Design and Implementation of Dynamic Key Based Stream Cipher for Cryptographic Processor

K.K.Soundra Pandian
Department of Electrical Engineering
Indian Institute of Technology Patna,
Patna, Bihar, India, 800013
Email: kpandian@iitp.ac.in

Saptadeep Pal
Department of Electrical Engineering
Indian Institute of Technology Patna
Patna, Bihar, India, 800013
Email: saptadeep.ee11@iitp.ac.in

Kailash Chandra Ray, *Member*, IEEE
Department of Electrical Engineering
Indian Institute of Technology Patna
Patna, Bihar, India, 800013
Email: kcr@iitp.ac.in

***Abstract*—In this paper, a new design for dynamic key based stream cipher is proposed for the hardware cryptographic applications such as data transmission and information security. Unlike the static key based existing stream ciphers, the novelty of this proposed stream cipher is based on dynamic key, generated by Toeplitz hash function which is used as a key for RC4 stream cipher. Further, this key is used to generate the dynamic hardware key for the cryptographic processor. The proposed design is implemented on targeting a commercially available xilinx spartan 3E xc3s500e-4fg320 FPGA device. This implementation of the proposed methodology accomplishes data throughput of 520 Mbps with the initial latency of 17 clock cycles, at a maximum clock frequency of 65 MHz with the total estimated power consumption of 131 mW. The excellence of the generated random hardware key values of our proposed method is validated using NIST statistical test suites. The proposed design in this paper is considerably improved in terms of frequency, throughput performance and area consumed compared to existing design.**

Keyword*: *Hash Function; Key Generation; Hardware Security; RC4 Stream Cipher; HDL; FPGA

I. Introduction

Information technology has become the backbone of modern society, which makes data security an important aspect of research. Now-a-days almost all the confidential facts about the financial status, customers, products, research, customers, or employees, are stored and handled on computers, or transferred to other devices through an accessible physical medium. Information needs to be shared in such a format, which cannot be exploited in a feasible amount of time by any adversary without the knowledge of the key parameters used to encrypt the information. Now-a-days, cryptography has been strongly corroborated in information science using the statistical information theory and number theory [1], [2]. With the emergence of the Internet of Things (IoT) applications, research with focus on robust yet simple crypto-systems have gained importance.

Cryptography can be categorized by the type of encryption operations, the way in which plain text is processed with the numbers of keys used. To secure the data, the encryption algorithm should be fixated to maintain the resistance against the adversary. The updating of encryption and decryption system is necessary as and when the new attack properties are effective. In the security system mechanism of cryptography, the key value ought to be by irreproducible and secret, and it should be challenging for the adversary to decipher the mechanism. Besides the size of key being short, the adversary can interpret the mechanism through brute-force method to retrieve the key value.

In a recent trend, the modern cryptography system utilizes the secure key mechanism of symmetric [3] with asymmetric key cryptography [4] to provide the complete solution to secure the data with authentication. Possessing the trend intact, the paper confesses on the cryptography system which exploits to generate the secured key of symmetric mechanism with the hash function [5]. It provides a virtuous arrangement of performance, efficiency, security, and implements ability. It depends upon the necessity, the researcher may compromise between the security level, throughput and area consumption. The symmetric key encryption plays a vital role, to encrypt huge amount of data effectively. However, the symmetric key based cryptography has some solemn issues, namely key management and key distribution [6]. The asymmetric key cryptography [3] was proposed, in order to resolve the key distribution solemn issues in the symmetric key encryption. On the other hand, the family of hash function is an algorithm that maps the data of arbitrary length to the fixed length data. Due to the one-wayness of hash functions, and less hardware complexity, this is the preferred solution to many cryptographic applications [7] where authentication of receiving data is the goal.

In a communication system using RC4 algorithm, the secret master key is shared between the receiver and the transceiver. In this paper, a method of secured encryption and decryption using RC4 algorithm and Toeplitz hash function in the integrated form is proposed where the key is generated using a Toeplitz matrix based hash value generator. This method is designed using Hardware Description Language (HDL) and implemented on the commercially available FPGA device. Further, the randomness test analysis is performed using statistical test methods provided by NIST.

The rest of the paper is organized as follows; Section II highlights the cryptographic algorithms such as RC4 and Toeplitz Hash function. The development of proposed model and methodology of dynamic key based RC4 stream cipher along with hash function are presented in section III. Simulation, implementation and randomness testing analysis are discussed in section IV and finally, section V concludes the paper.

