The quantum circuit construction for S-boxes without ancilla qubits: A more detail analysis

DOI: https://doi.org/10.54654/isj.v1i21.1031

Nguyen Van Long, Hoang Dinh Linh, Le Quoc Dat

***Abstract*— This article presents a comprehensive analysis of Denisenko's method for constructing quantum circuits for S-boxes without using ancilla qubits. We elaborate on this methodology in terms of practical aspects that can be applied to S-boxes or permutations of arbitrary sizes. In addition, we created an automatic tool in C++ language that allows to construction of quantum circuits for any 8×8 S-box without ancilla qubits. Furthermore, a quantum circuit for an 8-bit S-box of the MKV block cipher standard in the civilian sector of Vietnam is also provided using this tool.**

***Tóm tắt* — Bài báo trình bày về một phân tích chi tiết phương pháp xây dựng mạch lượng tử cho S-hộp không dùng qubit bổ sung của Denisenko. Nhóm tác giả đã cụ thể hóa phương pháp theo khía cạnh thực hành để có thể áp dụng cho S-hộp hoặc hoán vị kích thước bất kỳ. Ngoài ra một công cụ tự động trên ngôn ngữ C++ được xây dựng cho phép xây dựng mạch lượng tử cho S-hộp 8 bit bất kỳ mà không sử dụng qubit bổ sung. Hơn nữa, mạch lượng tử cho S-hộp 8 bit của mã khối MKV là mã khối sử dụng trong lĩnh vực dân sự của Việt Nam cũng được phân tích sử dụng cộng cụ này.**

***Key words*— quantum circuit, S-box, quantum resources, ancilla qubits, MKV block cipher**.

***Từ khóa*— mạch lượng tử, S-hộp, tài nguyên lượng tử, qubit bổ sung, mã khối MKV**.

I. INTRODUCTION

As we know, to carry out quantum computation-based attacks, the first problem that needs to be addressed is constructing quantum circuits that execute symmetric cryptographic algorithms. The approach to building a quantum circuit for an entire cryptographic algorithm is through the quantum circuit of its component transformations. This paper gives a comprehensive view on the solution for constructing quantum circuits for the nonlinear S-box component in block ciphers. S-boxes are nonlinear, thus their quantum implementation typically requires more resources, such as the number of qubits, compared to other linear components in a symmetric cryptographic algorithm.

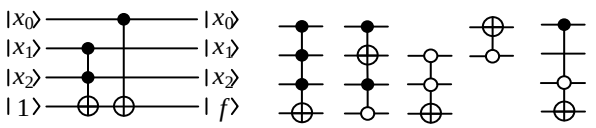
There are some public works relative to problem of quantum circuit construction for S-boxes. This construction generally depends on algebraic structure of each S-box. For example, S-box that is based on inverse mapping is used in many cryptographic algorithms such as AES [1], Camellia [2], Whirlpool [3], or on other algebraic structure [21], ... There are many research works in which the mothod of building the quantum circuit for this S-box are propsed and the method is continuously improved. The first implementation in 2016 by Grassl et al. in [4] required 40 qubits. In 2020, Langenberg et al. built the circuit quantum and the required number of qubit is 32 [5]. Then, the circuit of Zou et al.'s implementation in [6] requires 22 qubits. This number of qubit can be find in the implementation method of Sun's work [7] or Li's work [8]. In 2023, Zhenqiang Li et al.'s presented this type of S-box on the composite field GF($(2^4)^2$) instead of on GF(2^8). After that, they used LIGHTER-R tool [10] that allows to construct a quantum circuit for 4-bit permutation to get quantum circuit for original S-box. As a result, this approach requires only 16 qubit. This number of qubits is the best published for S-hop based on inverse mapping. However, all above method have to ancilla qubits.

This manuscript is received on May 04, 2024. It is commented on May 13, 2024 and is accepted on June 6, 2024 by the first reviewer. It is commented on June 27, 2024 and is accepted on June 29, 2024 by the second reviewer.

For the problem of constructing quantum circuits for S-boxes, many research works have been published. This construction generally heavily depends on the algebraic structure of each S-box. For instance, the 8-bit S-boxes based on the inverse mapping are utilized in various cryptographic algorithms such as AES [1], Camellia [2], Whirlpool [3], ... There are numerous research efforts on constructing quantum circuits for these S-boxes, and these methods are continuously improved. The initial implementation in 2016 by Grassl et al. [4] required 40 qubits. In 2020, Langenberg et al. [5] implemented a quantum circuit requiring only 32 qubits. Zou et al. [6] achieved an implementation with only 22 qubits, and similar results can also be found in the methods by Sun [7] or Li [8]. In 2023, Zhengqiang Li et al. [9] based their design of the S-box on the composite field $GF(2^4)^2$ instead of $GF(2^8)$, and subsequently used the LIGHTER-R toolkit [10] to construct quantum circuits for 4-bit permutations. Their method achieved the best-known quantum qubit count of 16 qubits for an S-box based on inverse mapping. However, these implementations all involve the use of ancilla qubits.

Another approach for constructing quantum circuits for S-boxes is to use the LIGHTER-R toolkit [10], or the improved version, DORCIS [11]. These toolkits allow the construction of quantum circuits without using ancilla qubits, but their limitation is that they are only applicable to S-boxes of size 4 bits or smaller. When the size of the S-box increases, for instance, the common 8-bit S-boxes, these toolkits cannot be applied.

In [12], the authors use the algebraic normal form (ANF) of the component Boolean functions of the S-box to construct a quantum circuit and estimate the corresponding quantum resource. Accordingly, a boolean function f with m bits can be implemented using $m + 1$ qubits and N_f generalized $CNOT(C \mid t)$ gates, where qubit target t is controlled by a set of qubits C such that $t \notin C$, and N_f is a number of monomials in the ANF representation of Boolean function f. Fig. 1-*a*) is a quantum circuit example of a Boolean function $f(x_0, x_1, x_2) = 1 \oplus x_0 \oplus x_1 x_2$ using generalized $CNOT(C \mid t)$ gates.



a) *b)*

Figure 1. a) Quantum circuit of f with CNOT(C | t) gates. b) Example of some generalized CNOT(C | t) gates

According to this approach, implementation of an $m \times m$ S-box would require $2m$ qubits and $\sum_{i=0}^{m-1} N_{f_i}$ $CNOT(C \mid t)$ gates. It can be seen that the number of these gates does not exceed the total number of monomial terms in the ANF representation of the Boolean functions of the considered S-box.

