Security of Infocommunication Systems and Internet of Things

2023 Vol 1, No 1

https://doi.org/10.31861/sisiot2023.1.01005

Received 08 May 2023; revised 12 June 2023; accepted 26 June 2023; published 30 June 2023

Ukrainian National Encryption Standards for FPGA Based Embedded Systems

Oleh Krulikovskyi[1,2,3,*], Serhii Haliuk[3], Ihor Safronov[3] and Ivan Gorbenko[4]

[1]Integrated Center for Research, Development and Innovation in Advanced Materials, Nanotechnologies and Distributed Systems for Fabrication and Control, Stefan cel Mare University of Suceava, Suceava, Romania

[2]Faculty of Electrical Engineering and Computer Science, Stefan cel Mare University of Suceava, Suceava, Romania

[3]Department of Radioengineering and Information Security, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine

[4]V. N. Karazin Kharkiv National University, Kharkiv, Ukraine

*Corresponding author (E-mail: o.krulikovskyi@chnu.edu.ua)

ABSTRACT The paper presents the hardware implementation based on FPGA of the main cryptographic transformations of the symmetric transformation algorithm of DSTU 7624:2014 and the stream cipher of DSTU 8845:2019, which are the national encryption standards of Ukraine. In the case of DSTU 7624: 2014 developed and implemented a hardware implementation for multiplication of two polynomials modulo $x^8 + x^4 + x^3 + x^2 + 1$ in the form of a combinational circuit that allows to execute the MixColumn transformation by one cycle. SubBytes transformation is implemented based on asynchronous read-only memory. For stream cipher, DSTU 8845:2019 the nonlinear function T are implemented as subtitution byte operation in the form of precalculated cells of ROM memory. The multiplication function by α and α^{-1} in Galois field arithmetic $GF(2^{64})$ is realized based on ROM and combinational logic. The control of the modes of operation of the shift register with linear feedback is performed based on a FSM. Both hardware implementations of encryption standards have been verified by the authors according to the specified data in the standard, and their HDL code can be provided by the authors for further research to interested parties.

KEYWORDS DSTU 7624:2014; DSTU 8845:2019; FPGA; Embedded Systems.

I. INTRODUCTION

Block symmetric ciphers are a key element in ensuring the confidentiality of information in telecommunication systems. To be effective across a variety of software and hardware platforms, these ciphers must be stable and high-performing, with low resource requirements. This is especially important in embedded systems, which are increasingly used in industrial, consumer, medical, automotive, cyber-physical systems, IoT devices, and more [1,8,9]. However, embedded systems are limited by computing power and energy costs [8-11]. To protect information in these systems, cryptographic algorithms must be effectively implemented within these constraints. With the introduction of new national standards for block symmetric ciphers, DSTU 7624:2014 and stream cipher DSTU 8845:2019 developed under the supervision of Prof. I. Gorbenko and A. Kuznetsov [2-4,7], it is essential to evaluate their implementation on general-purpose microcontroller cores and programmable logic integrated circuits (FPGA). While DSTU 7624:2014 and DSTU 8845:2019 have been studied on general-purpose microcontroller cores [1], their effectiveness on modern FPGA platforms has yet to be explored. By implementing these ciphers on FPGA, it would be possible to utilize them in high-speed data processing, storage systems, and IoT.

The proposed hardware implementation of the symmetric cryptographic algorithms, DSTU 7624:2014 and DSTU 8845:2019, for FPGA-based embedded systems, is aimed at optimizing the performance and resource requirements of these ciphers while ensuring their effective implementation on a wide range of embedded systems. The primary goal of this work is to create FPGA-based hardware implementations of these ciphers for use in IoT devices.

The work is organized as follows: in section I, the introduction, section II describes the main operations of the encryption standards DSTU 7624:2014 and DSTU 8845:2019. Implementation of the main transformations of the DSTU 7624: 2014 and DSTU 8845:2019 for FPGA in section III. Conclusions in section IV.

II. UKRAINIAN NATIONAL ENCRYPTION STANDARDS

A. DSTU 7624: 2014.

DSTU 7624: 2014 "Kalyna" is a modern symmetric block cipher that provides high security and robustness against attacks. The cipher was adopted as the national encryption standard of Ukraine in 2015 [4], and it is widely used in various applications, including information security and cryptography research. The design of DSTU 7624:2014 is based on the well-known AES cipher [5], but it introduces some significant modifications that enhance its security properties.

One of the main differences between DSTU 7624:2014 and AES is the use of different randomly generated S-blocks, as opposed to the same S-block used in each round of AES. This modification increases the complexity of the cipher and makes it more resistant to attacks. Additionally, DSTU 7624:2014 employs alternating addition with cyclic keys using modulo 2 and 2^{64}, which further enhances the security of the cipher.

DSTU 7624:2014 supports various key and block sizes, including 128, 256, and 512 bits [4], and its block diagram is shown in Fig. 1.

