=FirstState.H; (*Applied Hadamard gates to each qubit,
  i.e., multiplied the vector FirstState by matrix H*)
```

$$State=\left\{\frac{1}{2\sqrt{2}},\frac{1}{2\sqrt{2}},\frac{1}{2\sqrt{2}},\frac{1}{2\sqrt{2}},\frac{1}{2\sqrt{2}},\frac{1}{2\sqrt{2}},\frac{1}{2\sqrt{2}},\frac{1}{2\sqrt{2}}\right\}$$

```
i=0;
Print["Iteration no., i , probability of success= ", N[(State[[Target-
  Value]]*
  State[[TargetValue]])],"probability of failure= ",
  N[(1-State[[TargetValue]]*State[[TargetValue]])] ];
```

Iteration no. 0, probability of success = 0.125, probability of failure = 0.875;

```
(*Change of sign of the amplitude of the target value*)
State[[TargetValue]]=(-1)*State[[TargetValue]];
Print["changed the sign of the target value: \n",State];
```

$$State = \left\{\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}, -\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{2}}\right\}$$

```
State=State.matrixD; (*Applied the Grover diffusion, multiplied State
   by matrix D*)
Print["Applied the Grover diffusion, multiplied State by matrix D:
   \n",State];
```

$$State = \left\{\frac{1}{4\sqrt{2}}, \frac{1}{4\sqrt{2}}, \frac{1}{4\sqrt{2}}, \frac{1}{4\sqrt{2}}, \frac{1}{4\sqrt{2}}, \frac{1}{4\sqrt{2}}, \frac{5}{4\sqrt{2}}, \frac{1}{4\sqrt{2}}\right\}$$

```
i=1;
Print["Iteration no., i , probability of success= ", N[(State[[Target-
   Value]]*
   State[[TargetValue]])],"probability of failure= ", N[(1-State[[Target-
   Value]]*
   State[[TargetValue]])] ];
```

Iteration no. 1, probability of success = 0.78125, probability of failure = 0.21875;

```
NumberOfGroverIterations=30;
(*defined the number of iterations, for example, 30*)
For[i=2,i<=NumberOfGroverIterations,i++,
State[[TargetValue]]=(-1)*State[[TargetValue]];
   (*changed the sign of the target value*)
State=State.matrixD; (*Applied the Grover diffusion, multiplied State
   by matrix D*)
p=State[[TargetValue]]*State[[TargetValue]]
Print["Iteration no., i , Probability of success = ", N[p]," Probability
   of failure= ", N[1-p]];
]
```

Let us calculate the probability of successful search for the target value (TargetValue = 7) depending on the number of iterations of Grover's algorithm on three qubits (see Table 6). The optimal number of Grover iterations in this example is estimated as $\left[\frac{\pi}{4}\sqrt{\frac{2^3}{1}}\right]$ = [2.221441469079183] = 2.

APPENIX C: Search for One of the two Target Values of Grover's Algorithm

Let $N = 2^{10}$, $M = 2$, and the target value is TargetValue $\in$ {151, 223}. Determine the probability of success in the search for a target value depending on the number of iterations in Grover's algorithm (see Table 7). The optimal number of Grover iterations in this example is estimated as $\left[\frac{\pi}{4}\sqrt{\frac{2^{10}}{2}}\right]$ = [17.771531752633464] = 18.

Table 6

Number of Grover iteration	Probability of success of Grover iteration, i.e., probability to obtain TargetValue=7 as a result of measurement of qubits	Probability of failure of Grover iteration, i.e., probability to obtain TargetValue≠7 as a result of measurement of qubits.
1	0.78125	0.21875
2	**0.945313**	**0.0546875**
3	0.330078	0.669922
4	0.012207	0.987793
5	0.547974	0.452026
6	**0.999786**	**0.000213623**
7	0.576973	0.423027
8	0.0194569	0.980543
9	0.302891	0.697109
10	**0.931266**	**0.068734**
11	0.804925	0.195075
12	0.144965	0.855035
13	0.106316	0.893684
14	0.756614	0.243386
15	**0.957837**	**0.0421627**
16	0.357846	0.642154
17	0.0066241	0.993376
18	0.51881	0.48119
19	**0.998078**	**0.00192151**
20	0.605709	0.394291
21	0.0283488	0.971651
22	0.276378	0.723622
23	**0.915746**	**0.0842543**
24	0.827558	0.172442
25	0.166144	0.833856
26	0.0889775	0.911022
27	0.7311	0.2689
28	**0.968798**	**0.0312024**

Table 7

Number of Grover iteration	Probability of success of Grover iteration, i.e., probability to obtain one of the values TargetValue = {151, 223} as a result of measurement of qubits	Probability of failure of Grover iteration, i.e., probability to obtain a value other than from TargetValues as a result of measurement of qubits
1	0.0174867	0.982513
2	0.0480693	0.951931
3	0.0927473	0.907253
4	0.150127	0.849873
5	0.218419	0.781581
6	0.295493	0.704507
7	0.378945	0.621055
8	0.466173	0.533827
9	0.554456	0.445544
10	0.641041	0.358959
11	0.723227	0.276773
12	0.79845	0.20155
13	0.864365	0.135635
14	0.918916	0.0810837
15	0.960402	0.0395983
16	0.987528	0.0124724
17	*0.999448*	*0.000551974*
18	**0.995791**	**0.0042088**
19	0.976671	0.0233288
20	0.942684	0.0573158
21	0.89489	0.10511
22	0.83478	0.16522
23	0.764229	0.235771
24	0.685436	0.314564
25	0.60086	0.39914
26	0.513139	0.486861
27	0.425007	0.574993
28	0.339214	0.660786

Listing 2. Program implementation in Wolfram Mathematica