II. Cryptography Algorithms

An efficient computational function is the hash function, which maps an indiscriminate length of binary strings of sequence to a fixed length of binary sequences of string called hash value. Hash functions are generally used in cryptography for message authentications [7], the hash values of messages are encrypted and appended to the digital signature as the message, can confirm integrity as well as authentication in the crypto systems. The one-wayness property of the hash function determines the cryptographically strong random number designed based on the hash function. In RC4 stream cipher crypto processor system, key scheduling and random number, generation [8] components are executed to generate the key stream for the encryption and decryption process. The integration of utilizing the hash function for the stream ciphering process is well elucidated in detail here.

A. RC4 Algorithm

RC4 stream cipher [9] is proposed and designed by Ron Rivest for the RSA security of data. The improved and effective version of the RC4 stream cipher is postulated by Jian Xie et al [10] and Weerasinghe [11]. The RC4 stream cipher consists of substitution (S)-box S, an array of length M, every position of S can store one byte of information. To scramble this permutation of all the possible 8-bytes, secret key k of l-bytes size (typically, $5 \leqslant l \leqslant 16$) is required. An array L of length M clutches the core key, with the secret key k repeated as $L[i] = k[i \bmod l]$, for $0 \leq i \leq M-1$. In a communication system, the secret master key is shared

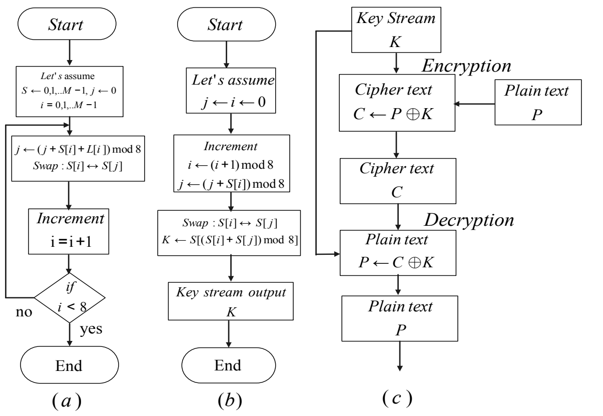
Start
Let's assume
$S \leftarrow 0,1,..M-1,\ j \leftarrow 0$
$i = 0,1,..M-1$
$j \leftarrow (j + S[i] + L[i]) \bmod 8$
$Swap : S[i] \leftrightarrow S[j]$
Increment
i = i + 1
if
i < 8
no
yes
End
(a)
Start
Let's assume
$j \leftarrow i \leftarrow 0$
Increment
$i \leftarrow (i+1) \bmod 8$
$j \leftarrow (j + S[i]) \bmod 8$
$Swap : S[i] \leftrightarrow S[j]$
$K \leftarrow S[(S[i] + S[j]) \bmod 8]$
Key stream output
K
End
(b)
Key Stream
K
Encryption
Cipher text
$C \leftarrow P \oplus K$
Plain text
P
Cipher text
C
Decryption
Plain text
$P \leftarrow C \oplus K$
Plain text
P
(c)

Figure 1: Algorithmic State Machine to Construct RC4 Stream Cipher: (a) Key Scheduling Phase, (b) Pseudo Random Generation Phase (c) Stream Cipher

between the receiver and the transceiver. At the transmitter, encryption is done using simple XOR operation (C = P $\oplus$ K) to get the cipher text (C) and the decryption at the receiver side is done using the XOR operation of the cipher text (P = C $\oplus$ K) to retrieve the original plain text (P).

The Pseudo-Random Generation (PRG) and the Key Scheduling (KS) phases are the main components of RC4 hardware design [2] algorithm is represented in the algorithmic state machine as shown in Figure. 1. Key scheduling phase itself consists of two sub-stages, namely *Initialization* and *Key setup*. In the initialization stage, an M-element $S-box$ is generated where M lies in the range from 0 to 2^{n-1}, where n is the size of the key. The initialized $S-box$ is then permuted based on the provided key L in the key setup stage where a M number of iterations takes place. Then the PRG Phase uses this $S-box$ to generate pseudo random key stream bytes, which are of an arbitrary number. The steps and process of KS, using the Toeplitz hash function and its significance is elucidated in [9], [12].