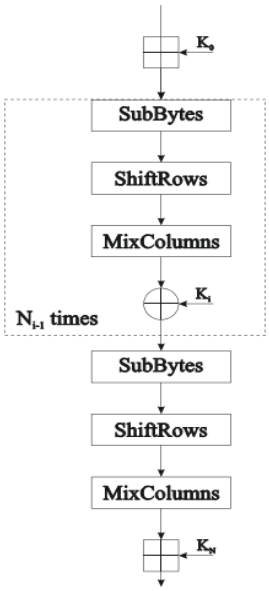


FIG. 1. Block diagram of the DSTU 7624: 2014.

The following features characterize the encryption scheme:

- whitening is performed by summing the input data block with a subkey K_0 by modulo 2^{64};
- for cycles from 1 to N_r-1 (where N_r is number of rounds and $N_r \in (10, 14, 18)$) cycle keys are entered by XOR with data block;
- the last encryption cycle consists of AES transformations: SubBytes, ShiftRows, MixColumns, and summing data block with a cycle subkey K_N by modulo 2^{64};
- for MixColumns transformation used field GF (2^8) constructed from the primitive polynomial $p(y) = x^8 + x^4 + x^3 + x^2 + 1$;
- in DSTU 7624:2014 four different S-box are used instead of 1 in AES.

In the encryption process using DSTU 7624:2014, $N_r + 1$ cycle keys K_i where $i = 0, \ldots, N_r - 1$. are used. The subkeys are generated during the key deployment, which is a part of the algorithm that produces round keys from the original key. The size of each subkey is the same as the size of the plaintext block and the current state of the cipher. In Fig. 2, the scheme of subkeys deployment is shown, where each subkey is used once in the encryption process. The key schedule algorithm for DSTU 7624:2014 is designed in such a way that it generates a unique subkey for each round of the encryption process. The number of rounds is determined by the key size, and for DSTU 7624:2014, it is 10 rounds for a 128-bit key, 14 rounds for a 256-bit key, and 18 rounds for a 512-bit key. This key schedule algorithm adds an additional layer of security to the cipher, making it resistant to attacks that attempt to reconstruct the original key from the subkeys.

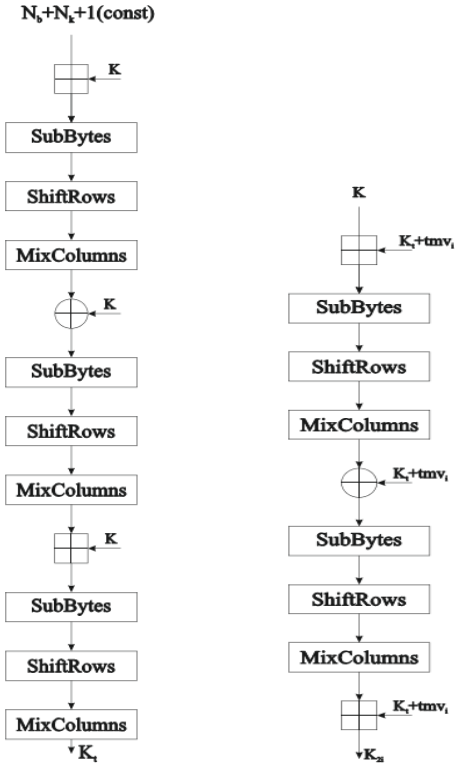


FIG. 2. Block diagram of subkeys deployment: *a* – intermediate key K_t; *b* – cycle keys K_i.

To generate an intermediate key K_t at the input of the algorithm, we provide the value of the master key K and the constant $const = N_b + N_k + 1$ where $N_b, N_k \in (2, 4, 8)$ and performing the necessary transformations according to Fig. 2 *a*, we obtain an intermediate key K_t with size $64 \times N_b$ bits, which is equal to the size of the input information block.

The value of the constant tmv_0 needs to be specified to generate cycle keys. By performing the transformation in accordance with Fig. 2b, using the intermediate key K_t, constant tmv_0, and master key K, a cyclic key with a zero index is obtained. Cycle keys with odd index $K_{(2i+1)}$ are formed based on keys with even index $K_{(2i)}$ by cyclic shift to the left on $2N_k + 3$ bytes, ie $K_{(2i+1)} = K_{(2i+1)} <<< (2N_k + 3)$. When forming the next key with an even index, the constant tmv_0 is divided on N_b 64-bit words $(w_0, w_1, \ldots, w_{N_b-1})$. Next, every word is logically shifted to the left by one digit, ie $w_i = (2 * w_i) \bmod 2^{64}$.

B. DSTU 8845: 2019. DSTU 8845:2019 is a stream cipher that follows the classic summing generator scheme, similar to the SNOW 2.0, SNOW 3.0, and SNOW V generators [5-6, 7]. It incorporates all the basic operations of ciphers from the SNOW family while replacing the AES encryption round with the nonlinear substitution function T, which implements the permutation of finite field elements GF(2^{64}) using components of the national cryptographic standard DSTU 7624:2014.

To ensure a high and ultra-high level of security, DSTU 8845:2019 uses a 256-bit initialization vector IV and a 256-bit or 512-bit secret key K, considering the possible application of quantum cryptanalysis. The developers of DSTU 8845:2019 focused on modern 64-bit computer

systems, selecting a word size of 8 bytes. Byte records use a representation from significant to less significant bits. The keystream generator for DSTU 8845:2019 in gamma mode is illustrated in Fig. 3.

