International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 03, March-2020

Design of a Lightweight Stream Cipher: BOKHARI 256

Prof. M. U. Bokhari, Shabbir Hassan
Aligarh Muslim University

Abstract: **- Symmetric key cryptography is the most commonly used primitive and stream cipher, in particular, meets the requirement of such algorithms. Developing a software-based synchronous stream cipher is reduced to building a pseudo-random sequence generator with specified cryptographic properties. It is pertinent to note that the implementation of such reliable cryptographic primitives is extremely complicated in practice that requires small processing capacity, low volume, low area, and low power consumption. In the recent past, a few lightweight stream ciphers have been implemented for real applications. In this paper, we have proposed software-based synchronous lightweight stream ciphers, BOKHARI 256, mainly aimed for resource-constrained devices such as Radio Frequency Identification Devices (RFID) Tags, Wireless Sensor Node (WSN) and devices with limited processing capabilities, memory, power resources and the limited computational unit, insufficient wireless bandwidth and low ability to communicate and other general-purpose devices such as credit card, smart card, Personal Digital Assistant (PDA), etc. BOKHARI 256 has designed using the feature of some well-known stream ciphers like FRUIT, LIZARD, SOBER and Grain 128 to improve the performance. The cipher BOKHARI 256 uses a 256-bit key and initialization vector (IV) that is known to be safe and can withstand several cryptographic attacks. BOKHARI 256 uses eight variant stage LFSRs and a linear feedback function $\mathrm{f(x)}$ for providing 256-bit security while the internal states and bit permutation use commutation relations of bitstream over $\mathrm{GF}(2^8)$. We used some new design paradigm to fulfill the requirement of a low-cost environment as mentioned above.**

Keywords: - Galois Field, LFSRs, Generator functions, cryptographic attacks, S-boxes.

A. INTRODUCTION

As we know that a shift registers can be easily implemented by computer hardware due to their less duty cycle and synchronous behavior. A linear feedback shift register (LFSR) is used to generate a wide range of pseudo-random sequences is the simplest type of feedback shift register. Because of LFSR linearity, knowing only $\mathbf{2^n}$ output bits, we can test the LFSR using the algorithm given by Mandal, Kalikinkar, Xinxin Fan that generates any output sequence [1]. Klapper and Goresky [2, 3] proposed a new type of pseudo-random binary sequence generator called Feedback with Carry Shift Register (FCSR). FCSR had a shift register, feedback function, and a small amount of memory. In addition to the current memory contents, the register bits are added to the number. The parity bit (depending on $\sum \boldsymbol{mod}(\mathbf{2})$) of the total is fed back into the first cell whereas the first cell is initialized with the higher-order bits depending on the location of $|\lfloor \sum \boldsymbol{mod}(\mathbf{2}) \rfloor|$ are fed back into the first cell and this new memory value is maintained [4, 33]. The key component used in the design of **BOKHARI 256** is shown in Figure I. The key parameters associated with FCSR's is relation integer $'\boldsymbol{q}'$ which is an odd positive integer $\forall\, \boldsymbol{q} \in \mathbb{Z}$. It represents the number of register cells in the structure. Further, assume that $\boldsymbol{q} = \boldsymbol{p^e}\ \forall\, \boldsymbol{e} \geq \mathbf{2}$ and $'\boldsymbol{p}'$ is an odd prime number such that $\mathbf{2}$ is the primitive $\boldsymbol{modulo}(\boldsymbol{p})$, and $\mathbf{2^{p-1} - 1}$ leaves remainder nonzero when is divided by $\boldsymbol{p^2}$, so the maximal run length of the period of this polynomial $'\boldsymbol{p}'$ is found:

