Optimization of S-boxes GOST R 34.12-2015 "Magma" quantum circuits without ancilla qubits

Denisenko D.V., Nikitenkova M.V.

04.06.2019

RusCrypto 2019

We have to implement cryptoalgorithms (AES, SHA et al.) in the form of quantum circuits for applying quantum algorithms (Grover, Simon) to a cryptoalgorithms.

How to implement existing cryptographic algorithms in the form of quantum circuits?

Simplified-DES

Simplified-DES – two-round Feistel Network $E_{SDES} : V_{10} \times V_8 \to V_8$ with key $K \in V_{10}$.

Quantum exhaustive key search with simplified-DES as a case study, [1]

- SDES implementation – 60 qubits;
- Grover's key search with quantum simulator *libquantum* – 61 qubits.

Denisenko D.V., Nikitenkova M.V. *Application of Grover's Quantum Algorithm for SDES Key Searching*, [2]

- SDES implementation – 18 qubits;
- Grover's key search with quantum simulator *quipper* – 19 qubits.

- The work [2] showed that the minimum estimate of the number of qubits for finding the SDES key by Grover's quantum algorithm ($18 + 1 = 19$ qubits) is achievable;
- provides detailed examples of the application of the Grover algorithm, source code for implementations in Wolfram Mathematica and the quantum simulator *quipper*.

Quantum circuits for implementation of cryptographic transformations

To apply quantum algorithms to cryptoalgorithms, such as ciphers, it is necessary to present the encryption function $E : V_n \times V_m \to V_m$ in the form of a quantum circuit.

Denisenko D.V., Marshalko G.B., Nikitenkova M.V., Rudskoy V.I., Shishkin V.A. *Estimating the complexity of the Grover's algorithm for key search of block Ciphers Defined by GOST R 34.12-2015*, [3], used the approach with the representation of coordinate functions in the form of quantum circuits.

n-bit strings transform	Number of ancilla qubits	Number of quantum gates
$P \oplus Key$	0	n
$P + Key \; mod \; 2^n$	1	$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{25}{6}n + 8$
S-box	n	dependent on S-box, $> n$
Linear	n	$\leq n(n-1)$
Cyclic shift ($\lll$)	0	0

Table 1: The amount of resources for the implementation of elementary transformations

GOST R 34.12-2015 «Kuznyechik»

To implement one iteration of $E : V_{128} \times V_{128} \to V_{128}$ in the form of a quantum circuit required $128 + 128 + 128 + 128 = 512$ qubits (figure 1).

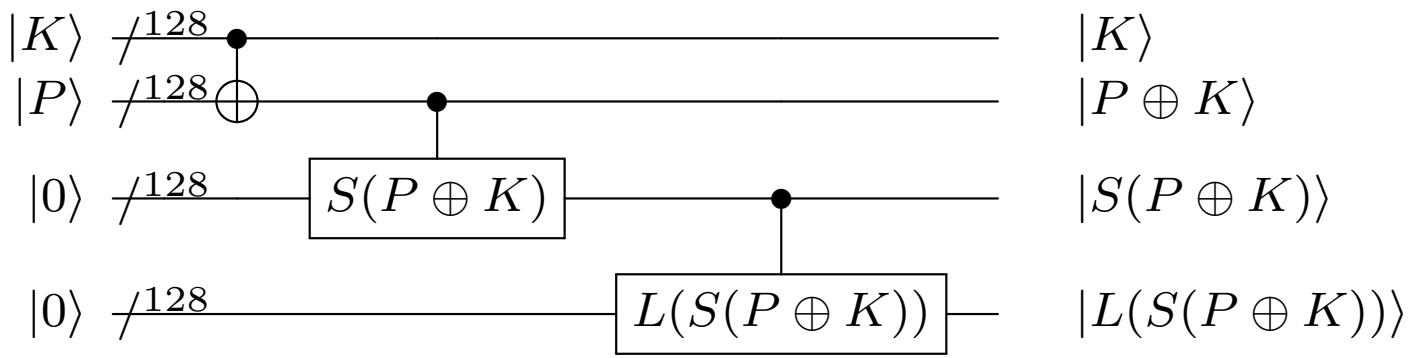


Figure 1: Quantum circuit of iteration GOST R 34.12-2015 «Kuznyechik»

ГОСТ Р 34.12-2015 «Магма»

To implement one iteration of $E : V_{32} \times V_{64} \to V_{64}$ in the form of a quantum circuit required $32 + 32 + 32 + 32 + 32 + 1 = 161$ qubits (figure 2).