```
(*A case when there are several sought numbers*)
TargetValues={151,223}; (*sought numbers*)
NumberOfQubits=10; (*determined the number of qubits*)
H= HadamardMatrix[2^NumberOfQubits]; (*Initiated the Hadamard matrix*)
Numb=2^NumberOfQubits; (*introduced an additional variable for a shorter
 expression*)
matrixD=ConstantArray[ConstantArray[2/Numb,Numb],Numb]-IdentityMa-
 trix[Numb];
(*Initiated the matrix D (the Grover diffusion)*)
FirstState=ConstantArray[0,2^NumberOfQubits];
FirstState[[1]]=1; (* Initiated the initial state*)
State=FirstState.H;
(* Applied Hadamard gates to each qubit, i.e., multiplied the vector
 FirstState by matrix H*)
```

$$\mathrm{p} = \sum_{k=1}^{\mathrm{Length[TargetValues]}} \mathrm{State[[TargetValues[[}k\mathrm{]]]]} * \mathrm{State[[TargetValues[[}k\mathrm{]]]]};$$

```
i=0;Print["Iteration no., i , probability of success = ", N[p],",
 probability of failure= ", N[1-p]];
(*Change of sign of the amplitude of the target value*)
For[i=1,i<=Length[TargetValues],i++,
State[[ TargetValues[[i]] ]]=(-1)*State[[ TargetValues[[i]] ]]; ];
State=State.matrixD;
(*Applied the Grover diffusion, multiplied the vector State by matrix D*)
```

$$\mathrm{p} = \sum_{k=1}^{\mathrm{Length[TargetValues]}} \mathrm{State[[TargetValues[[}k\mathrm{]]]]} * \mathrm{State[[TargetValues[[}k\mathrm{]]]]};$$

```
Print["Iteration no., i , probability of success = ", N[p],", probabil-
 ity of failure= ", N[1-p]];
NumberOfGroverIterations=30;
(*defined the number of Grover iterations*)
For[i=2,i<=NumberOfGroverIterations,i++,
For[j=1,j<=Length[TargetValues],j++,
State[[TargetValues[[j]]]]=(-1)*State[[TargetValues[[j]] ]];];
State=State.matrixD;
```

$$\mathrm{p} = \sum_{k=1}^{\mathrm{Length[TargetValues]}} \mathrm{State[[TargetValues[[}k\mathrm{]]]]} * \mathrm{State[[TargetValues[[}k\mathrm{]]]]};$$