The disadvantage of this approach consists of using the generalized $CNOT(C|t)$ gates with many control qubits. As we know, S-boxes used in cryptographic algorithms often have high algebraic degrees. For example, an 8-bit S-box in MKV cipher or other ciphers has an algebraic degree 7 which means that in the ANF representation, its boolean functions have at least one monomial of degree 7. Thus, it requires at least one generalized $CNOT(C|t)$ gate with 7 control qubits. According to results in ([13], Fig. 4.9 – page 182), the quantum circuit of the Toffoli gate (generalized CNOT gate with 2 control qubits) has 16 basic quantum gates (Hadamard, phase, controlled-Not and $\pi/8$ gates). Thus, the question is what is the depth of the quantum circuit and how many basic quantum gates are there in this circuit? This is an open problem, currently, we have not found public works that allow to efficiently decompose generalized CNOT gates with many control qubits, even with only 3 control qubits. Therefore, the estimates from the above approach are, to our knowledge, theoretically meaningful. On the other hand, the quantum circuit of each Boolean function in this method requires one ancilla qubit

Note. In the notion of generalized $CNOT(C \mid t)$ gates and generalized controlled-$\tilde{U}$ gates [14, 15], position of targer bit t is not important, and

controlled qubits in the set *C* may be $|0\rangle$ or $|1\rangle$. For example, these some gates are all generalized *CNOT*(*C* | *t*) (see Fig.1-*b*).

In 2019, Denisenko, in his research [15], proposed a solution that allows the construction of quantum circuits for S-boxes of any size without the need for ancilla qubits. That is, for an S-box of *m* bits, only *m* qubits are required. However, Denisenko's published results are quite brief and need further elaboration both in terms of the mathematical foundation and algorithmic analysis.

Our Contribution. We provide a detailed analysis of the theoretical basis and the specific implementation steps of the solution for constructing quantum circuits for S-boxes without using ancilla qubits, as described in [15]. Specifically, we will examine the relationship between Gray codes, 2x2 unitary matrices, and the corresponding quantum circuits. Additionally, we will review and discuss some of Denisenko's improvements for optimizing quantum circuits, which were not mentioned in his publication. Notably, we have also developed an automated tool that enables the construction of quantum circuits for any S-box by decomposing its unitary matrix into a product of 2x2 unitary matrices. In this process, we propose a solution for constructing Gray codes and a method for optimizing these codes to enhance implementation efficiency.

The rest of the article is as organized as follows. In Section II, we present the definitions, notations and brief descriptions of S-box in MKV cipher. Next, in Section III, we describe how we can construct a reversible quantum implementation of arbitrary S-boxes without an ancilla qubit. The relationship between the Gray code, two-level unitary matrices and the corresponding quantum circuit will be considered in detail in this Section. In the end of Section III, we discussed on some reduction principles resources of quantum circuit. Finally, we conclude in Section IV.

II. PRELIMINARIES

Before going into the technicalities of methods for constructing quantum circuits that implement S-boxes without ancilla qubits, we present some notations, and definitions and briefly describe the S-box of MKV block cipher [16]. Let S denote the $m \times m$ bit S-boxes that take a *m*-bit input value and return a *m*-bit output value. Let the matrix U corresponding to S-box. However, U is a $2^m \times 2^m$ permutation matrix, then it is a unitary matrix. Let $V_i, i = 1,2,3, \ldots$ be a set of two-level unitary matrices that is obtained by the decomposition of U which we will show in Sect. III. In the article, the concept of level 2 unita matrix is used with the following definition:

Definition 1. *Two-level unitary matrix is a unitary matrix that acts non-trivially only on two vector components.*

Suppose $V = \left(v_{s,t}\right)_{d\times d}$ is a two-level unitary matrix. Then V has the same matrix elements as those of the unit matrix except for certain four elements $v_{s,s}, v_{s,t}, v_{t,s}$ and $v_{t,t}$, where $0 \le s, t \le d-1$. An example of a two-level unitary matrix is

$$V = \begin{pmatrix} \mathbf{0} & 0 & 0 & \mathbf{1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathbf{1} & 0 & 0 & \mathbf{0} \end{pmatrix},$$

where $i = 0$, $j = 3$ (4 bold values in V).

Our immediate goal is to construct a circuit performing a two-level unitary matrix, built from generalized CNOT gates and generalized controlled-$\tilde{U}$ gates. Then we need to make use of the Gray code denoted by *G*. Suppose we have distinct binary numbers, *s* and *t*. Next, the definition of *G* is presented.

Definition 2. *A Gray code* $G = \{g_i\}$ *connecting s and t is a sequence of binary numbers, starting with s and concluding with t, such that adjacent members of the list differ in exactly one bit.*

The block cipher MKV proposed by the Vietnam Government Information Security Committee is used in the civil area of Vietnam [16]. The MKV is an iterated block cipher and specifies four main transformations AddRoundKey, SubCells, MixWords and XWords. The SubCells transformation is

defined as message partitioning into bytes followed by non-linear bijective mapping of each byte using 8×8 bit S-box. This S-box based on butterfly structure [17-19] is used in the round function and key schedule scheme of MKV cipher.

III. The method for implementing S-boxes without ancilla qubits

The method proposed by Denisenko in [15] allows construction quantum circuit of any $m \times m$ bit S-boxes. It consists of the following 4 steps:

1. Find a unitary matrix $U_S \in \mathbb{C}_{2^m,2^m}$ corresponding to the chosen $m \times m$ bit S-box. U_S is a $2^m \times 2^m$ binary matrix.

2. Find the set of two-level matrices $V_0, V_1, \cdots, V_{k-1}$ of U_S such that:

$$U_S = V_0 \times V_1 \times \cdots \times V_{k-1}$$

3. Construct a quantum circuit for each two-level matrix $V_i, 0 \le i \le k-1$.

4. Combining the quantum circuits received in step 3 to obtain a quantum circuit performing unitary matrix U_S. This is the quantum circuit for the given S-box.

However, in [15] the authors do not analyze the implementation steps in detail. Before delving into a more detailed analysis of step 3, we will review step 2 according to the previously published work (see [13], section 4.5.1, page 189).

A. Unitary matrix decomposition into a product of two-level matrices

Consider a unitary matrix *U* which acts on an *n*-dimensional Hilbert space. The essential idea behind this decomposition may be understood by considering the case when *U* is 3×3 (see [13], Sect. 4.5.1, page 189), so suppose that *U* has the form:

$$U = \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & j \end{pmatrix} \quad (1)$$

We will find two-level unitary matrices V_1, V_2, V_3 such that:

$$V_3 V_2 V_1 U = I \quad (2)$$

From (2) we have:

$$U = V_1^\dagger V_2^\dagger V_3^\dagger \quad (3)$$

where $V_i^\dagger = (V_i^T)^*$, for example $\begin{pmatrix} v_0 & v_3 & v_6 \\ v_1 & v_4 & v_7 \\ v_2 & v_5 & v_8 \end{pmatrix}^\dagger = \begin{pmatrix} v_0^* & v_3^* & v_6^* \\ v_1^* & v_4^* & v_7^* \\ v_2^* & v_5^* & v_8^* \end{pmatrix}$, V_1, V_2, V_3 are all two-level unitary matrices, and it is easy to see that their inverses $V_1^\dagger, V_2^\dagger, V_3^\dagger$ are also two-level matrices. The notation v^* is that if $v = x + iy$, then $v^* = x - iy$, where x and y is real numbers. Now we present the approach to obtain these V_1, V_2, V_3.