$$\nabla(\boldsymbol{q}) = \boldsymbol{p^e} - \boldsymbol{p^{e-1}}$$

The relation between the cardinality of the registers $'\boldsymbol{r}'$ and the connection integer $'\boldsymbol{q}'$ is exponentially depended to one another and is given by $\mathbf{2^r - 1} = \boldsymbol{q}$ and this implies that $\mathbf{log_2}(\boldsymbol{q}+\mathbf{1}) = \boldsymbol{r}.$ Mathematically it is found that for any value of $'\boldsymbol{q}'$, the expression $\mathbf{log_2}(\boldsymbol{q}+\mathbf{1})$ is real, so it is batter to accept the floor of this much amount. Hence the above relation gets expected value $\lfloor \mathbf{log_2}(\boldsymbol{q}+\mathbf{1}) \rfloor = \boldsymbol{r}.$ According to Niederreiter and Harald, the notion of the keystream's 2-adic complexity is also an important measure of the security of a *Stream Cipher* [5], it is possible to increase the 2-adic complexity of LFSR sequences by adding correct Boolean functions in the LFSR's output, but this issue has not received serious attention. Thus the purpose of this research work is to aim at exploring the possibility of designing a *Stream Cipher* model **BOKHARI-256** using LFSRs for *resource-constrained devices* and protocols like Embedded Binary **HTTP**, **WEB-HTTP** and Compressed **HTTP** over **PAN**. *Resource-constrained* are likely to be extended to embedded computing systems implemented within the Internet of Things (IoT).

B. BACKGROUND STUDY

In order to propose a lightweight stream cipher, the following lightweight cryptographic model of the eSTREAM candidates has been studied. Traditional ciphers for stream encryption and pertinent security issues are presented in [29]. As referred in said work, ciphers **Rivest Cipher 4 (RC4)** [30], **A5/1** and **E0** although not completely compromised, but they suffer from serious security issues and should not be used in new applications any more [31, 34].

I.FRUIT

In Fast Software Encryption (FSE), **Vahid Amin Ghafari** and **et al** [6] have presented a new idea for the design of *Stream Ciphers* **FRUIT** with a shorter internal state in 2015. **FRUIT** uses a secret key not only in the initialization phase of the model but also in the key generation phase of the cipher. The Fruit is proven to be much safe and ultralight than Grain. The main building block of the **FRUIT** uses Linear Feedback Shift Register and Non-Linear FSR. The size of LFSR and NFSR is 80 bits and 64 bits long respectively, while the internal state size of the cipher is found to be at least twice as of the security level required to resist the classic cryptographic attacks such as correlation attack, differential attack, side-channel attack, cube attack, time-memory-data trade-off attack [6, 7]. The main purpose behind this proposed model **FRUIT** is to indicate how a secret key can be used to achieve minimal cost in terms of area (Gate Equivalent), efficiency and attack immune. The internal state of the **FRUIT** is based on 43-bit LFSR and 37-bit NFSR, the left counter of 7-bit and an 8 bit right counter. The input vector of the Fruit is an 80-bit secret key and a publicly define initialization vector of 70-bit long. Experimentally it has observed that the maximum number of keystream bits that can be generated from a single key and IV is 2^{43} bit long [8].

II.LIZARD

In 2017, **Matthias Hamann**, **Matthias Krause1** and **et al** [9] have presented a model "**LIZARD**: A Lightweight *Stream Cipher* for Power-constrained Devices". Lizard, a lightweight *Stream Ciphers* for the energy-constrained environment as mentioned above. The hardware performance originates from a complex combination of a Grain-like architecture, a newly proposed design principle for the state initialization of *Stream Ciphers* that offers confirmed security against **TMD** trade-off attacks aimed at key recovery attacks. *Lizard uses 120-bit keys, 64-bit IVs and has a total of 121 bit of internal state.* The authors have also claimed that Lizard provides 80-bit security against key recovery and 60-bit security against any distinguishing attack. It is noted that the Lizard is only a unique **LWC** model that differs from the key length of the design paradigm that claims that the *achieve security level is directly proportional to the key length of the cipher*.

III.SOBER

SOBER is a family of *Stream Ciphers* originally developed by **Greg Rose** and et al. [10] of **QUALCOMM** Australia in 1997. The name is an artificial acronym for the enabled register Seventeen Octet Byte. The cipher was originally intended as a substitute for broken ciphers in cellular telephony. **SOBER** was the first cipher, comprising a 17-byte linear-feedback shift register (LFSR), a form of ruination called stuttering, and a nonlinear filter output. The shift register's specific design turned out to be vulnerable to guess and determine attack [37].