B. Toeplitz Hash Function

The hash values are generated by multiplying a message string with random binary matrix. Let E be a $x \times y$ matrix with boolean values, where y and x are message size and hash length in bits respectively. Let Y be a message consisting of y bits and boolean multiplication of the matrix E by the column matrix of the message Y as the hash value $h(E)$. The matrix is uniquely determined for the first row and column; the hash function H uses the Toeplitz matrix requires $x + y$ - 1 bits define the whole matrix. The hash function H family is $\oplus$ - linear if for all arguments E, E' satisfy $h(E \oplus E') = h(E) \oplus h(E')$, are feedback for the functions in H family, where $h \in_R H$ [13]. The hash function is called ϵ-balanced for $\epsilon = 2^{-x}$ [14] if,

$$P_{r_h}(h(E) = c) \leqslant \epsilon, \forall\ E \neq 0,\ c \qquad (1)$$

where, $P_{r_h}(h(E))$ represents the probability of the event $h(E)$. The characteristic property of the Toeplitz matrix is that each value of the column in the binary matrix is achieved by moving down the values of the preceding column and adding the new value element to the first element of the column [15], thus in the Toeplitz matrix each successive element of the column embodies the successive stages of the linear type of feedback shift register. LFSR with the feedback polynomial of degree n, which is used to define the Toeplitz matrix, with the initial seed value of n bits. The hardware architecture for LFSR based Toeplitz matrix for 8-bit hash value is shown in Figure. 2. The constructed LFSR based Toeplitz matrix is ϵ-balanced for $\epsilon \leq \frac{y}{2^{x-1}}$. The steps to construct the hash value can be well explained as below.

The irreducible polynomial of degree n over GF (2) is represented as $g(y)$, and LFSR generates the sequence of bit a_0, $a_1 \cdots$ with the feedback polynomial $g(y)$ and the initial seed value is represented as a_0, a_1, $\cdots$, a_n [16]. Thus for every irreducible polynomial $g(y)$ and initial value of state $a \neq 0$, the hash function is associated with the message Y consisting of y bits of binary length as the linear function of the combination. Let's consider the irreducible feedback polynomial for generating the hash value using the LFSR polynomial $g(y) = x^8 + x^4 + x^3 + x^2 + 1$ and the initial value

