Sādhanā (2020)45:256
https://doi.org/10.1007/s12046-020-01489-w

Performance analysis of current lightweight stream ciphers for constrained environments

SUBHRAJYOTI DEB* and BUBU BHUYAN

Department of Information Technology, North-Eastern Hill University, Shillong, India
e-mail: subhrajyotideb1@gmail.com; b.bhuyan@gmail.com

MS received 29 August 2019; revised 14 June 2020; accepted 27 July 2020

Abstract. Maintaining an adequate balance between security and other performance metrics like memory requirement, throughput and energy requirement in a resource-constrained environment is a major challenge. The National Institute of Standards and Technology (NIST), in its latest lightweight cryptography report, suggested the suitability of symmetric ciphers in constrained devices. In this paper we have performed statistical security analyses of six state-of-the-art stream ciphers, namely Lizard, Fruit, Plantlet, Sprout, Grain v1 and Espresso, with the help of randomness test, structural test, autocorrelation test and avalanche test. We have also carried out the performance analysis of these ciphers in detail after porting the optimized code of the ciphers to a low-cost microcontroller, namely ATmega 328P. The selection of the device is based on its acceptability in the Internet of Things (IoT)-based network. The statistical security, performance metrics and comparative analysis suggest the suitability of the selected ciphers for providing security in constrained environments.

Keywords. Stream cipher; block cipher; lightweight cryptography; IoT; randomness test; constrained device.

1. Introduction

Providing security services in a resource-constrained environment is a significant research challenge, and presently a spurt of research activities has taken place in this direction [1–6]. This rise in research interests is a result of several promising applications like smart city applications, remote health monitoring systems, disaster management systems, etc., which require secured communication among heterogeneous resource-constrained distributed devices. Addressing security issues in such a resource-constrained environment is the significant hurdle for successful implementation of these applications in a wider scale. The cryptographic primitives that are used to provide security in these networks should be lightweight in terms of requirements of memory, energy, etc. Further, these primitives should also be easily implementable in hardware. Out of the various security issues, maintaining the balance between the level of security and computational cost in a resource-constraint environment is a prominent one that has caught the attention of many researchers [7–9]. Asymmetric key cryptographic techniques are not suitable for the constrained environment as they require large computational processing, memory space and energy. Recently the lightweight symmetric cryptography has received considerable attention, because it provides high-level security in constrained environments like Internet of Things (IoT) [10]. The NIST of the United States started a 'lightweight cryptography' project, and it was suggested that the symmetric ciphers are suitable for providing high-grade security in low-end devices.[1] It may be noted that lightweight symmetric ciphers are classified into two types, namely block cipher and stream cipher, and they try to resolve the problem of confidentiality when the data is communicated over a hostile environment.

After several evaluation phases, Grain stream cipher has been identified as one of the best candidates of the ECRYPT eSTREAM project [11]. After eSTREAM competition a few lightweight block ciphers like Hight, Present, Katan, Midori, Gift and Klein emerged, which could compete with Grain cipher in terms of low hardware requirements. To maintain the trade-off between the level of data security and performance measures, researchers have been working on different types of lightweight block ciphers to improve the performance [12]. Furthermore it has been claimed that the mentioned block ciphers require chip area of nearly 1000 gate equivalents (GE), which make it suitable for passive low-end devices. Therefore, in the last few years, lightweight block ciphers were getting priority over stream ciphers in providing security in lightweight applications.

*For correspondence
Published online: 12 October 2020

[1]Report on Lightweight Cryptography [online]. Website https://csrc.nist.gov/CSRC/media/Publications/nistir/8114/draft/documents/nistir_8114_draft.pdf [accessed 09 January 2019].

With the pioneering research of Armknecht and Mikhalev [13], they proposed 'Sprout' lightweight stream cipher by reducing the size of the internal state. The smaller internal states result in the reduction of the chip area up to nearly 800 GE. With the development of Sprout, the acceptability of stream cipher in lightweight applications increases to a considerable extent [14–16]. Inspired by the Sprout cipher design, several new lightweight linear feedback shift register (LFSR)-based stream cipher's like Fruit [17], Lizard [18] and Plantlet [19] were introduced. Concurrently, Espresso cipher has emerged for the fifth-generation (5G) mobile communication system [20]. Although a few high-end cryptanalytic attacks on Sprout algorithm are mentioned in the literature [21–23], the security of the ciphers in lightweight environment should be judged based on different statistical tests like randomness test, autocorrelation, avalanche effect, etc. Inspired from this discussion, we are motivated to work on current lightweight stream ciphers and the main contributions of this paper are as follows:

- Various statistical tests were performed on a few selected stream ciphers, namely Lizard, Fruit, Plantlet, Sprout, Grain and Espresso, and their suitability for cryptographic applications was established.
- The codes of stream ciphers have been optimized to make them suitable for a constrained environment. Further, we have evaluated the performance of the selected ciphers in low-cost resource-constrained ATmega328P device.

In addition, the selected stream ciphers performances have been compared to the existing lightweight block ciphers such as Simeck, Kasumi, Klein, Present, Hight and Spec. In this work, we have followed the recent study on block ciphers [7, 9, 12]. Overall, experimental results and comparison reveal that selected stream ciphers are suitable for small computing devices.

The rest of this paper is structured as follows. Section 2 presents some of the background studies of the present work. Section 3 provides the statistical security analysis on stream ciphers and results thereof. Section 4 presents the performance analysis of the selected stream ciphers and their comparison with the peers. Finally, a conclusion has been presented in section 5.

2. Literature review

Recently lightweight cryptographic schemes received great momentum in providing security in IoT applications [1], including smart city projects, wireless body area network (WBAN), information and communication technology (ICT), tracking, surveillance, monitoring, etc. Several research papers have referred to security issues in IoT-based networks [2, 24]. The participating devices of IoT appliances work in a multitasking mode, i.e. the same devices act as transmitters as well as sensors. Moreover, the algorithms used to provide security should be lightweight and also their throughput should be high. As an example, an IoT-based application on the remote forest or the farming field may not have any charging facility. To handle the afore-mentioned issues, three essential strategies are suggested by Yang *et al* [25]. First, provide minimum security requirements to the low-end devices. Second, the device may use an expanded battery capacity. Third is energy harvesting by the device itself, which means to harvest energy from physical sources like sunshine, heat, weather, wind, etc. However, the suggested methods [25] are expensive for low-end devices like radio frequency identification (RFID).

It is broadly acknowledged that symmetric ciphers are the potential candidate for providing security in a constrained environment. In this paper we have compared various block ciphers, stream ciphers and hash functions based on a few identified parameters like security services, hardware and software implementation, error propagation, data buffering, energy, etc., as tabulated in table 1.

In recent years, several designers have proposed block ciphers to provide confidentiality and integrity protection in the constrained environment [26–30]. As discussed some research studies [7, 9, 12] put effort to construct lightweight block cipher by reducing block size, the number of rounds, simplifying the substitution-box (S-box), etc., to achieve low power dissipation. Biryukov and Perrin [31] presented a comprehensive survey on the symmetric ciphers and reviewed design strategies that make a scheme lightweight. They also discussed that the quality of randomness of a cipher depends on its nonlinear functions like S-box, Boolean function, etc. However, cryptographically secure nonlinear functions are computationally intensive comprising of many variables with repetitive operations in XOR, AND gates.

