Lizard Cipher for IoT Security on Constrained Devices

Ari Kusyanti[1], Rakhmadhany Primananda[2], Kalbuadi Joyo Saputro[3]
Department of Information Technology
Universitas Brawijaya
Malang, Indonesia

***Abstract*—Over the past decades, security become the most challenging task in Internet of Things. Therefore, a convenient hardware cryptographic module is required to provide accelerated cryptographic operations such as encryption. This study investigates the implementation of Lizard cipher on three Arduino platforms to determine its performance. This study is successful in implementing Lizard cipher on Arduino platform as a constrained devices and resulting 0.98 MB of memory utilization. The execution time of Lizard cipher is compared among Arduino variants, i.e Arduino Mega, Arduino Nano and Arduino Uno with ANOVA test. Tukey's HSD post-hoc test reveales that the execution time is significantly slower in Arduino Mega compared to Arduino Nano and Arduino Uno. This result will help IoT security engineers in selecting a lightweight cipher that is suitable for constraints of the target device.**

Keywords—Lizard cipher; IoT security; Arduino; ANOVA

I. Introduction

In recent years, Internet of Things (IoT) gets its popularity due to its integrated architecture that connects 'things' to share information between them. Microcontroller is one of the 'things' that is widely adopted for IoT considering its various functionalities. Since the increasing numbers of devices that connect to IoT, the possibility of security violation is also increasing [1] [2]. One of the solutions is by encrypting the data that can reduce the security risk. There is solution for the problem based on hardware which is called Hardware Security Modules (HSM). However HSM are expensive to be implemented in IoT.

For this purpose, three widely used target microcontrollers in the Internet of Things context, i.e. Arduino Uno, Nano and Mega are selected. Arduino is an easy to use and low cost device and equipped with open source platform. Arduino utilizes a hardware known as the Arduino development board and software for programming the code known as the Arduino IDE (Integrated Development Environment).

Apart from the targeted devices, the cipher itself has to be taken into consideration when implemented in Internet of Things environment. Lizard [3] is created with low-cost scenarios in mind. The design of Lizard is inspired from Grain v1 [4], but with the smaller inner state which is 121 bits and the larger key of 120 bits. It outperforms Grain v1 in memory area usage and power consumption; hence Lizard is suitable for power-constrained devices.

The objective of this study is to investigate the implementation of Lizard cipher on three Arduino platforms to determine its performance i, i.e. Arduino Mega, Arduino Uno and Arduino Nano. To evaluate the performance, the memory utilization and execution time of Lizard is observed. To analyze the data, ANOVA test is used to examine the difference implementation of Lizard on three Arduino.

II. Related Study

Based on previous studies, a number of cryptographic algorithms had been implemented on microcontrollers, such as [5] which evaluates the performance of public key cryptography such as RSA and ECC with various libraries on Arduino. The work of [6] had evaluated 19 lightweight ciphers on three microcontroller platforms: 8-bit AVR, 16-bit MSP430, and 32-bit ARM. The ciphers are all block cipher namely AES [7], Chaskey [8], Fantomas [9], HIGHT [10], LBlock [11], LEA [12], LED [13], Piccolo [14], PRESENT [15], PRIDE [16], PRINCE [17], RC5 [18], RECTANGLE [19], RoadRunneR [20], Robin [21], Simon [22], SPARX [23], Speck [24], and TWINE [25]. According to [26] stream ciphers tend to perform faster and more efficient than block ciphers. Therefore in this study, Lizard cipher as one of stream cipher is evaluated so that it can help IoT security engineers when selecting a lightweight ciphers that suits the requirements of the constraints of the target device.

III. Lizard Architecture and Arduino

This section will disscuss the overview of Lizard cipher [3] and the target devices i.e. Arduino Mega, Arduino Nano and Arduino Uno, also the test to analyze the data observed namely ANOVA test.

A. Lizard Cipher

In this section, the details of Lizard cipher are discussed. First, the components of the cipher in detail is described. Then, initialization state is specified and finally keystream generation process is described.