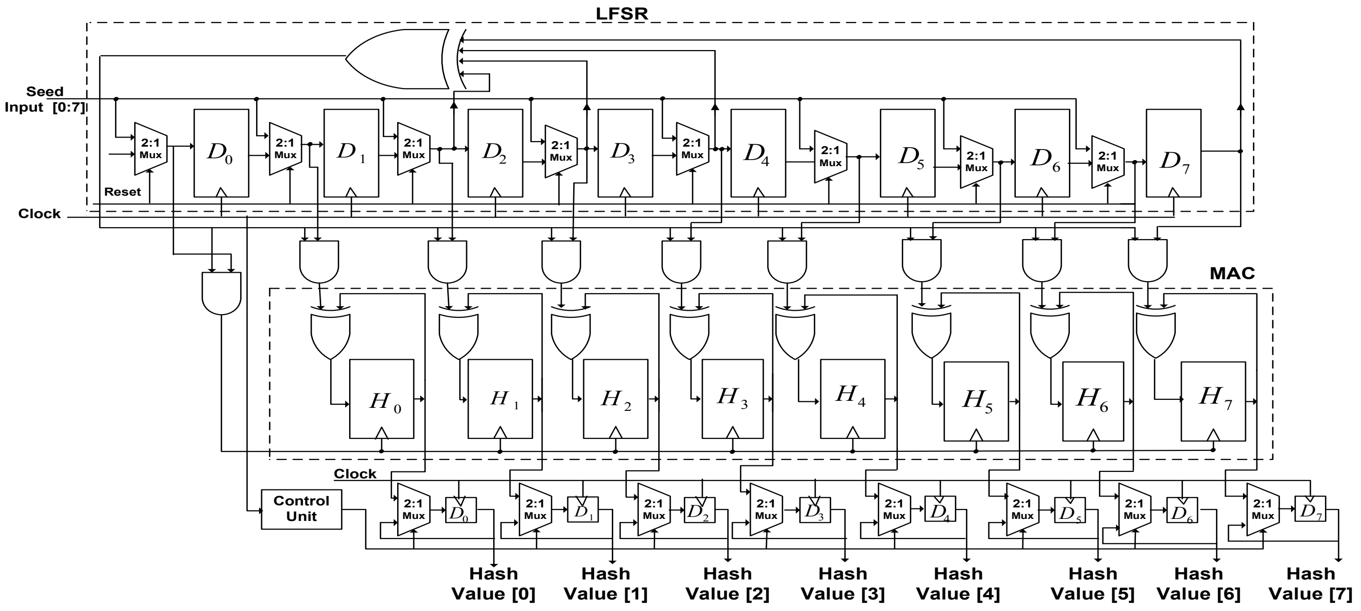

Figure 2: 8-bit Hardware architecture and Implementation of LFSR based Toeplitz hash function

of LFSR is considered as $(a_0, a_1, \cdots, a_7)$ = $(0\ 1\ 0\ 1\ 0\ 0\ 1\ 0)$. To determine the requirement of the hash function in the Toeplitz matrix, should satisfy x + y - 1 bit, the output sequence from LFSR should produce only 9-bits where x is the degree of $g(y)$ is 8 and the message bits y = 8 (thus, 8 + 8 - 1 = 15 bits). With the above irreducible polynomial $g(y)$ of degree 8 and with the initial seed value, the 15 bit sequence LFSR output is $(a_0, a_1, a_2, \cdots, a_{14})$ = $(0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0)$. Toeplitz matrix is formed by starting the initial value of LFSR,

LFSR Output Sequence =[0 1 0 1 0 0 1 0 1 0 0 0 1 0 0]

Toeplitz matrix (8 x 8) =

$$\begin{array}{cccccccc} \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{array}$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

for each values of the column in the binary matrix is achieved by moving down the values of the preceding column and adding the new value element to the first element of the column, thus in the toeplitz matrix each successive element of the column embodies the successive stages of the linear type of feedback shift register is achieved after every clock cycle to top of the each column.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$$

Let's assume the message bits be [1 1 0 0 1 1 1 1], thus multiplying the message in column matrix with the Toeplitz matrix which is obtained with the LFSR output sequence determines the Toeplitz matrix of (8 x 8) is given above and the hash value output is $[0\ 1\ 1\ 0\ 1\ 0\ 0\ 0]^T$. The hash value for the corresponding message bit, can be a secret key to RC4 as discussed.

III. Proposed Model and Methodology

The proposed model and method consist of the Toeplitz matrix based hash function, which provides hash value as a key to the RC4 stream cipher is shown in Figure. 3. The output

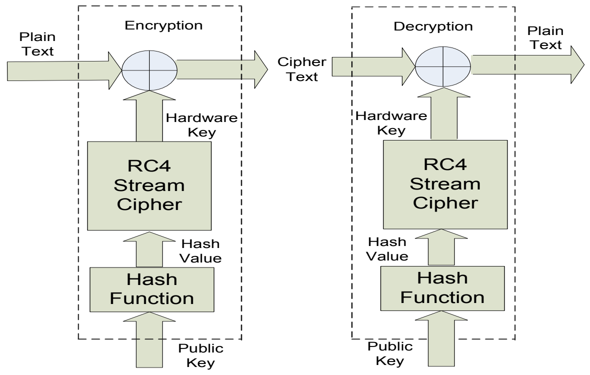

Figure 3: Proposed Methodology for Encryption and Decryption

key stream of the stream cipher is used to encrypt the plain text message using the XOR operation. The Toeplitz matrix based hash function requires the message which is to be encrypted, is used to generate the hash value. In this method, the public key (message) which will be shared between the encryption and decryption process. A continuous stream of message is

