IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 January 2024, accepted 22 February 2024, date of publication 1 March 2024, date of current version 19 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3372388

RESEARCH ARTICLE

SHC: 8-bit Compact and Efficient S-Box Structure for Lightweight Cryptography

SUNIL KUMAR[1,2], DILIP KUMAR[1], HEMRAJ LAMKUCHE[3], VIJAY SHANKAR SHARMA[2], HEND KHALID ALKAHTANI[4], (Member, IEEE), MUNA ELSADIG[4], (Member, IEEE), AND MARIYAM AYSHA BIVI[5]

[1]Department of CSE, National Institute of Technology Jamshedpur, Jamshedpur 303007, India
[2]Department of Computer and Communication, Manipal University Jaipur, Jaipur 303007, India
[3]School of Computing Science and Engineering, VIT Bhopal University, Bhopal 466114, India
[4]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh 11671, Saudi Arabia
[5]Department of Computer Science, College of Computer Science, King Khalid University, Gregar, Abha 61421, Saudi Arabia

Corresponding author: Muna Elsadig (Memohamedahmed@pnu.edu.sa)

This work was supported by Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia, through Researchers Supporting under Project PNURSP2023R384.

ABSTRACT The AES (Advance Encryption Standard) has made the development of new block ciphers unnecessary; it is now the de facto standard for most uses of block ciphers. However, the AES is still not well-suited for very limited contexts like RFID (Radio-Frequency Identification) tags and WSN(Wireless Sensor Networks), despite recent implementation advancements. In this study, we present SHC (Simple Hybrid Cipher), a new block cipher that uses a 64-bit block length and a 128-bit key length. It offers a hardware implementation that efficiently uses limited resources, making it ideal for use as a sensor in a WSN or an RFID tag. The core function of SHC depends on S-Box-based, composite field arithmetic technology, as it consumes relatively low cost on hardware implementation while still providing sufficient security as a solid encryption algorithm. The hardware implementation of SHC-64 requires 949 LUTs; it generates a maximum operating frequency of 515.995 MHz on the Xilinx-powered Artix-7 Field Programmable Gate Array (FPGA) development board. At the same time, the National Institute of Standards and Technology (NIST) recommended standard algorithm AES consumes 3645 LUTs and generates a maximum operating frequency of 277.369 MHz. The SHC-64 cipher also shows resistance against known cryptanalytics attacks.

INDEX TERMS AES, SHC, FPGA, block cipher, LU decomposition, cryptanalytic attack, hardware implementation.

I. INTRODUCTION

We use S- boxes in block cipher with the best cryptographic characteristics and economical hardware design. The S boxes, which work on 4-bit words, may have with smaller decreased size of variables which expands the optimum differential probability values together with linear correlation. In such cases, several turns are necessary to obtain a similar resistance to differential with linear attacks. In another way, building a lightweight cryptic cipher includes using a large S box, working with 8 bits just as an AES, instead of with less implementation cost. Under challenging problems, we can efficiently use S-Box, which uses fewer resources and has high safety value. But, for some mannered environments, the overall cost may be increased. Therefore, it is necessary to make a better S box application with the objective of successive S boxes adaptable to linear cum differential attacks. So, with this paper, we will build an inversion-dependent S box that will be easier to implement in hardware than an AES S box with the same cryptographic properties.

Mathematical functions over GF(2^m) can be thought of as the S-box, and it plays a crucial role in both the Substitution Permutation Network (SPN) and the Feistel structures (2^m).

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Wang.

An inversion of GF (2^8) and affine transformations of GF(2^m) are used by AES, CLEFIA, and Camellia, for example. The S-box circuits are critical to the hardware performance of these ciphers. There is a tremendous demand for lightweight S-box implementations because IoT devices are getting mainstream. IoT devices are generally resource-constrained. The subject of S-box hardware design is well-known. This is the simplest hardware version of the 8-bit S-box, but it takes up a lot of room. In light of these concerns, we must look into constructing an S-box using only logic gates, paying particular attention to the critical path delay and the overall area required. Cipher hardware and essential path time can be blocked by S-boxes, which have a considerable impact. Consequently, the S-efficient box's structure is critical to deciding the implementation's performance.

Lightweight block ciphers, for example, are designed to be implemented on ever-smaller devices. This goal can be achieved by developing algorithms that can only be used on a small number of devices at any given time. The use of cryptographic algorithms to secure cryptosystem data while requiring less hardware has been proposed [1], [2] In block ciphers like Noekeon, PRESENT, CLEFIA, block, AES [3], Camellia [4], and PRINCE, the substitution box is the most common and sophisticated sub-block (Xbox). CLEFIA and Camellia are two examples of block ciphers that, in addition to Halka and SMS4, can use the recommended S-box. It can also be used for future lightweight block ciphers. There are two main processes in this suggested S-box computation: field inversion and affine transformation. Nonlinearity, Differential approximation probability (DAP), The CMOS technologies at 180 nm and 65 nm make it possible to achieve critical path latency and area consumption for the structures. (Critical path latency). Hardware resources, temporal features, and security attributes are all adequate in the suggested system compared to the alternatives.

The Simple Hybrid Cipher (SHC) is expertly designed for environments with constrained resources, such as RFID tags and Wireless Sensor Networks (WSN), by utilizing a 64-bit block length and a 128-bit key length for optimal balance between security and performance. Unlike AES, which requires significant computational resources and power, SHC's efficient use of hardware, demonstrated by its implementation requiring only 949 Look-Up Tables (LUTs) compared to AES's 3645 LUTs, ensures minimal energy consumption and maximizes operating frequency, making it ideal for low-power devices. The core innovation lies in its S-Box design, employing composite field arithmetic for cost-effective hardware implementation while maintaining strong resistance against cryptanalytic attacks. This tailored approach addresses the challenges of AES in limited contexts by significantly reducing the hardware footprint and operational costs, ensuring SHC's suitability for securing next-generation IoT devices within these restricted environments.

II. LITERATURE REVIEW

The size of the data stored on a computer, the amount of power consumed by the device, and the device's price all need to be reduced, and an encryption technique is essential for this. such as electricity consumption, chip size, memory use, etc. While they are lightweight, lack of appropriate protection is one of the foremost issues behind their discard. However, several algorithms already adopt and applied in various areas of use, along with PRESENT [5], CLEFIA [6], KATAN/KTAANTAN [7], LE DED (light-encryption device), PHOTON [8], PICCOLO [9], PRINCE [10], SPONGENT [11], SIMON/ SPECK [12], LEA (low-speed encryption), PRIDE [13], TEA (minor encryption algorithm) (Wheeler and Needham 1994). This section summarizes the majority of modern lightweight block ciphers.

SFN (Feistel Network Substitution-Permutation) encoding method, both SPN structures and Feistel network structures, has been implemented (FN) [14]. Their device consisted of a 64-bit input block and a 96-bit key longitude, which were used by the rear 32-bit Key and the other 64-bit critical extension and round encryption as a control signal. Each 32-bit control signal performs a single round operation for key expansion, encryption, and decryption in the two structures' operational mode (SPN/FN). [15]. A key basic schedule was implemented without changing the key status, resulting in the need for smaller spaces for hardware deployment. Various round functions have been reprogrammed to make it possible to use the same small set of round keys repeatedly. Researchers [16], [17] created a lightweight block cipher algorithm based LiCi, including 31 consecutive rounds and four lightweight S-boxes. It supports 64-bit plaintext and 128-bit key length information. LiC used 1153 equivalent gate area (GE), 1944 storage bytes, and 30 mW power [18]. The advanced feature of SPN-based lightweight cryptography has been given the name BORON. It served a plaintext block that was 64 bits in size and had an input size of 128 or 80 bits. In addition to that, it has resistance against linear and differential assaults.

A 128-bit 1939 GE key and an 80-bit 1626 GE key are required with the BORON software and hardware platform. Reference [19] made several changes to the PRESENT design methodology and suggested an enhanced GIFT SPN block chip. PRESENT S-Box, which incorporated bit permutation with the DDT / S-Box Estimation Table, was utilized in place of a previous S-Box that was far smaller and more effective (LAT). This method proposed a GIFT-64 and a GIFT-128 with 28 and 40 rounds, each with a 128-bit steady of the key size. The scientists have implemented the 64-bit QTL block cipher algorithm to encrypt SPN and Feistel structures [20]. There were 64-bit and 128-bit key lengths. They decrypt using the same process as encryption but in the opposite order. Their solution didn't use key scheduling to reduce memory and power usage. The QTL hardware design uses 1025.52 and 1206.52 GE for 64-bit and 128-bit keys, respectively. QTL

isn't immune to statistical assaults on block ciphers. This is the method's biggest drawback [21], [22].

The plaintext and keys for 'RECTANGLE' are 80 and 128 bits. 80-bit main was 74.31 w and 1600 GEs. The 128-bit key mode used 2064 GEs and 72.15 w. Benefits include hardware-friendly architecture and faster development. Another group proposed adding an AKF small-block processor to the standard Feistel chip [23], [24]. Authors for AKF reintroduced ITUbee lightweight block cipher with an 80-bit block size and an 80-bit security key [25]. ITUBEE employed 20 rounds and some necessary whitening layers to make the game more secure. ITUBEE was designed to resist the significant attack that corresponded to it [26]. Combining SIMON and SPECK's advantages led to the SIMECK (SIMECK 32/64, SIMECK 48/96, and SIMECK 64/128) families (2015). Both CMOS 130 nm and 65 nm hardware were used. Random byte and bit-flip failure attacks can compromise SIMECK [27], [28]. The LED technique was given some help by a symmetric SPN-type block cipher proposed in Input is 64 bits, and keys can be either 64 bits (LED-64) or 128 bits (LED-128). Every three iterations, the LED used the same encryption method.

The author discusses several different S-boxes. Employs a cyclic group denoted by the symbol C255 to construct the proposed S-box. One way to build an S-box is to use the Box–Muller transforms in conjunction with polarization and the method for the central limit. It is proposed that the S-box be built using the bee waggle dance. An innovative new S-box approach was created, ''projective linear groups on the projective line and triangular permutation groups''. The author suggests a chaotic map and artificial bee colony optimization to create an S-box. In this way, the straightforward nature of the S-box approach to building is preserved.

Equal attention is given to both Feistel and MISTY structures in equilibrium. In we locate non-involutive 4-bit S-boxes, while in we locate involutive 4-bit S-boxes, both of which have an optimal bit-slice representation. The critical path delay for this 4-bit S-box is 7TX + 4TA, where TX and TA are the time delays of a 2-input AND gate and 2-input XOR gate, respectively. New tower fields are proposed to reduce the size of the AES combined S-box/inverse S-box design, which is then optimized for this new field and investigated the possibility of a more compact S-box. There is now the smallest AES S-box design available. The AES S-best-known box's implementation was given in two works. Although the composite field is employed it is the tower field that is taken into account over the more general framework of $F((2^2)^2)^2$ [29].

These works shift all linear operations from the input mappings to the output mappings of an isomorphic circuit. There have been numerous proposals for S-box circuits that are either more compact, quicker, or energy efficient than S-box. Despite its name, the S-box is not built for speed. Accordingly, high-velocity S-box layouts are provided. The paper proposes a hybrid method using polynomial and normal bases. With the MB, the inversion circuit's Area Delay can be reduced. Each PB, NB, and MB base represents an m-bit element of $F(2^m)$ [30]

Furthermore, each member of $F(2^m)$ can be represented in two independent representations using n (> m) bits. There exists an irreducible modular polynomial of degree m for non-redundant representations, while for redundant words, a modular polynomial of degree n can be reduced. Therefore, there is a larger pool of modular polynomials to choose from in redundant representations.

III. METHODOLOGY AND DESIGN OF SHC BLOCK CIPHER

In our study, the Simple Hybrid Cipher (SHC) employs S-Box-based Composite Field Arithmetic (CFA) technology, markedly enhancing the hardware efficiency of cryptographic processes while maintaining robust security for the encryption algorithm. This approach is particularly beneficial in constrained environments, such as RFID tags and Wireless Sensor Networks (WSN), where computational and power resources are limited. The use of CFA for the S-Box design in SHC enables a reduction in the hardware footprint by simplifying the complexity of the operations required for encryption and decryption processes. This is achieved through a strategic decomposition of algebraic operations into smaller, more manageable parts, allowing for efficient implementation on hardware with limited capabilities. By doing so, SHC can be implemented with a significantly lower number of Look-Up Tables (LUTs) compared to traditional algorithms like AES, which directly translates to reduced power consumption and faster processing times—critical factors in the aforementioned applications. Moreover, the security of SHC is not compromised for the sake of efficiency. The design choices surrounding the S-Box and the implementation of composite field arithmetic ensure that SHC retains resistance against common cryptanalytic attacks, such as differential and linear cryptanalysis. The optimization of the S-Box through CFA allows for the maintenance of high nonlinearity and low differential uniformity, which are key indicators of a cipher's ability to resist cryptanalytic attacks.