If $b = 0$ then set:

$$V_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (4)$$

If $b \neq 0$ then set

$$V_1 = \begin{pmatrix} \frac{a^*}{\sqrt{|a|^2+|b|^2}} & \frac{b^*}{\sqrt{|a|^2+|b|^2}} & 0 \\ \frac{b}{\sqrt{|a|^2+|b|^2}} & \frac{-a}{\sqrt{|a|^2+|b|^2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5)$$

Not that in either case of b, V_1 is a two-level unitary matrix. Thus, we calculate a product $V_1 U$ as follows:

$$V_1 U = \begin{pmatrix} a' & d' & g' \\ 0 & e' & h' \\ c' & f' & j' \end{pmatrix} \quad (6)$$

From equation (6) we see that the key point is that the middle entry in the left column is zero. Let "$'$" denote the other entries in the matrix (6). Their actual values do not matter. Now apply a similar procedure to find a two-level unitary matrix V_2 of the matrix in (6) such that $V_2 V_1 U$ has no entry in the bottom left corner. That is, if $c' = 0$ then set

$$V_2 = \begin{pmatrix} a'^* & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (7)$$

while if $c' \neq 0$ the set

$$V_2 = \begin{pmatrix} \frac{a'^*}{\sqrt{|a\prime|^2+|c\prime|^2}} & 0 & \frac{c'^*}{\sqrt{|a\prime|^2+|c\prime|^2}} \\ 0 & 1 & 0 \\ \frac{c\prime}{\sqrt{|a\prime|^2+|c\prime|^2}} & 0 & \frac{-a\prime}{\sqrt{|a\prime|^2+|c\prime|^2}} \end{pmatrix} \quad (8)$$

After that we find

$$V_2 V_1 U = \begin{pmatrix} 1 & d'' & g'' \\ 0 & e'' & h'' \\ 0 & f'' & j'' \end{pmatrix} \quad (9)$$

Since U, V_1 and V_2 are unitary, it follows that V_2V_1U is also unitary, and thus $d'' = g'' = 0$, since the first row V_2V_1U must have norm 1. Finally, set:

$$V_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & e''^* & h''^* \\ 0 & f''^* & j''^* \end{pmatrix} \quad (10)$$

From that we can verify that $V_3V_2V_1U = I$ and obtain $U = V_1^\dagger V_2^\dagger V_3^\dagger$.

In the general case suppose U is a $d \times d$ matrix. Then, in a similar above procedure to the 3×3 case, we can find two-level unitary matrices $V_1, V_2, \dots, V_{d-1}$ such that the matrix $V_{d-1}V_{d-2} \cdots V_2V_1U$ has a one in the top left-hand corner, and all zeroes elsewhere in the corresponding row and column. We then repeat this procedure for the $(d-1) \times (d-1)$ unitary submatrix that is obtained by deleting the first row and first column of the corresponding product matrix. Finally, we have:

$$U = V_1 V_2 \cdots V_k, \quad (11)$$

where the matrices $V_1, V_2, \dots, V_k$ are two-level unitary matrices, and $k \leq (d-1) + (d-2) + \cdots + 1 = \frac{d(d-1)}{2}$.

In [15] the author presents a similar method for any unitary matrix decomposition into a product of two-level matrices. A tool for this procedure in the Matlab language is proposed on the site:

http://cklixx.people.wm.edu/mathlib.html.

Thus, any unitary matrix on an d-dimensional Hilbert space may be written as a product of two-level unitary matrices.

B. Quantum circuit for unitary matrix

In this section, we show that the generalized CNOT gates and generalized controlled-$\widetilde{U}$ together can be used to implement an arbitrary two-level unitary operation on the state space of m qubits. Combining these circuits, we can implement an arbitrary unitary operation on m qubits, and therefore are generalized for quantum computation.

Let U denote a two-level unitary matrix on m qubits. By its definition, U acts non-trivially on the space spanned by the computational basis states $|s\rangle$ and $|t\rangle$, where $s = s_1 \dots s_m$ and $t = t_1 \dots t_m$ are the binary representations for s and t. Let $\widetilde{U}$ denote a non-trivial 2×2 submatrix of U. $\widetilde{U}$ can be thought of as a unitary operator on a single qubit.

Our main task is to create a quantum circuit implementing U based on a single qubit and generalized CNOT gates. In this method, we need to make use of Gray codes. Suppose we have distinct binary numbers, s and t. A Gray code $G = \{g_i\}$ connecting s and t is presented in definition 2. For example, $s = 101001, t = 110011$, we have the Gray code

$s =$ 1 0 1 0 0 1

1 0 1 0 1 1

1 0 0 0 1 1

$t =$ 1 1 0 0 1 1

Let $g_1, g_2, \dots, g_w$ denote the elements of Gray code connecting s and t, with $g_1 = s$ and $g_w = t$. Note that we can always find a Gray code such that $w \leq m + 1$ since s and t can differ at most m locations.

The main idea of the quantum circuit implementing U is as follows:

- (step 1) Perform a sequence of gates affecting the state changes $|g_1\rangle \rightarrow |g_2\rangle \rightarrow \cdots |g_{w-1}\rangle$;

- (step 2) Perform a controlled-$\widetilde{U}$ operation with the target qubit located at the single bit where g_{w-1} and g_w differ;

- (step 3) Perform a sequence of gates affecting the state changes $|g_{w-1}\rangle \rightarrow |g_{w-2}\rangle \rightarrow \cdots |g_1\rangle$, that is inverse transformation of operation in step 1.

Each of these steps can be easily implemented using simple operations, and the final result is an implementation of *U*. Now we will describe them in more detail. The first step is to swap the states $|g_1\rangle$ and $|g_2\rangle$. Suppose g_1 and g_2 differ at the *i*-th digit. Then we accomplish the swap by performing a controlled bit flip on the *i*th qubit, conditional on the values of the other qubits being identical to those in both g_1 and g_2. Next, we use a controlled operation to swap $|g_2\rangle$ and $|g_3\rangle$. We continue in this fashion until we swap $|g_{w-2}\rangle$ with $|g_{w-1}\rangle$. The effect of this sequence of *w* - 2 operations is to achieve the operation such that it tranforms $|g_i\rangle \rightarrow |g_{i-1}\rangle$ for $i = 2, \dots, w-1$ and $|g_1\rangle$ to $|g_{w-1}\rangle$. (Here *w* - 2 operations are equivalent to *w* – 2 gates):

$$|g_1\rangle \rightarrow |g_{w-1}\rangle$$
$$|g_2\rangle \rightarrow |g_1\rangle$$
$$|g_3\rangle \rightarrow |g_2\rangle$$

…

$$|g_{w-1}\rangle \rightarrow |g_{w-2}\rangle$$

Suppose $|g_{w-1}\rangle$ and $|g_w\rangle$ differ in the *j*-th bit. The next step is to perform a controlled-$\tilde{U}$ operation with the *j*-th qubit as target.