IV.The Grain 128

The **Grain** *Stream Cipher* was designed by **Martin Hell**, **Thomas Johansson1** and **Willi Meier** in 2004. Their design is aimed at hardware environments in which there is very minimal Gate count, energy consumption and memory is there. It is primarily focused on two shift registers and a function of non-linear output. The cipher has the upgrade option that at the cost of additional hardware the speed can be improved [11, 37]. The key size is 80 bits and no attack has been identified faster than exhaustive key search. The complexity and performance of hardware correspond with other hardware-oriented *Stream Ciphers* like **E0** and **A5/1** and are compare acceptable [12]. The **Grain** is a synchronous *Stream Ciphers* that operate on both LFSR and a non-linear filtering function. It adopts a bit-oriented architecture, which is appropriate for constrained hardware implementations that produce 1 bit/cycle. However, it also offers the option to increase its speed by increasing the length of the processing word; Grain provides higher levels of security compared with the well-known **A5/1** and **E0** ciphers while providing small hardware complexity. It requires about **1294 GE** for 1 bit/cycle and **3239 GE** for 16 bits/cycle [13, 14]. A software implementation is reported for the 1 bit/cycle rate that occupies **778** bytes of code and requires **107 366** cycles for initialization and **617** cycles to generate the output.

C. DESIGN SPECIFICATION OF BOKHARI 256

In this section, we have described the design specification and key generation phase of the proposed model **BOKHARI 256**. Design specification of **BOKHARI 256** mainly comprises four part:

I.Design Taxonomy

The cipher **BOKHARI 256** is based on a single bit memory element and XOR operation, primitive polynomial over Galois Field. Figure I represents key components used in this model [35].

II.The S-Boxes of BOKHARI 256

Substitution-boxes (S-box) are implemented as search tables that map $\boldsymbol{m}$ input bits to $\boldsymbol{n}$ output bits. In block ciphers, they are a common and well-studied choice, hence I have used this method of design in the proposed model **BOKHARI 256**. Some *Stream Ciphers* like [15, 16, 17] also uses S-boxes to improve their mapping non-linearity behavior and mask the *plaintext* bits with *ciphertext*. A special type of commutation relation based S-box have used in this model. The sequence obtained from bit permutation box at each clock cycles $\langle \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \ldots\ldots \rangle$ is $\langle \mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{9}, \mathbf{44}, \mathbf{265}, \mathbf{1854}, \ldots\ldots \rangle$ respectively. Since the next term of this sequence is un-predictable, thus it ensures that the next state of the S-boxes can't be determined. Thus the permuted bits are secure underside channel and linear masking attack.

III.Key Generation Phase

In order to generate the key for encrypting the text, first, all the LFSRs are filled with their default seed value and reset to zero. After that, both polynomials $\mathbf{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ are filled with the binary stream equivalent to the polynomial $\mathbf{g}(\boldsymbol{x}) = \boldsymbol{x}^7 + \boldsymbol{x}^5 + \boldsymbol{x}^4 + \boldsymbol{x} + \mathbf{1}$ $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{x}^5 + \boldsymbol{x} + \mathbf{1}$ which are equivalent to the seed $\mathbf{10110010}$ and $\mathbf{100010}$ respectively. At each pass of the outer loop (traverse with variable $'\boldsymbol{i}'$) the value attained by both the S-Boxes is operated by commutation relation CR and this commutated value is further passed to each LFSRs. At each clock's duty cycle, the same process is repeated and at the end of the loop, 256-bit keystream is generated. The complete signature of the proposed algorithm is something like $\boldsymbol{binary\ keyGen}(\boldsymbol{i}, \boldsymbol{LFSR_i}, \mathbf{g}(\boldsymbol{x}), \boldsymbol{h}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}))$ and the Boolean function $\boldsymbol{f}(\boldsymbol{x})$ is defined as:

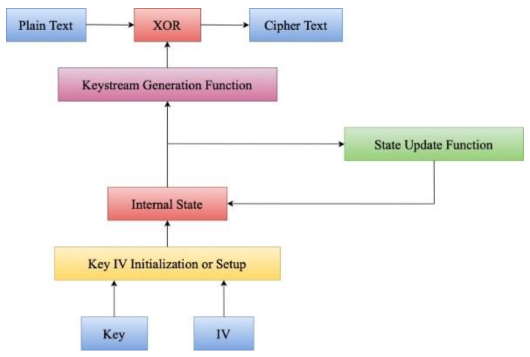

Figure I. Key components of model BOKHARI 256

$$\boxed{\boldsymbol{f}(\boldsymbol{x}) = \left(\boldsymbol{p}^{i\%8}(\boldsymbol{x}) \boxplus \boldsymbol{p}^{(i+1)\%8}(\boldsymbol{x})\right) \boldsymbol{modulo}(\mathbf{2}) \text{ for all value of } \boldsymbol{i} = \mathbf{0} \text{ to } \mathbf{7}}$$

Where the symbol $\boxplus$ represents the multiplication operation of polynomials under Galois Field $\boldsymbol{GF}(\mathbf{2^8})\boldsymbol{modulo}(\mathbf{2})$. The multiplication operation $\boxplus$ is closed under the Field containing $\mathbf{2^8}$ elements.