Components of Lizard cipher:

Lizard cipher has three main blocks, i.e NFSR (Non Linear Feedback Shift Register) i.e. NFSR1 and NFSR2, and output function a as depicted in Fig. 1. Lizard requires inputs of 120 bits key and 64 bits IV (Initiation Vector) and produces one bit keystream at a time.

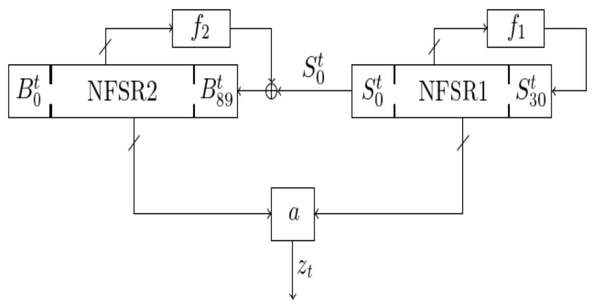

Fig. 1. Lizard Architecture.

NFSR (Non Linear Feedback Shift Register)

The first component on Lizard is NFSR which consist of two blocks, i.e NFSR1 and NFSR2. The difference between these two is in the feedback function. NFSR1's content is denoted as s_i, s_{i+1}, , , s_{i+79}, while NFSR2's is b_i, b_{i+1}, , , b_{i+63}. As for NFSR1, the update relation of NFSR1 is defined as

$$\begin{aligned}S_{30}^{t+1} = {} & S_0^t \oplus S_2^t \oplus S_5^t \oplus S_6^t \oplus S_{15}^t \oplus S_{17}^t \oplus S_{18}^t \oplus S_{20}^t \\ & \oplus S_{25}^t \oplus S_8^t S_{18}^t \oplus S_8^t S_{20}^t \oplus S_{12}^t S_{21}^t \\ & \oplus S_{14}^t S_{19}^t \oplus S_{17}^t S_{21}^t \oplus S_{20}^t S_{22}^t \\ & \oplus S_4^t S_{12}^t S_{22}^t \oplus S_4^t S_{19}^t S_{22}^t \oplus S_7^t S_{20}^t S_{21}^t \\ & \oplus S_8^t S_{18}^t S_{22}^t \oplus S_8^t S_{20}^t S_{22}^t \oplus S_{12}^t S_{19}^t S_{22}^t \\ & \oplus S_{20}^t S_{21}^t S_{22}^t \oplus S_4^t S_7^t S_{19}^t S_{21}^t \\ & \oplus S_4^t S_{12}^t S_{21}^t S_{22}^t \oplus S_4^t S_{19}^t S_{21}^t S_{22}^t \\ & \oplus S_7^t S_8^t S_{18}^t S_{21}^t \oplus S_7^t S_8^t S_{20}^t S_{21}^t \\ & \oplus S_7^t S_{12}^t S_{19}^t S_{21}^t \oplus S_8^t S_{18}^t S_{21}^t S_{22}^t \\ & \oplus S_8^t S_{20}^t S_{21}^t S_{22}^t \oplus S_{12}^t S_{19}^t S_{21}^t S_{22}^t\end{aligned}$$

Whilst update relation of NFSR2 is defined as

$$\begin{aligned}B_{89}^{t+1} = {} & S_0^t \oplus B_0^t \oplus B_{24}^t \oplus B_{49}^t \oplus B_{79}^t \oplus B_{84}^t \oplus B_3^t B_{59}^t \oplus B_{10}^t B_{12}^t \\ & \oplus B_{15}^t B_{16}^t \oplus B_{25}^t B_{53}^t \oplus B_{35}^t B_{42}^t \oplus B_{55}^t B_{58}^t \\ & \oplus B_{60}^t B_{74}^t \oplus B_{20}^t B_{22}^t B_{23}^t \oplus B_{62}^t B_{68}^t B_{72}^t \\ & \oplus B_{77}^t B_{80}^t B_{81}^t B_{83}^t\end{aligned}$$