A. DESIGN OF FEISTEL STRUCTURE

The 64-bit plaintext is fed into the suggested Feistel structure. After that, it splits the input into two 32-bit portions. The XL is made up of the leading 32 bits, while the XR is made up of the leading 32 bits to the right. There are a total of fourteen rounds in the structure. A different subkey from the P-array is utilized in every round.

Figure 1 represents the Feistel structure of the algorithm. At the start of the i^{th} round, XL is XORed with an i^{th} subkey, i.e., Pi, and the result is stored in XL. After that, the F-function takes this XL value as an argument. After that, we XOR the F-output functions with XR, feed it into the S-function, and then XOR it with XL. The next round is fed the resultant XL, which is subsequently exchanged for XR. When 14 rounds have passed, the undo swap switches XR and XL.

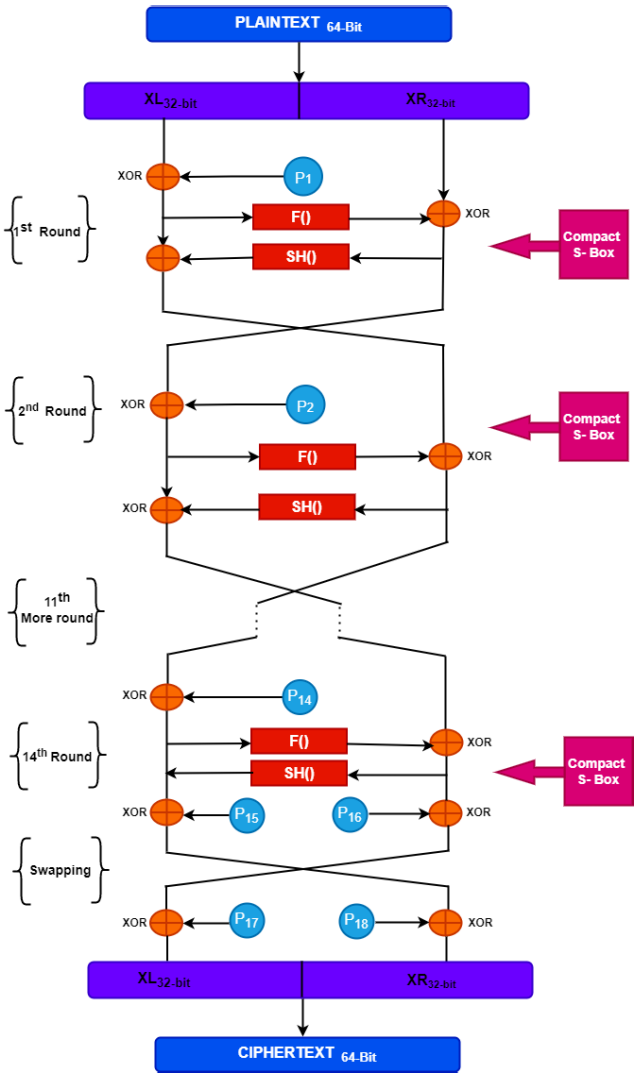


FIGURE 1. Proposed design of Feistel structure for SHC-64.

The XR and XL values are then XORed with the 15^{th},16^{th}, 17^{th}, and 18^{th} rounds using the key whitening subkey. For the 64-bit result, the 32-bit XL and XR numbers are added together. Decryption is identical to encryption, except that the key schedule is inverted.

B. LU DECOMPOSITION TECHNIQUE

In linear algebra, the LU Decomposition is the product of a lower and upper triangular matrix. There may also be a permutation matrix in this product. For solving a set of linear equations, we can compare the LU decomposition method to the matrix form of the Gaussian elimination method. LU decomposition technique takes a group of plaintext characters as input (say m) and substitutes ciphertext characters for them. The idea here is to take linear combinations of all the characters in one plaintext element. Several different varieties will be m. This will produce an (m×n) constant matrix S. LU decomposition method is used to generate the matrix containing keys, i.e., X=LU and g.c.d((det X) mod q, q) =1. The matrix containing the constant is then calculated using the formulae K=XP $\Longrightarrow$K =LUP. So, for the encryption process, the cipher (C) can be calculated as LC=K$\Longrightarrow$ C=L-1K, and in the decryption process, the plaintext (P) can be found using UP = C $\Longrightarrow$ P=U-1K.

C. LOW-COST S-BOX IMPLEMENTATION

A detailed explanation of the low-cost S-Box implementation is given below in several sections like 1, 2, and 3

1) **Cryptographic characteristics of S- S-boxes:** In different block ciphers, the S-box has its best use as a component. The S-box layer provides the foundation for the implementation cost, which includes area and CPD. To get an optimal result, we must minimize the cost of area and timing property of S- the box. In this paragraph, we will present an efficient optimization of area x delay to get 8- a bit S- box dependent upon inversion operation with less area and small CPD. An S box uses n-bit numbers as i/p and changes them into m-bit numbers o/p, where n and m may differ [31]. We can use a lookup table to implement the n x m S box with 2n words of m- bits. We can describe an S-box as a linear mapping from the fixed field F2n to the fixed area F2. A m x n S box can be defined as a vectorial Boolean function F: F2n We use differential uniformity analysis, algebraic degree, nonlinearity, and differential uniformity to check the strength of S boxes. In the future, we will present precise safety parameters that will be used to evaluate the security and safety of S boxes.
2) **The cryptographic architecture of the low-cost 8- bit S box:** So, in this work, we have used m×n S-Box as better hardware and time complex y the implementation of the proposed S-Box consumes low hardware cost for affine transformation and systematic Galois field inversion to minimize logic gates, a lookup table (LUT), and resource sharing is used to construct multiplication over GF (2^4) in the inversion circuit. The fundamental inversion process over GF (2^4) equality is optimally revised. The proposed architecture is divided into two parts. The first part has been improved to minimize area and delay. The other part contains two addition operations, one multiplication operation, one field squaring operation, and one multiplication operation using constant (λ), all performed over GF (2^4). The integrated block of the initial part of S-Box is linked to the form of a cohesive framework. As a result of the implementation of CMOS technology demonstrated in terms of the overall performance, the proposed S-box and the S-box over inversion are much faster than the conventional composite field arithmetic S-box over the Galois field. Most of the well-known S-boxes have less area, delay, and throughput. The proposed work of S-box provides high security and consumes little hardware cost in implementing S-box over FPGA devices which is a healthy option for block cipher algorithms.

3) **Design of low-cost 8- bit S-Box** We have used m×n S-Box with the integrated Boolean function of 8-bit S-box where F:F(2^n) is made by placing all elements with the inverse values and using it with an affine transformation.
Where m is the number of input bits, and n is the number of output bits. The values of m-bit and n-bit are different. Now for LUT implementation, the capacity of m and n varies. m holds the number of bits of the address, whereas n has the width of words depending on the used bit. The value of m and n are used in S-box implementation. In this research, we have used optimized composite field arithmetic operation over GF $(2^4)^2$ to implement the field inversion. Mixed-field arithmetic is used in the implementation of the proposed S-box to reduce the amount of hardware resources needed to calculate the S-box. The following two-step procedure is used to lower the cost of the S-box operation:
Step 1: Over the field, compute the field inversion over GF $(2^4)^2$.
Step 2: Inverting the output using a low area-efficient affine transformation operation.
Let us take the arbitrary element in the field GF $(2^4)^2$. m×n S box function GF (2m), where m is the no of a bit of address – GF $(2^4)^2$ is used to implement field inversion. Thus, to reduce the cost of hardware implementation, we have specifically used below:

1) Compute the field inversion over GF $(2^4)^2$
2) Applying an affine transformation to the output of the inversion (as shown in Figure 3)

Let us take m_i x+m_j to be an arbitrary element in GF $(2^4)^2$, Where $m_{i=}(m_i\ 3,\ m_i\ 2, m_i\ 1,\ m_i\ 0\ )_2$ the most significant bit is in the 8-bit input and $m_j = (m_j\ 3,\ m_j\ 2, m_j\ 1,\ m_j\ 0\ )_2$ the least significant bit in the 8-bit input, therefore $m_i, m_j \epsilon\ GF(2^4)$ and $m_i, m_j \epsilon\ GF(2)$ The Galois field GF $(2^4)^2$ is constructed as a field extension over the Galois field GF (2^4) by using the irreducible polynomial f(x)=x^2+Kx+Y where K, Y GF (2^4). now the inversion can be calculated by inversion of n_ix +n_j = (m_ix + m_j)$^{-1}$ where n_i, n_j belongs to GF(2^4). The above Equation can be generated by using extended Euclidian algorithms as follows:

(m_ix + m_j)$^{-1}$ = m_i ($m_i{}^2$Y + $m_i m_j$K+$m_j{}^2$)$^{-1}$ x + (m_j + m_iK) ($m^2{}_i$Y+$m_i m_j$K + $m^2{}_j$)$^{-1}$

$$\xrightarrow{Choosing\ K=1}$$

$$\begin{aligned} Y &= \lambda\ (m_i x + m_j)^{-1} \\ &= m_i(m^2 i\lambda + b\left(m_i + m_j\right))^{-1}x \\ &\quad + (m_j + m_i)(m_i^2\lambda + m_j(m_i + m_j))^{-1} \\ &= m_i D^{-1}x + (m_j + m_i)D^{-1} \end{aligned}$$

where, $D = \left(m_i + m_j\right)m_j + \lambda m_i^2$, To calculate the output bit, we have $n_j = m_i D^{-1}$, and $n_j = (m_j + m_i)D^{-1}$. Thus, the irreducible polynomials used in the optimized composite field arithmetic operation in the Galois field consist of $GF(2^2)c \rightarrow GF(2^2)^2$ using $F(x) = x^2 + x + 1$, and $GF(2^2) \rightarrow GF(2^2)^2\ f(x) = x^2 + x + \phi$ and GF(2^4)GF(2^4)^2 $f(x) = x^2 + x + \lambda$ Where the value of ϕ and λ are chosen in such a way as to minimize the area consumption in hardware architecture, therefore we have to use the constant value of ϕ and λ to get the lowest area utilization are 10_2 and $(1100)_2$, i.e. 2_N and 12_N respectively.

Figure 2 depicts the proposed architecture of an 8-bit S-box, an 8-bit inverse S-box (S-box^{-1}), and a coupled S-box and S-box1 (S-box/S-box^{-1}). A shared inversion block is shared by S-box/S-box^{-1}, the integrated S-box/S-box^{-1}. We have proposed a transformation that can be optimized with hardware implementation. Sub-blocks' multiplication and their inversion squaring must be defined over GF (2^4) with polynomial $f(x) = x^2 + x + 1$. Further, we will have to implement optimization with the design of the S box by reducing the delay on the critical path. Therefore, the method depends on optimizing a few sub-blocks, which may combine the square of the sub-block and multiply with λ. Thus, the gate count and delay of the critical path will be reduced. Hence, we can find the advantage of the S box as having the most minor consumption and delay of time will also be low.

The operation of the S box is computed by inverting the function GF (2^4) and transformation (AT). The first inverse transformation (AT-1) is initiated for the S box using GF (2^8) The affine transformation, the initial flip operation over in GF (2^8), is accomplished in the S-box operation immediately. The inversion is also performed following the inverse affine transformation in the S-box (S-box1) inverse. The hardware configuration of the field elements and the affine transformation greatly impact the S-box's complexity. The Equation for affine transformation is AT(A) = B, where B = M×A⊕ C_1 is computed, where A, M, C_1 is given below:

$$\mathrm{M} = [8 \times 8]$$

$$A\begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = B\begin{bmatrix} 1\,0\,0\,0\,1\,0\,0\,1 \\ 1\,1\,0\,0\,0\,1\,0\,0 \\ 0\,1\,1\,0\,0\,0\,1\,0 \\ 0\,0\,1\,1\,0\,0\,0\,1 \\ 1\,0\,0\,1\,1\,0\,0\,0 \\ 0\,1\,0\,0\,1\,1\,0\,0 \\ 0\,0\,1\,0\,0\,1\,1\,0 \\ 0\,0\,0\,1\,0\,0\,1\,1 \end{bmatrix} \times \begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Here A = (a_0, a_2, a_3, a_4, a_5, a_6, a_7) B = (b_0, b_2, b_3, b_4, b_5, b_6, b_7), C_1 = (10000010) m and n is 8×8 matrix. The inverse S-Box used inversion is performed after the inverse of affine transformation (AT^{-1}), where AT^{-1} is calculated by computing AT^{-1}(B). where B = N×A⊕ C_2. Where B is computed by using N, A, and constant C_2 as given