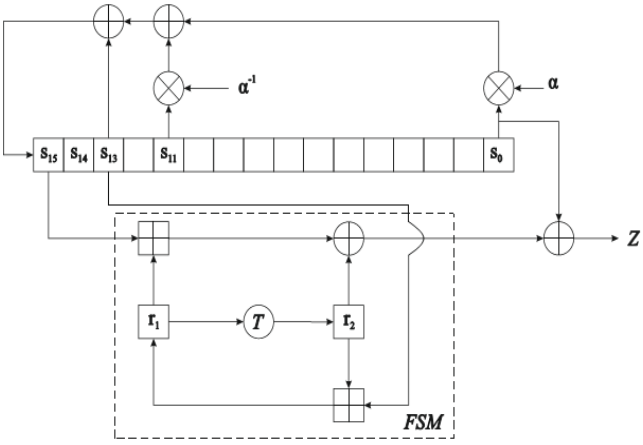


FIG. 3. Keystream generator DSTU 8845: 2019 in the mode of generating a *gamma* at time *i*.

Fig. 3 shows that the generator consists of a shift register with linear feedback and a FSM that performs the nonlinear transformation T. The input data (encryption key K and initialization vector IV) is used to initialize the state variable $S_i (i \geq 0)$, which consists of two components which include [12]:

- 16 variables $s^{(i)}$- cells of the shift register with linear feedback: $s^{(i)} = (s_{15}^{(i)}, s_{14}^{(i)}, s_{13}^{(i)}, s_{12}^{(i)}, s_{11}^{(i)}, s_{10}^{(i)}, s_9^{(i)}, s_8^{(i)}, s_7^{(i)}, s_6^{(i)}, s_5^{(i)}, s_4^{(i)}, s_3^{(i)}, s_2^{(i)}, s_1^{(i)}, s_0^{(i)})$;
- two registers of a finite automata $r^{(i)}$: $r^{(i)} = (r_2^{(i)}, r_1^{(i)})$.

At the output, we get the keystream (gamma), which is formed from 8-byte words Z_i.

From Fig. 3 follows that the taps of the shift register with linear inverse feedback are constructed from a primitive polynomial over the field GF (2^{64}): $f(x) = x^{16} + x^{13} + \alpha^{-1}x^{11} + \alpha$, where α is the root of the primitive polynomial over the field GF (2^8): $gz(z) = z^8 + \beta^{170}z^7 + \beta^{166}z^6 + \beta^2 z^5 + \beta^{224}z^4 + \beta^{70}z^3 + \beta^2$.

The field GF (2^8) as in DSTU 7624: 2014 is constructed from the primitive polynomial $p(y) = x^8 + x^4 + x^3 + x^2 + 1$ on the field GF(2), and the coefficients $g(z)$ are given through the degree of the primitive element β of the field GF (2^8), ie β is the root of the polynomial $p(y)$.

III. IMPLEMENTATION OF THE MAIN TRANSFORMATIONS OF THE DSTU 7624: 2014 AND DSTU 8845:2019 FOR FPGA

A. BASIC TRANSFORMATION DSTU 7624: 2014.

The column of 8 bytes (Fig. 4 a) is the main element of transformations in DSTU 7624: 2014. Depending on the size of the input block and key, the matrix of state will be formed using a certain number of columns. For example, in the case of a 128-bit block size and key, the matrix of state will consist of 2 columns, as shown in Fig. 4a. Similarly, for a block size of 256 and 512 bits, the matrix of state will contain 4 and 8 columns, respectively, as shown in Fig. 4 *b* and 4 *c*. To demonstrate the operation, we will use a block size of 128 bits.

w_0	w_8	w_0	w_8	w_{16}	w_{24}	w_0	w_8	w_{16}	w_{24}	w_{32}	w_{40}	w_{48}	w_{56}		
w_1	w_9	w_1	w_9	w_{17}	w_{25}	w_1	w_9	w_{17}	w_{25}	w_{33}	w_{41}	w_{49}	w_{57}		
w_2	w_{10}	w_2	w_{10}	w_{18}	w_{26}	w_2	w_{10}	w_{18}	w_{26}	w_{34}	w_{42}	w_{50}	w_{58}		
w_3	w_{11}	w_3	w_{11}	w_{19}	w_{27}	w_3	w_{11}	w_{19}	w_{27}	w_{35}	w_{43}	w_{51}	w_{59}		
w_4	w_{12}	w_4	w_{12}	w_{20}	w_{28}	w_4	w_{12}	w_{20}	w_{28}	w_{36}	w_{44}	w_{52}	w_{60}		
w_5	w_{13}	w_5	w_{13}	w_{21}	w_{29}	w_5	w_{13}	w_{21}	w_{29}	w_{37}	w_{45}	w_{53}	w_{61}		
w_6	w_{14}	w_6	w_{14}	w_{22}	w_{30}	w_6	w_{14}	w_{22}	w_{30}	w_{38}	w_{46}	w_{54}	w_{62}		
w_7	w_{15}	w_7	w_{15}	w_{23}	w_{31}	w_7	w_{15}	w_{23}	w_{31}	w_{39}	w_{47}	w_{55}	w_{63}		

a) *b)* *c)*

FIG. 4. Matrix of state: a) 128 bit, b) 256 bit, c) 512 bit.