Output Function

The final block of output function is defined as

$$z_t \coloneqq \mathcal{L}_t \oplus Q_t \oplus T_t \oplus T'_t$$

where

$$\mathcal{L}_t \coloneqq B_7^t \oplus B_{11}^t \oplus B_{30}^t \oplus B_{40}^t \oplus B_{45}^t \oplus B_{54}^t \oplus B_{71}^t$$

$$Q_t \coloneqq B_4^t B_{21}^t \oplus B_9^t B_{52}^t \oplus B_{18}^t B_{37}^t \oplus B_{44}^t B_{76}^t$$

$$\begin{aligned}T_t \coloneqq {} & B_5^t \oplus B_8^t B_{82}^t \oplus B_{34}^t B_{67}^t B_{73}^t \oplus B_2^t B_{28}^t B_{41}^t B_{65}^t \\ & \oplus B_{13}^t B_{29}^t B_{50}^t B_{64}^t B_{75}^t \oplus B_6^t B_{14}^t B_{26}^t B_{32}^t B_{47}^t B_{61}^t \\ & \oplus B_1^t B_{19}^t B_{27}^t B_{43}^t B_{57}^t B_{66}^t B_{78}^t\end{aligned}$$

$$T'_t \coloneqq S_{23}^t \oplus S_3^t S_{16}^t \oplus S_9^t S_{13}^t B_{48}^t \oplus S_1^t S_{24}^t B_{38}^t B_{63}^t$$

Initiaization state

The state initialization process can be divided into 4 phases, namely

a) Phase 1: Key and IV Loading

b) Phase 2: Grain-like Mixing

c) Phase 3: Second Key Addition

d) Phase 4: Final Diffusion

Prior to generating any keystream, there are four phases that has to be completed. Firstly, the cipher has to be initialized with the key and the IV. Let the bits of the key, k, denoted as k_i, $0 \le i \le 119$ and the bits of the IV denoted as IV_i, $0 \le i \le 63$. The initialization process is performed as follows.

$$B_j^0 = \begin{cases} K_j \oplus IV_j, & for\ j \in \{0, \dots, 63\}, \\ K_j, & for\ j \in \{64, \dots, 89\}, \end{cases}$$

$$S_i^0 \coloneqq \begin{cases} K_{i+90}, & for\ i \in \{0, \dots, 28\}, \\ K_j \oplus 1, & for\ i = 29, \\ 1, & for\ i = 30. \end{cases}$$

Secondly, the Grain-mixing like process, the cipher is clocked 128 times without producing any running key. In this process, the output function is fed back and xored with the input, both to the NFSR1 and to the NFSR2 as shown in Fig. 2. Similiarly for the phase 3, second key addition process is performed. After this initialization process is complete, keystream is generated in phase 4.

Key Generation state:

After completing all four stages, the first keystream bit that is used for plaintext encryption is z_{257}. The length of the keystream depends on the length of plaintext, since it will be encrypted bit by bit to produce ciphertext.

B. Arduino

Arduino is a microcontroller which can be easily used, low-cost and has capability of mini-computer [27]. Arduino builds upon a hardware known as the Arduino development board and software for developing the code known as the Arduino IDE (Integrated Development Environment). There are a variety of Arduino models with various features and enhancement in their latest versions that add more components. In this study, three types of Arduino are used, i.e. Arduino Mega, Arduino Uno, Arduino Nano.

The main difference among these Arduino is the concern of this study which lies on the processor. Arduino Mega 2560 features an ATmega2560 at its heart, while Arduino Nano is equipped with ATmega168 and Arduino Uno is based on ATmega328 MCU.

Table I presents the main characteristics of the target devices used, while in the next paragraphs provide a brief description of each Arduino type.

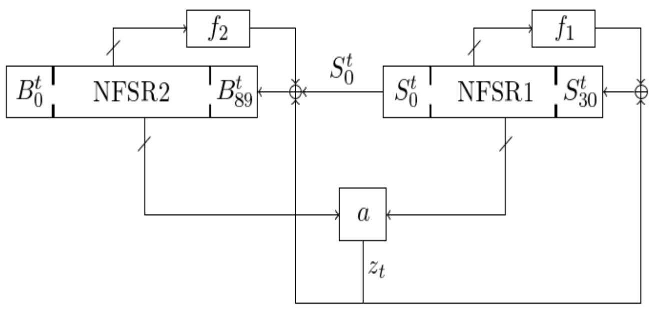

Fig. 2. Lizard State Iniziation.