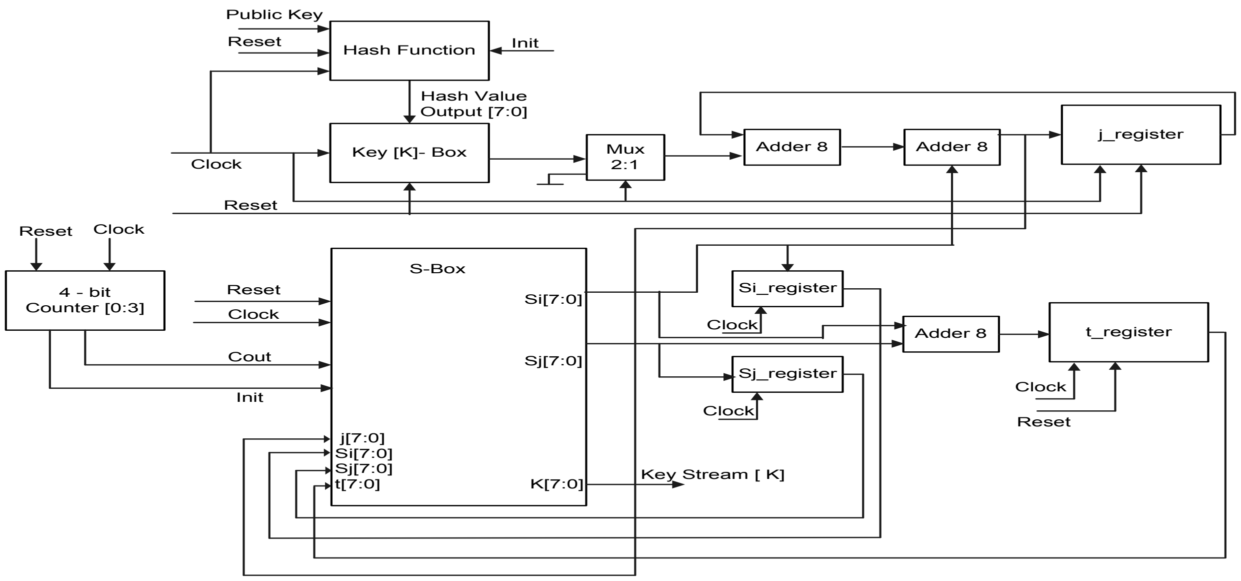

Figure 4: Schematic of Proposed Design

fed as input to the hash function block to generate the stream of the hash value. This operation is processed using the public key which is saved in the register memory bank and it is fed to generate the hash value by right shifting the operator bit by bit [17].

LFSR based PRNG's output bits are multiplied with the input bits, which can be achieved using AND gate and passed on to the MAC (multiply-and-accumulate) unit (using AND-OR logic) so as to accumulate it over a specified period of time (n clock cycles). The output is pushed every n^{th} clock cycle using a multiplexer which is controlled by a control unit. The control unit consists of a conventional counter and it produces a transition signal after every y clock cycle, where y is the size of the message to be hashed. This output is then provided as key (L) to the stream cipher block.

In a conventional $RC4$ stream cipher, the key stream is usually fixed which is either a public key or a private hardware key [18]. However, in this proposed design, we have constructed the dynamic key stream where the key k stream is replenished on every n cycle by a hash value of the public key as shown in the Figure. 4, schematic structure of proposed design. We have used an LFSR based Toeplitz matrix is used to generate the hash value. In general, a fixed $x \times y$ Toeplitz matrix is used to generate the hash value where the LFSR is reset after every computation of the hash value. But since the input to the hash value generator is the public key k in every cycle, this method would generate the same hash value every time.

In this methodology, the hash value itself changes over time since the LFSR based Toeplitz matrix is not reset except for a global reset. Thus the Toeplitz matrix only repeats after the total number of 2^n cycles, where n is the number of bits of the LFSR. The motivation of having a dynamic key for the $RC4$ algorithm is to increase the security of the whole crypto system. Moreover, since hash function is one-way, it is considerably infeasible to get the public key even if a cryptanalytic attacker is able to find a particular key k. Thus, it will again be very difficult to trace the next key stream value which is generated from the hash function. Initially, the crypto system waits for the n cycles to get the hash value generated. Once the hash value is ready, the RC4 algorithm is initiated to generate the key stream. After the first n clock cycles the hash function and RC4 algorithm operate in parallel. Henceforth, for every n clock cycle a new hash value is generated and replaces the value of the Key (L), thus realizing the dynamic key for the RC4 stream cipher.

IV. Simulation, Implementation and Test Analysis

At the initial stage of the process in LFSR, the seed value is loaded into the registers, using the clock. The seed input values are then loaded to the flip-flops and to the corresponding output, once the input value is high. The previous state output value is XOR_{ed} with the current state value and it is stored as the present state value, during the active low input state. With the initial seed input value of the LFSR, the hash value itself changes over time since the seed value for Toeplitz is not reset except for a global reset.