IV.Key IV Initialization Phase

In this phase, the secret key (of 256 bit long) and the initialization vector (of 256 bit long) are loaded to the model and initialize the cipher to clock eight times starting from $\mathbf{0}$ to $\mathbf{7}$ at each clock pulse, key-stream bits $\boldsymbol{K_i}$ and initialization vector's bit-stream $\boldsymbol{IV_i}$ are produces for all $\mathbf{0} \leq \boldsymbol{i} < \mathbf{256}$. At each phase, $\mathbf{8}$ bits pattern are produced by each $\boldsymbol{LFSR_i}$ thus $\mathbf{8} \times \mathbf{8} = \mathbf{64}$ bit patterns are produced in each clock pulse. So a total of $\mathbf{64} \times \mathbf{8}$ bits pattern produces in a complete trigger comprises $\mathbf{8}$ pulses that take initialization time $\mathbf{2.37}$ cycle per bytes.

D. PROPOSED MODEL BOKHARI 256

The main building blocks of **BOKHARI 256** are LFSR and *Galois Field* $\boldsymbol{GF}(\boldsymbol{q})$, Primitive Polynomials over the Galois Field $\boldsymbol{GF}(\boldsymbol{p^m})$, and a Boolean feedback function $\boldsymbol{f}(\boldsymbol{x})$. The design specification of the cipher is used as a keystream generator that generates a pseudo-random bit sequence which is called **PRNG** [36]. At each clock's duty cycle, 8-bit keystream is **XOR**'ed with an equivalent byte of *plaintext* $'\boldsymbol{P_T}'$ data to produce encrypted *ciphertext* $'\boldsymbol{C_T}'$. Each element $'\boldsymbol{e_i}'$ of $'\boldsymbol{C_T}'$ can be reversed back to $'\boldsymbol{e_j}'$ by using the same keystream with **XOR** operation. It is known that using an $'\boldsymbol{n}'$ dimensional state transition matrix, a $'\boldsymbol{n}'$ stage LFSR can be easily implemented to design *Stream Ciphers*. Such a matrix is called the *LFSR's State Transition Matrix*. We also found that the characteristic polynomial $\boldsymbol{p}(\boldsymbol{x})$ over Galois Field $\boldsymbol{GF}(\boldsymbol{p^m})$ variable $'\boldsymbol{x}'$, the state transition matrix is closely connected to the cycle lengths of the generated PRNG which is equal to $\mathbf{2^m - 1}$ where $'\mathbf{m}'$ is the number of shift registers [18]. In the proposed model **BOKHARI 256,** a new method of linearization for LFSRs, namely the basis Boolean network approach is used [19] to define the output of the dependent variables. The Boolean function used in **BOKHARI 256** over dependents $'\boldsymbol{x}'$ and $\boldsymbol{m} = \mathbf{8}$ is defined as:

$$\boldsymbol{f}(\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_{m-1}}) = (\boldsymbol{x_0} \oplus \boldsymbol{x_1}\boldsymbol{x_2}) \oplus (\boldsymbol{x_1} \oplus \boldsymbol{x_2}\boldsymbol{x_3}) \oplus \dots\dots \oplus (\boldsymbol{x_{m-3}} \oplus \boldsymbol{x_{m-2}}\boldsymbol{x_{m-1}})$$