```
Print["Iteration no., i , probability of success = ", N[p],", probabil-
 ity of failure=", N[1-p]];]
```

The probability of success after 17 Grover iterations turns out to be slightly greater than the probability of success after 18 Grover iterations. This does not contradict anything because $\left[\frac{\pi}{4}\sqrt{\frac{N}{M}}\right]$ is the upper bound for the number of iterations (see [12, p. 318]).

APPENDIX D: Program Implementation of Grover's Algorithm for the SDES Key Search by a Plaintext–Ciphertext Pair on a Quipper Quantum Simulator

The program implementation represents three files:

(1) QSDES.hs—implementation of an SDES algorithm,

(2) Grover.hs—implementation of Grover's algorithm on the basis of QSDES,

(3) Main.hs—start of simulation of the quantum circuit.

Here is the content of these files.

QSDES.HS

```
module QSDES where
- Block of quantum implementation of the SDES encryption circuit

import Quipper
import QuipperLib.Simulation
import System.Random
import Quipper.Printing
import Quipper.QData

- Summation with a key
sum_qubit :: ([Qubit], [Qubit]) -> Circ [Qubit]
sum_qubit ([k0, k1, k2, k3], [x0, x1, x2, x3]) = do
qnot_at x0 ‘controlled‘ [k0]
qnot_at x1 ‘controlled‘ [k1]
qnot_at x2 ‘controlled‘ [k2]
qnot_at x3 ‘controlled‘ [k3]
return [x0, x1, x2, x3]
- S0
s0_box :: ([Qubit], [Qubit]) -> Circ [Qubit]
s0_box ([x0, x1, x2, x3], [s0, s1]) = do
comment “S0_box”
qnot_at s0 ‘controlled‘ [x3]
qnot_at s0 ‘controlled‘ [x0, x1, x2, x3]
qnot_at s0 ‘controlled‘ [x0, x2]
x0 <- gate_X x0
qnot_at s0 ‘controlled‘ [x0, x1]
x2 <- gate_X x2
qnot_at s1 ‘controlled‘ [x0, x2]
x0 <- gate_X x0
qnot_at s1 ‘controlled‘ [x0, x1, x2, x3]
x1 <- gate_X x1
qnot_at s1 ‘controlled‘ [x0, x1]
x3 <- gate_X x3
qnot_at s1 ‘controlled‘ [x0, x3]
x1 <- gate_X x1
x2 <- gate_X x2
x3 <- gate_X x3
return [s0, s1]

- S1
s1_box :: ([Qubit], [Qubit]) -> Circ [Qubit]
s1_box ([x0, x1, x2, x3], [s2, s3]) = do
comment “S1_box”
qnot_at s2 ‘controlled‘ [x1]
qnot_at s3 ‘controlled‘ [x0, x2, x3]
x3 <- gate_X x3
qnot_at s3 ‘controlled‘ [x0, x3]
qnot_at s2 ‘controlled‘ [x0, x3]
x1 <- gate_X x1
```

```
qnot_at s2 ‘controlled‘ [x0, x1, x2, x3]
qnot_at s3 ‘controlled‘ [x2, x3]
x2 <- gate_X x2
x3 <- gate_X x3
qnot_at s2 ‘controlled‘ [x2, x3]
x0 <- gate_X x0
x1 <- gate_X x1
qnot_at s3 ‘controlled‘ [x0, x1, x3]
x0 <- gate_X x0
x2 <- gate_X x2
return [s2, s3]

—— round 1 ——

round1 :: ([Qubit], [Qubit]) -> Circ [Qubit]
round1 ([k0, k1, k2, k3, k4, k5, k6, k7, k8, k9], [x0, x1, x2, x3, x4,
x5, x6, x7]) = do
comment “ROUND_1"

[x6, x3, x7, x4] <- sum_qubit ([k0, k6, k8, k3], [x6, x3, x7, x4])
[x0, x1] <- s0_box ([x6, x3, x7, x4], [x0, x1])
[x6, x3, x7, x4] <- sum_qubit ([k0, k6, k8, k3], [x6, x3, x7, x4])