TABLE. I. ARDUINO TYPE

Parameter	Nano	Uno	Mega
Processor	ATmega168	ATmega 328P	ATmega2560
Clock Speed	16 MHz	16 MHz	16 MHz
Flash Memory	16 kB	32 kB	256 kB
EEPROM	512 bytes	1 kB	4 kB
SRAM	1 kB	2 kB	8 kB
Voltage Level	5 V	5 V	5 V

The first target, Arduino Nano with ATmega168 as the processor has a maximum frequency of 16 MHz and with 1kB SRAM and 16 KB flash memory. The second target is Arduino Uno with ATMega328P as its heart. The processor has 2kB SRAM and 32kB flash memory. Lastly, Arduino Mega with ATmega 2560 processor is equipped with 8 KB SRAM and 256 KB flash memory. All of these three operate at V = 5 V.

C. *A One-Way Analysis of Variance (ANOVA)*

In the process of examining the relationship between variables, ANOVA is used to compare the means of two or more groups on the dependent variable. ANOVA a hypothesis-testing technique compares the variance between samples to variation within each sample. When the value of between variation is bigger than the within variation, hence the means of different samples is not equal. On the other hand, when the between and within variations are the same, hence there is no significant difference between sample means.

Before analyzing the data with ANOVA, the population has to follow a normal distribution, hence it is categorize as a parametric tests. On the contrary, an ANOVA test cannot be used to test the equality of the sample means. In this case, a non-parametric test is used since it does not rely on distributional assumptions. There are several assumptions that have to be met:

a) Assumption of independence: The samples have to be selected randomly independent to one another.

b) Assumption of normality: The population has to follow a normal distribution.

c) Assumption of homogeneity of variance: The variance of population has to be equal.

In ANOVA there are two hypotheses which are null hypothesis (H0) and alternative hypothesis (Ha). The null hypothesis for an ANOVA assumes the population means are equal, while alternative hypothesis assumes that at least one mean is not statistically equal.

In summary, an ANOVA test is used to determine at least one mean is different. However, an additional test must be conducted to determine which mean(s) is/are different which is called post-hoc test. There are several techniques for testing the differences between means, Tukey's HSD is used as the post-hoc test that helps to describe further data analysis technique for this study.

IV. IMPLEMENTATION

Arduino is an open source microcontroller board [27] that can be used to program by using free development software. The Arduino uses a simplified version of C/C++ programming language. Integrated Development Environment (IDE) is used to program and configure the Arduino which can be downloaded from the official website of Arduino freely.

The implementation of Lizard cipher can be an invaluable part of an overall security solution in IoT which provides accelerated cryptographic operations. Lizard cipher is successfully implemented in various Arduino by using Arduino IDE. It is able to generate keystream and is tested against test vectors. The experiment run 800 times for each Arduino with various combinations of Key and IV as inputs which generates keystream.

Descriptive statistics is conducted to observe the distribution of the data. Table II displays the summary of the descriptive statistics.

The average of execution time is 55813.4850 microsecond for Arduino Mega, 55596.5700 microsecond for Arduino Uno, and 55596.5700 microsecond for Arduino Nano as can be seen in Fig. 3.

Meanwhile, memory utilization is 0.98 MB for each Arduino. With less than 1 MB of memory utilization indicated that with fewer resources for cryptography leaves more resources available to applications. The collected data then analyzed by using ANOVA to determine whether there is statistically significant difference of execution time among Arduino Mega, Arduino Uno and Arduino Nano.

The null hypothesis for an ANOVA assumes the population means are equal. Hence, the null hypothesis as:

Ho: The mean of execution time of Lizard cipher is statistically equal across the three types of Arduino.