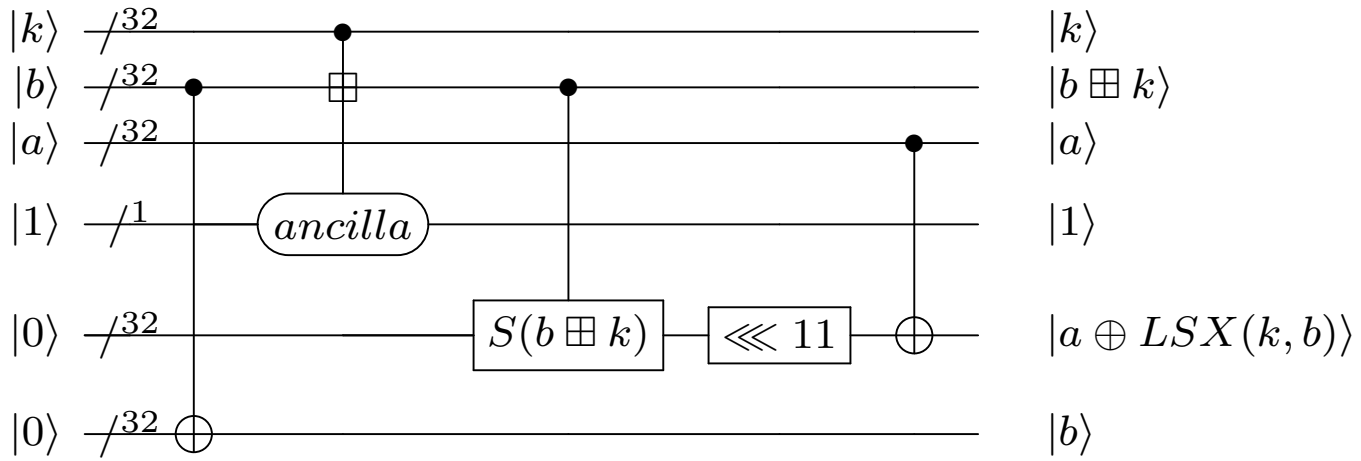


Figure 2: Quantum circuit of iteration GOST R 34.12-2015 «Magma».

Quantum circuit in fig. 2 specially constructed without reusing of qubits, it could be useful in another quantum computing model (measurement-based quantum computation, one-way quantum computer [7]).

GOST R 34.12-2015 «Magma»

To implement one iteration of $E : V_{32} \times V_{64} \to V_{64}$ with reusing of qubits required $32 + 32 + 32 + 32 + 1 = 129$ qubits (fig. 3).

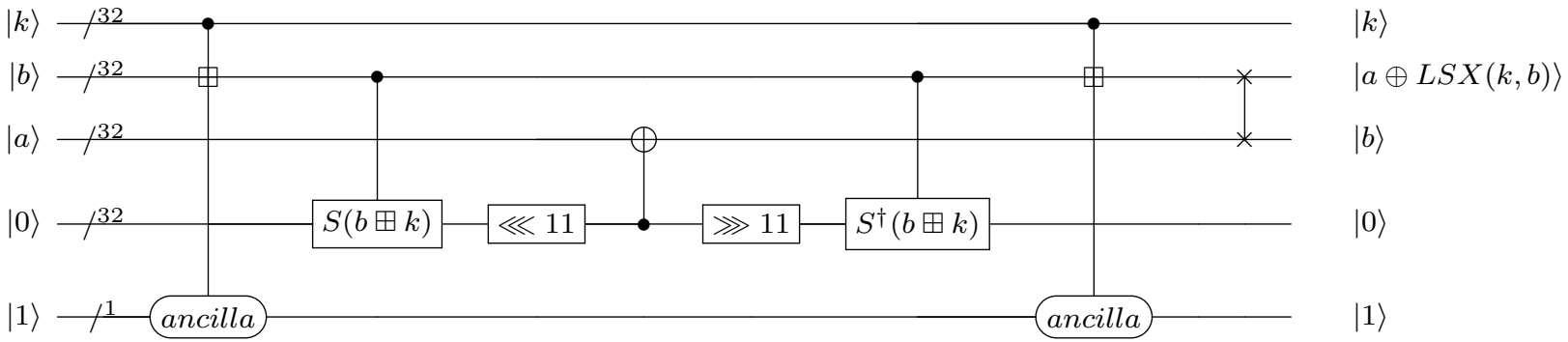


Figure 3: One iteration of GOST R 34.12-2015 «Magma» with reusing of qubits.

GOST R 34.12-2015 «Kuznyechik» with reusing of qubits

In the GOST R 34.12-2015 «Kuznyechik» there are 9 complete iterations and one more key XORing is applied.