Finally, we complete the *U* operation by undoing the swap operations: $|g_{w-1}\rangle \rightarrow |g_{w-2}\rangle$, $|g_{w-2}\rangle \rightarrow |g_{w-3}\rangle$, …, $|g_2\rangle \rightarrow |g_1\rangle$.

From the above analysis, we can generalize the relationship between Gray codes and 2-level unitary matrices of any size. This clarifies the theoretical basis of our research compared to what is presented in [14, 15]. For example, there is a 2-level unitary matrix.

$$U = \begin{pmatrix} a & \dots & 0 & \dots & c \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & \dots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ b & \dots & 0 & \dots & d \end{pmatrix},$$

where a, b, c and d are in the positions (s,s), (t,s), (s,t) and (t,t) corresponding. First, we consider effect of *U* under its multiplication by the vector. Then we have,

$$|s\rangle = |g_1\rangle \rightarrow a|s\rangle + b|t\rangle = a|g_1\rangle + b|g_w\rangle$$
$$|t\rangle = |g_w\rangle \rightarrow c|s\rangle + d|t\rangle = c|g_1\rangle + d|g_w\rangle$$

Other states remain unchanged because *U* is a permutation matrix that differs from the identity matrix of the same size in only four positions *a, b, c, d*.

Now we consider effect three above steps on the state $|s\rangle$. That is performing by operation: Step 3 × Step 2 × Step 1:

Step 1: Let $(|g_i\rangle, |g_{i+1}\rangle)$ denote the operation that performs swap $|g_i\rangle$ with $|g_{i+1}\rangle$. The product $(|g_{w-2}\rangle, |g_{w-1}\rangle) \cdots (|g_2\rangle, |g_3\rangle)(|g_2\rangle, |g_1\rangle)$ acts by following:

$$|g_1\rangle \rightarrow |g_{w-1}\rangle$$
$$|g_2\rangle \rightarrow |g_1\rangle$$
$$|g_3\rangle \rightarrow |g_2\rangle$$

…

$$|g_{w-1}\rangle \rightarrow |g_{w-2}\rangle$$

These states create a circle:

$$|g_{w-1}\rangle \leftarrow |g_1\rangle \leftarrow |g_2\rangle \leftarrow \cdots \leftarrow |g_{w-2}\rangle \leftarrow |g_{w-1}\rangle$$

,

while the other states remain unchanged.

Step 2: In this step we have

$$|g_{w-1}\rangle \rightarrow a|g_{w-1}\rangle + b|g_w\rangle$$
$$|g_w\rangle \rightarrow c|g_{w-1}\rangle + d|g_w\rangle$$

and the other states remain unchanged.

Step 3: In contrast to step 1, it will be performed in sequence

$$|g_{w-1}\rangle \rightarrow |g_1\rangle \rightarrow |g_2\rangle \rightarrow \cdots \rightarrow |g_{w-2}\rangle \rightarrow |g_{w-1}\rangle$$

and the other states remain unchanged.

After three steps, we have :

Step 1: $|g_1\rangle \rightarrow |g_{w-1}\rangle$

Step 2: $|g_{w-1}\rangle \rightarrow a|g_{w-1}\rangle + b|g_w\rangle$

Step 3: $a|g_{w-1}\rangle + b|g_w\rangle \rightarrow a|g_1\rangle + b|g_w\rangle$

And as result we have $|s\rangle = |g_1\rangle \rightarrow |g_{w-1}\rangle \rightarrow a|g_1\rangle + b|g_w\rangle$ after performing three steps. Similarly, if the input state is $|g_w\rangle = |t\rangle$ then we get

$$|g_w\rangle \rightarrow |g_w\rangle \rightarrow c|g_{w-1}\rangle + d|g_w\rangle \rightarrow c|g_1\rangle + d|g_w\rangle.$$

The state $|g_i\rangle$, $2 \le i \le w-1$ remains the same after three steps:

$$|g_i\rangle \rightarrow |g_{i-1}\rangle \rightarrow |g_{i-1}\rangle \rightarrow |g_i\rangle.$$

And the other states that is different from $|g_i\rangle$ remain the same too. That is, the abve three steps acts only on two state $|s\rangle$ and $|t\rangle$. In other words, this process perfoms *U*.

For $2 \le i \le w-1$, we have:

$$|g_i\rangle \rightarrow |g_{i-1}\rangle \rightarrow |g_{i-1}\rangle \rightarrow |g_i\rangle.$$

This means it remains unchanged. This indicates that the three steps implemented only affect two states $|s\rangle$ and $|t\rangle$, which corresponds to applying*U*.

For simplicity, we consider the example with 8×8 unitary matrix *U* that is defined as follows:

$$U = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix}, \quad (12)$$

where, $a, b, c, d \in \mathbb{C}$.

Notice that *U* acts non-trivially only on two states $|000\rangle$ and $|111\rangle$, and the square submatrix $\widetilde{U} = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ is two-level unitary. In this example, *m* = 3. According to definition 2, we have Gray code connecting $s = 000$ and $t = 111$:

$$\begin{array}{rc} & q_1 q_2 q_3 \\ g_1 = & 0\ 0\ 0 \\ g_2 = & 1\ 0\ 0 \\ g_3 = & 1\ 1\ 0 \\ g_4 = & 1\ 1\ 1 \end{array}$$

Note that Gray code connecting *s* and *t* is not unique. We can use other Gray code for considered example.

The above process can be illustrated in the following quantum circuit (Fig. 2).

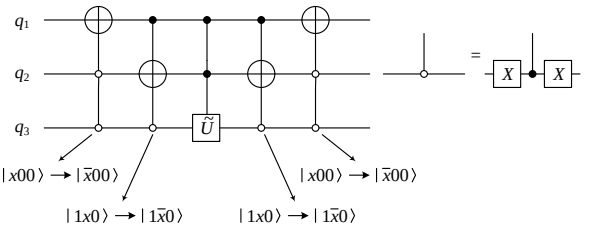


Figure. 2. Quantum circuit for U

Let us consider the effect on the states different from $|s\rangle = |000\rangle$ and $|t\rangle = |111\rangle$, for example, the $|101\rangle$ is not changed under *U* operation, that is trivial operation (Fig. 3). And for $|s\rangle = |000\rangle$, *U* perform it into $|t\rangle = |111\rangle$, that is a non-trivial operation (Fig. 4).