BOKHARI 256 *Stream Cipher* uses a 256-bit key and 256 initialization vector. It routines on $\mathbf{8}$ LFSRs of degree $\langle \mathbf{77}, \mathbf{81}, \mathbf{61}, \mathbf{24}, \mathbf{111}, \mathbf{33}, \mathbf{126} \rangle$ and $\langle \mathbf{31} \rangle$ whose feedback taps are defined by the primitive polynomials $\boldsymbol{p}(\boldsymbol{x})$ over Galois Field $\boldsymbol{GF}(\mathbf{2^8})$. Two substitution boxes of order **8x16** are used to achieve a proper mixture of bit patterns permutations. **BOKHARI 256** comprises two nonlinear feedback functions $\mathbf{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ that are cyclically connected to each other. The $\boldsymbol{i^{th}}$ state bit at the $\boldsymbol{t^{th}}$ duty cycle of the clock is represented by the letter $(\boldsymbol{C_T})_{\boldsymbol{i}}$ whereas the whole state of the cipher at the same clock is referred $\boldsymbol{C_T}$. The feedback function $\boldsymbol{f}(\boldsymbol{x})$, of all $\mathbf{8}$ LFSRs behave linearly in each clock cycle and the rest two non-linear feedback function $\mathbf{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ behaves nonlinearly for the same clock cycle. For different forthcoming phases of the cipher, the values of these two parameters get change and functionality becomes non-linear. So for different phases, the expression of the nonlinear feedback function changes depending on the values of min and max. LFSRs namely $\boldsymbol{LFSR_0}, \boldsymbol{LFSR_1}$, ... to $\boldsymbol{LFSR_7}$, each of them is associated with different primitive polynomials $\boldsymbol{p}(\boldsymbol{x})$ defined over the Galois Field $\boldsymbol{GF}(\boldsymbol{p^8})$ of degree $\langle \mathbf{77}, \mathbf{81}, \mathbf{61}, \mathbf{24}, \mathbf{111}, \mathbf{33}, \mathbf{126} \rangle$ and $\langle \mathbf{31} \rangle$. Furthermore, an array $\boldsymbol{A}[\mathbf{4}]$ of type integer is declared to specify the parameters for combining the lockup state of the LFSRs. At each clock cycle, the PRNG generates a random integer $\boldsymbol{ran_i}$ such that $\forall\, \boldsymbol{ran_i} < \boldsymbol{q}$ and the $\boldsymbol{GCD}(\boldsymbol{ran_i}, \boldsymbol{p}) = \mathbf{1}$, it means the

$$\begin{aligned}
\boldsymbol{p^{77}}(\boldsymbol{x}) &= \boldsymbol{x^{77}} + \boldsymbol{x^{71}} + \boldsymbol{x^{64}} + \boldsymbol{x^{49}} + \boldsymbol{x^{13}} + \boldsymbol{x^3} + \mathbf{1} \\
\boldsymbol{p^{81}}(\boldsymbol{x}) &= \boldsymbol{x^{81}} + \boldsymbol{x^{70}} + \boldsymbol{x^{52}} + \boldsymbol{x^{51}} + \boldsymbol{x^{40}} + \boldsymbol{x^{37}} + \boldsymbol{x^{35}} + \boldsymbol{x^{11}} + \boldsymbol{x^7} + \boldsymbol{x^3} \\
\boldsymbol{p^{61}}(\boldsymbol{x}) &= \boldsymbol{x^{61}} + \boldsymbol{x^{50}} + \boldsymbol{x^{23}} + \boldsymbol{x^5} + \mathbf{1} \\
\boldsymbol{p^{24}}(\boldsymbol{x}) &= \boldsymbol{x^{24}} + \boldsymbol{x^{23}} + \boldsymbol{x^{17}} + \boldsymbol{x^{16}} + \boldsymbol{x^{11}} + \boldsymbol{x^9} + \boldsymbol{x^5} + \boldsymbol{x^2} + \mathbf{1} \\
\boldsymbol{p^{111}}(\boldsymbol{x}) &= \boldsymbol{x^{111}} + \boldsymbol{x^{90}} + \boldsymbol{x^{52}} + \boldsymbol{x^{44}} + \boldsymbol{x^{36}} + \boldsymbol{x^{13}} + \boldsymbol{x^{11}} + \boldsymbol{x^7} \\
\boldsymbol{p^{33}}(\boldsymbol{x}) &= \boldsymbol{x^{33}} + \boldsymbol{x^{17}} + \boldsymbol{x^5} + \mathbf{1} \\
\boldsymbol{p^{126}}(\boldsymbol{x}) &= \boldsymbol{x^{126}} + \boldsymbol{x^3} + \mathbf{1} \\
\boldsymbol{p^{31}}(\boldsymbol{x}) &= \boldsymbol{x^{31}} + \boldsymbol{x^{30}} + \boldsymbol{x^{23}} + \boldsymbol{x^{16}} + \boldsymbol{x^{12}} + \boldsymbol{x^4} + \boldsymbol{x^3} + \boldsymbol{x^2}
\end{aligned}$$