At the input, we receive a block of information W of size 128 bits, which is represented as a matrix of size 2×8 bytes (Fig. 4 *a*). It should be noted that the state matrix is filled with bytes in columns: $W = \{w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, w_9, w_{10}, w_{11}, w_{12}, w_{13}, w_{14}, w_{15}\}$. Bit order is little-endian.

To generate an intermediate key, the value of the key K and the value of $const = N_b + N_k + 1 = 2 + 2 + 1$ (in the case of a 128-bit key) transmitted to the input and performed four rounds of encryption. The first round is the initial whitening that is described by:

$$C = \{c_0, c_1\} \quad (1)$$

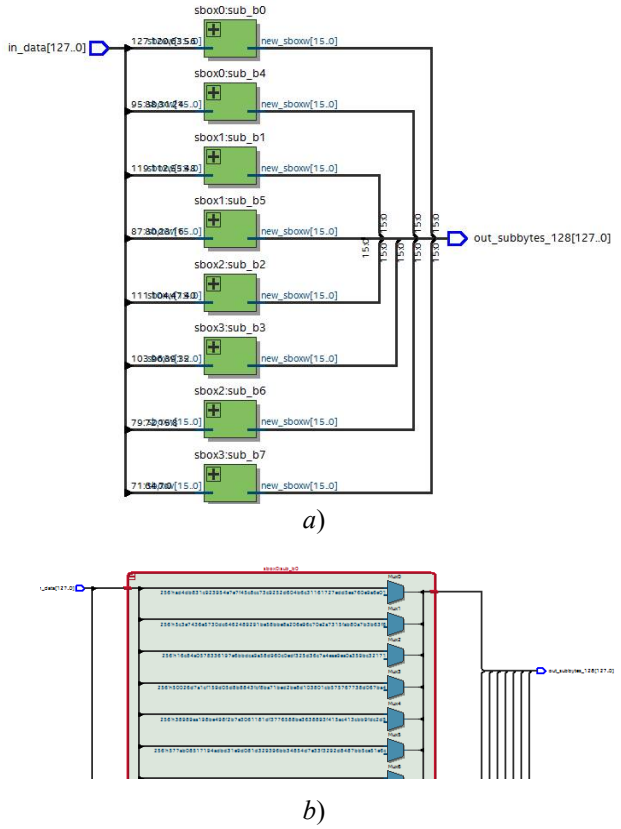
where:

$c_0 = (\{w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7\}) + (\{k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7\}) \, mod 2^{64}$ and $c_1 = (\{w_8, w_9, w_{10}, w_{11}, w_{12}, w_{13}, w_{14}, w_{15}\}) + (\{k_8, k_9, k_{10}, k_{11}, k_{12}, k_{13}, k_{14}, k_{15}\}) \, mod \, 2^{64}$

Taking the amount modulo 2^{64} can be executed by rejecting the transfer bit. This operation is trivial and can be implemented using a 64-bit carry adder.

After the initial whitening, in the second cycle, the current state is converted by *SubBytes* operation. The *SubBytes* operation consists in replacing each element $w_{i,j}$ of the state matrix W by $\pi_{i \bmod 4}(w_{i,j})$ where $\pi_m: V_8 \to V_8, m \in \{0, 1, 2, 3\}$ substitutions are given in appendix A of DSTU 7624: 2014 [4].

For example, if: $w = 0 \times 11$, then $\pi_1(0 \times 11) = 0 \times 15$. The peculiarity of DSTU 7624: 2014 is that it determines the use of four tables of substitutions, so it is advisable to implement such tables based on cells of read-only memory (ROM). From the condition, $i \bmod 4$it follows that the elements in one row use the same table of substitutions. That is the case of 128-bit block size causes the use of eight ROM blocks using multiplexers (Fig. 5).



a) *b)*

FIG. 5. Module *SubBytes* based on ROM in RTL Viewer: *a*) module SubBytes with 128-bit input, *b*) submodule *sbox0.*

Memory cells are organized as a two-dimensional array of 256 words of 1 byte each. This allows the *SubBytes* operation to be executed by one clock period.

The ShiftRows operation is a cyclic shift to the right of the rows of the state matrix, and is determined by the following relationship:

$$shift_i = \left\lfloor \frac{i*L}{512} \right\rfloor \quad (2)$$

where $L \in \{128,256,512\}$. Relation (2) for 128, 256, 512-bit block size can be visualized as follows:

w_0	w_8
w_1	w_9
w_2	w_{10}
w_3	w_{11}
w_{12}	w_4
w_{13}	w_5
w_{14}	w_6
w_{15}	w_7

a)

w_0	w_8	w_{16}	w_{24}
w_1	w_9	w_{17}	w_{25}
w_{26}	w_2	w_{10}	w_{18}
w_{27}	w_3	w_{11}	w_{19}
w_{12}	w_{20}	w_4	w_{28}
w_{13}	w_{21}	w_5	w_{29}
w_{14}	w_{22}	w_{30}	w_6
w_{15}	w_{23}	w_{31}	w_7

b)

w_0	w_8	w_{16}	w_{24}	w_{32}	w_{40}	w_{48}	w_{56}
w_{57}	w_1	w_9	w_{17}	w_{25}	w_{33}	w_{41}	w_{49}
w_{50}	w_{58}	w_2	w_{10}	w_{18}	w_{26}	w_{34}	w_{42}
w_{43}	w_{51}	w_{59}	w_3	w_{11}	w_{19}	w_{27}	w_{35}
w_{36}	w_{44}	w_{52}	w_{60}	w_4	w_{12}	w_{20}	w_{28}
w_{29}	w_{37}	w_{45}	w_{53}	w_{61}	w_5	w_{13}	w_{21}
w_{22}	w_{30}	w_{38}	w_{46}	w_{54}	w_{62}	w_6	w_{14}
w_{15}	w_{23}	w_{31}	w_{39}	w_{47}	w_{55}	w_{63}	w_7

c)