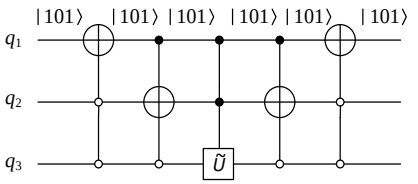


Figure 3. Trivial operation of *U* on $|101\rangle$

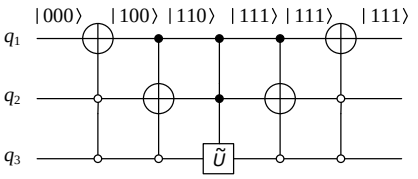


Figure 4. Non-trivial operation of *U* on $|000\rangle$

C. On a Gray code construction method

As in previous analyses, we can observe that for a pair of values *s* and *t* of size *m* bits, there exist (*w* – 1)! Gray codes connecting *s* and *t*, where *w* is the Hamming distance between *s* and *t*. Therefore, finding the shortest Gray code is equivalent to using the fewest quantum gates. In this section, we consider a solution approach. Our approach involves creating an array of pre-computed Gray code sequences, and then depending on the values of *s* and *t*, finding the shortest length Gray code. The procedure for constructing the array containing the possible Gray code sequences is implemented according to pseudocode 1.

Pseudocode 1. Generate an *m*-bit Gray code array.

Input: The value *m* represents the size of elements in the Gray code.

Output: An array containing 2^m elements ò the Gray code.

GenAllGrayCode(*m*):

AllGC ← ∅ // Empty array

AllGC ← {'0', '1'}

For *i* = 2 to 2^n do

$L \leftarrow \emptyset$ // Empty array

For *j* from *i* – 1 down to 0 do

$L \leftarrow L \cup \{AllGC[j]\}$

End for

$AllGC \leftarrow AllGC \cup L$

For *j* from 0 to *i* do

$AllGC[j] \leftarrow' 0' || AllGC[j]$

End for

For *j* from *i* to 2*i* do

$AllGC[j] \leftarrow' 1' || AllGC[j]$

End for

$i \leftarrow 2i$

End for

Return *AllGC*

Pseudo code takes the input as the size of elements in Gray code. The algorithm in the pseudo code generates a sequence of 2^m elements arranged such that any two consecutive elements have a Hamming distance of 1. This is one of the sorting methods for sequence of 256 byte such that two consecutive elements have a Hamming distance of 1. Gray code array AllGC is presented in Table I. In the case of an 8-bit S-box, the array generated from

TABLE I. GRAY CODE ARRAY ALLGC FROM PSEUDOCODE 1

00000000	00110000	01100000	01010000	11000000	11110000	10100000	10010000
00000001	00110001	01100001	01010001	11000001	11110001	10100001	10010001
00000011	00110011	01100011	01010011	11000011	11110011	10100011	10010011
00000010	00110010	01100010	01010010	11000010	11110010	10100010	10010010
00000110	00110110	01100110	01010110	11000110	11110110	10100110	10010110
00000111	00110111	01100111	01010111	11000111	11110111	10100111	10010111
00000101	00110101	01100101	01010101	11000101	11110101	10100101	10010101
00000100	00110100	01100100	01010100	11000100	11110100	10100100	10010100
00001100	00111100	01101100	01011100	11001100	11111100	10101100	10011100
00001101	00111101	01101101	01011101	11001101	11111101	10101101	10011101
00001111	00111111	01101111	01011111	11001111	11111111	10101111	10011111
00001110	00111110	01101110	01011110	11001110	11111110	10101110	10011110
00001010	00111010	01101010	01011010	11001010	11111010	10101010	10011010
00001011	00111011	01101011	01011011	11001011	11111011	10101011	10011011
00001001	00111001	01101001	01011001	11001001	11111001	10101001	10011001
00001000	00111000	01101000	01011000	11001000	11111000	10101000	10011000
00011000	00101000	01111000	01001000	11011000	11101000	10111000	10001000
00011001	00101001	01111001	01001001	11011001	11101001	10111001	10001001
00011011	00101011	01111011	01001011	11011011	11101011	10111011	10001011
00011010	00101010	01111010	01001010	11011010	11101010	10111010	10001010
00011110	00101110	01111110	01001110	11011110	11101110	10111110	10001110
00011111	00101111	01111111	01001111	11011111	11101111	10111111	10001111
00011101	00101101	01111101	01001101	11011101	11101101	10111101	10001101
00011100	00101100	01111100	01001100	11011100	11101100	10111100	10001100
00010100	00100100	01110100	01000100	11010100	11100100	10110100	10000100
00010101	00100101	01110101	01000101	11010101	11100101	10110101	10000101
00010111	00100111	01110111	01000111	11010111	11100111	10110111	10000111
00010110	00100110	01110110	01000110	11010110	11100110	10110110	10000110
00010010	00100010	01110010	01000010	11010010	11100010	10110010	10000010
00010011	00100011	01110011	01000011	11010011	11100011	10110011	10000011
00010001	00100001	01110001	01000001	11010001	11100001	10110001	10000001
00010000	00100000	01110000	01000000	11010000	11100000	10110000	10000000

pseudo code 1 consists of 256 elements as follows:

The value of the table is arranged according to the rule from top to bottom, left to right. For each pair *s*, *t*, there will exist many Gray codes connecting them. For example, with *s* = 00000010 and *t* = 00001011, from the Table I, we obtain:

g_0 = 00000010
g_1 = 00000110
g_2 = 00000111
g_3 = 00000101
g_4 = 00000100
g_5 = 00001100
g_6 = 00001101
g_7 = 00001111
g_8 = 00001110
g_9 = 00001010
g_{10} = 00001011

In this case, the Gray code connecting two values *s* and *t* can be $\{g_0, g_1, g_8, g_9, g_{10}\}$, $\{g_0, g_1, g_2, g_7, g_8, g_9, g_{10}\}$, $\{g_0, g_9, g_{10}\}$, … To minimize the number of quantum gates, we need the short Gray code. For this problem, we propose the following procedure.

Pseudocode 2. Find the optimal Gray code.

Input: Gray code connecting two elements *s* and *t* obtained from the array generated according to Pseudocode 1 – array GC[*N*], number of elements of this Gray code - *N*.

Output: Optimal Gray code and its corresponding number of elements.