ordered pair $(\boldsymbol{ran_i}, \boldsymbol{p})$ are co-prime in nature [32]. For every set of LFSRs in the model, we have computed some value $\boldsymbol{A}[\boldsymbol{i}]\ \forall\ \boldsymbol{0} \leq \boldsymbol{i} < \boldsymbol{4}$, such that each $\boldsymbol{A}[\boldsymbol{i}]$ is associated with the set of values are given in Table I.

$$\boldsymbol{\varphi}(\boldsymbol{p^e}) = (\boldsymbol{p}-\boldsymbol{1})\boldsymbol{p^{e-1}} = \boldsymbol{p^e}(\boldsymbol{1}-\frac{\boldsymbol{1}}{\boldsymbol{p}}) \quad \text{and} \quad \boldsymbol{p^e} = \boldsymbol{q} \text{ so, } \boldsymbol{\varphi}(\boldsymbol{p^e}) = \boldsymbol{q}\left(\boldsymbol{1}-\frac{\boldsymbol{1}}{\boldsymbol{q^e}}\right) = \boldsymbol{A}[\boldsymbol{2}]\left(\boldsymbol{1}-\frac{\boldsymbol{1}}{\boldsymbol{A}[\boldsymbol{0}]}\right)$$

After getting the random value (an arbitrary integer) $\boldsymbol{ran_i}$, the initial state (Seed) of the LFSR is computed. Period of the $m-sequence$ cipher generated by the proposed model is represented in Table I. *Period of the LFSR is calculated in MATLAB*, so the average period of each LFSR is 9.65 bits per clock is found. In case of Grain 128 [20], Sober 128 [21], BSF-128 [22, 23], A2U2 [24] and HITAG [25, 26], COZMO [27], FRUIT [8] and LIZARD [28] the average period of $\boldsymbol{m-sequence}$ bits generated by the LFSRs are as follows (refer to Table I).

Table I. List of period achieved by other primitives

Primitives	Period generated by Primitives
Grain 128 [20]	7.35
Sober 128 [21]	8.15
BSF 128 [22, 23]	7.50
A2U2 [24]	5.50
HITAG 2 [25, 26]	8.00
COZMO [27]	9.05
FRUIT [8]	9.00
LIZARD [28]	7.85
BOKHARI 256	**9.65**

I.Associated Data Processing Phase

After the initialization phase of the key's initialization vector gets over, the cipher shifts to the next phase of their production are referred to as the dependent processing step. In this phase, the cipher takes the dependent data as input and updates the state of the cipher without producing output bits. The general procedure of the scenario is something like:

$$\boxed{\boldsymbol{m_i} = (\boldsymbol{MIN_i} \ll \boldsymbol{MAX_i})\ \forall\ \boldsymbol{0} \leq \boldsymbol{i} < \boldsymbol{L}}$$

Where $'\boldsymbol{L}'$ denotes the length of linked data and $'\boldsymbol{m_i}'$ represents the corresponding bits associated with linked data. Update the state in each clocking for $\boldsymbol{i} = \boldsymbol{0}$ to $\boldsymbol{m} + \boldsymbol{255}$. The cipher continuously updates their state in each clock cycle ranging from $\boldsymbol{i} = \boldsymbol{0}$ to $\boldsymbol{m} + \boldsymbol{255}$.

II.Test Vectors of BOKHARI 256

Most *Stream Ciphers* are based on Linear Feedback Shift Registers (LFSRs) design that is less so in software, although they are powerful in hardware. **BOKHARI 256**'s design is based on **8** LFSRs of **8** stages each. For the state list, it uses 256 bytes (two S boxes each of 128 bytes of order **8x16**) ranging from the memory location SBoxL[0] to SBoxL[127] and SBoxR[128] to SBoxR[255], and 32 bits of key memory ranging from $\boldsymbol{K}[\boldsymbol{0}]$ to $\boldsymbol{K}[\boldsymbol{31}]$ and the same initialization vector. The proposed model **BOKHARI-256** has run under the environment of **JDK 1.6** and **JRE** on Windows 8.1 Professional in **MATLAB**, the following simulation results have obtained (refer to Table II). The keystream and IV are represented in binary form while the obtain *ciphertext* is represented in Hexadecimal and Unicode Characters is given in Table II.