A. Simulation and Implementation Results

The proposed method is simulated and synthesized using Verilog HDL on hardware FPGA device (xilinx spartan 3E xc3s500e-4fg320). The simulation result of generating a hash value for the corresponding message bit is presented as an example in the previous section, is shown in the Figure. 5, are further used to construct the random dynamic key from the 8-bit seed. The synthesized report for logic utilization is summarized in Table. I. The proposed integrated design of hash values of stream cipher to generate the secured and dynamic key stream is implemented on xilinx spartan 3E. The device utilization and timing constraints of the implemented design is shown in Table. I. The maximum operating frequency

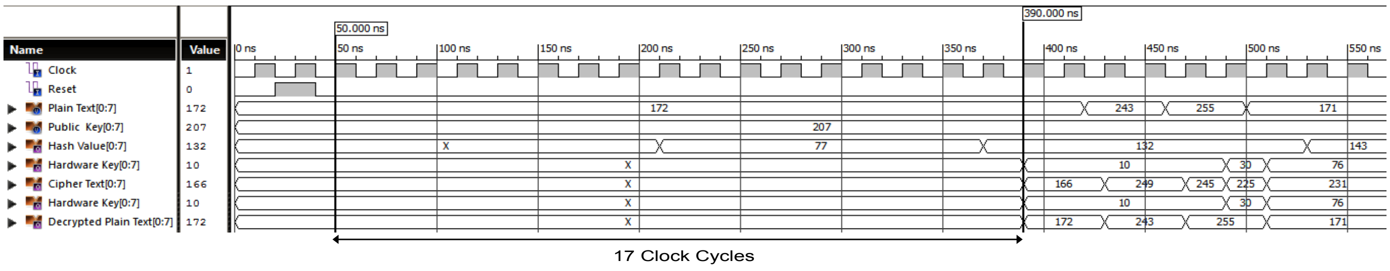

Figure 5: HDL Simulation Result of Proposed Design

of the proposed design is 65 MHz.

Table I
LOGIC UTILIZATION AND TIMING SUMMARY

Logic	**Utilization**
Number of Slice	422
Number of Slice Flip Flops	197
Number of 4 input LUTs	831
Number of bonded IOBs	26
Timing	**Utilization (ns)**
Minimum period	15.383

The proposed design and other design is also implemented on the same hardware platform (xilinx spartan 3E xc3s500e-4fg320 FPGA family) for the fair comparison. Thus, the implementation of the stream ciphers in reference to performance results and area consumed in terms of throughput and CLB slices respectively are presented in Table. II. As exemplified in Table. II, the leading throughput is achieved by the proposed RC4 stream cipher compared to existing family of stream ciphers, and it's measured in Megabits per second (Mbps). In order to generate the reliable and efficient system in the FPGA implementation design, cooling and power factor plays a vital role to determine the system functionality. Due to clocking, data and bit stream, the power and thermal characteristic vary intensely.

Table II
PERFORMANCE COMPARISON

Integrated Hash With Ciphers	**Area (slices)**	**Frequency (MHz)**	**Throughput (Mbps)**
$RC4^+$ [19]	483	43.43	347
VMPC [20]	372	56.13	449
RC4A [21]	643	58.75	470
Proposed RC4	422	65.01	520

Table III
THERMAL SUMMARY

Ambient Temp (°C)	25 °C
Junction Temp (°C)	27.7 °C
Max Ambient (°C)	82.3 °C
Θ J-A (°C/W)	20.4 °C/W
Airflow (LFM)	250

To avoid the pitfalls of FPGA design, the power and temperature analysis are performed on the proposed design and are analysed using xilinx X-power analyser tool and the results are shown in Table. III. Thus, the design consumes a total power of 131 mW with the junction temperature of 27.7 °C are within the desired limit for the proposed design, at the maximum process specification.

An efficient RC4 architecture is constructed to support variable key length and the architecture of an integrated crypto system in which the RC4 algorithm is initiated and key stream is generated have been proposed. The length of the key may be from 8-bit to 256-bit, disparate to the prior design [22], [23] which upkeep the key size of fixed length. Thus, the implementation of the RC4 stream cipher in reference to performance results and area consumed in terms of throughput and CLB slices respectively, the achieved throughput by the proposed RC4 stream cipher, is measured in Megabits per second (Mbps).