[x7, x4, x6, x3] <- sum_qubit ([k7, k2, k9, k5], [x7, x4, x6, x3])
[x2, x5] <- s1_box ([x7, x4, x6, x3], [x2, x5])
[x7, x4, x6, x3] <- sum_qubit ([k7, k2, k9, k5], [x7, x4, x6, x3])

return [x0, x1, x2, x3, x4, x5, x6, x7]

—— round 2 ——

round2 :: ([Qubit], [Qubit]) -> Circ [Qubit]
round2 ([k0, k1, k2, k3, k4, k5, k6, k7, k8, k9], [x0, x1, x2, x3, x4,
x5, x6, x7]) = do
comment “ROUND_2"

[x0, x1, x5, x2] <- sum_qubit ([k7, k2, k5, k4], [x0, x1, x5, x2])
[x6, x3] <- s0_box ([x0, x1, x5, x2], [x6, x3])
[x0, x1, x5, x2] <- sum_qubit ([k7, k2, k5, k4], [x0, x1, x5, x2])

[x5, x2, x0, x1] <- sum_qubit ([k9, k1, k8, k0], [x5, x2, x0, x1])
[x4, x7] <- s1_box ([x5, x2, x0, x1], [x4, x7])
[x5, x2, x0, x1] <- sum_qubit ([k9, k1, k8, k0], [x5, x2, x0, x1])

return [x0, x1, x2, x3, x4, x5, x6, x7]

- QSDES circuit (“direct”).
sdes :: ([Qubit], [Qubit]) -> Circ ([Qubit], [Qubit])
sdes (key, plaintext) = do
let [k0, k1, k2, k3, k4, k5, k6, k7, k8, k9] = key
let [x0, x1, x2, x3, x4, x5, x6, x7] = plaintext

 comment_with_label “Key”
 (k0, k1, k2, k3, k4, k5, k6, k7, k8, k9)
 (“k0", "k1", "k2", "k3", "k4", "k5", "k6", "k7", "k8", "k9")

 comment_with_label “Plaintext”
 (x0, x1, x2, x3, x4, x5, x6, x7)
 (“x0", "x1", "x2", "x3", "x4", "x5", "x6", "x7")

 [x0, x1, x2, x3, x4, x5, x6, x7] <- round1 ([k0, k1, k2, k3, k4, k5,
 k6, k7, k8, k9],
  [x0, x1, x2, x3, x4, x5, x6, x7])
 [x0, x1, x2, x3, x4, x5, x6, x7] <- round2 ([k0, k1, k2, k3, k4, k5,
 k6, k7, k8, k9],
  [x0, x1, x2, x3, x4, x5, x6, x7])

 swap x6 x0
 swap x3 x1
 swap x4 x2
 swap x5 x7

 return ([k0, k1, k2, k3, k4, k5, k6, k7, k8, k9], [x0, x1, x2, x3, x4,
 x5, x6, x7])

 - QSDES circuit (“direct”)
 sdes_reverse :: ([Qubit], [Qubit]) -> Circ ([Qubit], [Qubit])
 sdes_reverse = reverse_generic_endo sdes

 - Test QSDES circuit with a key superposition;
 - is used for test no. 2.
 sdes_key_superposition :: [Bool] -> Circ ([Qubit], [Qubit])
 sdes_key_superposition plaintext = do
 key_in_superposition <- qinit (replicate 10 False)
 mapUnary hadamard key_in_superposition
 plaintext <- qinit plaintext
 (key, cyphertext) <- sdes (key_in_superposition, plaintext)
 return (key, cyphertext)
```

GROVER.HS

```
module Grover where
- Grover’s algorithm for QSDES

import Quipper
import Quipper.QData
import QSDES
- Oracle for QSDES
sdes_oracle :: ([Qubit], [Qubit], [Bool], Qubit) -> Circ Qubit
sdes_oracle (key, plaintext, cyphertext_bool, oracle) = do
(key, cyphertext) <- sdes (key, plaintext)
```