OptimizeGrayCode(*N*, GC[*N*]):
$max_s \leftarrow 0, max_t \leftarrow 0$
If $N > 2$ then
  While ($max_s \neq 1$ *or* $max_t \neq 1$) do
    For *i* from 1 to *N* do
      If($wt[GC[i] \oplus GC[N-1]] = 1$) then
  $max_t \leftarrow N - i - 1$
  break
  End if
 End for
 For *i* from *N* – 1 down to 1 do
  If ($wt[GC[i] \oplus GC[0]] = 1$) then
  $max_s \leftarrow i$
  break
  End if
 End for
 If ($max_s \geq max_t$) then
  For *i* from max_s to *N* do
  $GC[i - max_s + 1] \leftarrow GC[i]$
  End for
  $N \leftarrow N - max_s + 1$
 Else
  $GC[N - max_t] \leftarrow GC[N-1]$
  $N \leftarrow N - max_t + 1$
 End if
 If *N* = 2 then
  break
 End if
  End while
  For *i* from 1 to *N* – 2 do
 For *j* from *i* + 2 to *N* do
  If ($wt[GC[i] \oplus GC[j]] = 1$) then
  Copy(*GC* + *i* + 1, *GC* + *j*, *N* – *j*)
  $N \leftarrow N - (j - i) + 1$
  End if
 End for
 End for
End if
Return *N*, *GC*.

We will not present detailed step-by-step operations of the algorithm in Pseudocode 2 but will instead outline its idea. The algorithm takes as input the array of Gray codes generated from Pseudocode 1, along with two values *s* and *t*. From the values of *s* and *t*, a sub-array is extracted from the original GC array. This sub-array represents the Gray code connecting *s* and *t*. However, this sub-array not only contains consecutive pairs with a Hamming distance of 1 but also includes non-consecutive pairs with the same Hamming distance. From here, we can reduce the length to obtain a shorter Gray code.

This approach represents our specific way to address this problem, acknowledging that there may exist more optimal solutions. Nevertheless, with element sizes of 8 bits, the procedure yields results very quickly.

D. Discussion on reduction rules and equivalent quantum circuits

In [15], the authors construct quantum circuits for S-boxes referring to the method presented in [13]. Following the approach in [13], the resulting quantum circuit will consist exclusively of general CNOT(C | t) gates and general controlled-$\widetilde{U}$ gates. The quantum circuit thus obtained for the matrix U in formula (13) will be:

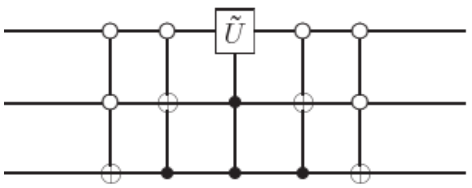


Figure 5. Quantum circuit for the matrix U in formula (12) using general CNOT(C | t) gates and general controlled-$\widetilde{\mathbf{U}}$ gates.

However, when applied to S-boxes, such as the 4-bit example of the Magma algorithm by Denisenko et al., a quantum circuit composed solely of generalized CNOT gates with 1 control qubit and a generalized CNOT(C | t) gate with 01 is obtained. The generalized controlled gate in this case is replaced by the CNOT(C | t) gate because the 2-qubit unitary matrix is permutation matrix. The CNOT gates obtained follow the simplification rule, which we will explain shortly.

In Table 3 of [15], the construction of a quantum circuit without using additional qubits for the 4-bit S-box $\pi_1 = (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15)$ is presented. According to this, its unitary matrix decomposes into a product of 9 2-qubit unitary matrices.

$$U_{\pi_1} = V_1 \cdot V_2 \cdot V_3 \cdot V_4 \cdot \ldots \cdot V_9$$

For example, with the matrix V_3 acting non-trivially on two states $s = |0100\rangle$ và $t = |1001\rangle$. Its corresponding Gray code is as follows, where the authors reverse the values of s and t. Reversing the order does not change the essence of the problem.

	q_1	q_2	q_3	q_4
$g_1 = t =$	1	0	0	1
$g_2 =$	1	1	0	1
$g_3 =$	1	1	0	0
$g_4 = s =$	0	1	0	0

With the Gray code obtained in this manner, following the approach in [13], the resulting quantum circuit is depicted in Figure 6 – a). Meanwhile, in [15], the quantum circuit provided by the authors appears as in Figure 6 – b). This rule is understood as follows: for gates that do not perform transformations between $|g_3\rangle$ and $|g_4\rangle$ or vice versa, they remain unchanged, while other gates only retain the control qubit on the qubit that executes the " $\oplus$ " operation in the generalized CNOT(C | t) gate that transforms $|g_3\rangle$ to $|g_4\rangle$. Consequently, we obtain a quantum circuit consisting solely of 1-qubit CNOT gates and 1 generalized CNOT(C | t) gate. Clearly, this simplified approach is more optimal. However, theoretically, we have not yet been able to prove the equivalence of these two circuits. This remains an open issue following the studies on primitive solutions presented in [13] and Denisenko's solutions in [15]. On another front, through experimentation, we can verify the equivalence of these circuits when considering all possible input states and their corresponding outputs. Similar results are also obtained for the 8 2-qubit unitary matrices in the decomposition of U_{π_1}. We refer to this simplification rule as the Local Reduction Rule within each 2-qubit unitary matrix.

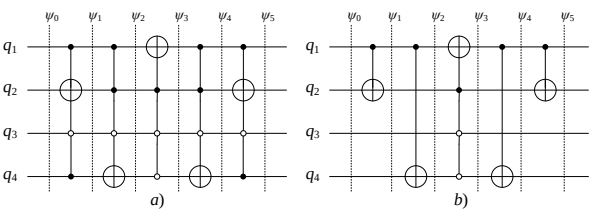


Figure 6. a) Quantum circuit for V_3 following the method in [13], b) Quantum circuit for V_3 according to Denisenko's simplification in [15]

In the Table II and Table III we presents some example that computes output state by quantum circuit in Fig. 6-a) and Fig. 6-b). As reusult, two scheme have same output state by the same given itput state.

TABLE II. EXAMPLE OF SEQENCE STATES CALCULATED BY FIG.6. INPUT STATE IS $|1101\rangle$

By scheme in Fig. 6-a)						
	Ψ_0	Ψ_1	Ψ_2	Ψ_3	Ψ_4	Ψ_5
q_1	1	1	1	1	1	1
q_2	1	0	0	0	0	1
q_3	0	0	0	0	0	0
q_4	1	1	1	1	1	1
By scheme in Fig. 6-b)						
	Ψ_0	Ψ_1	Ψ_2	Ψ_3	Ψ_4	Ψ_5
q_0	1	1	1	1	1	1
q_1	1	0	0	0	0	1
q_2	0	0	0	0	0	0
q_3	1	1	0	0	1	1

TABLE III. EXAMPLE OF SEQENCE STATES CALCULATED BY FIG.6. INPUT STATE IS $|1001\rangle$

By scheme in Fig. 6-a)						
	Ψ_0	Ψ_1	Ψ_2	Ψ_3	Ψ_4	Ψ_5
q_1	1	1	1	0	0	0
q_2	0	1	1	1	1	1
q_3	0	0	0	0	0	0
q_4	1	1	0	0	0	0
By scheme in Fig. 6-b)						
	Ψ_0	Ψ_1	Ψ_2	Ψ_3	Ψ_4	Ψ_5
q_0	1	1	1	0	0	0
q_1	0	1	1	1	1	1
q_2	0	0	0	0	0	0
q_3	1	1	0	0	0	0