As seen in Figure. 5, after the initial reset, the 8-bit hash value is generated after 8 clock cycles. This hash value is then fed to the Key[K]-box in the next clock cycle. The first key scheduling phase is then initiated which takes another 8 clock cycles to compute the [S]-box value for pseudo-random generation phase and thus the first hardware key is ready after 17 clock cycles from the initial reset. Henceforth, the hash operation, key-scheduling and pseudo-random generation phase runs simultaneously as a pipelined process and new pseudo-random hardware key is generated every clock cycle. Thus, the maximum throughput can be calculated from the maximum frequency and number of hardware key bits generated every clock cycle. The proposed design is synthesized on xilinx spartan 3E xc3s500e-4fg320 FPGA platform, and the design operates at the maximum frequency of 65 MHz and data throughput of 520 Mbps with the initial latency of 17 clock cycles.

B. Randomness Test Analysis

The forte of the stream ciphering approaches assessed by the NIST statistical randomness test suite [24] [25]. The monobit and autocorrelation tests are determined in a 20,000 consecutive bit stream. The monobit test is to determine the number of ones and zeros which follows the χ^2 distribution with freedom of degree one. The auto correlation test is to check the correlation between the sequence and shifted version of the same sequence, which follows the N (0,1) normal distribution for α = 0.01.

Quite a few random number bit streams (with the sequence length of q = 2048 bit stream) with the linear feedback shift register of initial seed is generated and run through the NIST test suite. An assessed probability (null hypothesis randomness) p-value $\geq \alpha$ where p is the tail probability, would postulate confidence of 99% in the randomness of a sequence [24] of the dynamic key value, where α of 0.01 indicates that except 1 sequence of 100 sequences to be rejected. For brevity, quite a few subset of the results is shown in Table. IV. Since the computed p-value is greater than 0.01, as pass the sequence to be random. Due to lack of space constraints, the other test suite process is not presented in detail here, hence the reader may refer to the test standards given in NIST [24]. The generated dynamic key sequence meritoriously passes the statistical test suites as shown in Table. IV.

Table IV
KEY SEQUENCE RANDOMNESS TEST ($\alpha = 0.01$)

Statistical Test Suites [24]	P-Value (Tail Probability)	Result*
Frequency	0.45247	Pass
Block Frequency	0.81141	Pass
Cumulative Sums	0.57476	Pass
Rank	0.74190	Pass
DFT(Spectral)	0.44097	Pass
Approximate Entropy	0.25646	Pass
Serial	0.66966	Pass
Linear Complexity	0.03574	Pass

* Result passes for value ($p \geq \alpha$) [24]

Statistical Test [25]	χ^2 and Normal Distribution (X-Value)	Result
Monobit	1.0952	Pass**
Auto Correlation	1.5135	Pass@

**Result passes for value ($X \leq 6.6349$) [25]
@Result passes for value ($X \leq 2.3263$) [25]

V. CONCLUSION

In this paper, a new design is proposed for generating the dynamic key based stream cipher using the Toeplitz hash function, which provides a highly secured stream cipher. The proposed design is implemented on commercially available FPGA device, which assures the throughput of 520 Mbps considerably improved than the existing design with an initial latency of 17 clock cycles. The NIST randomness test results have proven that the generated hardware key stream is indeed random. The comparison result shows that the proposed design with high throughput can be good candidate and suitable for the hardware cryptographic applications such as wireless data communications, IoT, sensor networks, and information security.

REFERENCES

[1] N. Sklavos P. Kitsos, G. Kostopoulos and O. Koufopavlou., "Hardware implementation of the RC4 stream cipher," *IEEE 46th Midwest Symposium on Circuits and Systems*, vol. 3, pp. 1363–1366, Dec 2003.

[2] S. Maitra S. Sen Gupta, K. Sinha and B. P. Sinha., "One Byte per Clock: A Novel RC4 Hardware," *Proc. of Indocrypt*, p. 347Ą0Č2363, 2010.

[3] W. Diffie and M.E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[4] G. Geetha, and M. Suresh Kumar, "Asymmetric Key Cipher Based on Non-Linear Dynamics," *International Conference on Emerging Trends in Engineering and Technology, (ICETET '08)*, pp. 1250–1254, July 2008.

[5] D. Gligoroski, R.S. Odegard, M. Mihova, S.J. Knapskog, A. Drapal, V. Klima, J. Amundse, and M. El-Hadedy, "Cryptographic hash function Edon-R'," *Proceedings of the 1st International Workshop on Security and Communication Networks (IWSCN'09)*, vol. 22, no. 6, pp. 1–9, May 2009.