Table II. Test vector result of BOKHARI 256

$\boldsymbol{P_T}$	**Generated *Ciphertext* ($\boldsymbol{C_T}$)**	**KEY/IV**
"exampleDem0"	A023243204B23201924B132A001924B105024B232019C320044B14B1132F190021950732	**1101100101000000 1010100101000010**
	& (•┘•┘ ,–(│♀–(♀┼┘0&(•┘–(└¶ •┘	

E. CONCLUSION

It is not a trivial task to provide a fair comparative analysis of the presented ciphers, as they are implemented in different platforms and the comparison metrics' are not directly correlated. Nevertheless, in the case of hardware and software implementation of the proposed model **BOKHARI 256**, the ciphers proposed by other authors also kept a restricted benchmark review. In performance metrics *"number of clock cycles, processing time, throughput, efficiency and bits per cycle "* the *Stream Cipher* **BOKHARI 256** found to be secure and appears to achieve better performance. After doing an extensive attack immune analysis, it has found that the proposed model **BOKHARI 256** can withstand many cryptographic attacks like guess-and-determine attacks, algebraic attack, side-channel attacks, the time-memory trade-off with huge precomputation, correlations attack and linear masking attack. The proposed model can be implemented in a real environment in order to achieve maximal performance.

REFERENCES

[1] Mandal, Kalikinkar, Xinxin Fan, and Guang Gong. "*Design and implementation of warbler family of lightweight pseudorandom number generators for smart devices*." ACM Transactions on Embedded Computing Systems (TECS) 15.1 (2016): 1-28.
[2] Klapper, Andrew, and Mark Goresky. "Feedback shift registers, 2-adic span, and combiners with memory." Journal of Cryptology 10.2 (1997): 111-147.
[3] Goresky, Mark, and Andrew Klapper. Algebraic shift register sequences. Cambridge University Press, 2012.
[4] Guo, Yuqian, et al. "Stability and set stability in distribution of probabilistic Boolean networks." IEEE Transactions on Automatic Control 64.2 (2018): 736-742.
[5] Niederreiter, Harald. "Some computable complexity measures for binary sequences." Sequences and their Applications. Springer, London, 1999. 67-78.
[6] Ghafari, Vahid Amin, Honggang Hu, and Chengxin Xie. "Fruit: ultra-lightweight stream cipher with shorter internal state." IACR. http://eprint. iacr. org/2016/355 (2016).
[7] Mikhalev, Vasily, Frederik Armknecht, and Christian Müller. "On ciphers that continuously access the non-volatile key." IACR Transactions on Symmetric Cryptology (2016): 52-79.