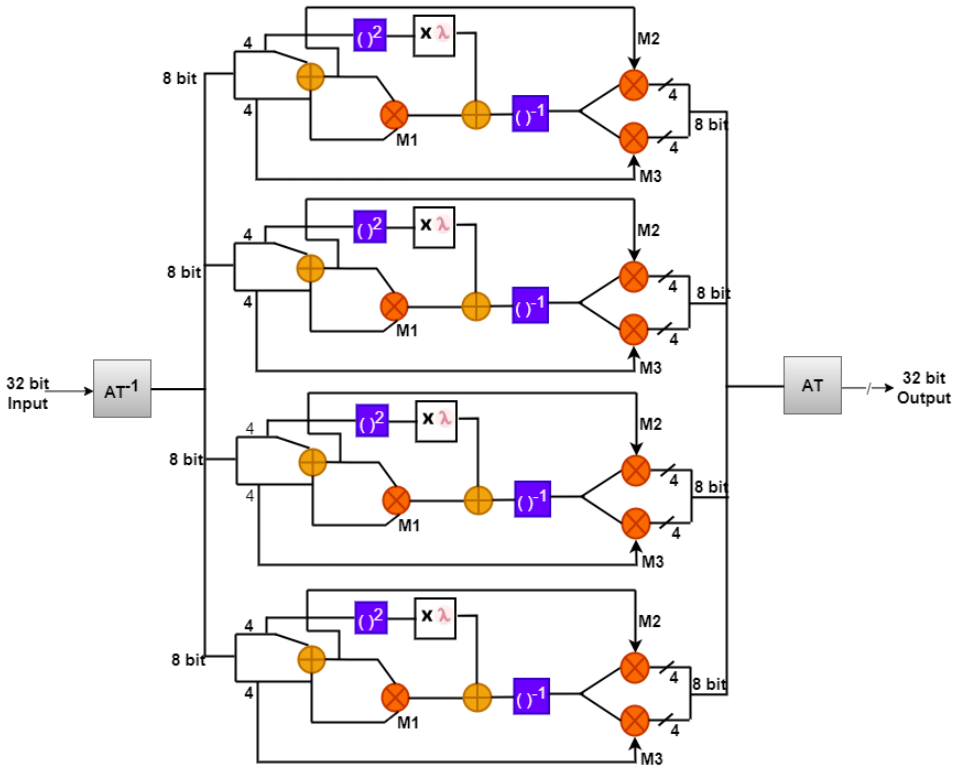


FIGURE 2. Design of low-cost 8- bit S-Box.

below:

$$\left[AT^{-1}\right] = \mathrm{N}\begin{bmatrix} 0\,1\,1\,0\,0\,0\,1\,0 \\ 0\,0\,1\,1\,0\,0\,0\,1 \\ 1\,0\,0\,1\,1\,1\,0\,0 \\ 0\,1\,0\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,0\,1 \\ 1\,0\,0\,0\,1\,1\,0\,1 \\ 1\,1\,0\,0\,0\,0\,0\,0 \end{bmatrix} \times \begin{bmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Here B is an 8×8 matrix, and constant C_2 is (10101111). Below is the formula for calculating affine transformation (AT) B = (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0), similarly for computing the inverse affine transformation (AT^{-1}) is A = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0). The transformation of information between affine transformation and inverse affine transformation is given below in the following equations:

$$\begin{aligned}
a_0 &= (b_7 \oplus b_3 \oplus b_0)'\ b_0 \ = (a_4 \oplus a_1 \oplus a_0) \\
a_1 &= (b7 \oplus b_6 \oplus b_2)'b_1 \ = (a_5 \oplus a_2 \oplus a_1) \\
a_2 &= (b_4 \oplus b_1 \oplus b_0)'b_2 \ = (a_6 \oplus a_3 \oplus a_2) \\
a_3 &= (b_5 \oplus b_2 \oplus b_1)'b_3 \ = (a_7 \oplus a_4 \oplus a_3) \\
a_4 &= (b_6 \oplus b_3 \oplus b_2)'b_4 \ = (a_5 \oplus a_4 \oplus a_0) \\
a_5 &= (b_7 \oplus b_4 \oplus b_0)'b_5 \ = (a_6 \oplus a_5 \oplus a_1) \\
a_6 &= (b_5 \oplus b_4 \oplus b_0)'b_6 \ = (a_7 \oplus a_6 \oplus a_2) \\
a_7 &= (b_6 \oplus b_5 \oplus b_1)'b_7 = (a_7 \oplus a_3 \oplus a_0)
\end{aligned}$$

The architecture of the proposed affine transformation AT(A) and inverse affine transformation AT^{-1}(B) requires several two-input XOR gates and two-input XNOR gates to implement on the hardware. The critical path delay of the transformation function, like affine transformation AT(A) and inverse affine transformation AT^{-1}(B), is = 2 times Tx, where x is the XOR gate delay.

1) **Multiplication Inversion using Galois Field:**
The multiplication over Galois field GF (2^4) is based on primitive polynomial f(x) where the function f(x) is described as follows: f(x) = x−4 + x+1. The overall structure of multiplication in the inversion is as two 4-bit inputs: A1 and A2. The critical path delay in two inputs, A1 and A2, and the multiplication output is 2Tx and 3Tx, respectively, where Tx signifies the time delay of two 4-bit input XOR gates, respectively, as shown in Figure 4. The structure uses 3-Quad bit field multiplication in the inversion system. So, to reduce the no of XOR gates, we use two multiplications with single operands. The input is supplied to the input A2 of field multiplication in the inversion system.
Thus, to minimize the critical path delay, we have used a similar multiplication supplied to the input A1 of field multiplication in the inversion system. The overall structure uses two multiplications with primitive polynomial f(x)., where f(x) = x−4 + x+1. Thus, the hardware resources needed for two distinct

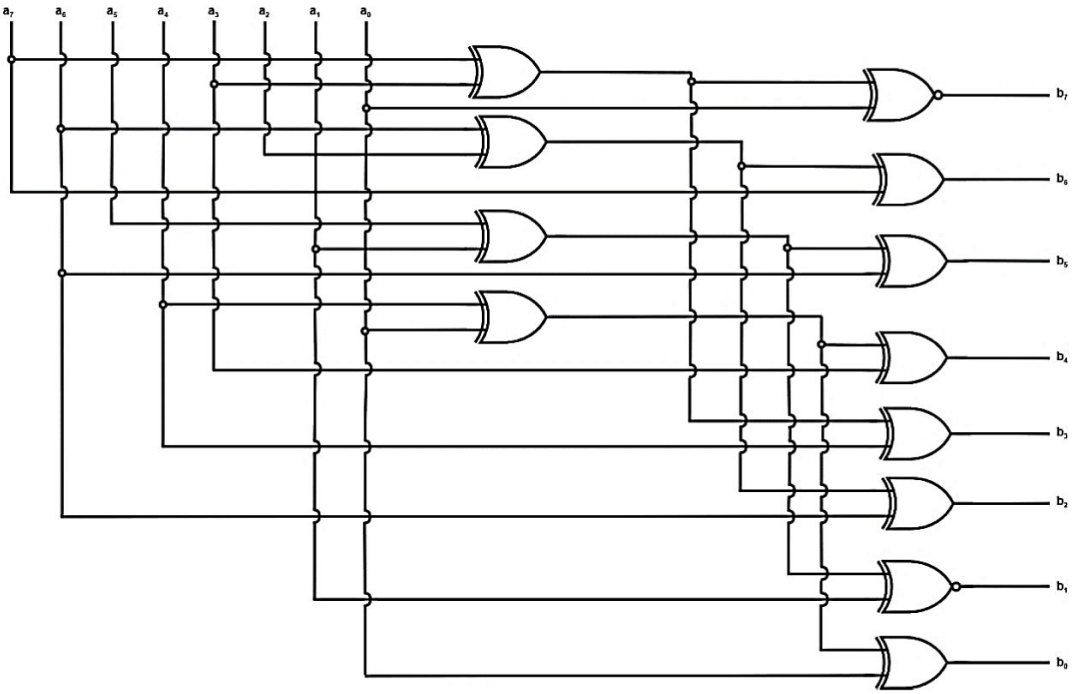


FIGURE 3. Affline transformation for SHC-64.

multiplications are two input Nand Gate and two input XOR Gate. However, the intended analytic of two multiplication required fewer hardware resources; hence less XOR gate and Nand gate are minimized in our proposed systems.

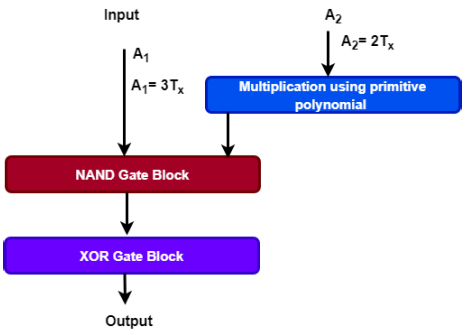


FIGURE 4. Optimized gate usage in SHC-64.

2) **Optimized architecture of low-cost 8-bit S-Box**

We have deployed hardware acceleration to the improved 8-bit S-box. In the S-Box, you can find a constant, a multiplication block that combines squaring and multiplication, and two addition operations(λ). The S-box's internal structure combines to form a coherent framework where we have used two-block multiplication by the constant (λ). The execution of field squaring is executed over Galois field GF (2^4). It is specifically used for hardware optimization only. The area squaring is implemented using 4-bit binary numbers, which are the elements of Galois field GF (2^4), along with primitive polynomial f(x) where f(x) = x−4 + x+1.

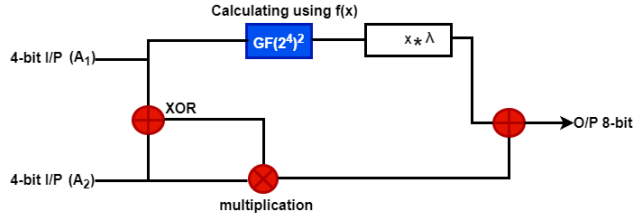


FIGURE 5. Optimized architecture of low-cost 8-bit S-Box.

To maximize the performance of the below S-box structure mentioned in Figure 5, the overall process initiated with 8-bit input is further divided into pairs of 4-bit inputs. Now we compute the value by using the XOR gate and then performing multiplication over we have Galois field GF (2^4) along with primitive polynomial f(x) where f(x) = x−4 + x+1. We have also used a pair of block multiplication using λ (a constant variable as discussed above). The above operation's output is again evaluated using the XOR gate over Galois field GF (2^4).

Thus, we can optimize the hardware implementation of our S-box structure which will help us achieve higher performance. Assuming that the critical path delay is evaluated to 2Tx, nearly 4 logic gates are needed to perform both squaring and multiplication operations over constant λ. The 4-bit output in Figure 6. above requires no logic gates following refinement in the unification of the S-box structure. As a result, our new

design s-box structure requires less hardware footprint in implementation and further improves critical path delay in the s-box structure.

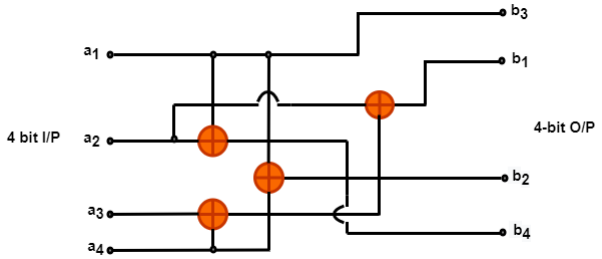


FIGURE 6. Inversion function over galois field.

3) **One nibble inversion function in GF (24):** The multiplicative inversion can be executed by using any of the techniques given below:

- The multiplicative inverse is calculated by mathematical inversion in the Galois field GF (2^4), similar to the XOR logic gate.
- The multiplicative inverse can also be implemented using lookup tables in hardware descriptive languages like VHDL or Verilog.
- The above inversion equation can also be deduced by using electronic gates like AND, OR and
- NOT gate excluding XOR gate in Galois field GF (2^4).

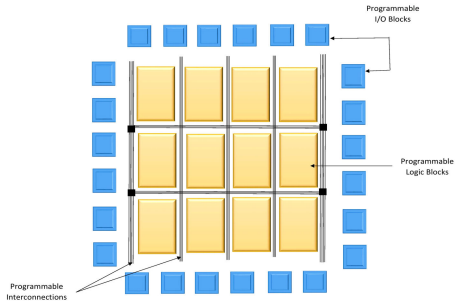


FIGURE 7. Inversion function over galois field.

The primitive polynomial f(x) where f(x) = x^{-4} + x+1. Uses $\lambda = \{1100\}_2$ in our implementation. From Figure 7, we state that the overall performance of multiplicative inversion consumes fewer logic bits and optimizes the critical path delay, further improving hardware area footprint and hardware efficiency by using the composite field arithmetic operation over the Galois field GF (2^4).