The other line of research is the development of light weight stream ciphers to make it suitable for a constraint environment. To overcome the inadequacy of block cipher, Fournel *et al* [32] considered the eSTREAM ciphers' performance in the wireless network.[2] Manifavas *et al* [14] presented a comprehensive review of stream ciphers for security and performance analysis in low-end embedded devices. Many stream cipher designs considered that the internal state size should be at least twice of its key size. Recent investigations were carried out to fit LFSR-based stream cipher in a low-cost resource-constrained environment by reducing the internal state size [12, 13, 31]. These investigations resulted in several well-known stream ciphers, namely Sprout [13], Fruit [17], Lizard [18] and Plantlet [19]. Many research studies

[2]eSTREAM: the ECRYPT Stream Cipher Project [online]. Website http://www.ecrypt.eu.org/stream/ [accessed 09 January 2019].

Table 1. Cryptographic solution associated with IoT applications.

Cipher	Category	Class	Type	Services	Scalar parameters for constrained domain						Reliability
					Speed	HI	SI	EP	Data buffering	Energy	
AES 128	Block cipher	Generic	CTR CBCMAC	Confidentiality, authentication	High	Complex	Efficient	No	More space	High	Low/high
Present[26]	Block cipher	Lightweight	CTR	Confidentiality	High	Complex	Efficient	No	More space	Limited	High
Keccak[42]	Hash	Generic	HMAC	Authentication	High	Complex	Inefficient	Limited	More space	High	Low/high
Spongent[43]	Hash	Lightweight	HMAC	Authentication	High	Complex	Inefficient	Limited	Limited space	Limited	High
Grainfamily[11]	Stream cipher	Generic	LFSR NFSR	Confidentiality, authentication	High	Easy	Efficient	No	Limited space	Limited	High
Sprout[13]	Stream cipher	Lightweight	LFSR NFSR	Confidentiality	High	Easy	Efficient	No	Limited space	Limited	High
Fruit[17]	Stream cipher	Lightweight	LFSR NFSR	Confidentiality	High	Easy	Efficient	No	Limited space	Limited	High
Lizard[18]	Stream cipher	Lightweight	LFSR NFSR	Confidentiality	High	Easy	Efficient	No	Limited space	Limited	High
Plantlet[19]	Stream cipher	Lightweight	LFSR NFSR	Confidentiality	High	Easy	Efficient	No	Limited space	Limited	High
Espresso[20]	Stream cipher	Lightweight	LFSR NFSR	Confidentiality	High	Easy	Efficient	No	Limited space	Limited	High

HI = Hardware implementation; SI = Software implementation; EP = Error propagation; Low/High = signifies conditional dependence.

[21–23, 33–37] have reported the susceptibility of these ciphers to various high-end attacks like fault attack, time memory attack, distinguishing attack, etc. However, these high-end attacks requiring large time and memory may not be considered for evaluating security in emerging networks comprising low-cost constrained devices. Instead, for lightweight cryptography the security evaluation should be done through statistical tests [38, 39].

A secure stream cipher is an important cryptographic primitive that generates a good quality pseudorandom sequence. Turan *et al* [40] experimentally proved that stream ciphers are useful for random number generation. Further, they mentioned that the randomness of a keystream might be judged by various statistical randomness tests only. Qasaimeh *et al* [9] considered numerous lightweight ciphers with good randomness results. On the other hand, they show that ciphers provide good performance using an Arduino-based microcontroller.

3. Statistical test results

Nonlinear feedback shift registers (NFSR) and an LFSR, including a nonlinear filter function, are used to construct pseudorandom generators for lightweight stream ciphers. Most of the current stream ciphers claim that they produce high-quality pseudorandom keystream with lesser computational burden. The statistical test plays an essential part in the design of the cryptosystem because it quickly detects the weaknesses of an output sequence [40].

The experimental works were performed using the following methodology.

Methodology: For the measurement of randomness test, we have used the NIST statistical test suite and AIS 20/31. All experimental works are performed on Mathematica 10, SageMath 7.0, IBM SPSS®, Arduino Studio and GNU Compiler Collection (GCC)-4.8. The configuration of the system used for the experiment was as follows: Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz, 3408 Mhz, 4 Core(s), 8 Logical Processor(s).

The following subsections summarizes the results of the statistical test on selected stream ciphers, namely Lizard, Fruit, Plantlet, Sprout, Grain v1 and Espresso. The statistical tests are carried out using NIST randomness test, structural test, autocorrelation analysis, keystream data analysis and avalanche test.

3.1 *Randomness test analysis*

In general, producing high-quality random numbers deterministically is a hard problem. Stream ciphers are inherently suitable for pseudorandom number generation. Here, we consider the randomness test of selected

lightweight stream cipher keystreams using NIST test suite.[3] For randomness test, we coded these stream ciphers in GCC and generated 10^6 (1 million) keystream bits. Note that the author of Fruit [17] proposed multiple versions of Fruit cipher (listed in ePrint web). The current version is known as Fruit-v2 [17]. Throughout the paper, we shall consider the original version of Fruit (20170719:094350).

In this experiment, the randomness test result is calculated by probability value (P-value) to determine whether the sequence satisfies the criteria. In this test, if the P-value is less than 0.01, the keystream data is supposed to be non-random; otherwise, it is judged to be random. The NIST test suite consists of 15 different statistical tests. Here, we briefly investigate the randomness test results of the pseudorandom sequence length of 10^6 bits generated by six stream ciphers [8, 16]. The P-values of the randomness test results are shown in table 2.

In 2011 the German Federal Office for Information Security (BSI) introduced a renewed version of an evaluation methodology called AIS 20/31 for random number generators [39]. It may help designers to examine security features in their design, and it also helps the evaluators of generators in the evaluation process. This randomness methodology includes eight statistical tests T0–T7, such as the poker test, the long run test and the uniform distribution test. Moreover, out of eight tests, only five statistical tests are applied to check randomness. It is noteworthy that no test has given the evaluation result as fail or reject. The afore-mentioned methodology in the random number generator design and evaluation has become the de facto standard in Europe [38]. In AIS 20/31 and NIST test suite result determines that the mentioned lightweight stream ciphers have successfully passed the randomness tests.

3.2 *Structural test analysis*

Structural test was introduced by Turan *et al* [40] and they reported four types of structural tests, namely i) key/keystream correlation test, ii) initialization vector (IV)/keystream correlation test, iii) frame correlation test and iv) diffusion test. In this work, we systematically use structural randomness tests to evaluate whether the keystream is random or not. The generated pseudorandom binary keystreams of the selected ciphers are used for the structural test. To perform the structural test, we have followed the construction techniques of [40]. The complete structural test report may be noted from [40, Sections 3–4]. The assessment of the mentioned test is performed by the chi-square goodness of fit test (χ^2). We have summarized the structural test results in table 3. Besides these, three important parameters of selected stream ciphers, namely key size, IV size and internal state size, are specified in table 3.