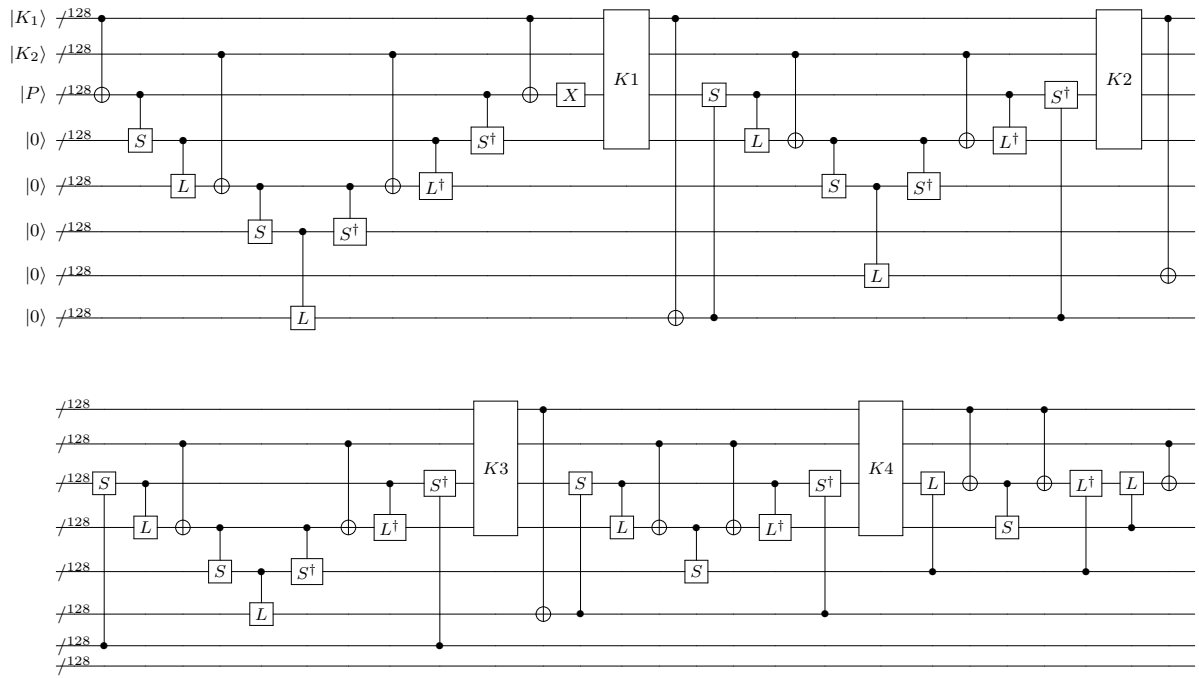


Figure 4: Quantum circuit of 10 rounds of «Kuznyechik» algorithm with reusing of qubits (on top - there are first 4 iterations of the algorithm, below - the remaining 5 full iterations and one - incomplete)

GOST R 34.12-2015 «Kuznyechik» key generation algorithm with reusing of qubits

In fig. 4 in blocks Ki, $(i = 1, 2, 3, 4)$ round keys of the «Kuznyechik» encryption algorithm are generated. Each block Ki includes 8 iterations of the quantum circuit shown in fig. 5.

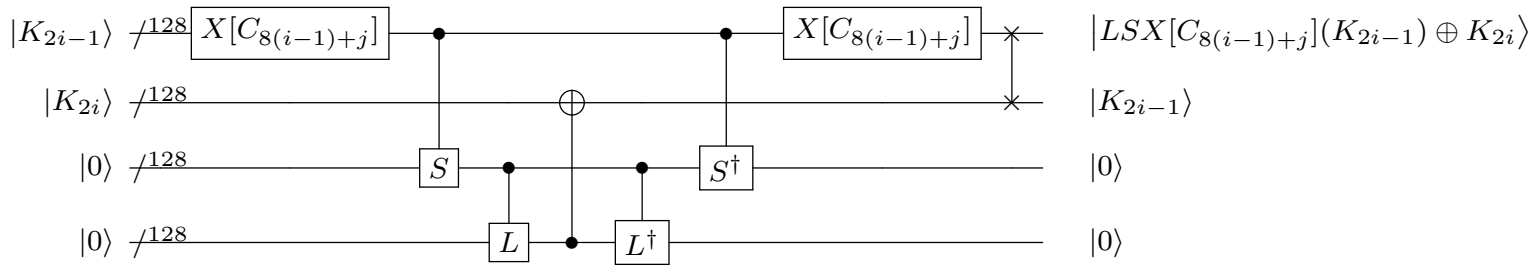


Figure 5: Quantum circuit for key generation algorithm of GOST R 34.12-2015 «Kuznyechik» with reusing of qubit $i = 1, 2, 3, 4, j = 1, 2, ..., 8$.

The implementation of S-boxes by quantum circuits without ancilla qubits

Quantum circuits that implement S-boxes GOST R 34.12-2015 «Magma» without ancilla qubits are first published in work:
Denisenko D.V. «Quantum circuits for S-box implementation without ancilla qubits» [4].