Another simplification rule that Denisenko employs in his research involves synthesizing quantum circuits for each 2-qubit unitary matrix component. For instance, with the matrix V_4 from U_{π_1}, the resulting circuit is as follows:

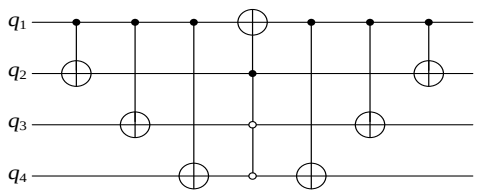


Figure 7. Quantum circuit for matrix $\mathbf{V_4}$

The quantum circuit for $V_3 \cdot V_4$ is then as follows:

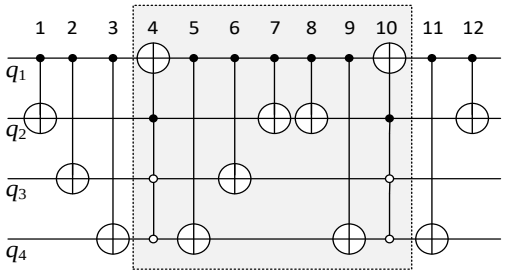


Figure 8. Quantum circuit for matrix $\mathbf{V_3} \cdot \mathbf{V_4}$

Observing Figure 8 with dashed lines, we note that the qubit values on the second wire remain unchanged before gate 7 and after gate 8, allowing gates 7 and 8 to be removed. Similarly, the qubit on the fourth wire retains its value before gate 5 and after gate 9, making gates 5 and 9 removable as well. Consequently, the resulting quantum circuit is:

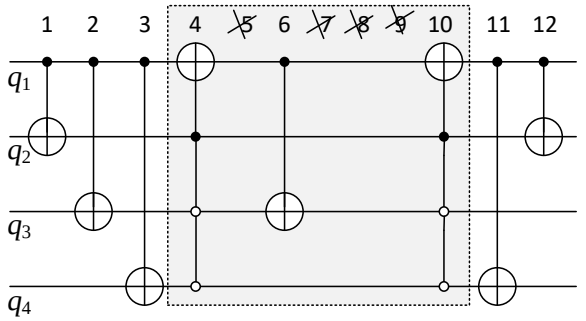


Figure 9. Reduced quantum circuit for $\mathbf{V_3} \cdot \mathbf{V_4}$

Clearly, the circuit in Figure 9 uses fewer quantum gates. We refer to this rule as the Global Reduction Rule, where reduction is achieved by combining 2-qubit unitary matrices together. Gates that can be reduced lie between two generalized CNOT($C \mid t$) gates that perform transformations in consecutive 2-qubit unitary matrices.

When applied to the entire product of the 9 2-qubit unitary matrices of U_{π_1}, we obtain a reduced quantum circuit that implements the 4-bit S-box π_1.

Another issue that we observe from the reduction results of Denisenko and colleagues in [15] pertains to the Global Reduction Rule. Specifically, to achieve reduction according to this rule, the effect of the gate $\widetilde{U}$ across consecutive 2-qubit unitary matrices must act on the same qubit wire. It is crucial to find combinations where sequences in $V_1 \times V_2 \times \cdots \times V_k$ have consecutive occurrences of $\widetilde{U}$ acting on the same qubit wire. As the dimension m increases, such as in the case of 8-bit S-boxes, exhaustively exploring all possible combinations becomes impractical. From this, we also see that satisfying the requirement for $\widetilde{U}$ to act on the same qubit wire across consecutive 2-qubit unitary matrices leads to the realization that not all shortest Gray codes will fulfill this

criterion. This certainly impacts the number of quantum gates required to implement the S-box.

IV. Experimental results

Based on the above results, we create an automatic tool on C++ to construct the quantum circuit for any 8-bit S-box without an ancilla qubit. These are commonly used sizes in many cryptographic primitives. Of course, we can construct the tool with any size of the S-box. The input of the tool is the S-box in the lookup table form, for example, S-box of MKV (see section 2 for a description of this S-box), and the output is a quantum resource and quantum circuit, corresponding. The tool includes following steps:

1. Compute the unitary matrix U of the S-box. U is permutation matrix.

2. Decompose the matrix U into a product of 2x2 unitary binary matrices.

3. For each 2x2 unitary matrix, determine the states on which it acts non-trivially (determine the values s and t). From this, construct the corresponding Gray code.

4. Construct a quantum circuit for each 2x2 unitary matrix.

5. Combine the quantum circuits from step 4 to obtain the quantum circuit for U, which is the circuit of the S-box.

6. Summarize the quantum resources.

We present some experimental results in Table IV below, where "7" and "1" in CNOT(7, 1) gate mean number of controlled qubits and target qubit corresponding. In this context we assume that "Local Reduction Rule" is true.

Table IV. Quantum resource of some 8-bit S-boxes without ancilla qubit

S-box of	Qubit	CNOT(7,1) gates	CNOT gates	Total gates	Refer
AES	8	251	958	1209	[9]
		251	1005	1256	Ours
Kuznyechik	8	250	1020	1270	[9]
		250	1087	1337	Ours
MKV	8	240	930	1170	Ours

As mentioned in the previous section, there are multiple Gray codes available to connect the two values s and t. In our view, the choice of which code to use will affect the implementation resources of the S-box. For the S-boxes of AES and Kuznyechik, we might not have selected the optimal set of Gray codes, resulting in a higher number of quantum gates compared to the results in [14, 15]. However, the main objective of this paper is to provide a detailed analysis of the steps to construct the quantum circuit for the S-box without using ancilla qubits.

IV. Conclusion

The article presents an approach to building quantum circuits that allows implementing S-box implementations of block codes without ancilla qubits. The mathematical aspects of each step of Denisenko's method [15] are explained in more detail. In addition, we created an automatic tool in C++ language that allows to construction of quantum circuits for any 8×8 S-box without ancilla qubits. The output of this tool is quantum resources and quantum circuits. The results of the article allow us to contribute to building quantum circuits for the full block cipher. That is the first step to evaluate the security of cryptographic primitives against attacks based on quantum computing, for example, attacks using Grover's algorithm.

Open Issues. 1) Proving the equivalence of quantum circuits implementing 2-qubit unitary matrices using the local reduction rule. 2) Developing methods to optimize the quantum gate reduction process according to the global reduction rule.

Acknowledgments. We would like to extend my sincere gratitude to Dr. Tran Duy Lai, former Director of the Institute of Cryptographic Science and Technology, for his invaluable insights and constructive feedback. His thoughtful comments have significantly strengthened the quality and clarity of the manuscript in proving the theoretical foundations.