IV. FPGA IMPLEMENTATION OF SHC BLOCK CIPHER

A. VHDL IMPLEMENTATION OF SHC BLOCK CIPHER

Any verified issue can be solved by an FPGA. FPGAs use Xilinx soft microprocessors due to their parallel architecture and gate equivalent efficiency. Digital controllers with excellent performance and cheap cost in the footprint area are required for commercial applications. VLSI technology enables high-performance, low-cost architecture. A unique hardware description language (HDL), such as Verilog HDL or VHDL, was developed to describe digital circuits. Developers may quickly test and model their digital circuit designs using the simulators for Verilog® and VHDL. Hardware implementations include FPGAs, ASICs, and GPUs. Because of FPGA's low cost and programmability, we chose to use it for the hardware implementation of our method. To facilitate the rapid rollout of WSN-compatible lightweight cryptography, RFID tags, aggregation networks, the Internet of Things, and pervasive computing, FPGA are almost the perfect runners. Internal memory expansion, such as Block RAM in Xilinx devices or system blocks in Artix-7 devices, is when a device gets internal memory. FPGA has many advantages since it can quickly build lookup tables and conversion functions on the device's limited RAM. We used a unique FPGA board with Digilent Nexys DDR4 Artix 7 power to implement our SHC algorithm. The timing analysis, behavioral simulation, and Verilog implementation are all synthesized using Xilinx Vivado 2022.3. The SHC algorithm is broken into many modules to ensure portability and coded using the Verilog computer language.

B. EFFICIENT HLS IMPLEMENTATION USING XILINX VIVADO

The programmable input-output, programmable logic, and programmable connectivity of these blocks within a single unit make up the FPGA architecture (see Figure 8. A data stream is directly customized into the design of an FPGA device. The lookup tables (LUTs), registers, slices, and arithmetic hardware, which include digital signal processing (DSP) blocks, are all integrated within the programmable gate array. LUTs can be configured to carry out logic operations very quickly. It can convert a bitwise AND action into a bitwise XOR operation. By repeatedly creating comparable computing hardware in an FPGA, parallelism is possible. The FPGA device carries out the task without a CPU because its architecture serves as a circuit in and of itself.

Additionally, we can set up the system to work effectively for both single-core and multi-core processing. The entire FPGA is stored in volatile memory (RAM or random-access memory), where the program remains in memory as long as the device is powered on. They should therefore be created each time a control is offered. When an FPGA device is being programmed, a design entry is used, in which the source code for a program is written in either Verilog programming or hardware description language (HDL). The code is then translated into device netlist format after being programmed. The entire circuit, including logical components, is represented in the netlist format. The design synthesis phase is the name of this conversion phase. Native Generic Circuit (NGC) files are created during synthesis and sent to the design implementation phase for additional processing.

Using an NGC file as an input, the design implementation process performs serial operations, including translation, map, and place and route. The translate function converts the design into NGD and maps the input netlists from the NGC file (native generic database). Physical pins on the FPGA board have been assigned to each available port. The user

constraints (UCF) file contains the translation phase's output. All subblocks are mapped to combinational logic blocks in the mapping phase, which follows the design implementation process (CLB). Place and route is the final stage of design implementation, during which every logic block is precisely mapped following the user-constrained file (UCF). Once the steps above have been completed, a connection wire is used to load a routed design into an FPGA device. The implemented method needs to be converted into a format that the FPGA device can support. BITGEN is a tool used to create a BIT file that is configured with an FPGA device to convert an implemented design into an FPGA-compatible format.

C. TRANSLATION DESIGN AND OPTIMIZATION

This research implemented the SHC cryptographic algorithm using a Digilent Nexys4 DDR FPGA board. The Nexys4 DDR board is a cutting-edge Artix-7TM Field Programmable Gate Array (FPGA) from Xilinx® powered digital circuit creation platform that is ready to use. The high-capacity FPGA board for the Artix-7 is powered by Xilinx and has the part number XC7A100T-1CSG324C. Its other features are significant external memories, USB ports, Ethernet, VGA connectors, some input switches, 16 LEDs, 7 segment LEDs, and numerous additional resource ports. Applications ranging from straightforward combinational circuits to potent embedded processors can be run on the Digilent Nexys4

Nexys4 DDR Artix-7 FPGA Board powered by Digilent	
Project Family	Artix-7
Part No.	XC7A100TCSG324-1
Process Nodes	28 *nm*
Target Language	HDL / Verilog
Development tool	Vivado® 2022.2
Device Name:	Digilent Nexys4 DDR
#Slices	15,850 logic slices 6-input LUT, 8 Flip-flops
RAM	4,860 Kbits
Clock	6 Clock tiles, each with a phase-locked loop (PLL)
DSP slices	240
Slice LUTs	63,400
Slice Registers	126800
F7 Muxes	31700
F8 Muxes	15850
Block RAM tile	135
Bonded IOB	210
PHY_CONTROL	6
Phaser_Ref	6
In/Out FIFO	24/24
IBUFDS	202
Delay	300
IO_LOGIC	210
Internal Clock Speed	450 MHz

DDR chip. The Digilent Nexys4 DDR FPGA device is manufactured with numerous integral peripherals, including a temperature sensor, digital microphone, accelerometer, speaker amplifier, and multiple input-output devices, which enables the Digilent Nexys4 DDR FPGA device to be used for a wide range of projects without requiring any additional hardware (DigilentInc, 2014). High-performance logic is installed on the Artix-7 XC7A100TCSG324-1 FPGA device powered by Xilinx to maximize performance. It provides a lot of capacity for improved project development as well. The table below lists the key characteristics of the Digilent Nexys4 DDR Artix-7.

D. OPTIMIZED LOGIC IMPLEMENTATION

The Digilent Nexys4 DDR Artix-7 FPGA board is compatible with most advanced development tools, including Xilinx ISE and High-performance development tool Xilinx™ Vivado® 2022.2 edition, includes ChipScope™ for better device management. This FPGA development board is empowered with 128 MiB DDR2 SDRAM memory. It also delivers automatic overcurrent and overvoltage protection on the input power supply. This device is also plugged in with the list of manufactured ports and peripherals, including 16 user switches, a Micro SD card connector, 16 user LEDs, a 12-bit VGA output, Two 4-digit 7-seg display, PWM audio outputs, a USB-UART Bridge, PDM microphone, Two tri-color LEDs, 3-axis accelerometer, Temperature sensor, Digilent USB-JTAG port, 10/100 Ethernet, USB HID Host, 128MiB DDR2, Five pushbuttons, Serial Flash, JTAG port for external cable, Four PMOD ports, FPGA configuration reset button, PMOD for XADC signals, CPU reset button (for soft cores), and Programming mode jumper.

E. DEVICE PROGRAMMING AND TESTING

Our SHC processor and hardware implementation of the cryptographic algorithm is created using the Xilinx Vivado tool, which Xilinx Developer offers. The updated Graphical User Interface of Xilinx Vivado® 2022.3 is available (GUI). The UltraFast Create Methodology, integrated into Vivado 2022, enables developers to design projects at a breakneck speed, enhancing the design cycle and maximizing productivity. The simplest way to develop a solution using hardware description language or Verilog programming is with Vivado, which compiles source code to guarantee logical programming efficiency. One key aspect that draws us from Xilinx ISE to Vivado is that it has a particular feature to integrate C and C++ applications and automatically transform them into hardware description language or Verilog on the fly. Programs written in C, C++, and Synthetic can be natively targeted for Xilinx-powered devices using Vivado's built-in standard libraries without creating RTL (Resistor-Transistor Logic). High-level synthesis (HLS) is a feature of Vivado that guarantees high performance and productivity. In its IDE (integrated development environment), Xilinx Vivado has four primary components: the Vivado Simulator, High-Level Synthesis, IP Integrator, and TCL (Tool command language) store.

F. BEHAVIORAL SIMULATION

To translate our design logic into a functional project, implementation in Xilinx Vivado follows a straightforward procedure. When you start a new project, you are prompted to enter the working project's name and location. After that, you must choose the type of flow and source code file. You

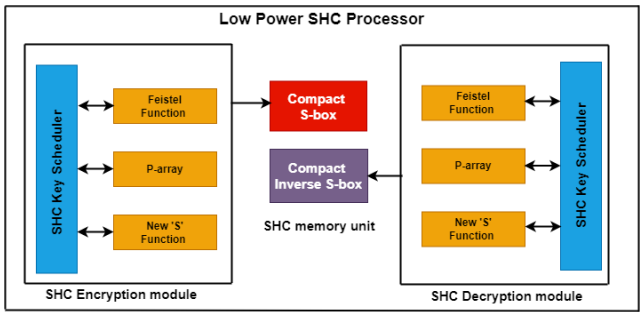


FIGURE 8. Low powered SHC-64 implementation.

can select the type of project you want to build from various options, including RTL projects, post-synthesis projects, I/O planning projects, imported projects, and example projects. The RTL project enables us to execute RTL analysis, Synthesis implementation, design planning, and process add programming source code files, design block designs in IP integrator, and generate IP. The source code file, which allows for the specification of HDL or Verilog source code files, netlists, block designs, and IP files, is a crucial component in designing projects. Finding the Design Constraints File (SDC or XDC) for Physical and Timing Constraints is the following stage in the development phase. The development board we selected for designing our projects is specified by Vivado in the following method; in this instance, we chose the Digilent Nexys4 DDR Artix-7 FPGA board. By creating BITSTREAM using the hardware manager tool, Vivado transforms our design into a device-supported format after the synthesis and implementation phases are complete.

G. FUNCTIONAL SIMULATION

This research aims to construct a low-cost processor for our SHC cryptographic method. We outline a novel strategy for implementing the S-box at a reasonable cost utilizing composite field arithmetic (CFA) technology. The initial key and S-box settings are utilized during the encryption procedure.

A pipelined implementation of the compact S-box was used to achieve parallelism and excellent performance in key generation. The Field Programmable Gate Array (FPGA) chip implements the SHC algorithm and its low-cost processor design. Our method significantly outperformed current techniques when we compared them. SHC algorithm hardware was integrated into a Digilent Nexys4 DDR Artix-7 FPGA chip. The SHC algorithm processor's main module is depicted in Figure 8. above; these modules are connected via connections and relations.

H. TIMING SIMULATION

We deployed two Digilent Nexys4 DDR Artix-7 FPGA devices to test the hardware implementation of the SHC encryption and decryption algorithm. One of these two Artix-7 FPGA devices is used for information transmission, and the other is for information reception. The Universal Asynchronous Receiver Transmitter (UART) module is used for information transmission and reception. A full-duplex asynchronous system that can connect with the physical ethernet card attached to our Artix-7 FPGA board is provided by a UART module. The sending and receiving pins are set up to deliver feedback on the link status and data activity (DigilentInc, 2014).

To initiate the connection, the UART module must be configured with several parameters, including Baud Rate, Number of Data Bits, Parity Check, Stop Bits, and Endianness. After conversion, the UART module delivers digitally encrypted data to embedded hardware or wireless sensors. The embedded devices or wireless sensors at the receiving station will receive encrypted data and send it to the UART module for additional processing. The UART module's baud rate generator enables information to be received and delivered back to the SHC decryption algorithm for deciphering the original statement given by a sender. UART sequentially transfers data using bytes of digital information, enabling error-free data processing. The UART module will set a flag indicating that new information is available during the transfer of data from the receiver. It may also cause a CPU interrupt to request that the host processor transmit the data it has just received.

Figure 9 shows the suggested SHC algorithm in an experimental configuration for real-time transmission utilizing a Digilent Nexys4 DDR Artix-7 FPGA board.

In this work, the bit-error-rate (BER) checker, which is used to examine errors that occurred during the transmission of information, is initiated by the FPGA platform in the transmitting block. To start a BER test, the receiving station also retains data from the transmitter. Information won't be decrypted if there is an intertwining event in the data; instead, an error counter will be increased. To prevent timing errors, ChipScope is utilized to collect encrypted data as it is being transmitted at a maximum clock frequency of 100 MHz. ChipScope is also a logic analyzer to record the programming logic in VHDL or Verilog coding meticulously. Integrated Bus Analyzer (IBA) and Integrated Logic Analyzer (ILA) are used to capture the internal signals and internal buses, respectively (IBA). To examine the internal movement created during information transmission and reception and the programming logic, we employed the 64-bit Xilinx ISE Design Suite ChipScope. This utility enables us to interact with numerous Artix-7 FPGA device components.

V. RESULTS AND FINDINGS

A. SIMULATION RESULTS

First, the proposed architecture of the SHC algorithm powered with a compact S-box was implemented using a Digilent-powered Artix-7 FPGA development board. The proposed design is described using the Verilog HDL. The development system is Xilinx Design tool Vivado 2022 edition. The overall implementation of our proposed lightweight block cipher

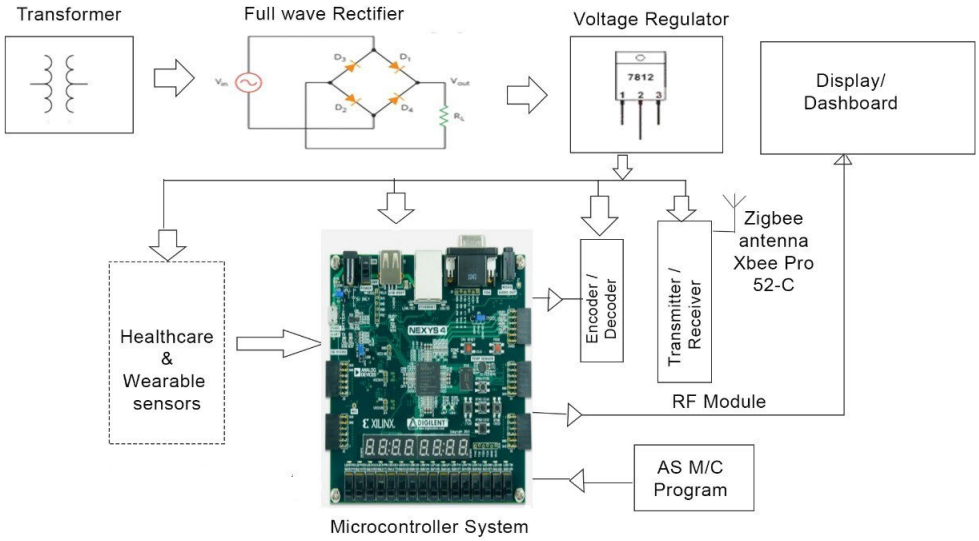


FIGURE 9. Proof of concept using SHC-64.