[6] M. Bala Krishna, and M.N. Doja, "Symmetric key management and distribution techniques in wireless ad hoc networks," *International Conference on Computational Intelligence and Communication Networks (CICN'11)*, pp. 727–731, Oct. 2011.

[7] Tsudik, G, "Message authentication with one-way hash functions," *Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '92)*, vol. 3, pp. 2055–2059, 4-8 May 1992.

[8] K. Zeng, C. -H. Yang, D. -Y. Wei, and T.R.N. Rao, "Pseudorandom bit generators in stream-cipher cryptography," *IEEE Transactions on Computers*, vol. 24, no. 2, pp. 8–17, Feb. 1991.

[9] S.S. Gupta, A. Chattopadhyay, K. Sinha, S. Maitra, and B.P Sinha, "High-Performance Hardware Implementation for RC4 Stream Cipher," *IEEE Transactions on Computers*, vol. 62, no. 4, pp. 730–743, 2013.

[10] Jian Xie and Xiaozhong Pan., "An improved RC4 stream cipher," *International Conference on Computer Application and System Modeling (ICCASM'10)*, vol. 7, pp. 156–159, Oct 2010.

[11] T.D.B Weerasinghe., "An Effective RC4 stream cipher," *International Conference on Industrial and Information System (ICIISA'13)*, pp. 69–74, Aug 2013.

[12] Chang, D., Gupta, K., Nandi, M, "RC4-Hash: A New Hash Function based on RC4 (Extended Abstract)," *In: Barua, R., Lange, T. (eds.) INDOCRYPT '06. LNCS, Springer, Heidelberg*, vol. 4329, 2006.

[13] Krawczyk H., "LFSR-based hashing and authentication," *In: Advances in cryptology - crypto'94. Lecture notes in computer science, Springer-Verlag*, vol. 839, pp. 129–139, 1994.

[14] J.L. Carter and M.N. Wegman, "Universal Classes of Hash Functions," *JCSS*, vol. 18, pp. 143–154, 1979.

[15] P.P.Deepthi and P.S. Sathidevi, "Design, Implementation and analysis of hardware efficient stream ciphers using LFSR based hash functions," *Computers and Security*, vol. 28, no. 3-4, pp. 229–241, 2009.

[16] K.K.Soundra Pandian and K. C. Ray, "Five Decade Evolution of Feedback Shift Register: Algorithms, Architectures and Applications," *International Journal of Communication Networks and Distributed Systems (accepted for publication, in press)*, Feb. 2015.

[17] S. Pal, K.K. Soundra Pandian, and K.C. Ray, "FPGA implementation of stream cipher using Toeplitz Hash function," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI '14)*, Sept. 2014, pp. 1834–1838.

[18] S.H.M. Kwok, and E.Y. Lam, "Effective Uses of FPGAs for Brute-Force Attack on RC4 Ciphers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 8, pp. 1096–1100, Aug. 2008.

[19] Goutam Paul, Subhamoy Maitra, "RC4 Stream Cipher and Its Variants Series," *Discrete Mathematics and Its Applications, CRC Press*, p. 229, 2011.

[20] B. Zoltak, "VMPC one-way function and stream cipher," *In B. Roy and W. Meier, editors, Fast Software Encryption,Lecture Notes in Computer Science,Springer-Verlag*, vol. 3017, pp. 210–225, 2004.

[21] Maximov, Alexander, "Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness of RC4 Family of Stream Ciphers," *Fast Software Encryption, Lecture Notes in Computer Science, Springer Berlin Heidelberg*, pp. 342–358, 2005.

[22] Alan J. Hu, "A CPLD-Based RC4 Cracking System," *Proc.of Canadian Conference on Electrical and Computer Engineering*, May 1999.

[23] T.Hamalainen P.Hamalainen, M.Hannikainen and J.Snarinen, "Hardware Implementation of the Improved WEP and RC4 Encryption Algorithms for Wireless Terminals," *Proc.of European Signal Processing Conference (EUSIPCO), Tampere, Finland*, pp. 2289–2292, 5th - 8th Sept. 2000.

[24] R. Rueppel, "An Effective RC4 stream cipher," *available online at http://csrc.nist.gov/rng., Springer Berlin Heidelberg*, 2001.

[25] A. Menezes, P. van Oorschot, S. Vanstone, "Handbook of Applied Cryptography," *CRC Press*, 1996.