Quantum circuits for implementation of cryptographic transformations (new)

If in the structure of $E : V_n \times V_m \to V_m$ there isn't operation $\boxplus mod 2^t$, $t > 1$, then $n + m$ logical qubits are enough.

For $\boxplus$ mod 2^t operation, where $t > 1$, may require 1 additional qubit and $\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{25}{6}n + 8$ quantum gates (see [8]).

If it is possible to apply the quantum Fourier transform, the modular addition operation can be implemented without the use of ancilla qubits [9].

n-bit strings transform	Number of ancilla qubits	Number of quantum gates
$P \oplus Key$	0	n
$P + Key \ mod \ 2^n$	1	$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{25}{6}n + 8$
S-box	0	depend on S-box, $> n$
Linear	0	$> n$
Cyclic shift ($\lll$)	0	0

Table 2: The amount of resources for the implementation of elementary transformations

Cryptographic transformations of X, S and L can be realized in the form of quantum circuits without using ancilla qubits.

Sufficient number of logical qubits for implementation GOST R 34.12-2015 and AES.

GOST R 34.12-2015 «Magma»	$256 + 64 = 320$
GOST R 34.12-2015 «Kuznyechik»	$256 + 128 = 384$
AES-128	$128 + 128 = 256$
AES-192	$192 + 128 = 320$
AES-256	$256 + 128 = 384$

The minimum number of logical qubits required to hash function implementation in the form of a quantum circuit is defined by the maximum length of the internal state of the hash function.

Sufficient number of logical qubits for implementation SHA-2, SHA-3 and GOST R 34.11-2012

Algorithm	Minimum number of qubits for quantum circuit
SHA-2 (224, 256)	512
SHA-2 (384, 512)	1024
SHA-3	1600
GOST R 34.11-2012	1024

Example

Let's construct a quantum circuit that implements

$$\pi_1 = (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15).$$

The substitution $\pi_1 \in S(V_4)$. Denote $y = \pi_1(x)$, $x, y \in V_4$. The states $|x\rangle, |y\rangle$ are vector-columns from $L_{\mathbb{C}^{2^4}}$, the action of the operator $U|x\rangle = |y\rangle$ is a multiplication of the column vector $|x\rangle$ by the matrix $U \in \mathbb{C}_{2^4,2^4}$.

Definition 1

Let $N = 2^n$, $n \in \mathbb{N}$, and $e_1, e_2, \ldots, e_N$ be the basis of the vector space $L_{\mathbb{C}^N}$ over field of complex numbers $\mathbb{C}$. The unitary matrices $U \in \mathbb{C}_{2^n,2^n}$, nontrivially acting on no more than two basis vectors $e_1, e_2, \ldots, e_N$, are called *two-level unitary matrices* (see [5], section 4.5.1).

1. The unitary matrix for π_1:

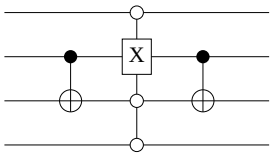
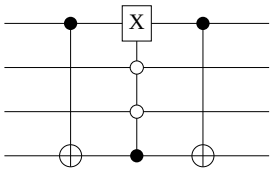
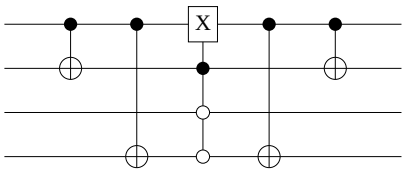
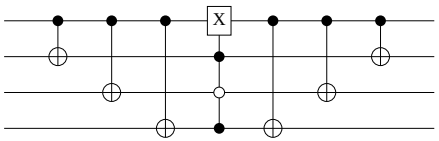
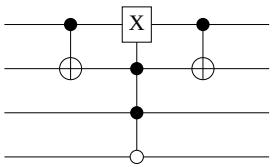
$$U_{\pi_1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

2. The matrix U_{π_1} can be represented as a product of two-level unitary matrices:

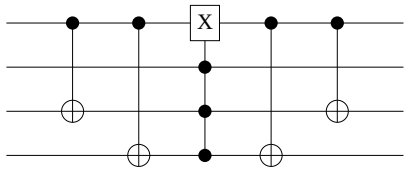
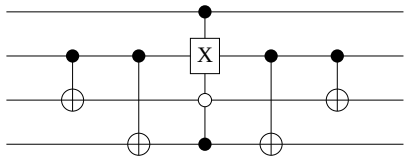
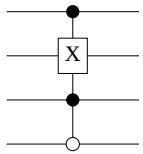
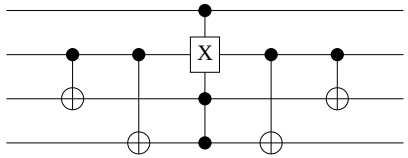
$$U_{\pi_1} = V_1 \cdot V_2 \cdot V_3 \cdot V_4 \cdot V_5 \cdot V_6 \cdot V_7 \cdot V_8 \cdot V_9.$$

$\pi_1 \rightarrow$ quantum circuit without ancilla qubits

The table contains two-level matrices $V_1, \ldots, V_9$, participating in the decomposition U_{π_1}, states s and t, on which two-level matrices act nontrivially, and quantum circuits implementing two-level matrices $V_1, \ldots, V_9$.