FIG. 6. Circular shift for: *a*) 128 bit, *b*) 256 bit, *c*) 512 bit.

In the case of a fixed data block size, the hardware ShiftRows operation can be implemented simply by selecting bits from the bus. In the case of a variable block, it will be necessary to use several circuits based on multiplexers.

When implementing additions modulo 2 and 2^{64} based on four input LUTs (Look-up table) 128 LUTs will be used. The *SubBytes* operation will require 3328 LUT when using a 128-bit state matrix and 8 ROM blocks for Cyclone V FPGA.

B. MIXCOLUMS OPERATION IN DSTU 7624: 2014. The *MixColumn* operation in DSTU 7624: 2014 is to convert each element of the state matrix by the formula:

$$w_{i,j} = (u >>> i) \otimes W_j \quad (3)$$

where: $\otimes$ is the scalar multiplication of vectors, $W_j$is the column of the state matrix, $v = (0x01, 0x01, 0x05, 0x01, 0x08, 0x06, 0x07, 0x04)$.

The operation of multiplication and addition is executed in the finite field formed by the irreducible polynomial $Y(x) = x^8 + x^4 + x^3 + x^2 + 1$ or $0x11d$ in hexadecimal. The vector v forms a circulating matrix of the *MDR* code and consists of a sequence of byte constants in the hexadecimal representation, which is interpreted as elements of the field GF (2^8), with a cyclic shift relative to the vector over the finite field. The *MixColumn* operation can be defined as:

$$C_j = \begin{bmatrix} 0x01 & 0x01 & 0x05 & 0x01 & 0x08 & 0x06 & 0x07 & 0x04 \\ 0x04 & 0x01 & 0x01 & 0x05 & 0x01 & 0x08 & 0x06 & 0x07 \\ 0x07 & 0x04 & 0x01 & 0x01 & 0x05 & 0x01 & 0x08 & 0x06 \\ 0x06 & 0x07 & 0x04 & 0x01 & 0x01 & 0x05 & 0x01 & 0x08 \\ 0x08 & 0x06 & 0x07 & 0x04 & 0x01 & 0x01 & 0x05 & 0x01 \\ 0x01 & 0x08 & 0x06 & 0x07 & 0x04 & 0x01 & 0x01 & 0x05 \\ 0x05 & 0x01 & 0x08 & 0x06 & 0x07 & 0x04 & 0x01 & 0x01 \\ 0x01 & 0x05 & 0x01 & 0x08 & 0x06 & 0x07 & 0x04 & 0x01 \end{bmatrix} \begin{bmatrix} w_{0,j} \\ w_{1,j} \\ w_{2,j} \\ w_{3,j} \\ w_{4,j} \\ w_{5,j} \\ w_{6,j} \\ w_{7,j} \end{bmatrix} \quad (4)$$

where C_j vector. Multiplication of bytes from the columns of the state matrix by the vector v can be implemented in the form of a combinational scheme.

Multiplying any nonzero element of the field GF (2^8) by 2 can be implemented as a logical shift of bits to the left by 1 position, resulting in a product of 9 bits. Next, from this result need to subtract modulo $x^8 + x^4 + x^3 + x^2 + 1$. These operations can be written as follows:

$$k_2(w_{i,j}) = w_{i,j} * 2 = \{w_{i,j}[6:0], 1'b0\} \oplus (8'h1d \wedge \{8\{w_{i,j}[7]\}\}) \quad (5)$$

where: $\wedge$ is a conjunction operation. Relation (5) describes the combinational scheme shown in Fig. 7.

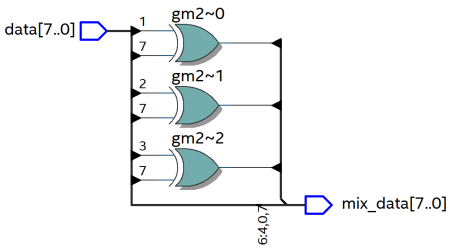


FIG. 7. Multiplication of the nonzero element of the field GF(2^8) by 2 modulo *Y(x).*

Taking into account (5), multiplication by 3 any nonzero element can be written as:

$$k_3(w_{i,j}) = w_{i,j} * 3 = k_2(w_{i,j}) \oplus w_{i,j} = (w_{i,j} * 2) \oplus w_{i,j} = (\{w_{i,j}[6:0], 1'b0\} \oplus (8'h1d \wedge \{8\{w_{i,j}[7]\}\})) \oplus w_{i,j} \quad (6)$$

Accordingly, the combination scheme described (6) will look like:

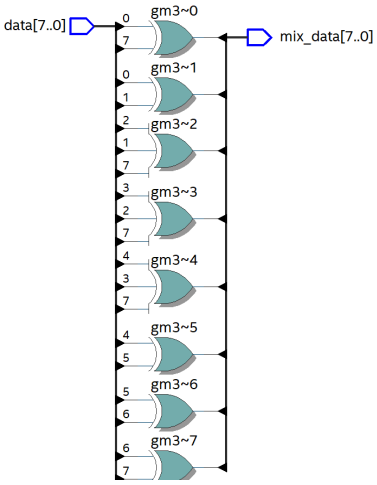


FIG. 8 Hardware implementation of (6).