TABLE. II. DESCRIPTIVE STATISTIC

Execution Time	N	Mean	Std. Deviation
Arduino Mega	800	55813.4850	151.26459
Arduino Nano	800	55596.5700	151.29735
Arduino Uno	800	55596.5700	151.29735
Total	2400	55668.8750	182.56201

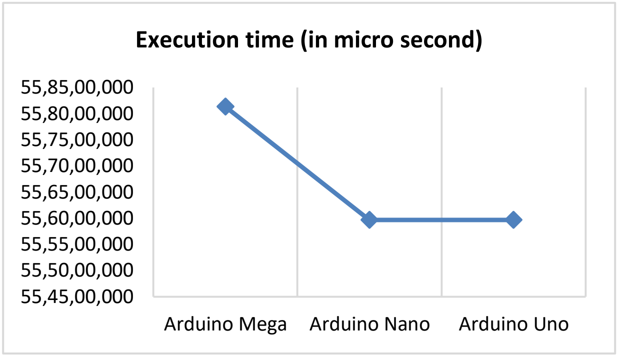

Fig. 3. Execution Time of Lizard.

The alternative hypothesis is:

Ha: At least one mean of execution time of Lizard cipher is not statistically equal.

There are several assumptions that has to be met before conducting ANOVA, i.e. assumption of independence, assumption of normality, and assumption of homogeneity of variance. As for assumption of independence, the sample has to be random and the value of one observation is not related to any other observation. This assumption is met since the data are taken from each Arduino individually without interfering one another. The second assumption is assumption of normality which indicates that the data are normally distributed. Based on the result of Kolmogorov-Smirnov test, the data is normal, so it can be proceed further. The last assumption is assumption of homogeneity of variance. By using Levene's F test the variances of the distributions in the data is equal. The result of ANOVA test is presented in Tables III to V. An Alpha level of .05 is used for all analyses. Based on Table III, the test for homogeneity of variance is not significant [LeveneF(2, 2397) = 0.00, p > .05] indicating that this assumption underlying the application of ANOVA is met.

ANOVA is used to examine whether the execution time of Lizard cipher is a function of the three Arduino variant. The independent variable represented the three different types of Arduino: 1) Arduino Mega; 2) Arduino Uno; and 3) Arduino Nano. The dependent variable is the execution time of Lizard cipher.

This study is successful in attaining main goal in determining whether Lizard cipher can be implemented in three variant of Arduino platform. It presents promising results that Lizard cipher occupies less than 0.98 MB of memory utilization for cryptography in Arduino which leaves enough memory available for other functionalities. The execution time of Lizard cipher is observed and compared among Arduino variants, i.e. Arduino Mega, Arduino Nano and Arduino Uno. There is a significant difference in execution time of Lizard cipher between Arduino Mega and Arduino Uno (*p* = 0.00), as well as between Arduino Mega and Arduino Nano (*p* = 0.00). However, there are no differences between Arduino Nano and Arduino Uno (*p* = 1.00).

TABLE. III. TEST OF HOMOGENEITY OF VARIANCES

Levene Statistic	df1	df2	Sig.
.000	2	2397	1.000

TABLE. IV. ANOVA EXECUTION TIME

Source	Sum of Square	Df	Mean Square	F	Sig.
Between Groups	25094462 .520	2	12547231 .260	548 .211	.000
Within Groups	54861539 .980	2397	22887 .584		
Total	79956002 .500	2399			

TABLE. V. POST-HOC TEST TUKEY'S HSD

(I) Arduino	(J) Arduino	Mean Difference (I-J)	Std. Error	Sig.
Mega	Nano	216.91500*	7.56432	.000
	Uno	216.91500*	7.56432	.000
Nano	Mega	-216.91500*	7.56432	.000
	Uno	.00000	7.56432	1.000
Uno	Mega	-216.91500*	7.56432	.000
	Nano	.00000	7.56432	1.000

Tukey's HSD procedures are used to determine which pairs of the three group means differed. Tukey's HSD is used since the assumption of homogeneity is met. According to Table V, Tukey's HSD post hoc test reveales that the execution time is significantly slower in Arduino Mega (55813.4850 ± 151.26459, p = .000) and Arduino Nano (55596.5700 ± 151.29735, p = .000) compared to Arduino Uno (55596.5700 ± 151.29735). There is no significant difference in execution time of Lizard cipher between Arduino Uno and Arduino Nano (p = 1.00).