$V_1 = \{200, 4000, 2000, 1000, 800, 400, 8000, 100, 80, 40, 20, 10, 8, 4, 2, 1\}$	$s = \|0110\rangle$ $t = \|0000\rangle$	
$V_2 = \{8000, 80, 2000, 1000, 800, 400, 200, 100, 4000, 40, 20, 10, 8, 4, 2, 1\}$	$s = \|0001\rangle$ $t = \|1000\rangle$	
$V_3 = \{8000, 4000, 2000, 1000, 40, 400, 200, 100, 80, 800, 20, 10, 8, 4, 2, 1\}$	$s = \|0100\rangle$ $t = \|1001\rangle$	
$V_4 = \{8000, 4000, 2000, 1000, 800, 20, 200, 100, 80, 40, 400, 10, 8, 4, 2, 1\}$	$s = \|0101\rangle$ $t = \|1010\rangle$	
$V_5 = \{8000, 4000, 2000, 1000, 800, 400, 20, 100, 80, 40, 200, 10, 8, 4, 2, 1\}$	$s = \|0110\rangle$ $t = \|1010\rangle$	

$\pi_1 \rightarrow$ quantum circuit without ancilla qubits

$V_6 = \{8000, 4000, 2000, 1000, 800, 400, 200, 8, 80, 40, 20, 10, 100, 4, 2, 1\}$	$s = \|0111\rangle$ $t = \|1100\rangle$	
$V_7 = \{8000, 4000, 2000, 1000, 800, 400, 200, 100, 80, 2, 20, 10, 8, 4, 40, 1\}$	$s = \|1001\rangle$ $t = \|1110\rangle$	
$V_8 = \{8000, 4000, 2000, 1000, 800, 400, 200, 100, 80, 40, 2, 10, 8, 4, 20, 1\}$	$s = \|1010\rangle$ $t = \|1110\rangle$	
$V_9 = \{8000, 4000, 2000, 1000, 800, 400, 200, 100, 80, 40, 20, 8, 10, 4, 2, 1\}$	$s = \|1011\rangle$ $t = \|1100\rangle$	

Since $|y\rangle = U|x\rangle$, $|y\rangle = V_1 \cdot \ldots \cdot (V_8 \cdot (V_9 \cdot |x\rangle))$, the quantum circuit of U_{π_1}:

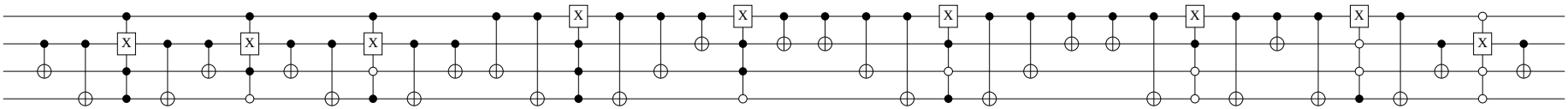


Figure 6: Quantum circuit $\pi_1 = (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15)$.

Estimations from Nielsen M.A., Chuang I.L., [5], sec. 4.5

Nielsen M.A., Chuang I.L., [5], sec. 4.5

- "... an arbitrary unitary matrix on an n qubit system may be written as a product of at most $2^{n-1}(2^n - 1)$ two-level unitary matrices."
- "Suppose U is a two-level unitary matrix on an n qubit quantum computer.
  ...
  Thus, implementing U requires $O(n^2)$ single qubit and CNOT-gates."

Estimations for π_1

- $n = 4$;
- $2^{4-1}(2^4 - 1) = 2^7 - 2^3 = 128 - 8 = 120$;

For implementation π_1 required only 9 unitary two-level matrices $V_1 \dots V_9$.

S-box

$$\pi_1 = (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15)$$

decomposed into the product of three cycles:

$$\pi_1 = s_1 \cdot s_2 \cdot s_3 = (0, 6, 5, 10, 4, 9, 14) \cdot (1, 8) \cdot (7, 12, 11)$$

Nontrivial transitions in cycles (coincide with transitions of states by $V_1 \dots V_9$).
$data_1 = \{(6, 0), (4, 9), (5, 10), (6, 10), (9, 14), (10, 14)\}$;
$data_2 = \{(1, 8)\}$;
$data_3 = \{(7, 12), (11, 12)\}$;

Optimal placement of transitions: (10,14)(11,12)(9,14)(7,12)(1,8)(6,10)(5,10)(4,9)(6,0)

Optimization of π_1 quantum circuit

Input: Substitution $s \in S(V_n)$.
Output: Quantum circuit for $s \in S(V_n)$ without ancilla qubits.