Accordingly, for other coefficients from the vector v, and taking into account the rules of transformations in GF (2^8), the equations describing the corresponding combinational circuits, will look like:

$$k_4(w_{i,j}) = k_2(k_2(w_{i,j})) \quad (7)$$

$$k_5(w_{i,j}) = k_4(w_{i,j}) \oplus w_{i,j} \quad (8)$$

$$k_6(w_{i,j}) = k_2(k_3(w_{i,j})) \quad (9)$$

$$k_7(w_{i,j}) = k_6(w_{i,j}) \oplus w_{i,j} \quad (10)$$

$$k_8(w_{i,j}) = k_2(k_4(w_{i,j})) \quad (11)$$

So, taking into account (5-11), equation 4 takes the following form:

$$C_j = \begin{bmatrix} w_{0,j} \oplus w_{1,j} \oplus k_5(w_{2,j}) \oplus w_{3,j} \oplus k_8(w_{4,j}) \oplus k_6(w_{5,j}) \oplus k_7(w_{6,j}) \oplus k_4(w_{7,j}) \\ k_4(w_{0,j}) \oplus w_{1,j} \oplus w_{2,j} \oplus k_5(w_{3,j}) \oplus w_{4,j} \oplus k_8(w_{5,j}) \oplus k_6(w_{6,j}) \oplus k_7(w_{7,j}) \\ k_7(w_{0,j}) \oplus k_4(w_{1,j}) \oplus w_{2,j} \oplus w_{3,j} \oplus k_5(w_{4,j}) \oplus w_{5,j} \oplus k_8(w_{6,j}) \oplus k_6(w_{7,j}) \\ k_6(w_{0,j}) \oplus k_7(w_{1,j}) \oplus k_4(w_{2,j}) \oplus w_{3,j} \oplus w_{4,j} \oplus k_5(w_{5,j}) \oplus w_{6,j} \oplus k_8(w_{7,j}) \\ k_8(w_{0,j}) \oplus k_6(w_{1,j}) \oplus k_7(w_{2,j}) \oplus k_4(w_{3,j}) \oplus w_{4,j} \oplus w_{5,j} \oplus k_5(w_{6,j}) \oplus w_{7,j} \\ w_{0,j} \oplus k_8(w_{1,j}) \oplus k_6(w_{2,j}) \oplus k_7(w_{3,j}) \oplus k_4(w_{4,j}) \oplus w_{5,j} \oplus w_{6,j} \oplus k_5(w_{7,j}) \\ k_5(w_{0,j}) \oplus w_{1,j} \oplus k_8(w_{2,j}) \oplus k_6(w_{3,j}) \oplus k_7(w_{4,j}) \oplus k_4(w_{5,j}) \oplus w_{6,j} \oplus w_{7,j} \\ w_{0,j} \oplus k_5(w_{1,j}) \oplus w_{2,j} \oplus k_8(w_{3,j}) \oplus k_6(w_{4,j}) \oplus k_7(w_{5,j}) \oplus k_4(w_{6,j}) \oplus w_{7,j} \end{bmatrix} \quad (12)$$

So, we got the equation (12), describing the combination circuit which can be easily implemented in a hardware description language (Verilog, VHDL) for any FPGA chip architecture.

C. DSTU 8845:2019. In DSTU 8845:2019, the nonlinear substitution function Q^T (see Fig. 3) implements the permutation of elements of a finite field GF (2^{64}) using the *MixColumn* operation defined above by (12) for DSTU 7624:2014. Effective calculation of C_j can be realized based on the next rule:

$$Q^T = T_0[w_0] \oplus T_1[w_1] \oplus T_2[w_2] \oplus T_3[w_3] \oplus T_4[w_4] \oplus \oplus T_5[w_5] \oplus T_6[w_6] \oplus T_7[w_7], \quad (13)$$

where $T_i[w_i]$ are defined in [7] and can be realized as precalculated substitution tables on ROM (Fig. 9).

The use of constant tables allows to significantly reduce the number of operations, in particular, the nonlinear substitution function is calculated by seven XOR operations over 64-bit blocks.