SHC algorithm focuses on the power-constrained device and embedded systems. For various application situations, specific requests exist on the usage and implementation objectives. The section describes the findings and implementation results of the SHC algorithm for low-cost hardware implementation using the Artix-7 FPGA development board. The FPGA execution of a low-cost SHC cryptographic algorithm module. We have employed SHC-encryption and SHC-decryption cores. For each, we explored our design approaches regarding look-up tables (LUTs) for our compact S-box components embedded with CFA technology. The SHC method was implemented in hardware, which was used to generate the experimental results. With the aid of hardware implementation, the overall logical and structural architecture can be broken down into smaller, more manageable pieces. Synthesis and automation on VHDL and RTL are used to test and verify these components (Register-transfer Level) automatically. We used VHDL to create a synchronous digital circuit with data-signal-based logic operations and information flow across hardware registers. To determine how well our SHC encryption algorithm runs on low-power devices, we have implemented it on the Digilent Nexys Artix-7 FPGA development board.

B. HARDWARE FOOTPRINT AREA OF S-BOX

Our SHC cryptographic implementation has 32 IOBs (Input-Output Blocks), as shown in Figure 10. The interface of our SHC algorithm emphatically relies upon the target application by utilizing extra I/O pins for a parallel key input. We did not code the key inside the cryptographic module to minimize the overhead of control logic. SHC algorithm is an autonomous cryptographic module where the key is provided remotely. From that point of view, our hardware implementation decision offers the best adaptability compared to standard AES, Blowfish, and IDEA symmetric cryptosystems. The Digilent Nexys4 DDR Artix-7 provides high compatibility with Xilinx® Vivado® design suite, which also includes ChipScope™, so our design can be implemented with no extra cost.

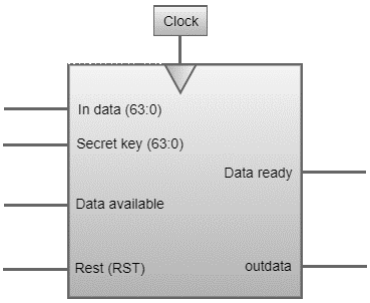


FIGURE 10. I/O Interface of SHC-64 Cryptosystem.

C. HARDWARE FOOTPRINT AREA OF SHC CIPHER

The entire SHC algorithm control logic is precisely designed using a finite state machine over starting, intermediate, and final states, as shown in Figure 11. The operation of the SHC algorithm begins with resetting the interface to initiate the first round of the Feistel function with two inputs of the SHC algorithm, the plaintext of 64-bit and secret key of 64-bit, which was pre-computed with the help of a compact s-box and 32-bit p-array, the compact s-box data are read from pipelined registers. In contrast, the other data is read from equivalent registers. The SHC uses a 64-bit multiplexer which selects appropriate input plaintext and corresponding secret key in the initial round of the Feistel function. *D* flip flops are used for SHC-64 bit to synchronize the key schedule's output and a round function's output. The compact S-box runs parallel with the Feistel function to generate complex ciphertext in each round. The output of 1st round acts as

input to the next round, and it continues till the execution of 14 straight rounds to generate the complete ciphertext. The key schedule works with the Feistel function for each round $1 \leq j \leq 14$

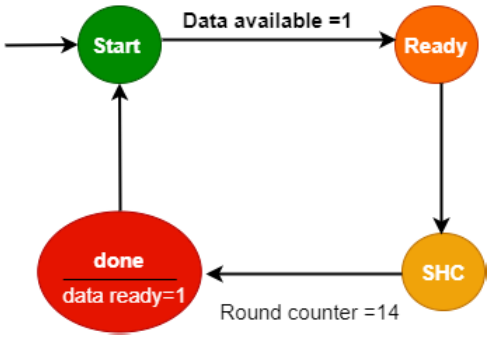


FIGURE 11. Finite state machine for SHC-64 Cryptosystem.

The execution of permutation and substitution of bits are remarkably clear in hardware, which is simple wiring of input bits. The non-linear SHC S-box module is implemented using composite field arithmetic (CFA) technology which consumes less hardware footprint when compared with conventional S-box. A parallel pipelined implementation of the S-box achieves high performance without destroying extra overhead implementation. The S-box, fueled by CFA, is the lightweight block cipher's main strength in the SHC. An inverter allows us to offer a new H function that operates at a very high speed. The H function receives the result of the F function as an input and inverts the 32 bits immediately to generate new output. The decryption process of the SHC algorithm is an inverse operation of its encryption process. The decryption process begins with swapping extra p array keys. which will continue for 14 rounds to generate original plaintext information. The low-power SHC processor was implemented over Artix-7 to handle encryption and decryption processes along with block-level parallel execution of compact S-box and inverse S-box which runs parallel with SHC's Feistel structure.

The hardware implementation of the SHC cryptosystem was executed on Artix^{TM}-7 FPGA development board bearing code number Xilinx® XC7A100TCSG324-1, as shown in Figure 12.

The hardware implementation of the Simple Hybrid Cipher (SHC-64) on the Xilinx-powered Artix-7 FPGA development board demonstrates a remarkable blend of efficiency and performance, particularly when compared to the Advanced Encryption Standard (AES) and other block ciphers. The SHC-64 requires only 949 Look-Up Tables (LUTs) for its implementation, showcasing a significantly smaller hardware footprint compared to AES, which is known to require more extensive resources for similar levels of security. Furthermore, SHC-64 achieves a maximum operating frequency of 515.995 MHz, indicating superior processing speed which is crucial for real-time encryption tasks in resource-constrained environments such as RFID tags and Wireless Sensor Networks (WSN).

The simulated experimental results were carried out using Xilinx Vivado 2022 edition on Microsoft Windows 10 operating system-powered laptops with 64-bit architecture on Intel® Processors. Table 1 shows the detailed specification of the core processor used in this research.

TABLE 1. System specifications used in the implementation of the SHC cryptosystem.

System Specifications		
Processor Name	Intel Core i5-10505	Intel Core i7-11700K
Processor Number	i5-10505	i7-11700K
Lithography	14 nm	14 nm
Total Cores	6	8
Total Threads	12	16
Turbo Boost Frequency	4.60 GHz	4.90 GHz
Processor Base Frequency	3.20 GHz	3.60 GHz
Cache	12 MB	16 MB
Bus Speed	8 GT/s	8 GT/s
TDP	65 W	125 W
Memory Types	DDR4-2666	DDR4-3200
Max Memory Bandwidth	41.6 GB/s	50 GB/s
Laptop RAM	12 GB	12 GB
SSD	500 GB	500 GB
Graphics	Nvidia	Nvidia

The SHC cryptosystem was implemented on the Digilent Artix-7 FPGA development board and the most potent experimental board used explicitly for research and development projects. The board is manufactured to deliver high performance. For low-cost cryptosystem implementation, the board comprises more GPIO (general purpose input-output pins) to help us connect multiple healthcare sensor devices to exchange secure communication between sensors and IoT gateways through the FPGA controller. Table 2 shows device specifications that allow the cryptographic operation to perform on the fly because the key can reside in inbuilt RAM bits of the Artix-7 FPGA board.

Table 3 outlines the hardware performance of our proposed SHC cryptosystem using an Artix-7 FPGA board. The performance metrics were achieved after successful synthesis on Xilinx Vivado 2022 edition. The entire simulation and hardware programming involves several function executions, including post place, post map, and post route mapping timing report. The development board of the family Artix-7 FPGA is specifically used to minimize power consumption and deliver high-end performance with minimal implementation cost.

The proposed implementation of the SHC cryptosystem with a compact S-box is enacted using the CFA mechanism. The pipelined registers of the Artix-7 FPGA device help us map the substitution box's arithmetic operations to execute its operations at a very high speed to gain optimal performance with minimal LUT consumption. The overall LUT consumption is less than the standard symmetric cryp-

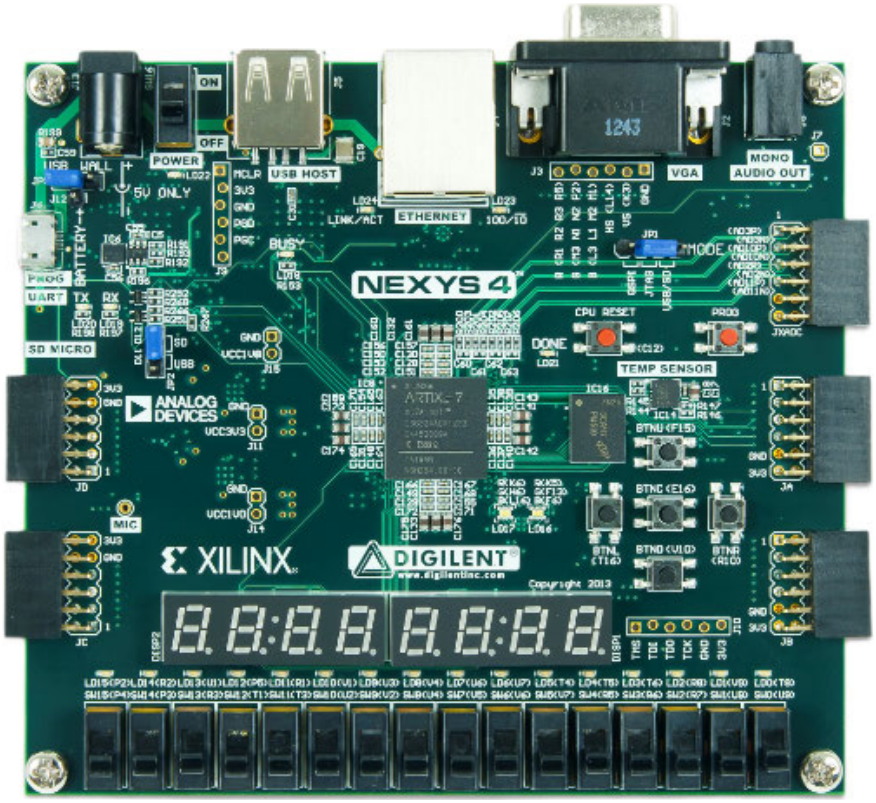


FIGURE 12. Digilent Artix-7 FPGA development board for SHC cryptosystem.

tosystem implemented on the Artix-7 FPGA development board.

D. DEVICE UTILIZATION OF SHC CIPHER

Table 4 illustrates the overall resource utilization achieved through the hardware implementation of the SHC cryptosystem. The proposed SHC implementation has a hardware footprint area cost equivalent to 1.49% of the total hardware resources available on the Artix-7 FPGA development board. The total slices utilized for logic are 949, which is 1.49% of the available 63400 LUTs.

E. COMPARATIVE ANALYSIS OF LIGHTWEIGHT CIPHERS

The Xilinx Vivado IDE's timing analysis of the SHC cipher allows for simple verification of clocks, constant clocks, pulse width clocks, unconstrained internal endpoints, delays, multiple clocks, generated clocks, loops, partial input delay, partial output delay, latch loops. Table 5 shows that the IP Integrator is the first step in Xilinx Vivado's initialization, simulating, analyzing, synthesizing, analyzing, and implementing are all steps along the way of creating a bitstream. To continue the analysis, the generated bit file is read by specialist software, which then uses it to put a script file (also a bit file) into the required FPGA hardware. In our specific circumstance, we generated using Xilinx Vivado 2022 and programmed them into dedicated hardware, i.e.

TABLE 2. Digilent Artix-7 FPGA board specifications.

FPGA Device	Specifications
FPGA Board	Digilent Artix-7 FPGA
Code	Xilinx XC7A100TCSG324-1
1 MSPS On-chip ADC	Yes
Logic Cells	101,440
Logic Slices	15,850
DDR2 (MiB)	128
Flip-flops	65,200
Block RAM (Kbits)	4,860
DSP Slices	240
GTP 6.6Gb/s Transceivers	8
I/O Pins	300
Distributed RAM	1188
GPIO	Yes
XADC Blocks	1
Maximum Temperature	125°C
Programming	JTAG, USB FLASH
LEDs (Output)	16
Switches (Input)	16

Artix 7 FPGA development board, using the Digilent Adept desktop application.