1. Represent s as a product of independent cycles and remove fixed points. As a result, we obtain $k \in \mathbb{N}$ independent cycles, $s = s_1 s_2 \dots s_k$;
2. Consider each cycle s_i, $i \in \overline{1,k}$ as an independent substitution, find the decomposition of the unitarian matrix U_{s_i} corresponding to the cycle $s_i, i \in \overline{1,k}$, into the product of two-level matrices (see [4], [5]):
$$U_{s_i} = V_1^i \cdot \dots \cdot V_t^i;$$
3. By the founded matrices $V_1^i \cdot \dots \cdot V_t^i$ we could determine all possible pairs of states
$$data_i = \{(x_j^i, y_j^i) : V_j^i \left|x_j^i\right\rangle = \left|y_j^i\right\rangle, i \in \overline{1,k}, j \in \overline{1,t}, x_j^i \neq y_j^i\}.$$
Lists $data_i$ can be simply written according to the cycles of $s_i, i \in \overline{1,k}$, by restoring the transition table of s_i. Description of formation of lists $data_i$ through two-level matrixes is given in order to define $data_i$ strictly and unambiguously.
4. For each pair $(x_j^i, y_j^i) \in data_i$, $i \in \overline{1,k}, j \in \overline{1,t}$ define the list of bits numbers $numb_{(x_j^i, y_j^i)} \subset \{1, 2, \dots n\}$, where x_j^i is different from y_j^i.

Optimization of π_1 quantum circuit

5. Denote $data = \bigcup_{i=1}^{k} data_i$. It is required to sort the elements of $data$ in such a way that

$$\sum_{w=1}^{|data|-1} |numb_{(x_w,y_w)} \cap numb_{(x_{w+1},y_{w+1})}| \to max,$$

moreover, the transitions (x_j^i, y_j^i) obtained through the cycle s_i must preserve the relative order (otherwise, instead of the cycle s_i we will get some other cycle s'). This stage can be implemented using random search with restrictions. As a result of this sorting of $data$ we get $data'$.

6. To each element $(x', y') \in data'$ still correspond some two-level matrix, i.e. each element of $data'$ could be easily implemented using some simple quantum circuit (see [4], [5]). The implementation of two-level matrices in the form of quantum circuit can be ambiguous. Let $d_{(x',y')}$ be the Hamming distance between the binary representations x' and y', then there exists exactly $d_{(x',y')}!$ various quantum circuits consisting from quantum gates CNOT and generalized CNOT(C|t), that implement the transition (x', y') (see [5]). Among them there are only $d_{(x',y')}$ quantum circuits that differ significantly, which are determined by the number t of the controlled qubit in quantum gates CNOT(C|t) that occurs during the implementation of two-level matrices.

Optimization of π_1 quantum circuit

7. We assume that the quantum circuit that implements the transitions of $data'$ could be optimized by independent parts. Then the search for an optimized quantum circuit for $s \in S(V_n)$ can be organized using the following procedure:
   1. Initialize an array of $|data'|$ elements $memory = \{1, \ldots, 1\}$. In $memory[iter]$ we will write the number of implementation of the two-level matrix that implements the transition $(x'_{iter}, y'_{iter}) \in data'$, $iter \in \overline{1, |data'|}$. Set $iter = 1$.
   2. Set the search depth, for example, $depth = 3$. Until $iter < |data'|$ do:
      1. Search for a quantum circuit with minimum length that implements transitions $(x'_{iter}, y'_{iter}), \ldots, (x'_{iter+depth-1}, y'_{iter+depth-1})$ by iterating over all possible variants of quantum circuits that implements $(x'_{iter}, y'_{iter}), \ldots, (x'_{iter+depth-1}, y'_{iter+depth-1})$. Write the founded numbers of implementations two-level matrices to $memory[iter], \ldots, memory[iter + depth - 1]$.
      2. $iter = iter + depth$; The depth of the search is determined by the available computing power.
   3. Repeat this procedure starting at $iter = 2$ and the same value of $depth$.
8. As a result, we obtain a quantum circuit that implements $s \in S(V_n)$ without using ancilla qubits, with the minimal number of quantum gates.

The result of applying the algorithm to the GOST R 34.12-2015 "Magma"

The table 3 presents the results of our realization of algorithm.