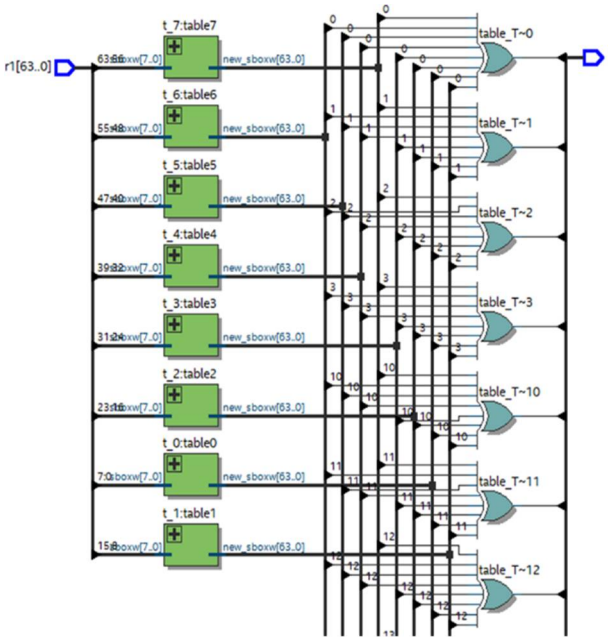


FIG. 9 Hardware implemented module of nonlinear transformation function T.

The concept of implementing an encryption system based on the developed hardware IP cores for the Cyclone 5 FPGA is shown in Fig. 10.

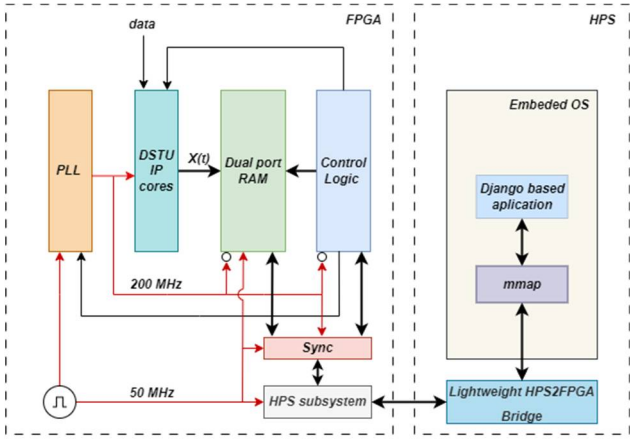


FIG. 10 Hardware implemented IP cores for DSTU 7624:2014 of of nonlinear transformation function T.

The mmap module can be used to interact through the control panel based on the Django application with the encryption module.

IV. CONCLUSION

In the work IP cores for FPGA-based systems based on the algorithm of symmetric block transformation, DSTU 7624: 2014 and stream cipher DSTU 8845:2019 are developed and verified.

In the case of DSTU 7624: 2014 developed and implemented a hardware implementation for the multiplication of two polynomials modulo $\boldsymbol{x^8 + x^4 + x^3 + x^2 + 1}$ in the form of a combinational circuit that allows the MixColumn operation in one cycle. SubBytes transformation is implemented based on asynchronous ROM.

For stream cipher, DSTU 8845:2019 are developed multiplication of two polynomials modulo $x^8 + x^4 + x^3 + x^2 + 1$ in the form of precalculated tables of RAM modules, which allows the MixColumn operation for the function nonlinear substitution T for one period of clock. The multiplication function by α and α^{-1} in arithmetic GF (2^{64}) is realized based on asynchronous permanent storage devices and combinational logic. The control of the modes of operation of the shift register with linear feedback is performed based on a finite state machine. The control of the source keystream and the feedback elements is performed based on a combinational scheme.

Future work will consist of the creation of drivers for real-time Linux-based solutions for Cyclone V SoC FPGA and exploring performance of the system for comparing with modern solutions.

AUTHOR CONTRIBUTIONS

Author contributions: O.K. – investigation and validation, resources, FPGA implementation and testing, writing-original draft preparation; S.H. and I.S.- investigation and validation, resources; I.G. - conceptualization, methodology. writing-original draft preparation, supervision.

COMPETING INTERESTS

The authors declare no competing interests.

REFERENCES

[1] Ya. R. Sovin, V. I. Otenko, E. F. Shtefanyuk “Effective implementation of the DSTU 7624:2014 block symmetric encryption algorithm ("Kalyna") for 8/16/32-bit embedded systems” Modern information protection, № 3, pp. 6-16, 2017.

[2] A.A. Kuznetsov, D.V. Ivanenko, E.P. Kolovanova “Perspective block cipher «Kalyna» modelling” Applied Radio Electronics, vol. 13, № 3, pp. 201–207, 2014.

[3] Gorbenko I., Kuznetsov A., Lutsenko M. and Ivanenko D. The research of modern stream ciphers // 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, pp. 207-210, 2017.

[4] DSTU 7624:2014. Information Technology. Cryptographic protection of information. Algorithm of symmetric block transformation. [Text]. - Enter 01–07–2015. - K.: Ministry of Economic Development of Ukraine, 2015.

[5] O. O. Kuznetsov, V. O. Frolenko, E. S. Yeromin, D. V. Ivanenko “Research of cross-platform implementations of stream symmetric ciphers” Radioengineering, № 193. – pp. 94–106, 2014.

[6] Patrik Ekdahl, Thomas Johansson, Alexander Maximov, Jing Yang, A new SNOW stream cipher called SNOW-V, IACR Trans Symmetric Cryptol 2019 (3) (2019) 1–42, 10.13154/tosc.v2019.i3.1-42.