3.3 *Autocorrelation analysis*

Autocorrelation analysis is most relevant for LFSR-based stream ciphers. The keystream sequence of the stream cipher should be indistinguishable from truly random sequences. Binary strings with proper autocorrelation features play an essential role in secure applications. In this paradigm, the attacker attempts to establish some probabilistic relation between the key bits and output. Let $\mathbf{s}^N = (\mathbf{s}_0, \ldots, \mathbf{s}_{N-1})$ be a series of an infinite sequence. The autocorrelation of a sequence $\mathbf{s}^N$ is defined as follows:

$$\mathcal{AC}(\tau) = \frac{1}{N}\sum_{\iota=1}^{N-1}(-1)^{(\mathbf{s}_\iota + \mathbf{s}_{\iota+\tau})}, \quad 0 \le \tau \le \mathbf{s} - 1. \qquad (1)$$

The autocorrelation $\mathcal{AC}(\tau)$ measures the similarity between $\mathbf{s}^N$ and the shift of $\mathbf{s}^N$ by τ. From the cryptographic point of view, autocorrelation assures that the attacker cannot determine the correlations among shifted versions of the identical bitstream.

Example 1 Let us consider an LFSR assign with seed value $[1, 1, 0, 1]$. Run this LFSR 20 times; the corresponding output $(n) = [1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0]$. Now, $\mathcal{AC}(n, p, k) = \frac{4}{15} = 0.2666$ where n is a periodic sequence of elements of $\mathbb{F}_2$ and p is the period of n. Here, k is an integer value. In this example, n = 20 (output sequence), $p = 15$ and $k = 7$.

The autocorrelation between the sequences at τ = 0 is one. The measurement of autocorrelation varies from +0.06 to –0.05 for value $\tau = 100$. Our investigation reveals the absence of linear correlation in keystream data. In figure 1, graphically, we present pairs of autocorrelation indices for the 64-Hex bits (256-bit) keystream sequence of the selected ciphers.

It is seen that stream ciphers pass autocorrelation tests for randomly generated keystreams. However, [23] shows that in Sprout cipher key recovery is possible from partial information of the internal state.

3.4 *Keystream data analysis*

This part focuses on the statistical properties of the output keystream of the mentioned ciphers. Let us consider the first 40 hexbits sequences like $(0, 1, 2, \ldots, e, f)$ produced by the cipher. Initially, frequency distribution has been computed from the output data. Following this, some metrics such as coincidence index, normalized Shannon entropy

[3] A statistical test suite for pseudorandom number generators for cryptographic applications [online]. Website https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf [accessed 09 January 2019].

Table 2. NIST SP800-22 test results of binary keystream generated by Lizard, Fruit, Plantlet, Sprout, Grain v1 and Espresso.

	Lizard		Fruit		Plantlet		Sprout		Grain v1		Espresso	
Test	*P*-value	Result	*P*-value	Result	*P*-value	Result	*P*-value	Result	*P*-value	Result	*P*-value	Result
Frequency	0.350485	Pass	0.122325	Pass	0.924076	Pass	0.032923	Pass	0.122325	Pass	0.304126	Pass
Block frequency	0.008879	Pass	0.350485	Pass	0.637119	Pass	0.897763	Pass	0.455937	Pass	0.739918	Pass
Cumulative sums	0.350485	Pass	0.739918	Pass	0.514124	Pass	0.401199	Pass	0.122325	Pass	0.162606	Pass
Runs	0.534146	Pass	0.911413	Pass	0.181557	Pass	0.574903	Pass	0.834308	Pass	0.834308	Pass
Longest run	0.350485	Pass	0.574903	Pass	0.739918	Pass	0.637119	Pass	0.883171	Pass	0.739918	Pass
Rank	0.534146	Pass	0.213309	Pass	0.000070	Weak	0.000233	Pass	0.011791	Pass	0.010988	Pass
FFT	0.991468	Pass	0.739918	Pass	0.004301	Pass	0.171867	Pass	0.045675	Pass	0.171867	Pass
Non-overlapping template	0.550133	Pass	0.474986	Pass	0.657933	Pass	0.236810	Pass	0.616305	Pass	0.455937	Pass
Overlapping template	0.350485	Pass	0.350485	Pass	0.851383	Pass	0.289667	Pass	0.236810	Pass	0.108791	Pass
Universal	0.000621	Weak	0.000513	Weak	0.015826	Pass	0.000079	Weak	0.007315	Weak	0.002347	Weak
Approximate entropy	0.350485	Pass	0.739918	Pass	0.191687	Pass	0.213309	Pass	0.439621	Pass	0.426931	Pass
Serial1	0.911413	Pass	0.897763	Pass	0.574903	Pass	0.554420	Pass	0.000474	Weak	0.171867	Pass
Serial2	0.137154	Pass	0.12911	Pass	0.262249	Pass	0.924076	Pass	0.162606	Pass	0.759756	Pass
Linear complexity	0.035174	Pass	0.031254	Pass	0.383827	Pass	0.637119	Pass	0.040108	Pass	0.045675	Pass
Random excursions												
$x = -4$	0.251463	Pass	0.554420	Pass	0.334538	Pass	0.210562	Pass	0.012071	Pass	0.236810	Pass
$x = -3$	0.325103	Pass	0.217941	Pass	0.080519	Pass	0.883171	Pass	0.637119	Pass	0.350485	Pass
$x = -2$	0.172692	Pass	0.179539	Pass	0.678686	Pass	0.322461	Pass	0.236810	Pass	0.574903	Pass
$x = -1$	0.533659	Pass	0.759756	Pass	0.122325	Pass	0.714961	Pass	0.228764	Pass	0.574903	Pass
$x = 1$	0.031894	Pass	0.262249	Pass	0.574903	Pass	0.327129	Pass	0.050845	Pass	0.145326	Pass
$x = 2$	0.302581	Pass	0.785327	Pass	0.213309	Pass	0.419012	Pass	0.205897	Pass	0.883171	Pass
$x = 3$	0.125127	Pass	0.319018	Pass	0.206158	Pass	0.090936	Pass	0.191687	Pass	0.139921	Pass
$x = 4$	0.196302	Pass	0.479361	Pass	0.367453	Pass	0.145328	Pass	0.856907	Pass	0.191687	Pass
Random excursions variant												
$x = -9$	0.102526	Pass	0.547637	Pass	0.071177	Pass	0.746572	Pass	0.309056	Pass	0.275709	Pass
$x = -8$	0.309056	Pass	0.706149	Pass	0.100508	Pass	0.845773	Pass	0.644060	Pass	0.971699	Pass
$x = -7$	0.115387	Pass	0.520767	Pass	0.253551	Pass	0.205897	Pass	0.630178	Pass	0.100508	Pass
$x = -6$	0.372502	Pass	0.071177	Pass	0.500934	Pass	0.035955	Pass	0.077290	Pass	0.468595	Pass
$x = -5$	0.127148	Pass	0.280306	Pass	0.329332	Pass	0.129620	Pass	0.228764	Pass	0.637119	Pass
$x = -4$	0.324180	Pass	0.366918	Pass	0.262249	Pass	0.021505	Pass	0.840081	Pass	0.092784	Pass
$x = -3$	0.245072	Pass	0.339799	Pass	0.064149	Pass	0.275709	Pass	0.834308	Pass	0.017912	Pass
$x = -2$	0.309056	Pass	0.016717	Pass	0.275709	Pass	0.372502	Pass	0.791880	Pass	0.671779	Pass
$x = -1$	0.329332	Pass	0.249284	Pass	0.159613	Pass	0.759756	Pass	0.862344	Pass	0.372502	Pass
$x = 1$	0.028181	Pass	0.072663	Pass	0.339799	Pass	0.413032	Pass	0.623240	Pass	0.664861	Pass
$x = 2$	0.407091	Pass	0.124717	Pass	0.041890	Pass	0.314042	Pass	0.437274	Pass	0.027558	Pass
$x = 3$	0.468595	Pass	0.077290	Pass	0.437274	Pass	0.828458	Pass	0.209577	Pass	0.378138	Pass
$x = 4$	0.010482	Pass	0.096578	Pass	0.494392	Pass	0.236810	Pass	0.437274	Pass	0.581770	Pass
$x = 5$	0.042808	Pass	0.050845	Pass	0.588652	Pass	0.867692	Pass	0.209577	Pass	0.468595	Pass
$x = 6$	0.191687	Pass	0.350485	Pass	0.253551	Pass	0.102526	Pass	0.280306	Pass	0.378138	Pass
$x = 7$	0.449672	Pass	0.935716	Pass	0.989179	Pass	0.037566	Pass	0.165646	Pass	0.205897	Pass
$x = 8$	0.588652	Pass	0.554420	Pass	0.685579	Pass	0.266680	Pass	0.090936	Pass	0.401199	Pass
$x = 9$	0.345115	Pass	0.108791	Pass	0.514124	Pass	0.085587	Pass	0.443451	Pass	0.389567	Pass