F. PERFORMANCE COMPARISON OF SHC, AES, AND BLOWFISH CIPHER

The execution of encryption and decryption using the SHC algorithm was carried out on an Artix-7 FPGA development board from the Xilinx family bearing the model number XC7A100TCSG324. The SHC-64 accepts an input of 64 bits,

TABLE 3. Performance of SHC cryptosystem using Artix-7 FPGA development board.

Hardware Profiles	Performance metrics
Symmetric Cipher	*SHC*
Architecture	LWC
Block Size	64-bit
Key Size	128-bit
Number of Rounds	14
Structure	Feistel Network
FPGA Board	Digilent Nexys Artix-7
Device Family	XC7A100TCSG324
Package	Xilinx CGS324
Speed	-1
IDE	Xilinx Vivado 2022.2
The operated Design Mode used	RTL
Maximum Operating Frequency	515.995 MHz
Number of Slices-M	134
Number of Slice-L	205
Number of LUTs	949
Number of bonded IOBs	32
Number of GCLKs	2
Latency	28
Throughput	2.362 (Gbps)
Critical Path Delay	1.938 ns
Power Utilization	2.56012 mW
Efficiency(Throughput/Slices)	6.96

TABLE 4. Artix-7 FPGA device utilization for SHC-64.

Constraints	Available	Used	Percentage
Slices LUTs	63400	949	1.49 %
F7 MUX	31700	81	0.25 %
F8 MUX	15850	19	0.11 %
Bonded IOB	210	32	15.24 %
Slices-L	15850	205	1.29 %
Slices-M	15850	134	0.84 %

and out of those 64 bits, 16 bits (LSB bits) were supplied by using the 16 input switches available on the FPGA device.

The ON state of the switch is analogous to the binary number 1, while the OFF state of the switch is analogous to the binary number 0. Similarly, the 16-bit least significant bit of the 64-bit encrypted output in binary form was displayed on 16 user LEDs pins to display the 16-bit LSB as ciphertext. LEDs light up on the FPGA board to indicate the state of the decryption bits, shown in Figures 13 and 14 below.

G. TIMING ANALYSIS OF SHC CIPHER

The SHC algorithm is best described by its key properties, including its incredibly high-speed operations, lightweight, and compact size. By increasing the number of rounds in the Feistel network, the SHC algorithm can support key lengths ranging from 128 bits to 256 bits. Applications such as wireless sensor networks (WSN), ubiquitous computing, the Internet of Things (IoT), RFID tags, and aggregation networks are excellent candidates for their use. Any file and any size of the file can be encrypted using an application that is used to deploy the software implementation of the SHC cryptosystem.

TABLE 5. Timing analysis of SHC cipher using Xilinx Vivado 2022 edition.

Constraints	Elapsed Time
XDC timing constraints	00:00:20
Logic Optimization (Retarget)	00:00:00.382
Constant Propagation (to load pins)	00:00:00.433
Sweep	00:00:00.492
BUFG optimization	00:00:00.620
Shift Register Optimization	00:00:00.637
Power Optimization	00:00:00.070
Netlist Obfuscation Task	00:00:00.003
XDEF routing	00:00:00.134
Global Placement Core	00:00:05
Area Swap Optimization	00:00:05
Pipeline Register Optimization	00:00:05
Delay and Skew Optimization	00:01:37
Constraint Validation Runtime	00:00:01
Write Bitstream	00:00:36
Global Vertical Routing Utilization	0.19872 %
Global Horizontal Routing Utilization	0.186346 %

The OpenMP module that came pre-installed on the professional edition of Microsoft Visual Studio 2022 was used to complete the SHC implementation process. In addition, we have used a variety of file types, including Text (TXT), Joint Photographic Expert Group (JPEG), Zip Compressed (ZIP), Document (DOC), Portable Document Format (PDF), Audio (MP3), and Video (MP4) files, each of which has a unique file size. To generate the encrypted version of a file, the application used for file encryption for SHC serial and SHC parallel accepts any file of any type as input. It requires a static secret key for the encryption process. The SHC-Parallel version was put through its paces on 6-core and 8-core CPUs while being tested with an encryption software application. To achieve high performance, the parallel design of the SHC algorithm was carefully constructed with the help of the OpenMP architecture. Additionally, the source code was optimized with the loop unrolling technique. The SHC-Sequential algorithm is contrasted with its parallel counterpart, the SHC-Parallel algorithm, regarding the encrypted file. Table 6 and 7 illustrates the difference in the amount of time required to execute the SHC-64 cryptosystem with another standard block cipher like AES and Blowfish algorithm.

H. POWER UTILIZATION ANALYSIS OF SHC CIPHER

A significant number of researchers have successfully implemented lightweight block ciphers over FPGA. A good number of them concentrated on achieving the highest possible level of performance. In contrast, others attempted to optimize their designs for the least amount of money, space, and power consumption. Additionally, lightweight block ciphers were created expressly only for hardware applications. We evaluate the performance of our SHC algorithm implementation compared to the version of other lightweight block ciphers based on FPGA implementations. Table 8 demonstrates that implementing a lightweight block cipher for low-cost FPGA cores, SHC-64, offers the smallest hardware footprint area for performance and the

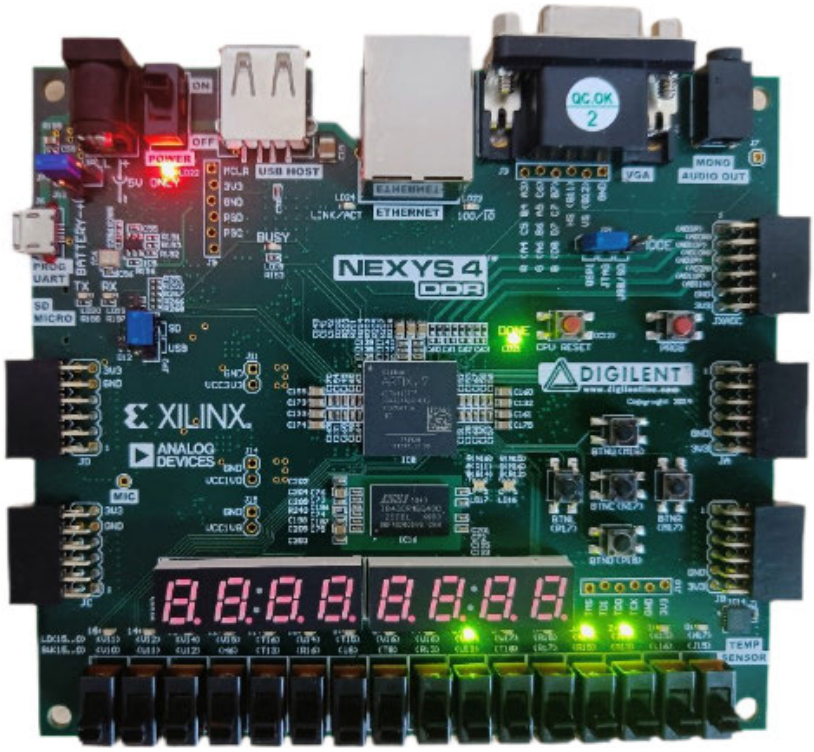


FIGURE 13. LSB input provided using switches on Artix-7 FPGA development board for encrypting information.

highest hardware efficiency compared with standard block ciphers such as AES, PRESENT, HIGHT, and others. This is demonstrated by the fact that the hardware footprint area for implementing SHC-64 is the smallest. The complexity of the SHC encryption algorithm, which operates on a fixed block size and takes approximately the same amount of time for input, is written as O(1). Our SHC encryption algorithm typically has a time complexity of O(m), where m is the message size block of data that needs to be encrypted. This time complexity can vary depending on the mode of operation. The results of the experiments show that our SHC encryption algorithm has a high-performance level compared with the AES algorithm in terms of the efficiency of the hardware, specifically throughputs and area cost. The performance narrative is based on the throughput rate, measured in inputs, outputs, and iterations.

I. CORRELATION AND ENTROPY USING SHC CIPHER

Our hardware version of the SHC encryption algorithm was far superior in terms of latency, throughput, area, and power consumption. This was something that we were able to find. The SHC algorithm enables data processing at high electronic speeds while maintaining an adequate level of security. Additionally, we have analyzed the power consumption of several lightweight blocks based on the FPGA design. For each block, the overall amount of power consumption is computed. The Xilinx Vivado 2022.3 edition

TABLE 6. Comparative analysis of various lightweight ciphers on Intel Core i5 CPU-based system.

Tested on Intel Core i5-10505		SHC Serial	SHC Parallel	Execution time (ms)	
Key Used: SHC64CIPHER					
File Type	File Size	Single Core	6 Core	AES (serial)	Blowfish (serial)
TEXT	102 KB	0.02856	0.0238	0.09710	0.0714
JPEG	324 KB	0.08996	0.07496	0.30586	0.2249
ZIP	866 KB	0.14326	0.11938	0.48708	0.3581
WORD	224 KB	0.07558	0.06298	0.25697	0.1889
PDF	512 KB	0.09255	0.07712	0.31467	0.2313
AUDIO	1.04 MB	0.23549	0.19624	0.80066	0.5887
VIDEO	2.56 MB	0.56696	0.47246	1.92766	1.4174

can automatically estimate the power consumption of the encryption and decryption processes of the SHC algorithm. Compared to the AES and BLOWFISH algorithms, the power consumption of the SHC algorithm, which is 6.82 mW, is significantly lower. AES consumes 18.5 mW, CLEFIA consumes 13.7 mW, and Blowfish consumes 29.86 mW. Figure 15 shows the block-level RTL schematic diagram of the SHC hardware implementation on the Artix-7 FPGA development board.

VI. SECURITY ANALYSIS

The method proposed was planned with simple but logically operational solid measures. The security threat review of the proposed solution is explained below, explaining the protection mechanism against different attacks.

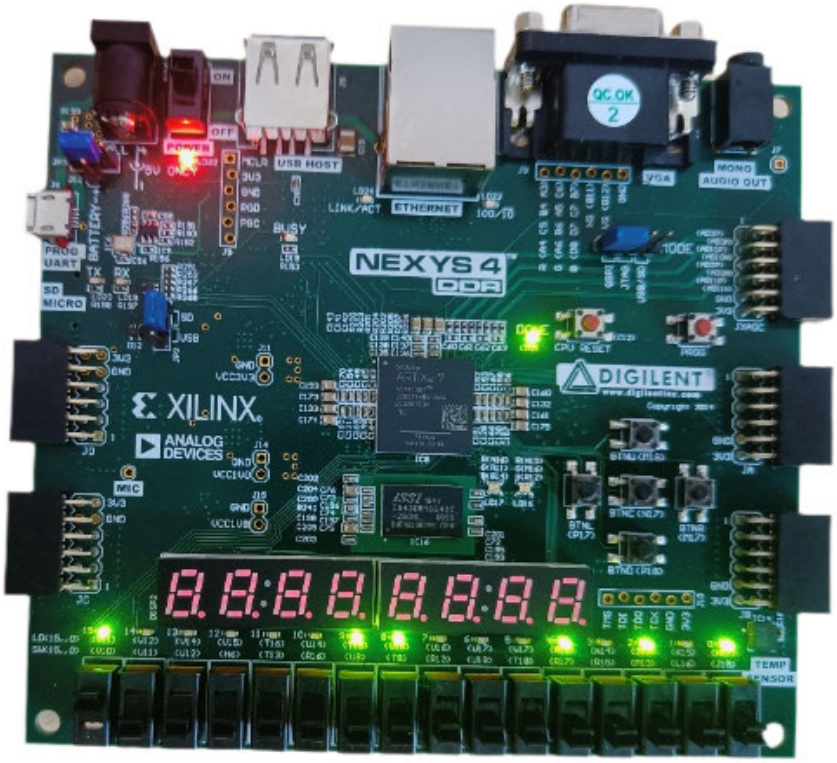


FIGURE 14. The SHC-64 encryption process executed on Artix-7 FPGA development board.

TABLE 7. Comparative analysis of various lightweight ciphers on Intel Core i7 CPU-based system.

Tested on Intel Core i7-11700		SHC Serial	SHC Parallel	Execution time (ms)	
Key Used: SHC64 CIPHER					
File Type	File Size	Single Core	8 Core	AES (serial)	Blowfish (serial)
TEXT	102 KB	0.02753	0.02238	0.0936	0.0688
JPEG	324 KB	0.08879	0.07218	0.3018	0.2219
ZIP	866 KB	0.13998	0.11380	0.4759	0.3499
WORD	224 KB	0.07558	0.06144	0.2569	0.1889
PDF	512 KB	0.09145	0.07434	0.3109	0.2286
AUDIO	1.04 MB	0.22959	0.18665	0.7806	0.5739
VIDEO	2.56 MB	0.55121	0.44813	1.8741	1.3780

A. LINEAR CRYPTANALYSIS

This form of attack is referred to as a plaintext attack. In this case, the attacker knows some unknown plaintext and its ciphertext. The primary purpose is to retrieve the Key that is used during encryption. An attacker uses the Equation to create a linear link between the plaintext and its respective ciphertext. to determine the likelihood of satisfying the equation [29], [30].

$$a_1 \bigoplus a_2 \bigoplus \ldots . a_n \bigoplus b_1 \bigoplus b_2 \bigoplus \ldots b_n = 0 \quad (1)$$

Equation (1) applies to an encryption method for the plaintext where the number of bits is greater than 1 and less than 2n. If the cipher is not protected inside this text, then the cipher would be regarded as weak and attackable. The difference in probability from values 1 to 2 is called a bias. A bias value close to 1 in 2 represents improved linear attack security. For most linear terms, the linear likelihood is precisely 1 to 2, and the bias value is (1 to 2) – (1 to 2) = 0. Therefore, the proposed technique protects against the linear attack (known as the known-plaintext attack).