REFERENCES

[1] Daemen, J. and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*. 2002: Springer.

[2] Aoki, K., et al. *Camellia: A 128-bit block cipher suitable for multiple platforms—design andanalysis*. in *Selected Areas in Cryptography*. 2001. Springer.

[3] Barreto, P. and V. Rijmen. *The Whirlpool hashing function*. in First open NESSIE Workshop, Leuven, Belgium. 2000.

[4] Grassl, M., et al. *Applying Grover's algorithm to AES: quantum resource estimates*. in Post-Quantum Cryptography. 2016. Springer.

[5] Langenberg, B., et al.. Steinwandt, *Reducing the cost of implementing the advanced encryption standard as a quantum circuit*. 2020. 1: p. 1-12.

[6] Zou, J., et al. *Quantum circuit implementations of AES with fewer qubits*. in Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II 26. 2020. Springer.

[7] Huang, Z. and S. Sun. *Synthesizing quantum circuits of AES with lower t-depth and less qubits*. in International Conference on the Theory and Application of Cryptology and Information Security. 2022. Springer.

[8] Li, Z., et al., *Novel quantum circuit implementation of Advanced Encryption Standard with low costs*. 2022. 65(9): p. 290311.

[9] Li, Z., et al., *New record in the number of qubits for a quantum implementation of AES*. 2023. 11: p. 1171753.

[10] Dasu, V.A., et al. *LIGHTER-R: optimized reversible circuit implementation for sboxes*. in 2019 32nd IEEE International System-on-Chip Conference (SOCC). 2019. IEEE.

[11] Chun, M. and A. Baksi, *Dorcis: Depth optimized quantum implementation of substitution boxes*. 2023.

[12] Денисенко, Д., et al., *Оценка сложности реализации алгоритма Гровера для перебора ключей алгоритмов блочного шифрования ГОСТ Р 34.12-2015*. 2019. 155(4): p. 645-653.

[13] Nielsen, M.A. and I.L. Chuang, *Quantum computation and quantum information*. 2010: Cambridge university press.

[14] Denisenko, D. and M. Nikitenkova. *Optimization of S-boxes GOST R 34.12-2015 "Magma" quantum circuits without ancilla qubits*. in 8th Workshop on Current Trends in Cryptology, Svetlogorsk, Russia. 2019.

[15] Denisenko, D.V., *Quantum circuits for S-box implementation without ancilla qubits*. 2019. 128: p. 847-855.

[16] Vietnam Government Information Security Commission. https://mod.gov.vn/vn/chi-tiet/sa-ttsk/sa-tt-qpan/lay-y-kien-du-thao-tcvn-cho-thuat-toan-ma-khoi-mkv-trong-linh-vuc-mat-ma-dan-su. 2024.

[17] Avraamova, O.D., et al., *A compact bit-sliced representation of Kuznyechik S-box*. 2021. 12(2): p. 21-38.

[18] Fomin, D.B., *New classes of 8-bit permutations based on a butterfly structure*. 2019. 10(2): p. 169-180.

[19] Nam, T.S., N. Van Long, and N.B. Cuong. *An Optimized Bit-Slice Implementation of Secure 8-Bit Sbox Based on Butterfly Structure*. in 2023 15th International Conference on Knowledge and Systems Engineering (KSE). 2023. IEEE.

[20] Li, C.-K. and R. Roberts, *Decomposition of unitary matrices and quantum gates*. 2013. 11(01): p. 135.

[21] Long, N. V., & Đức, L. D. (2020). Đề xuất S-hộp có tính chất mật mã tốt cho hoán vị của hàm băm Keccak. *Journal of Science and Technology on Information Security*, *1*(11), 32-45. https://doi.org/10.54654/isj.v1i11.93.

ABOUT THE AUTHOR

Nguyen Van Long
Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission
E-mail: nvlong.bcy@gmail.com
Education: He received an Engineer's degree in Information Security of Telecommunication Systems at FSO Academy, Russian Federation in 2008; Ph.D degree at FSO Academy, Russian Federation in 2015.
Recent research direction: Cryptography, Coding theory and Information Security.

Tên tác giả: **Nguyễn Văn Long**

Cơ quan làm việc: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ

Email: nvlong.bcy@gmail.com

Quá trình đào tạo: Nhận bằng Kỹ sư chuyên ngành An toàn thông tin Hệ thống Viễn thông tại Học viện FSO, Liên bang Nga năm 2008; Tiến sĩ tại Học viện FSO, Liên bang Nga năm 2015.

Hướng nghiên cứu hiện nay: Mật mã, lý thuyết mã hóa và bảo mật thông tin.

Hoang Dinh Linh
Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission
Email: hoangdinhlinh@bcy.gov.vn.
Education: In 2014, he graduated from the Advanced Mathematics Program, obtaining a Bachelor's degree in Mathematics at the VNU University of Sciences, Vietnam National University, Hanoi. In 2021, he received a Master's degree in Applied Mathematics at the VNU University of Sciences, Vietnam National University, Hanoi.
Recent research direction: Researching, designing, and evaluating the security of symmetric encryption algorithms. Standards for assessing the randomness of random number generators for cryptography.

Tên tác giả: **Hoàng Đình Linh**

Cơ quan làm việc: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ

Email: hoangdinhlinh@bcy.gov.vn

Quá trình đào tạo: Năm 2014 tốt nghiệp Cử nhân Toán học tại Trường Đại học Khoa học Tự nhiên, Đại học Quốc gia Hà Nội. Năm 2021 nhận bằng Thạc sĩ Toán ứng dụng tại Trường Đại học Khoa học, Đại học Quốc gia Hà Nội.

Hướng nghiên cứu hiện nay: Thiết kế và đánh giá tính bảo mật của các thuật toán mã hóa đối xứng, tiêu chuẩn đánh giá tính ngẫu nhiên của bộ tạo số ngẫu nhiên cho mật mã.

Le Quoc Dat
Workplace: Institute of Cryptographic Science and Technology, Vietnam Government Information Security Commission
E-mail: datkma97@gmail.com
Education: He received a B.S. degree in Cryptographic Information Security of Special Communication Systems at FSB Academy, Russian Federation in 2022.
Recent research direction: Cryptography, Coding theory and Information Security.

Tên tác giả: **Lê Quốc Đạt**

Cơ quan làm việc: Viện Khoa học - Công nghệ mật mã, Ban Cơ yếu Chính phủ

Email: datkma97@gmail.com

Quá trình đào tạo: Nhận được bằng Cử nhân về Bảo mật thông tin mật mã của các hệ thống truyền thông đặc biệt tại Học viện FSB, Liên bang Nga năm 2022.

Hướng nghiên cứu hiện nay: Mật mã học, lý thuyết mã hóa và bảo mật thông tin.