Random excursions test is a series of eight tests, and Random excursions variant is a series of eighteen tests, and each of the state is represented as x

(NSE), standard deviation, variance and standard error of the mean (SEM ±) have been used to assess the statistical properties of keystream data. Statistical results are shown in table 4. The tabulated result indicates that the keystream sequence is uniformly distributed, and capable of resisting statistical attacks.

3.5 *Avalanche test*

The main benefit of the avalanche effect is that it measures the strength against real-time attacks like brute force attacks. Depending on the cryptographic structure, it is desirable that small changes in the input produces

Table 3. Structural test for the selected stream cipher.

Cipher name	Key	Initialization vector	Internal state	Key/keystream	IV/keystream	Frame correlation	Diffusion test
Lizard [18]	120-bit	64-bit	121-bit	0.132568	0.440241	0.439479	0.295863
Fruit [17]	80-bit	70-bit	80-bit	0.164247	0.220413	0.176764	0.249981
Plantlet [19]	80-bit	90-bit	110-bit	0.151183	0.412489	0.512967	0.354638
Sprout [13]	80-bit	80-bit	89-bit	0.255946	0.055267	0.073154	0.239995
Grain v1 [11]	80-bit	64-bit	160-bit	0.110995	0.164114	0.218232	0.269991
Espresso [20]	128-bit	96-bit	256-bit	0.203666	0.047232	0.042885	0.239994

significant changes in the output. A mapping function from $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2^m$ (n-bit to m-bit binary sequence) presents the avalanche effect as follows:

$$\sum_{0 \le j \le 2^{n-1}} D_j = m \cdot (2^{n-1}) \tag{2}$$

where $D_j = \mathsf{wt}(f(x_j) \oplus f(x_j \oplus \xi_i^n))$ for each i, $x_j \in \mathbb{F}_2^n$, the weight of f is denoted by wt and ξ_i^n is 1-bit error of n-bit sequence.

Example 2 Assume that 16-bit key is 0110010101110011. Now, we considered that the most significant bit is flipped like 1110010101110011. After the keystream generation, the actual keystream is converted to 0111001001100101. After changing 1 bit, the reconstructed keystream is obtained as 0**01101100101000**. Here the avalanche effect is demonstrated by measuring the number of the flipped bits in the reconstructed keystream (shown as bold data), and it is divided by the number of the bits in the keystream. Therefore, the avalanche effect is considered to be 50%.

The avalanche effect can be satisfied when an average of 50% of the output bits exhibit a difference with the change in one input bit. Here the avalanche test is computed by two variants: i) key and keystream and ii) key and ciphertext sequence by simply 1-bit change. Further, our avalanche result shows the extent of difference after alteration and is shown in table 5. Tabulated results reveal that mentioned stream ciphers achieve a good avalanche effect in both cases.

The overall statistical results show that the selected stream ciphers are effective for various statistical securities.

4. Performance analysis

The constrained environment [2] is extensively covered in many emerging fields of networking, like IoT. As discussed, the IoT devices are resource-constrained in nature and intended to work for a long time. Arduino is a low-cost open-source microcontroller [41]. It is widely used for the development of low-level programming. Additionally, Arduino is used in constrained networks like ZigBee, WiFi, Bluetooth, Z-Wave, ANT, etc. Here, we have analysed the performance of the selected ciphers using Arduino ATmega 328P resource-constrained microcontroller. The specification of the microcontroller is presented in table 6. The low-cost microcontroller Arduino has a single-core CPU, which we applied for controlling the parallel unit. For multitasking, it asynchronously controls the encryption of the output and decryption of the input. Further, this device can be connected to a computer through the USB port or alternatively connected with an AC-to-DC adapter and battery. The following subsection focuses on the result of the stream cipher's performance (in terms of time, energy efficiency, program size, throughput and speed), and discussion. Furthermore, details of the performance metrics are described in table 7.

4.1 *Results*

In this work, we have studied some stream ciphers (such as Lizard, Fruit, Plantlet, Sprout, Grain v1, Espresso, RC4, Trivium and Mickey) and block ciphers (such as Present, Klein, Kasumi, Hight, Simeck and Speck).

One of the main hurdles in achieving performance in Arduino is the utilization and deployment of cryptographic codes. Arduino has only 32 kilobytes of flash memory, 2 kilobytes of static random-access memory (SRAM) and 1 kilobyte of electrically erasable programmable read-only memory (EEPROM), which are very less compared with a computer-based system. In this condition, if one implements AES cipher to Arduino, cipher needs 10 rounds of processing for 128-bit keys. Every round of processing comprises one single-byte-based substitution step, a row-wise permutation step, a column-wise mixing step and the addition of the round key. The method of encryption will need some memory space to save temporary results and the last encrypted results, which go in flash memory. Cipher secret key is stored in the EEPROM. The problem becomes more aggravated when we use a large program code to port into devices having low resource availability.