B. DIFFERENTIAL CRYPTANALYSIS

This form of attack is referred to as a plaintext attack identified. This attack focuses on the high likelihood of output differences concerning a particular input difference [29], [30]. For example, X is a set of bits of the proposed S-box input plaintext such as X = [x1, x2…xn], and Y is the corresponding set of ciphertext bits such as Y = [y1, y2…yn]. The discrepancy between the two complaint texts is shown as α = x1 to x2, and β = y1 to y2 reflects the difference between their respective ciphertexts. For a given input difference, β is indicated as differential pair (α, β). Suppose β for the given α is a distinctive form of attack. The difference pair probability must be reduced to protect the cipher under differential attack or ciphertext assault. Several combinations of 16-bit 'α' were checked for the corresponding β values with different 16-bit plaintexts.

SHC-64's S-Box design is a critical component in its defense against linear and differential cryptanalysis. The composite field arithmetic (CFA) used in the S-Box construction enhances non-linearity and reduces differential uniformity, making it difficult for attackers to establish linear

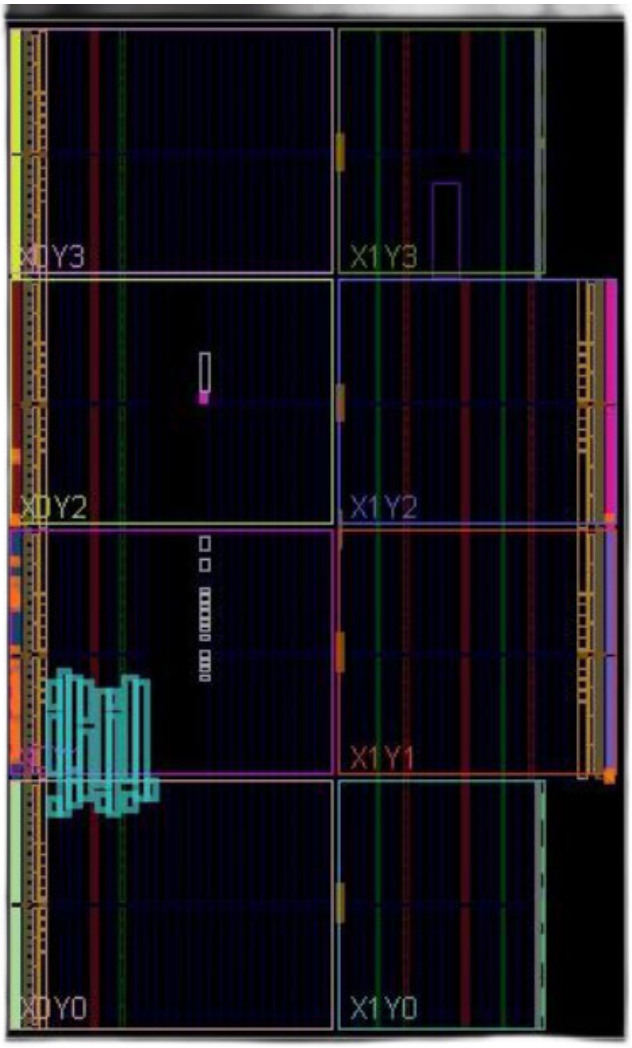


FIGURE 15. RTL Schematic of SHC-64 cipher.

or differential characteristics that could be exploited in these attacks.

Consequently, the difference pair (α, β) was observed for the proposed method. The maximum likelihood of differential observed is 9 to 216. However, the most satisfactory probability of differentiation is 2 bis 216, which is very acceptable for any encryption technique. Therefore, it is possible to conclude that the algorithm proposed is safe from differential attack.

C. BICLIQUE ATTACK

It is a variant of the cryptanalysis meet-in-the-middle (MITM) method. A biclique structure is used to maximize the number of rounds the MITM attack could target. Biclique attack, just like MITM, applies both to cipher blocks and (referred) hash functions. A full AES and a complete IDEA [41] breakdown can be done using a biclique attack but with little gain over brute force. The KASUMI cipher and preimage resistance of the Skein-512 and SHA-2 hash functions were also applied. The only one-key assault on AES is publicly known to attack the maximum number of rounds. Previous known attacks were usually applied with algorithm variants in which rounds were reduced (typically to 7 or 8).

In Biclique Attack Mitigation the structure of SHC-64, including its key scheduling and block cipher design, is constructed to resist biclique attacks, which are advanced forms of meet-in-the-middle (MITM) attacks. By carefully designing the algorithm's internal structure and employing a secure key expansion mechanism, SHC-64 minimizes the effectiveness of biclique attacks, which are known to be potent against several standard encryption algorithms.

D. AVALANCHE EFFECT

A cryptographic security review calculates the best encryption ratio. This research shows that changing even one bit in the plaintext or ciphering key can change the ciphertext significantly. When at least 50% of ciphertext bits expand, it's considered an avalanche. More avalanche implies better protection. Equation (6.5.1). shows the avalanche effect. The hamming distance for a plaintext block was randomly set to 5 bits during observation. Table 3 compares SHC to lightweight encryption algorithms based on test results. Table 6.4 shows the same observation using Key and plaintext. The avalanche effect for plaintext and Key in SHC tests.

$$AE = \frac{Number\ of\ changed\ bit\ in\ ciphertext}{Number\ of\ bits\ in\ ciphertext} \times 100\% \quad (2)$$

In Avalanche Effect the SHC-64 is designed to ensure a strong avalanche effect, where a single bit change in the plaintext or key results in a significant and unpredictable change in the ciphertext. This property is vital for securing against various forms of cryptanalysis, as it ensures that the cipher's output is highly sensitive to its input, thereby obscuring patterns that could be used in cryptanalytic attacks.

E. INTERPOLATION ATTACK

Following the two attacks, several new block ciphers were introduced that proved safe against differential and linear attacks. The KN-Cipher and the SHARK Cipher were developed as iterative block ciphers. The resistance of both the lightweight and standard block ciphers is tested via interpolation attacks. An S-box is an algebraic function used in interpolation operations; this function might be quadratic or polynomial above GF (2^8). An adversary can use the LaGrange interpolation method to define the coefficients without the confidential key information if given the input as a linked plaintext and associated Key over an encryption process. Because our lightweight S-box uses CFA technology, an attack of this kind is improbable.

F. BRUTE FORCE ATTACK

A cryptanalytic attack by a brute force attack can decrypt encrypted data [42], [43], except for the unconditionally secure encryption technique (information-theoretic security). Such an attack may be used if other vulnerabilities in an

TABLE 8. Performance comparison of lightweight cryptosystems.

Algorithm	Block size	FPGA device	Frequency	T'puts	Slices	Efficiency	Reference
SHC	64	Artix-7	515.995	2.362 (Gbps)	339	6.96	*This Chapter*
PRESENT	64	Spartan-III	254	508	271	1.87	Guo et. al., [32],2008
ICEBERG	64	Virtex-II	-	1016	631	1.61	Rouvroy et. al., [33],2004
SEA	64	Virtex-II	145	156	424	0.368	Standaert et. al., [34], 2006
AES	128	Spartan-III	196.1	25,107	17,425	1.44	Good et.al., [35], 2005
HIGHT	64	Spartan-III	163.7	65.48	91	0.72	Yalla et. al., [36],2009
LED	64	Artix-7	378	21.6	37	0.58	Nall et.al., [37],2014
XTEA	64	Spartan-III	62.6	35.77	254	0.14	Kaps et. al., [38], 2008
SIMON	64	Spartan-III	136	3.60	36	0.10	Aysu et. al., [39], 2014
PRINT	64	Spartan-III	147.73	147.7	210	0.703	Matsukawa et. al., [40],2016
PRINT	64	Artix-7	293.51	293.5	139	2.11	Matsukawa et.al., [40], 2016

encryption scheme (if any) cannot be taken advantage of that would facilitate the job. The brute force attacks work by measuring any possible combination of passwords and checking them to see if they are correct. As the length of the password increases, the time and the average computing power needed to find the right password increase exponentially. With rising key sizes, resources needed for a brute attack are growing exponentially, not linearly. While U.S. export regulations traditionally limit the key length to 56-bit symmetrical keys (e.g., the Data Encryption Standard), these limits no longer apply, so modern symmetrical algorithms usually use 128- to 256-bit keys that are computationally more robust.

The choice of 64-bit block size and 128-bit key length in SHC-64, combined with its efficient yet secure cryptographic operations, ensures that brute force attacks are infeasible. The algorithm's design requires computational efforts that grow exponentially with the key size, making it resistant to brute force attempts.

G. WEAK KEYS ATTACK

The weak keys embody a small part of the total room. Hackers can compromise a system by encrypting plaintext data using a randomly generated key. A solid cipher has no weak keys [44]. Complex cipher creation is made possible by the SHC encryption algorithm's linearity, which relies on its CFA technology. In making the ciphertext, SHC does not use the actual Key. Instead, XOR is used to cascade the Key into F. Fixed F-function has a pure nonlinear feature, and it may be made resistant to weak key attacks by choosing the right key.

H. RELATED KEYS ATTACK

An attack in cryptography called "related-key" is when the attacker can watch the cipher work under several different keys whose values are unknown, but where the attacker knows how the keys are related to each other in a certain way. For example, the attacker may figure out that the keys always have the same last 80 bits, even if they have no idea what the bits are. There is no way an attacker could get a cryptographer to encrypt plain text with many different secret keys linked in some way.

In this attack, the attacker has some idea about some mathematical connection (or they try to find it) that connects the keys. An attacker would have a challenging time convincing a human cryptographer to encrypt plaintexts using several secret keys that are somehow linked. Therefore, this model appears impractical at first glance.

Related-key Cryptanalysis implies that the attacker learns the encryption of some plaintexts under the initial unknown Key, K, and some related keys (e.g., $K = g(K)$). As the name suggests, a chosen related-key attack involves a specific method by which the Key will be altered. It's important to remember that the attacker only knows or picks the key relationship, i.e., g(k), but not the key values themselves.

VII. CONCLUSION

In this study, we present SHC-64, a novel, efficient block cipher. There are many different kinds of attacks on RFID systems that are making them vulnerable. The most significant difficulty with RFID systems is providing enough protection against such attacks. Because of the considerable amount of processing power, storage space, and other resources needed for their implementation, the modern encryption algorithms developed for high-end devices are unsuitable for RFID systems. RFID systems have strict limits on the allowed power consumption and the available encrypted store space size. One of the best ways to keep data safe under such conditions is to use lightweight cryptographic methods. Our target was a cipher with the same level of security as ciphers with 64-bit block sizes and 128-bit keys but with a much smaller footprint. Interestingly, SHC-64 shares with many compact block ciphers the exact implementation needs. SHC-64 was created to work well in low-resource settings, such as those seen in RFID tags or other small, pervasive devices. Based on our research into the system's security, we've concluded that SHC-64 is safe to use. To top it all off, SHC-64 only needs 949 LUTs in its area-optimized implementation, which is 72.61 % less space than the NIST-recommended AES encryption technique. For this reason, we consider it to be of theoretical and practical significance. We urge researchers to analyze the SHC-64 algorithm critically, as with all new submissions.

REFERENCES

[1] A. Majumdar, T. Debnath, S. K. Sood, and K. L. Baishnab, "Kyasanur forest disease classification framework using novel extremal optimization tuned neural network in fog computing environment," *J. Med. Syst.*, vol. 42, no. 10, pp. 1–16, Oct. 2018.

[2] A. Majumdar, N. M. Laskar, A. Biswas, S. K. Sood, and K. L. Baishnab, "Energy efficient e-healthcare framework using HWPSO-based clustering approach," *J. Intell. Fuzzy Syst.*, vol. 36, no. 5, pp. 3957–3969, May 2019.

[3] T. K. L. Hui, R. S. Sherratt, and D. D. Sánchez, "Major requirements for building smart homes in smart cities based on Internet of Things technologies," *Future Gener. Comput. Syst.*, vol. 76, pp. 358–369, Nov. 2017.

[4] G. Zhou, Z. Liu, W. Shu, T. Bao, L. Mao, D. Wu, and Feng-Qiu, "Smart savings on private car pooling based on Internet of Vehicles," *J. Intell. Fuzzy Syst.*, vol. 32, no. 5, pp. 3785–3796, Apr. 2017.

[5] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," in *Proc. 9th Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Vienna, Austria, Springer, 2007, pp. 450–466.