[8] Amin Ghafari, Vahid, and Honggang Hu. "Fruit-80: a secure ultra-lightweight stream cipher for constrained environments." Entropy 20.3 (2018): 180.
[9] Hamann, Matthias, Matthias Krause, and Willi Meier. "Lizard–a lightweight stream cipher for power-constrained devices." IACR Transactions on Symmetric Cryptology (2017): 45-79.
[10] Masoodi, Faheem, Shadab Alam, and M. U. Bokhari. "SOBER Family of Stream Ciphers: A Review." International Journal of Computer Applications 23.1 (2011): 1-5.
[11] Hell, Martin, Thomas Johansson, and Willi Meier. "Grain: a stream cipher for constrained environments." IJWMC 2.1 (2007): 86-93.
[12] Hell, Martin, et al. "The Grain family of stream ciphers." New Stream Cipher Designs. Springer, Berlin, Heidelberg, 2008. 179-190.
[13] Bjørstad, T. E. "Cryptanalysis of grain using time/memory/data tradeoffs." on Estream Phase 3 (2013).
[14] Hell, Martin, Thomas Johansson, and Willi Meier. "Grain: a stream cipher for constrained environments." IJWMC 2.1 (2007): 86-93.
[15] Berbain, Côme, et al. "Sosemanuk, a fast software-oriented stream cipher." New stream cipher designs. Springer, Berlin, Heidelberg, 2008. 98-118.
[16] Adams, Carlisle, and Stafford Tavares. "The structured design of cryptographically good S-boxes." Journal of cryptology 3.1 (1990): 27-41.
[17] Kumar, Naveen, et al. "BEAN: a lightweight stream cipher." Proceedings of the 2nd international conference on Security of information and networks. 2009.
[18] Zhong, Jianghua, and Dongdai Lin. "A new linearization method for nonlinear feedback shift registers." Journal of Computer and System Sciences 81.4 (2015): 783-796.
[19] Good, Tim, and Mohammed Benaissa. "Hardware results for selected stream cipher candidates." State of the art of stream ciphers 7 (2007): 191-204.
[20] Zhang, Haina, and Xiaoyun Wang. "Cryptanalysis of Stream Cipher Grain Family." IACR Cryptology ePrint Archive 2009 (2009): 109.
[21] Hawkes, Philip, and Gregory G. Rose. "Primitive Specification for SOBER-128." IACR Cryptology ePrint Archive 2003 (2003): 81.
[22] Bokhari, M. U., and Shadab Alam. "BSF-128: a new synchronous stream cipher design." Proceeding of international conference on emerging trends in engineering and technology. 2013.
[23] Bokhari, M. U., and Shabbir Hassan. "A comparative study on lightweight cryptography." Cyber Security. Springer, Singapore, 2018. 69-79.
[24] David, Mathieu, Damith C. Ranasinghe, and Torben Larsen. "A2U2: a stream cipher for printed electronics RFID tags." 2011 IEEE International Conference on RFID. IEEE, 2011.
[25] Immler, Vincent. "Breaking HITAG2 revisited." International Conference on Security, Privacy, and Applied Cryptography Engineering. Springer, Berlin, Heidelberg, 2012.
[26] Verstegen, Aram, Roel Verdult, and Wouter Bokslag. "Hitag 2 hell–brutally optimizing guess-and-determine attacks." 12th {USENIX} Workshop on Offensive Technologies ({WOOT} 18). 2018.
[27] Hamann, Matthias, Matthias Krause, and Willi Meier. "Lizard–a lightweight stream cipher for power-constrained devices." IACR Transactions on Symmetric Cryptology (2017): 45-79.
[28] Bonnerji, Rhea, et al. "COZMO-A new lightweight stream cipher." (2017).
[29] Klein, Andreas. "*Linear feedback shift registers*." Stream Ciphers. Springer, London, 2013. 17-58.
[30] Paul, Goutam, and Subhamoy Maitra. RC4 stream cipher and its variants. CRC press, 2011.
[31] Galanis, Michalis D., et al. "Comparison of the hardware architectures and FPGA implementations of stream ciphers." Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004. IEEE, 2004.
[32] Shabbir Hassan and Mohammad Ubaidullah Bokhari. "*Computing in Cryptography*." 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2016. ISSN 0973-7529; ISBN 978-93-80544-20-5.
[33] Shabbir Hassan, M.U. Bokhari and Md. Zeyauddin. "*Radio Frequency Identification Tag: A Review.* " 2017 4th International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2017. ISSN 0973-7529; ISBN 978-93-80544-24-3
[34] Bokhari, M. U., and Shabbir Hassan. "*A comparative study on lightweight cryptography*." Cyber Security. Springer, Singapore, Cyber Security, Advances in Intelligent Systems and Computing 729. 2018. 69-79. https://doi.org/10.1007/978-981-10-8536-9_8
[35] Hassan, Shabbir and Mohammad Ubaidullah Bokhari, (2019), "*Analysis and Design of LFSR Based Cryptographic Algorithm*." Journal of Advances and Scholarly Researches in Allied Education (JASRAE), ISSN 2230-7540, Vol. 16, Issue No. 9, June-2019.
[36] Hassan, Shabbir and Mohammad Ubaidullah Bokhari, "*Design of Pseudo Random Number Generator using Linear Feedback Shift Register.* " International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-9 Issue-2, December, 2019.
[37] Shabbir Hassan, Prof. M. U. Bokhari, presented a paper entitled "*Lightweight Cryptography: A Review*", Recent Trends in Mathematical and Computational Science (NCRTMCS), January 2015, pp-78.