We have optimized C codes of the selected stream ciphers before porting to ATmega328P. Ciphers C program code have been optimized in such a way that it becomes smaller in

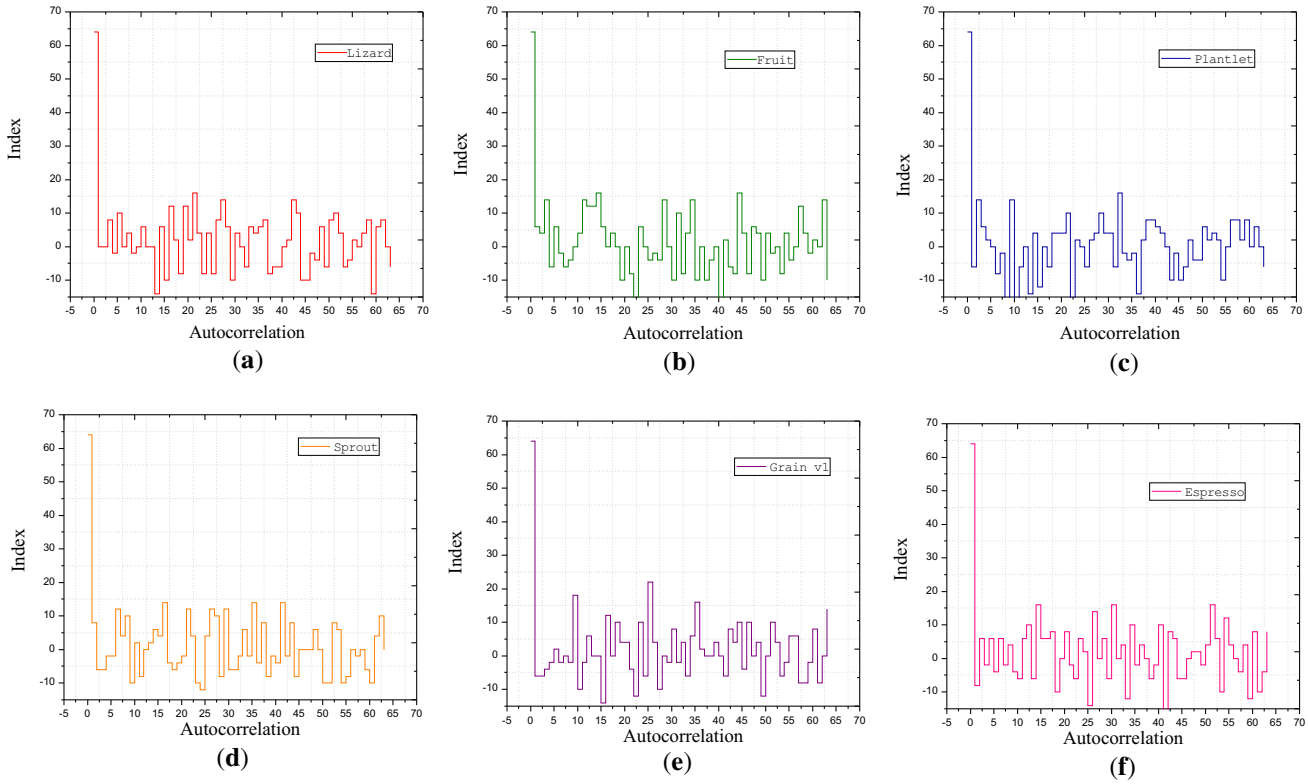

Figure 1. Autocorrelation results for (**a**) Lizard, (**b**) Fruit, (**c**) Plantlet, (**d**) Sprout, (**e**) Grain v1 and (**f**) Espresso.

Table 4. Statistical data analysis of the keystream.

Properties	Lizard	Fruit	Plantlet	Sprout	Grain v1	Espresso
Coincidence index	0.0641	0.05641	0.04838	0.05769	0.05384	0.06794
Normalized entropy	0.94502	0.94927	0.9698	0.94745	0.9587	0.94745
Standard deviation	0.03857	0.03415	0.03455	0.03535	0.03162	0.03455
Variance	0.00148	0.00116	0.00081	0.00125	0.001	0.00119
Variance (bias = true)	0.00138	0.00109	0.00076	0.00117	0.00093	0.0011
SEM ±	0.00996	0.00853	0.00958	0.00883	0.0079	0.00958

Table 5. Avalanche test for selected ciphers.

Name of the cipher	Avalanche effect% in key–keystream	Avalanche effect% in key–ciphertext
Lizard	49.9989%	48.9999%
Fruit	49.9899%	51.0169%
Plantlet	49.9933%	49.9899%
Sprout	49.9333%	52.0179%
Grain v1	50.0134%	49.9899%
Espresso	49.9869%	51.0133%

Avalanche effect% in key–keystream = (no. of the changing bits in the keystream/no. of bits in the keystream × 100%), and the same rule applies for avalanche effect% in key–ciphertext.

size, consumes less memory and executes faster to perform some input–output operations. Initially, we tried to identify the part of the code whose execution consumes more time. In the following paragraphs, we have briefly mentioned the optimization steps of stream ciphers codes to fit in the constrained environment and improve its performance.

A `for()` loop usually consumes considerable amount of time in microcontroller as discussed in [41]. In order to save variables between loops in Arduino, we have used the keyword `volatile` just before the integer type declaration `int`. In this way, we have been able to reduce the program memory requirement by 146 bytes. Next, through port manipulation we have achieved faster input-output control and there by saving the memory requirement. To measure the state of different input–output ports, three registers D, B and C are used [41].

For instance an optimized implementation of XOR operator is shown here. Let us consider $x = 1100$, $y = 1010$ and the result is 0110. The condition for the XOR operation is if both bits are the same, the resulting bit is a 0. If the bits are different, the result is a 1.

Table 6. Specification of ATmega328P.

Name	Value
Program memory type	Flash
Program memory size	32 (kB)
CPU speed	20 (MIPS/DMIPS)
SRAM	2 (kB)
EEPROM	1 (kB)
Timers	2 × 8-bit, 1 × 16-bit
Number of comparators	1
Temperature range	− 40 to 85 (°C)
Clock speed	16 (MHz)
Pin count	32
Voltage	5 (V)

Table 7. Performance measures.

Metric	Description
Code size	Memory size to store the cipher code and constants. In general, it remains in the flash memory.
RAM size	Memory size to store the intermediate states during the execution of the cipher code.
Throughput	Number of encrypted bits per second (kbps).
Speed	Number of cycles/byte to encrypt or decrypt per block.

```
void setup() {
  DDRB = DDRB | B00100000; // set PB11 as output
  Serial.begin(9600);
}

void loop() {
  PORTB = PORTB ^ B00100000; // invert PB11
  delay(100);
}
```

In this way, other bitwise operators like and, or, not, shift left, shift right can be efficiently performed by applying one or two registers. Compared with original ciphers codes, our optimized code size has been reduced by ≈ 18% for each of the ciphers' keystream generation processes. We have examined the optimized code in ATmega 328P, and measured the performance metrics like computation time (for keystream generation), energy, throughput and speed. Further, our result is also compared to the recent works (i.e., some block ciphers such as Simeck, Kasumi, Klein, Present, Hight and Speck) [7, 9, 12].

Any low-end device requires less computation time for output generation. We have compared the computational time to generate 256 byte output data for selected block and stream ciphers. Figure 2a illustrates the computation time results for the mentioned block and stream ciphers, namely Lizard, Fruit, Plantlet, Sprout, Grain v1, Espresso, RC4, Trivium, Mickey, Present, Klein, Kasumi, Hight, Simeck and Speck. As seen from figure 2a, lightweight ciphers require lesser computation time than the conventional block and stream ciphers. Specifically, figure 2a illustrates that Lizard, Sprout and Plantlet require remarkably less computation time compared with other ciphers.