[7] Information Technology. Cryptographic protection of information. Algorithm of symmetric flow transformation. DSTU 8845:2019, 2019.

[8] Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A. and Uhsadel, L., “A survey of lightweight-cryptography implementations” IEEE Design & Test of Computers, 24(6), pp. 522-533, 2007

[9] Rinne S., Eisen-barth T., Paar C. “Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers” ECRYPT Workshop Software Performance Enhancement for Encryption and Decryption, pp. 33-43, 2007.

[10] W. Diehl, F. Farahmand, P. Yalla, J. -P. Kaps and K. Gaj, "Comparison of hardware and software implementations of selected lightweight block ciphers," 2017 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, Belgium, 2017, pp. 1-4, doi: 10.23919/FPL.2017.8056808.

[11] Mohammed El-Hajj, Ahmad Fadlallah, "Analysis of Lightweight Cryptographic Algorithms on IoT Hardware Platforms", 2022 32nd International Telecommunication Networks and Applications Conference (ITNAC), pp.121-126, 2022.

Oleh Krulikovskyi

Assistant professor at Radio Engineering and Information Security Department of Yuriy Fedkovych Chernivtsi National University. His research field covers digital signal processing, FPGA and hardware cryptography. Author of more than 20 publications.

Serhii Haliuk

Assistant professor at Radio Engineering and Information Security Department of Yuriy Fedkovych Chernivtsi National University. His research interest covers the development of the different components of hidden communication systems. Author of more than 20 publications.

Ihor Safronov

PhD student at Radio Engineering and Information Security Department of Yuriy Fedkovych Chernivtsi National University. His research field covers digital signal processing, PCB layout and cicrcuit design. Author of more than 5 publications.

Ivan Gorbenko

Professor of the Department of Security of Information Systems and Technologies of V.N. Karazin Kharkiv National University, doctor of technical sciences, professor.

Field of scientific interests: cryptographic protection of information, cryptographic systems and protocols, post-quantum cryptography.

Co-developer of four Ukrainian national standards of cryptographic transformations.

Author of more than 300 publications, including 18 educational books and monographs, 85 patents.

Українські національні стандарти шифрування для вбудованих систем на базі FPGA

Олег Круліковський [1,2,3,*], Сергій Галюк[3], Ігор Сафронов[3], Іван Горбенко[4]

[1] Інтегрований центр досліджень, розробок та інновацій у галузі передових матеріалів, нанотехнологій і розподілених систем для виготовлення та керування, Сучавський університет імені Штефана чел Маре, Сучава, Румунія

[2] Факультет електротехніки та комп'ютерних наук, Сучавський університет імені Стефана чел Маре, Сучава, Румунія

[3] Кафедра радіотехніки та інформаційної безпеки, Чернівецький національний університет імені Юрія Федьковича, Чернівці, Україна

[4] Кафедра безпеки інформаційних систем і технологій, Харківський національний університет імені В. Н. Каразіна,. Харків, Україна.

*Автор-кореспондент (Електронна адреса: o.krulikovskyi@chnu.edu.ua)

АНОТАЦІЯ У роботі представлено апаратну реалізацію на базі FPGA основних криптографічних перетворень алгоритму блокового симетричного криптографічного перетворення ДСТУ 7624:2014 та потокового симетричного перетворення ДСТУ 8845:2019, які є національними стандартами криптографічних перетворень України. У випадку ДСТУ 7624: 2014 апаратна реалізація множення двох поліномів за модулем $x^8 + x^4 + x^3 + x^2 + 1$ виконана у вигляді комбінаційної схеми, яка дозволяє виконувати MixColumn перетворення за один цикл. Перетворення SubBytes реалізовано на основі постійного запам'ятовуючого пристрою (ПЗП). Керування модулями розгортання проміжних ключів та шифрування реалізовано на базі скінченого автомата. Для потокового шифру ДСТУ 8845:2019 нелінійна функція T реалізована як байтова операція підстановки у вигляді попередньо обчислених комірок ПЗП. Функція множення на α і α^{-1} в арифметиці поля Галуа GF (2^{64}) реалізована на основі ПЗП та комбінаційної логіки. Управління режимами роботи регістра зсуву з лінійним зворотним зв'язком здійснюється на основі скінченого автомата. Для підвищення швидкодії виконання криптографічних перетворень авторами використано конвеєрний принцип побудови IP ядер. Робота модулів апаратних реалізацій стандартів криптографічних перетворень ДСТУ 7624:2014 та ДСТУ 8845:2019 верифікована авторами відповідно до вказаних контрольних векторів у стандартах, і їх код HDL може бути наданий авторами для подальших досліджень зацікавленим сторонам. Апаратна реалізація IP ядер шифрів є кросплатформеною і може бути адаптованою на різноманітні FPGA із відповідними ресурсами. Також, в роботі представлено для подальших досліджень концепцію реалізації системи шифрування на основі розроблених апаратних IP-ядер для FPGA Cyclone 5 із використанням вбудованої операційної системи на базі ядра Linux для подальшої інтеграції їх до вбудованих систем або пристроїв IoT.

КЛЮЧОВІ СЛОВА ДСТУ 7624:2014; ДСТУ 8845:2019; FPGA; вбудовані системи.