[6] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-bit blockcipher CLEFIA," in *Proc. 14th Int. Workshop Fast Softw. Encrypt.*, Springer, 2007, pp. 181–195.

[7] C. De Cannière, O. Dunkelman, and M. Knežević, "KATAN and KTANTAN—A family of small and efficient hardware-oriented block ciphers," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Springer, 2009, pp. 272–288.

[8] J. Guo, T. Peyrin, and A. Poschmann, "The PHOTON family of lightweight hash functions," in *Proc. 31st Annu. Cryptol. Conf. Adv. Cryptol.*, Santa Barbara, CA, USA, Springer, 2011, pp. 222–239.

[9] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "*Piccolo*: An ultra-lightweight blockcipher," in *Proc. 13th Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Nara, Japan, Springer, 2011, pp. 342–357.

[10] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. YalçinC, "PRINCE—A low-latency block cipher for pervasive computing applications," in *Proc. 18th Int. Conf. Theory Appl. Cryptol. Inf. Secur. Adv. Cryptol.*, Beijing, China, Springer, 2012, pp. 208–225.

[11] A. Bogdanov, M. Knežević, G. Leander, D. Toz, K. Varici, and I. Verbauwhede, "SPONGENT: A lightweight hash function," in *Proc. 13th Int. Workshop Cryptograph. Hardw. Embedded Syst.*, vol. 6917, Nara, Japan, Springer, 2011, pp. 312–325.

[12] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The Simon and SPECK lightweight block ciphers," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[13] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D.-G. Lee, "LEA: A 128-bit block cipher for fast encryption on common processors," in *Proc. 14th Int. Workshop Inf. Secur. Appl.*, Jeju Island, South Korea, Springer, 2013, pp. 3–27.

[14] M. R. Albrecht, B. Driessen, E. B. Kavun, G. Leander, C. Paar, and T. Yalçin, "Block ciphers—Focus on the linear layer (feat. PRIDE)," in *Proc. 34th Annu. Cryptol. Conf. Adv. Cryptol.*, Santa Barbara, CA, USA, Springer, 2014, pp. 57–76.

[15] S. Banik, A. Bogdanov, T. Isobe, K. H. Shibutani, H. Hiwatari, T. Akishita, and F. Regazzoni, "Midori: A block cipher for low energy," in *Proc. 21st Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Auckland, New Zealand, Springer, 2015, pp. 411–436.

[16] L. Li, B. Liu, Y. Zhou, and Y. Zou, "SFN: A new lightweight block cipher," *Microprocessors Microsyst.*, vol. 60, pp. 138–150, Jul. 2018.

[17] B. Koo, D. Roh, H. Kim, Y. Jung, D.-G. Lee, and D. Kwon, "CHAM: A family of lightweight block ciphers for resource-constrained devices," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, Springer, 2017, pp. 3–25.

[18] J. Patil, G. Bansod, and K. S. Kant, "LiCi: A new ultra-lightweight block cipher," in *Proc. Int. Conf. Emerg. Trends Innov. (ICEI)*, Feb. 2017, pp. 40–45.

[19] G. Bansod, N. Pisharoty, and A. Patil, "BORON: An ultra-lightweight and low power encryption design for pervasive computing," *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, no. 3, pp. 317–331, Mar. 2017.

[20] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "Gift: A small present: Towards reaching the limit of lightweight encryption," in *Proc. 19th Int. Conf. Cryptogr. Hardw. Embedded Syst.*, Taipei, Taiwan, Springer, 2017, pp. 321–345.

[21] L. Li, B. Liu, and H. Wang, "QTL: A new ultra-lightweight block cipher," *Microprocessors Microsyst.*, vol. 45, pp. 45–55, Aug. 2016.

[22] S. Sadeghi, N. Bagheri, and M. A. Abdelraheem, "Cryptanalysis of reduced QTL block cipher," *Microprocessors Microsyst.*, vol. 52, pp. 34–48, Jul. 2017.

[23] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede, "RECTANGLE: A bit-slice lightweight block cipher suitable for multiple platforms," *Cryptol. ePrint Arch.*, 2014.

[24] A. Chaurasia, V. S. Sharma, C. L. Chowdhary, S. Basheer, and T. R. Gadekallu, "Non-Gaussian traffic modeling for multicore architecture using wavelet based rosenblatt process," *IEEE Access*, vol. 11, pp. 38523–38533, 2023.

[25] F. Karakoç, H. Demirci, and A. E. Harmancı, "AKF: A key alternating feistel scheme for lightweight cipher designs," *Inf. Process. Lett.*, vol. 115, no. 2, pp. 359–367, Feb. 2015.

[26] F. Karakoç, H. Demirci, and A. E. Harmanci, "ITUbee: A software oriented lightweight block cipher," in *Proc. 2nd Int. Workshop Lightweight Cryptogr. Secur. Privacy*, Gebze, Turkey, Springer, 2013, pp. 16–27.

[27] G. Yang, B. Zhu, V. Suder, M. D. Aagaard, and G. Gong, "The simeck family of lightweight block ciphers," in *Proc. 17th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Springer, Sep. 2015, pp. 307–329.

[28] V. Nalla, R. A. Sahu, and V. Saraswat, "Differential fault attack on SIMECK," in *Proc. 3rd Workshop Cryptography Secur. Comput. Syst.*, Jan. 2016, pp. 45–48.

[29] A. Majumdar, A. Biswas, K. L. Baishnab, and S. K. Sood, "DNA based cloud storage security framework using fuzzy decision making technique," *KSII Trans. Internet Inf. Syst.*, vol. 13, no. 7, 2019.

[30] H. M. Heys, "A tutorial on linear and differential cryptanalysis," *Cryptologia*, vol. 26, no. 3, pp. 189–221, Jul. 2002.

[31] S. Khan, M. S. Ibrahim, M. Ebrahim, and H. Amjad, "FPGA implementation of secure force (64-bit) low complexity encryption algorithm," *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 12, pp. 60–69, Nov. 2015.

[32] X. Guo, Z. Chen, and P. Schaumont, "Energy and performance evaluation of an FPGA-based SoC platform with AES and PRESENT coprocessors," in *Proc. 8th Int. Workshop Embedded Comput. Syst., Archit., Model., Simul.*, Samos, Greece, Springer, 2008, pp. 106–115.

[33] G. Rouvroy, F.-X. Standaert, J.-J. Quisquater, and J.-D. Legat, "Compact and efficient encryption/decryption module for FPGA implementation of the AES rijndael very well suited for small embedded applications," in *Proc. Int. Conf. Inf. Technol., Coding Comput.*, vol. 2, 2004, pp. 583–587.

[34] F. X. Standaert, G. Piret, N. Gershenfeld, and J. J. Quisquater, "SEA: A scalable encryption algorithm for small embedded applications," in *Proc. 7th IFIP Smart Card Res. Adv. Appl.*, Tarragona, Spain, Springer, 2006, pp. 222–236.

[35] T. Good and M. Benaissa, "AES on FPGA from the fastest to the smallest," in *Proc. 7th Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Springer, 2005, pp. 427–440.

[36] P. Yalla and J.-P. Kaps, "Lightweight cryptography for FPGAs," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, Dec. 2009, pp. 225–230.

[37] N. N. Anandakumar, T. Peyrin, and A. Poschmann, "A very compact FPGA implementation of LED and PHOTON," in *Proc. 15th Int. Conf. Cryptol. Prog. Cryptol.*, New Delhi, India, Springer, 2014, pp. 304–321.

[38] J. P. Kaps, "Chai-tea, cryptographic hardware implementations of XTEA," in *Proc. 9th Int. Conf. Cryptol.*, Springer, Dec. 2008, pp. 363–375.

[39] A. Aysu, E. Gulcan, and P. Schaumont, "SIMON says: Break area records of block ciphers on FPGAs," *IEEE Embedded Syst. Lett.*, vol. 6, no. 2, pp. 37–40, Jun. 2014.

[40] T. Matsukawa, T. Okabe, E. Suzuki, and Y. Sato, "Hierarchical Gaussian descriptor for person re-identification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1363–1372.

[41] D. Khovratovich, G. Leurent, and C. Rechberger, "Narrow-bicliques: Cryptanalysis of full IDEA," in *Proc. 31st Annu. Int. Conf. Theory Appl. Cryptograph. Techn. Adv. Cryptol.*, Cambridge, U.K., Springer, 2012, pp. 392–410.

[42] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*, Springer, 2009.

[43] S. Kumar Sharma, A. Chaurasia, V. S. Sharma, C. L. Chowdhary, and S. Basheer, "GEMM, a genetic engineering-based mutual model for resource allocation of grid computing," *IEEE Access*, vol. 11, pp. 128537–128548, 2023.

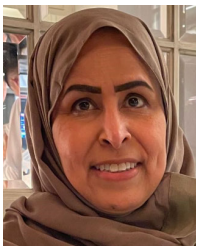
[44] J. Daemen, "Cipher and hash function design strategies based on linear and differential cryptanalysis," Ph.D. dissertation, KU Leuven, Leuven, Belgium, 1995.

SUNIL KUMAR received the bachelor's degree in computer engineering and the master's degree in computer engineering from RGPV University, in 2009 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering, National Institute of Technology Jamshedpur. His research interests include lightweight cryptography, the cloud-IoT security, machine learning, and social network analysis. He has been serving as a Reviewer for *IJISP* journal (IGI Global) for more than 20 research articles.

DILIP KUMAR received the B.Tech. degree in CSE from BIT Sindri, Jharkhand, the M.Tech. degree in computer science from NIT Rourkela, and the Ph.D. degree from the National Institute of Technology (NIT) Jamshedpur, India. He is currently an Assistant Professor with NIT Jamshedpur, India. His research experience is around 23 years. His research interests include optimization techniques, heuristic techniques, machine learning, the IoT, and cloud computing.

HEMRAJ LAMKUCHE received the degrees from North Maharashtra University and Bharathiar University, and the Ph.D. degree from Symbiosis International University. With a strong foundation in mathematics and computer science, he completed his Ph.D. degree. He is currently a Distinguished Cryptographer with more than 11 years in the field, made pioneering contributions to cybersecurity. His research, marked by numerous international publications and patents, focuses on designing secure protocols, considering practical constraints, and privacy. His expertise spans cryptography, VAPT, information security, digital twins, and blockchain technology. He is proficient in solidity, Python, and C++. Recognized for his training programs for armed forces and civil services, his innovative work shapes modern encryption techniques, influencing the next generation in cryptography. He is a member of IACR, Nvidia CUDA Developer, and holds IEEE professional membership.

VIJAY SHANKAR SHARMA received the B.E., M.E., and Ph.D. degrees from the MBM Engineering College, Jodhpur, which one of the oldest engineering college of India. He is currently an Assistant Professor (Senior Scale) with the Department of Computer and Communication Engineering, Manipal University Jaipur, India. He has teaching experience of more than ten years. He has been published 19 research papers in international and national journals/conferences. His research interests include networking and simulation, AI, ML, big data analytics, Hadoop, and the theory of computation.



HEND KHALID ALKAHTANI (Member, IEEE) received the Bachelor of Science degree in computer science from the School of Engineering and Applied Science, The George Washington University, in 1992, the Master of Science degree with concentration in information management from the Department of Engineering Management, The George Washington University, in 1993, and the Ph.D. degree in information security from the Department of Computer Science, Loughborough University, in 2018. She is currently pursuing the Ph.D. degree with the Information Systems Department, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University (PNU). She is also an Associate Professor with the Information Systems Department, College of Computer and Information Sciences, PNU. She has 23 years of work experience as a Lecturer, and worked as a Computer Center President and a Statistic Center President in faculty collages. She received an Award from SIDF Academy: Leading Creative Transformation in Critical Time Program, Stanford University, Center for Professional Development.

MUNA ELSADIG (Member, IEEE) received the Ph.D. degree in information technology, network security form Universiti Teknologi PETRONAS (UTP), Malaysia. She is currently an Assistant Professor with the Department of Information System, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University (PNU), Saudi Arabia. Her research interests include blockchain technology, cyber security, healthcare quality, intelligent security systems in the IoT, cloud, and risk assessment. She is a reviewer of many international journals and conferences.

MARIYAM AYSHA BIVI received the B.Sc. degree in computer science from Madurai Kamaraj University, in 1997, the M.C.A. degree from Madras University, in 2000, and the M.Phil. degree in computer science from Periyar University, in 2005. She is currently a Lecturer with the Department of Computer Science, King Khalid University, where she has been, since September 2007. From 2000 to 2007, she was with the Justice Basheer Ahmed Sayeed College for Women, eventually as a Senior Scale Lecturer. She was also a part-time Lecturer with the University of Madras, from 2000 to 2007, an Academic Counsellor with Indira Gandhi National Open University, from 2002 to 2007, and a Guest Lecturer with the Bharathi Women's College, in November 2002. She has published many technical papers in journals and conferences. She worked and contributed in the fields of machine learning, the IoT, and vehicular network. She is also working on machine learning, image processing, data mining, big data, and the IoT.