```
qnot_at oracle ‘controlled‘ cyphertext .==. cyphertext_bool
(key, plaintext) <- sdes_reverse (key, cyphertext)
return oracle
- Circuit inversion about the mean or Conditional Phase Flip (CPF)
inversion_about_mean :: [Qubit] -> Circ [Qubit]
inversion_about_mean top_qubits = do
comment “start inversion about mean”
mapUnary hadamard top_qubits
mapUnary gate_X top_qubits
let pos = (length top_qubits) - 1
let target_qubit = top_qubits !! pos
let controlled_qubit = take pos top_qubits
hadamard_at target_qubit
qnot_at target_qubit ‘controlled‘ controlled_qubit
hadamard_at target_qubit
mapUnary gate_X top_qubits
mapUnary hadamard top_qubits
comment “end inversion about mean”
return top_qubits
- Grover’s algorithm for QSDES
grover_search_circuit_sdes :: (Int, [Bool], [Bool]) -> Circ ([Bit], Bit)
grover_search_circuit_sdes (iterations_num, plaintext, cyphertext) = do
key <- qinit (replicate 10 False)
plaintext <- qinit plaintext
oracle <- qinit True
mapUnary hadamard key
hadamard_at oracle
- Start of Grover iterations
let index = iterations_num
for 1 (index) 1 $ \i -> do
comment “start grover iteration”
oracle <- sdes_oracle (key, plaintext, cyphertext, oracle)
key <- inversion_about_mean key
comment “after grover iteration”
endfor
- Measurement of qubits, return of the result
hadamard_at oracle
(key, oracle) <- measure (key, oracle)
- cdiscard oracle
return (key, oracle)
```

MAIN.HS

```
module Main where
- Main block. Start of examples

import System.Random
import Data.Time
import Quipper
import Quipper.Printing
import QuipperLib.Simulation
import Quipper.QData
 import QSDES
 import Grover

 - START OF THE PROGRAM
 main :: IO ()
 - main = test1_circuit
 main = test3_exec –25 Grover iterations, finding the key distribution.

 - Functionality test. Test of the correctness of composing the circuit
 test1_circuit :: IO ()
 test1_circuit = do
 putStrLn “QSDES functionality test:”
 print_generic GateCount sdes ((replicate 10 qubit),(replicate 8 qubit))
 print_generic PDF sdes ((replicate 10 qubit),(replicate 8 qubit))

 test1_exec :: IO ()
 test1_exec = do
 putStrLn “QSDES functionality test:”
 print_generic GateCount sdes ((replicate 10 qubit),(replicate 8 qubit))
 g <- newStdGen
 print $ run_generic g (0.0 :: Double) sdes ([True,True,False,False,False,True,True,True,True,False], [False,False,True,False,True,False,False,False])
 print $ run_generic g (0.0 :: Double) sdes ([True,True,False,False,False,True,True,True,True,False], [True,False,False,False,True,True,False,True])
 print $ run_generic g (0.0 :: Double) sdes ([True,True,False,False,False,True,True,True,True,False], [True,True,True,True,False,False,True,False])
 print $ run_generic g (0.0 :: Double) sdes ([True,True,False,False,False,True,True,True,True,False], [False,True,False,True,False,True,True,True])

 - Testing the circuit with key qubits in superposition.
 test2_circuit :: IO ()
 test2_circuit = do
 putStrLn “QSDES results when key is in superposition:”
 print_generic PDF (sdes_key_superposition [True,False,False,True,True,False,True,False])

 test2_exec :: IO ()
 test2_exec = do
 putStrLn “QSDES results when key is in superposition:”
 - t1 <- getZonedTime
 - putStrLn $ formatTime defaultTimeLocale “%FT%T%z” t1
 g <- newStdGen
```

```
print $ sim_generic undefined (sdes_key_superposition
([True,False,False,True,True,False,True,False]))
- t2 <- getZonedTime
- putStrLn $ formatTime defaultTimeLocale "%FT%T%z" t2