S-box	data' – sequences of elementary qubit states transformations
π_0	(14,15)(13,14)(11,14)(10,13)(9,14)(8,14)(7,9)(6,11)(12,0)(4,10)(3,6)(2,6)(1,4)
π_1	(10,14)(11,12)(9,14)(7,12)(1,8)(6,10)(5,10)(4,9)(6,0)
π_2	(1,3)(9,1)(13,9)(5,15)(7,13)(10,7)(6,10)(14,6)(2,5)(4,2)(8,14)(11,4)(0,11)
π_3	(14,15)(13,14)(12,15)(11,13)(9,12)(8,15)(7,15)(6,15)(5,13)(4,13)(3,8)(1,8)(12,0)
π_4	(5,1)(0,7)(2,5)(8,0)(14,2)(4,8)(13,4)(10,3)(11,14)(12,11)(15,12)
π_5	(12,13)(11,13)(10,11)(9,10)(8,11)(7,10)(5,15)(6,12)(4,9)(2,15)(3,6)(5,0)(1,13)
π_6	(13,15)(9,14)(12,13)(6,14)(5,9)(11,15)(10,11)(4,6)(3,5)(8,15)(1,14)(7,12)(8,0)
π_7	(7,3)(1,7)(0,1)(4,0)(8,4)(6,8)(11,6)(14,11)(2,14)(15,2)(9,15)(12,9)(13,12)

Table 3: Data from algorithm 1 for implementing S-boxes without ancilla qubits.

Before optimization:

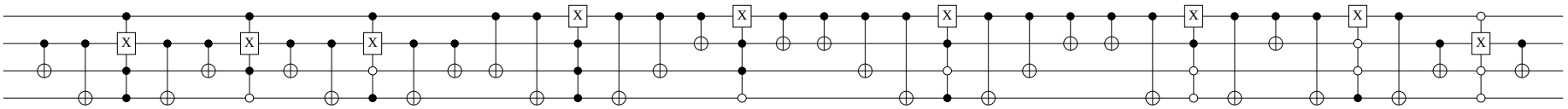


Figure 7: Quantum circuit for $\pi_1 = (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15)$.

After optimization:

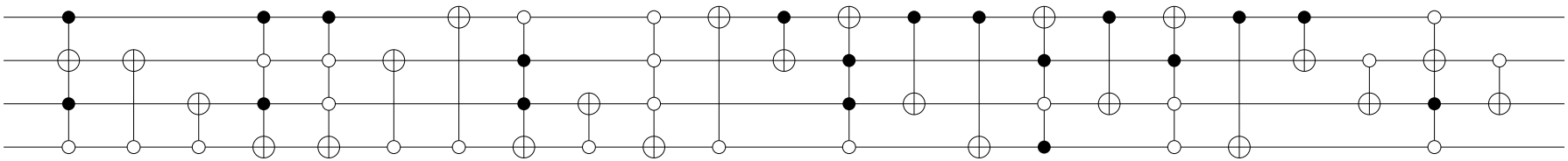


Figure 8: Quantum circuit for $\pi_1 = (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15)$.

The more cycles, the shorter QC that implements the S-box may be!

Table 4: A comparison of the number of quantum gates in quantum circuits that implement the S-boxes of GOST R 34.12-2015 «Magma»

S-box	Number of cycles in S-box	Total gates in old QC	Total gates in new QC
π_0	2	33	29
π_1	3	29	23
π_2	2	37	27
π_3	1	29	29
π_4	3	31	23
π_5	2	35	29
π_6	2	31	25
π_7	1	31	29

Estimations from *Quantum Computing: Progress and Prospects*

Cryptosystem	Key size, bit	Security, bit	Logical Qubits Required	Physical Qubits Required	Time Required to Break System
AES	128	128	2953	4.61×10^6	2.61×10^{12} years
	192	192	4449	1.68×10^7	1.97×10^{22} years
	256	256	6681	3.36×10^7	2.29×10^{32} years
RSA	1024	80	2290	2.56×10^6	3.58 hours
	2048	112	4338	6.2×10^6	28.63 hours
	4096	128	8434	1.47×10^7	229 hours
ECDLP, (NIST P-256 NIST P-386 NIST P-521)	256	128	2330	3.21×10^6	10.5 hours
	386	192	3484	5.01×10^6	37.67 hours
	512	256	4719	7.81×10^6	95 hours
SHA256	N/A	72	2403	2.23×10^6	1.8×10^4 years

Table 5: See [10], table 4.1.