The energy requirement is also one of the essential criteria for the IoT domain. In this part, we have computed the total energy requirement for selected cipher's keystream generation (calculated as per the specification);[4] it ran at a voltage of 5 V, and the resistance value of 150 Ω. The simulation result is depicted in figure 2b. As seen from figure 2b, the energy requirement gradually increases while the keystream bit size increases. We have also computed the program size (kilobyte) of the selected stream ciphers, which are ported in the microcontroller and graphically depicted in figure 2c. Notably, Grain v1 roughly requires the maximum and Lizard requires the minimum resources in terms of computation time, program size and energy requirement.

In the resource-constrained environment, low computational power in low-end devices has made the encryption and decryption procedures more challenging to implement at the device level. Thus, throughput metric plays a vital role in ensuring the performance of the scheme. Selected stream cipher's throughput has been verified by generating large keystream. Specifically, throughput is calculated by measuring the average amount of data (bytes) encrypted or decrypted per unit time and expressed as follows:

$$\text{throughput} = \frac{\text{no. of bytes encrypted/decrypted}}{\text{time}}. \quad (3)$$

Based on the equation as mentioned earlier, throughput has been computed for stream and block ciphers. Figure 3 illustrates the selected ciphers throughput values with respect to the size of the file. In this work, throughput has been shown in kilobytes per second (kbps). Figure 4 illustrates the throughput result of Simeck, Kasumi, Klein, Present, Hight and Speck. Several experiments are performed in the Arduino studio platform to evaluate the throughput performance of the selected block and stream ciphers. From throughput plots it is observed that when the file size increases, the selected stream cipher provides higher throughput than the block cipher. Notably, it is

[4]Atmega328/p datasheet [online]. Website http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf [accessed 09 January 2019].

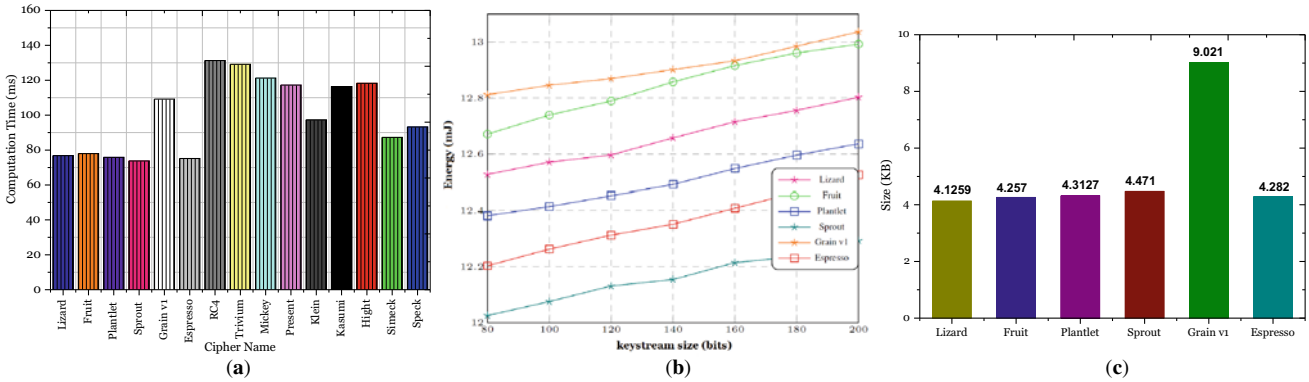

Figure 2. (**a**) Computation time for block and stream cipher, (**b**) effect of energy and (**c**) program size.

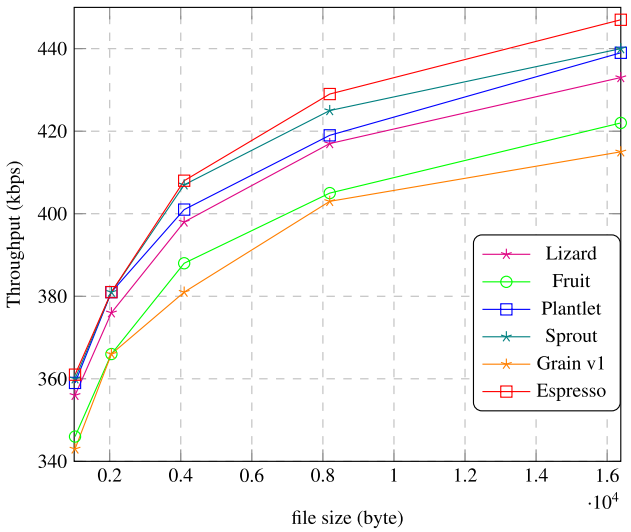

Figure 3. Throughput for stream ciphers.

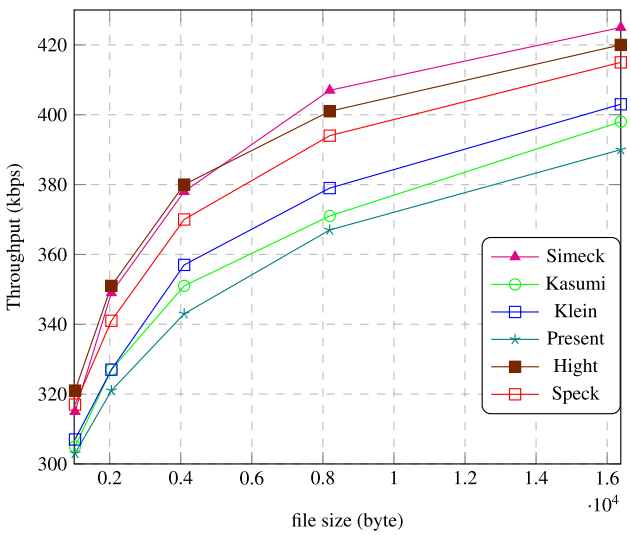

Figure 4. Throughput for block ciphers.

found that the Lizard cipher obtained high performance compared with that of other ciphers during all the intervals.

In this part, numerical analysis of selected stream ciphers' speed has been carried out. In general, speed analyses are presented to identify the performance of the ciphers. Speed $\mathcal{S}$ can be expressed by the following equation:[5]

$$\text{speed } \mathcal{S} = \sum \mathcal{S}_i \pi_i \tag{4}$$

where $\mathcal{S}_i$ is the cipher speed for the plaintext of i bytes and π_i is the proportion of plaintexts of i bytes to encipher.

To increase the cipher speed, it is recommended to use a processor with high computational capacity. Here, speed has been calculated by measuring the encrypted packet per unit time. In this case, encryption packets size is 350 packets/40 bytes. Figure 5a depicts speed performance (bytes/μs, and packets/μs) of the selected stream ciphers. A comparison of speed performance with current block ciphers has been made, and illustrated in figure 5b. It is noticed that our selected stream ciphers provide similar results compared to mentioned lightweight block ciphers [9]. Block ciphers and stream ciphers speed evaluation plots merely indicate that Hight, Present and Grain v1 give slightly lower speed, whereas Lizard has ideal speed across different trials.