- Search for the key pair of public and private texts.
- Example with one true keys.
test3_circuit :: IO()
test3_circuit = do
putStrLn "Quantum exhaustive key search (1 key):"
let parameters = (25,[False,False,False,True,False,False,False,False],
[False,False,True,True,False,False,True,True])
print_generic GateCount (grover_search_circuit_sdes parameters)
- print_generic PDF (grover_search_circuit_sdes parameters)

test3_exec :: IO()
test3_exec = do
putStrLn "Quantum exhaustive key search (1 key):"
startTime <- getCurrentTime
let parameters = (25,[False,False,False,True,False,False,False,False],
 [False,False,True,True,False,False,True,True]) – O.T.,C.T.
g <- newStdGen
- print $ run_generic g (0.0 :: Double) (grover_search_circuit_sdes
parameters)
print_generic GateCount (grover_search_circuit_sdes parameters)
print $ sim_generic undefined (grover_search_circuit_sdes parameters)
stopTime <- getCurrentTime
let deltaTime = show $ diffUTCTime stopTime startTime
putStrLn deltaTime

- Search for the key pair of the open and closed text.
- Example with two true keys.
test4_circuit :: IO()
test4_circuit = do
putStrLn "Quantum exhaustive key search (2 keys):"
let parameters = (18,[True,False,True,False,False,True,False,True],
 [False,False,True,True,False,True,True,False])
print_generic GateCount (grover_search_circuit_sdes parameters)
- print_generic PDF (grover_search_circuit_sdes parameters)

test4_exec :: IO()
test4_exec = do
putStrLn "Quantum exhaustive key search (2 keys):"
startTime <- getCurrentTime
let parameters = (18,[True,False,True,False,False,True,False,True],
 [False,False,True,True,False,True,True,False])
print_generic GateCount (grover_search_circuit_sdes parameters)
g <- newStdGen
```

```
- print $ run_generic g (0.0 :: Double) (grover_search_circuit_sdes
  parameters)
print $ sim_generic undefined (grover_search_circuit_sdes parameters)
stopTime <- getCurrentTime
let deltaTime = show $ diffUTCTime stopTime startTime
putStrLn deltaTime
```

REFERENCES

1. H. Bernien, S. Schwartz, A. Keesling, H. Levine, and A. Omran, Nature (London, U.K.) 551, 579 (2017); arXiv:1707.04344. doi 10.1038/nature2462210.1038/nature24622
2. J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, and P. Becker, Nature (London, U.K.) 551, 601 (2017); arXiv:1708.01044. doi 10.1038/nature2465410.1038/nature24654
3. https://www.ibm.com/blogs/research/2017/11/thefuture-is-quantum/.
4. M. Veldhorst, H. G. J. Eenink, C. H. Yang, and A. S. Dzurak, Nat. Commun. **8**, 1766 (2017); https://doi.org/10.1038/s41467-017-01905-6
5. C. S. Calude and E. Calude, arXiv:1712.01356v1.
6. J. Kelly, A Preview of Bristlecone, Google's New Quantum Processor, Quantum AI Lab. https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html. Accessed March 05, 2018.
7. P. W. Shor, J. Comput. **26**, 1484 (1997).
8. D. R. Simon, SIAM J. Comput. **26**, 1474 (1997).
9. M. Kaplan, G. Leurent, A. Leverrier, et al., Lect. Notes Comp. Sci. 9815, (2016).
10. L. K. Grover, in *Proceedings of the 28th ACM Symposium on Theory of Computing STOC 1996*, Ed. by G. L. Miller (ACM, 1996), p. 212.
11. G. Brassard, P. Hoyer, M. Mosca, et al., arXiv:quant-ph/0005055.
12. M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge Univ. Press, Cambridge, 2000).
13. M. Almazrooie, A. Samsudin, R. Abdullah, and K. N. Mutter, Springer Plus **5**, 1494 (2016). doi 10.1186/s40064-016-3159-4
14. Quantum Simulator Libquantum. http://www.libquantum.de.
15. A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron, arXiv:1304.5485v1.
16. S. Siddiqui, M. J. Islam, and O. Shehab, arXiv:1406.4481v2 [quant-ph].
17. Quantum Simulator Quipper. http://www.mathstat.dal.ca/~selinger/quipper/.
18. C. Shannon, Bell Syst. Tech. J. **28**, 656 (1949).

Translated by I. Nikitin