Company	Type	Technology	Now	Next Goal
D-Wave	Annealing	Superconducting	2048	5000
Fujitsu	Digital Annealer	Classical	1024	8192
Google	Gate	Superconducting	72	TBD
IBM	Gate	Superconducting	50	TBD
Intel	Gate	Superconducting	49	TBD
Univ. of Wisconsin	Gate	Neutral Atoms	49	TBD
Intel	Gate	Spin	26	TBD
IQOQI	Gate	Ion Trap	20	TBD
Rigetti	Gate	Superconducting	19	128
IonQ	Gate	Ion Trap	11	79
USTC (China)	Gate	Superconducting	10	20
NTT/Japan NII	Qtm Neural Network	Photonic	2048	>20,000
Univ. of Maryland / NIST	Quantum Simulator	Ion Trap	53	TBD
Harvard/MIT	Quantum Simulator	Rydberg Atoms	51	TBD
Huawei – HiQ Cloud	Software Simulator	Classical	42-169	N/A
Alibaba/Univ. of Michigan	Software Simulator	Classical	144	N/A
USTC/Origin QC	Software Simulator	Classical	64	N/A
University of Melbourne	Software Simulator	Classical	60	N/A
IBM Research	Software Simulator	Classical	56	N/A

Table 6: Quantum processors and simulators, updated 03.05.2019, part 1.

Company	Type	Technology	Now	Next Goal
ETH Zurich	Software Simulator	Classical	45	N/A
Intel–qHiPSTER	Software Simulator	Classical	43	N/A
Atos	Software Simulator	Classical	41	N/A
Microsoft–Azure	Software Simulator	Classical	40	N/A
Rigetti–Forest	Software Simulator	Classical	36	N/A
Microsoft–PC	Software Simulator	Classical	30	N/A
iARPA QEO	Annealing	Superconducting	N/A	100
NSF STAQ Pro	Gate	Ion Trap	N/A	>64
Silicon QC	Gate	Spin	N/A	10
CEA-Leti/INAC/	Gate	Spin	N/A	100

Table 7: Quantum processors and simulators, updated 03.05.2019, part 2.

Computer	1-Qubit Gate Fidelity	2-Qubit Gate Fidelity	Read Out Fidelity
IBM Q5 Tenerife	99.84%	95.98%	94.46%
IBM Q16 Melbourne	99.68%	92.84%	93.02%
IBM Q20 Poughkeepsie	99.89%	97.75%	TBD
IBM Q20 Tokyo	99.80%	97.16%	91.72%
IBM Q System One	99.96%	98.31%	TBD
Rigetti 16Q Aspen-1	97%	91%	93%
Rigetti 8Q Agave	96.15%	87.00%	83.84%
Rigetti 19Q Acorn	98.63%	87.50%	93.30%
IonQ 11 Qubit	>99%	>98%	99.80%

Table 8: Fidelity of quantum operations.

Thank you for attention!

[1] Almazrooie M., Azman S., Rosni A., Mutter K. *Quantum exhaustive key search with simplified-DES as a case study.* SpringerPlus 2016 5:1494. DOI:10.1186/s40064-016-3159-4., 2016.

[2] Денисенко Д.В., Никитенкова М.В. *Применение квантового алгоритма Гровера в задаче поиска ключа блочного шифра SDES*. ЖЭТФ, том 155, вып. 1, 2019. DOI:10.1134/S0044451019010036.

[3] Денисенко Д.В., Маршалко Г.Б., Никитенкова М.В., Рудской В.И., Шишкин В.А. *Оценка сложности реализации алгоритма Гровера для перебора ключей алгоритмов блочного шифрования ГОСТ Р 34.12-2015*. ЖЭТФ, том 155, вып. 4, 2019, DOI:10.1134/S0044451019040072.

[4] Денисенко Д.В. *О реализации подстановок в виде квантовых схем без использования дополнительных кубитов*, ЖЭТФ, том 155, вып. 6, DOI:10.1134/S004445101906004X, 2019.

[5] Nielsen M.A., Chuang I.L. Quantum computation and quantum information, Cambridge Univ. Press, http://csis.pace.edu/ctappert/cs837-18spring/QC-textbook.pdf, 2010

[6] Grassl M., Langenberg B., Roetteler M., Steinwandt R. *Applying Grover's algorithm to AES: quantum resource estimates.* arxiv.org/abs/1512.04965, 2015.

[7] R. Raussendorf, D.E. Browne, H.J. Briegel *Measurement-based quantum computation with cluster states.* DOI:10.1103/PhysRevA.68.022312, https://arxiv.org/abs/quant-ph/0301052, 2003.

[8] Kaye P. *Reversible addition circuit using one ancillary bit with application to quantum computing*, https://arxiv.org/abs/quant-ph/0408173v2, 2004.

[9] Thomas G. Draper. *Addition on a Quantum Computer*. Quantum Physics (quant-ph), https://arxiv.org/abs/quant-ph/0008033.

[10] *Quantum Computing: Progress and Prospects*. National Academies of Sciences, Engineering, and Medicine. 2018. The National Academies Press, Washington DC, https://doi.org/10.17226/25196.