4.2 *Discussion*

The NIST lightweight cryptography report initially focused on symmetric cryptography. In this study, we have also focused our attention on lightweight stream ciphers. In providing security in devices having different level of resource constrainedness, researchers have faced difficulties in maintaining the trade-off between security level

[5]C Lauradoux – throughput/code size trade-off for stream ciphers. The State of the Art of Stream Ciphers-SASC [online]. Website http://www.ecrypt.eu.org/stream/papersdir/2007/018.pdf [accessed 09 January 2019].

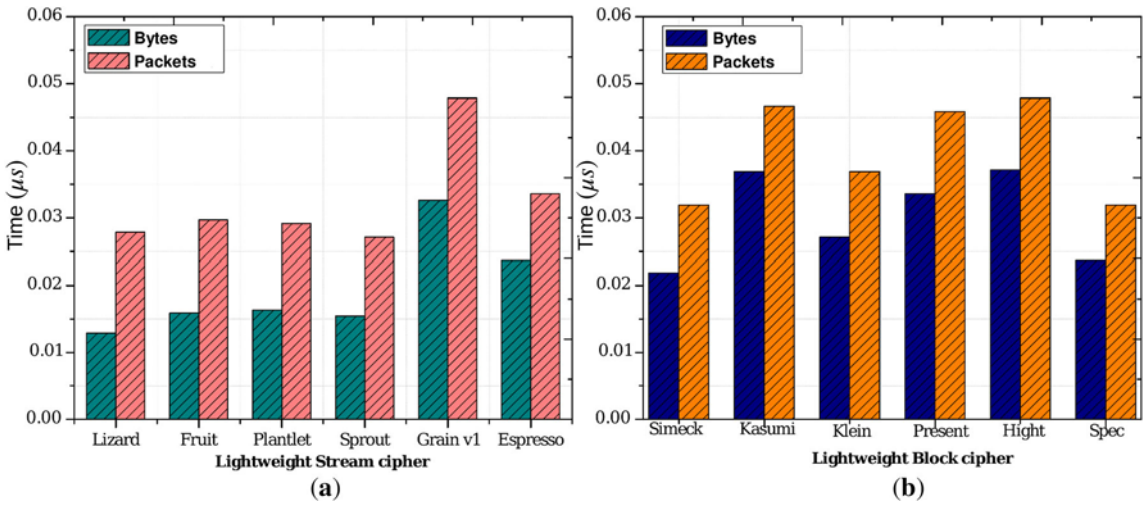

Figure 5. Speed evaluation: (**a**) selected stream ciphers and (**b**) block ciphers.

required and the various resource constraints like power, costs, etc.

As discussed in introduction, Sprout cipher provides the novel idea to design stream cipher with smaller internal state. In literature, it has been found that many high-end attacks are possible on Sprout stream cipher [17, 23]. However, high-end attacks require a lot of resources, and NIST has suggested these attacks may not be considered in lightweight cryptography.

80-bit security is considered to be the minimum security requirement for lightweight cryptographic applications [12]. In our statistical analysis, we have observed that Sprout has passed many of the statistical tests. Inspite of Sprout showing good statistical properties, there are attacks on Sprout [22, 36, 37] which can recover the secret key in less than 2^{80} attempts and hence Sprout is not acceptable for lightweight cryptography.

Stream ciphers may be broken by side-channel attacks, which employ partial information leaked through encryption method. However, most of the authors of stream ciphers based on shorter internal state claimed that the ciphers are capable of resisting side-channel attacks [13, 17–19].

The following observation is made based on the implementation of several stream ciphers and block ciphers by many methods.

Observation 1 *Accurate and comprehensive nature analysis of the selected stream cipher is a complicated task. It is noticed that the small state stream ciphers use a comparatively smaller internal state, but extremely complex functions are used for random number generation. Table 2 shows that the selected ciphers provide better randomness results. Moreover, the selected ciphers provide similar randomness result compared to some famous symmetric ciphers like RC4, Trivium, MICKEY, AES and DES (validated our claims via implementation). Additionally, selected lightweight stream ciphers can be used for an underlying challenge–response authentication protocol in the small computing devices* [18]. *From the complete experimentation, it is found that Lizard cipher design choices are better than other smaller state stream ciphers. Lizard provides (2n/3)-bound on the security against TMDTO key recovery attacks (where n is 120), i.e. 80-bit security, which is sufficient for lightweight applications.*

Our overall discussion may be useful for choosing cryptographic primitives in the resource-constrained environment.

5. Conclusion

The emergence of various promising lightweight applications has motivated academia–industries to develop lightweight cryptographic algorithms. The lightweight cryptographic algorithm should maintain the balance between the security and performance level. This paper shows that the stream cipher possesses an effective security solution for resource-constrained devices. Our analysis shows that the selected lightweight stream ciphers, viz. Lizard, Fruit, Plantlet, Sprout, Grain v1 and Espresso, provide high-level statistical security as demonstrated by the results of randomness test, structural test, autocorrelation test and avalanche test. The selected ciphers are also ported in low-cost Arduino ATmega328P microcontroller, and their performance is measured based on the metrics like computation time, energy, memory requirement and throughput. The comparative analysis reveals that the selected ciphers are better suited for resource-constrained devices.

References

[1] Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M and Ayyash M 2015 Internet of things: a survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutori* 17(4): 2347–2376

[2] Sfar A R, Natalizio E, Challal Y and Chtourou Z 2018 A roadmap for security challenges in the Internet of Things. *Digit.Commun. Netw.* 4(2): 118–137

[3] He D, Kumar N and Chilamkurti N 2015 A secure temporal-credential-based mutual authentication and key agreement scheme with pseudo identity for wireless sensor networks. *Inf. Sci.* 321: 263–277

[4] Jiang Q, Ma J, Wei F, Tian Y, Shen J and Yang Y 2016 An untraceable temporal-credential-based two-factor authentication scheme using ECC for wireless sensor networks. *J. Netw. Comput. Appl.* 76: 37–48

[5] Deb S, Bhuyan B and Gupta N C 2018 Design and analysis of LFSR-based stream cipher. In: *Proceedings of the International Conference on Computing and Communication Systems*, Shillong, India, pp. 631–639

[6] Ashokkumar C, Venkatesh M B, Giri R P, Roy B and Menezes B 2019 An error-tolerant approach for efficient AES key retrieval in the presence of cache prefetching—experiments, results, analysis

[7] Pei C, Xiao Y, Liang W and Han X 2018 Trade-off of security and performance of lightweight block ciphers in Industrial Wireless Sensor Networks. *EURASIP J. Wirel. Commun. Netw.* 2018(1): 117

[8] Deb S and Bhuyan B 2018 Performance evaluation of Grain family and Espresso ciphers for applications on resource constrained devices. *ICT Express* 4(1): 19–23

[9] Qasaimeh M, Al-Qassas R S and Tedmori S 2018 Software randomness analysis and evaluation of lightweight ciphers: the prospective for IoT security. *Multimed. Tools Appl.* 77(14): 18415–18449. *Sādhanā* 44(4): 88

[10] Bansod G, Raval N and Pisharoty N 2014 Implementation of a new lightweight encryption design for embedded security. *IEEE Trans. Inf. Forensics Secur.* 10(1): 142–151

[11] Hell M, Johansson T and Meier W 2007 Grain: a stream cipher for constrained environments. *Int. J. Wirel. Mobile Comput.* 2(1): 86–93

[12] Dinu D, Le Corre Y, Khovratovich D, Perrin L, Großschädl J and Biryukov A 2015 Triathlon of lightweight block ciphers for the internet of things. *J. Cryptogr. Eng.* 1–20

[13] Armknecht F and Mikhalev V 2015 On lightweight stream ciphers with shorter internal states. In: *Proceedings of the International Workshop on Fast Software Encryption*, Istanbul, Turkey, March 8–11. *Lecture Notes in Computer Science* 9054, pp. 451–470

[14] Manifavas C, Hatzivasilis G, Fysarakis K and Papaefstathiou Y 2016 A survey of lightweight stream ciphers for embedded systems. *Secur. Commun. Netw.* 9(10): 1226–1246

[15] Banik S, Isobe T and Morii M 2018 On design of robust lightweight stream cipher with short internal state. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 101(1): 99–109

[16] Deb S, Upadhya R and Bhuyan B 2018 Applicability of lightweight stream cipher in crowd computing: a detailed survey and analysis. In: Pathan A S K (Ed.) *Crowd Assisted Networking and Computing*, 1st ed. Boca Raton, Florida, USA: CRC Press, pp. 245–272

[17] Vahid Amin Ghafari H H and Chen Y 2016 Fruit-v2: ultra-lightweight stream cipher with shorter internal state. *IACR Cryptol. ePrint Arch.* 2016: 355

[18] Hamann M, Krause M and Meier W 2016 LIZARD – a lightweight stream cipher for power-constrained devices. *IACR Cryptol. ePrint Arch.* 2016: 926

[19] Mikhalev V, Armknecht F and Müller C 2016 On ciphers that continuously access the non-volatile key. *IACR Trans. Symmetric Cryptol.* 2: 52–79

[20] Dubrova E and Hell M 2017 Espresso: a stream cipher for 5G wireless communication systems. *Cryptogr. Commun.* 9(2): 273–289

[21] Esgin M F and Kara O 2015 Practical cryptanalysis of full sprout with TMD tradeoff attacks. In: *Proceedings of the International Conference on Selected Areas in Cryptography*, Sackville, Canada, August 10–12, pp. 67–85

[22] Maitra S, Sarkar S, Baksi A and Dey P 2015 Key recovery from state information of sprout: application to cryptanalysis and fault attack. *IACR Cryptol. ePrint Arch.* 2015: 236

[23] Banik S 2015 Some results on sprout. In: *Proceedings of Progress in Cryptology – INDOCRYPT*, Bangalore, India, pp. 124–139

[24] Mosenia A and Jha N K 2017 A comprehensive study of security of internet-of-things. *IEEE Trans. Emerg. Top. Comput.* 5(4): 586–602

[25] Yang Y, Wu L, Yin G, Li L and Zhao H 2017 A survey on security and privacy issues in Internet-of-Things. *IEEE Internet of Things J.* 4(5): 1250–1258

[26] Bogdanov A, Knudsen L R, Leander G, Paar C, Poschmann A, Robshaw M J, Seurin Y and Vikkelsoe C 2007 PRESENT: an ultra-lightweight block cipher. In: *Proceedings of Cryptographic Hardware and Embedded Systems – CHES*, Vienna, Austria, pp. 450–466

[27] Gong Z, Nikova S and Law Y W 2011 KLEIN: a new family of lightweight block ciphers. In: *Proceedings of Radio Frequency Identification: Security and Privacy Issues – RFIDSec*, Amherst, USA, pp. 1–18

[28] Hong D, Sung J, Hong S, Lim J, Lee S, Koo B S, Lee C, Chang D, Lee J, Jeong K and Kim H 2006 HIGHT: a new block cipher suitable for low-resource device. In: *Proceedings of Cryptographic Hardware and Embedded Systems – CHES*, Yokohama, Japan, pp. 46–59

[29] Yang G, Zhu B, Suder V, Aagaard M D and Gong G 2015 The Simeck family of lightweight block ciphers. *IACR Cryptol. ePrint Arch.* 2015: 612

[30] Beaulieu R, Shors D, Smith J, Treatman-Clark S, Weeks B and Wingers L 2013 The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptol. ePrint Arch.* 2013: 404

[31] Biryukov A and Perrin L 2017 State of the art in lightweight symmetric cryptography. *IACR Cryptol. ePrint Arch.* 2017: 511

[32] Fournel N, Minier M and Ubéda S 2007 Survey and benchmark of stream ciphers for wireless sensor networks. In: *Proceedings of the International Workshop on Information Security Theory and Practices*, Heraklion, Crete, Greece, pp. 202–214

[33] Maitra S, Sinha N, Siddhanti A, Anand R and Gangopadhyay S 2018 A TMDTO attack against Lizard. *IEEE Trans. Comput.* 67(5): 733–739

[34] Banik S 2014 Some insights into differential cryptanalysis of Grain v1. In: *Proceedings of the Australasian Conference on Information Security and Privacy*, Wollongong, NSW, Australia, March 8–11. *Lecture Notes in Computer Science* 8544, pp. 34–49

[35] Banik S, Barooti K and Isobe T 2019 Cryptanalysis of Plantlet. *IACR Trans. Symmetric Cryptol.* 3: 103–120

[36] Zhang B and Gong X 2015 Another tradeoff attack on sprout-like stream ciphers. *IACR Cryptol. ePrint Arch.* 2015: 94520210

[37] Esgin M F and Kara O 2015 Practical cryptanalysis of full sprout with TMD tradeoff attacks. *IACR Cryptol. ePrint Arch.* 2015: 289

[38] Petura O, Mureddu U, Bochard N, Fischer V and Bossuet L 2016 A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices. In: *Proceedings of the 26th International Conference on Field Programmable Logic and Applications (FPL)*, Lausanne, Switzerland, pp. 1–10

[39] Balasch J, Bernard F, Fischer V, Grujić M, Laban M, Petura O, Rožić V, Van Battum G, Verbauwhede I, Wakker M and Yang B 2018 Design and testing methodologies for true random number generators towards industry certification. In: *Proceedings of the 23rd IEEE European Test Symposium (ETS)*, Bremen, Germany, pp. 1–10

[40] Turan M S, Doganaksoy A and Calik C 2006 Statistical analysis of synchronous stream ciphers. In: *Proceedings of The State of the Art of Stream Ciphers – SASC 2006: Stream Ciphers Revisited*, Leuven, Belgium, pp. 84–93

[41] Wheat D 2012 Arduino software. In: *Arduino Internals*. Apress

[42] Kavun E B and Yalcin T 2010 A lightweight implementation of Keccak Hash function for radio-frequency identification applications. In: *Proceedings of Radio Frequency Identification: Security and Privacy Issues – RFIDSec*, Istanbul, Turkey, pp. 258–269

[43] Bogdanov A, Knezevic M, Leander G, Toz D, Varici K and Verbauwhede I 2011 spongent: a lightweight hash function. In: *Proceedings of Cryptographic Hardware and Embedded Systems – CHES*, Nara, Japan, pp. 312–325