V. CONCLUSION

This study is successful in attaining main goal in determining whether Lizard cipher can be implemented in three variant of Arduino platform to perform a number of important security-related functions. The result presents promising results that Lizard occupy less than 1 MB of memory utilization for cryptography in Arduino which will be enough memory available for other functionalities. The execution time of Lizard cipher are observed and compared among Arduino variants, i.e. Arduino Mega, ArduinoNao and Arduino Uno. There is a significant difference in execution time between Arduino Mega and Arduino Uno (p = 0.00), as well as between Arduino Mega and Arduino Nano (p = 0.00). However, there are no differences between Arduino Nano and Arduino Uno (p = 1.00). This result will assist IoT security engineers in choosing a lightweight cipher that suitable for constraint device.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] S. A. Kumar, T. Vealey, and H. Srivastava, "Security in internet of things: Challenges, solutions and future directions," in 2016 49th Hawaii International Conference on System Sciences (HICSS). IEEE, 2016, pp. 5772–5781.

[3] Hamann, M., Krause, M., & Meier, W., 2017. LIZARD – A Lightweight Stream Cipher for Power-constrained Devices. Germany: University of Mannheim.

[4] Hell, M., Johansson, T., Meier, W.: Grain – A Stream Cipher for Constrained Environments. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010 (2005) http://www.ecrypt.eu.org/stream https://www.arduino.cc/en/main/boards [accessed on January 2019].

[5] A. D. Elbayoumy and Simon J. Shepherd. Stream or Block Cipher for Securing VoIP?. International Journal of Network Security, Vol.5, No.2, PP.128–133, Sept. 2007 128.

[6] Daniel, D.; Le Corre, Y.; Khovratovich, D.; Perrin, L.; Grobschadl, J.; Biryukov, A. Triathlon of Lightweight Block Ciphers for the Internet of Things. 2015. Available online: http://orbilu.uni.lu/bitstream/10993/25565/1/209.pdf.

[7] National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES). FIPS Publication 197, available for download at http://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf, 2001.

[8] N. Mouha, B. Mennink, A. Van Herrewege, D. Watanabe, B. Preneel, and I. Verbauwhede. Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In A. Joux and A. M. Youssef, editors, Selected Areas in Cryptography — SAC 2014, volume 8781 of Lecture Notes in Computer Science, pages 306–323. Springer Verlag, 2014.

[9] V. Grosso, G. Leurent, F.-X. Standaert, and K. Varici. LS-designs: Bitslice encryption for efficient masked software implementations. In C. Cid and C. Rechberger, editors, Fast Software Encryption — FSE 2014, volume 8540 of Lecture Notes in Computer Science, pages 18–37. Springer Verlag, 2015.

[10] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A new block cipher suitable for low-resource device. In L. Goubin and M. Matsui, editors, Cryptographic Hardware and Embedded Systems — CHES 2006, volume 4249 of Lecture Notes in Computer Science, pages 46–59. Springer Verlag, 2006.

[11] W. Wu and L. Zhang. LBlock: A lightweight block cipher. In J. López and G. Tsudik, editors, Applied Cryptography and Network Security — ACNS 2011, volume 6715 of Lecture Notes in Computer Science, pages 327–344. Springer Verlag, 2011.

[12] D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D. Lee. LEA: A 128-bit block cipher for fast encryption on common processors. In Y. Kim, H. Lee, and A. Perrig, editors, Information Security Applications — WISA 2013, volume 8267 of Lecture Notes in Computer Science, pages 3–27. Springer Verlag, 2013.

[13] . J. Guo, T. Peyrin, A. Poschmann, and M. J. Robshaw. The LED block cipher. In Cryptographic Hardware and Embedded Systems — CHES 2011, volume 6917 of Lecture Notes in Computer Science, pages 326–341. Springer Verlag, 2011.

[14] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: An ultra-lightweight blockcipher. In B. Preneel and T. Takagi, editors, Cryptographic Hardware and Embedded Systems — CHES 2011, volume 6917 of Lecture Notes in Computer Science, pages 342–357. Springer Verlag, 2011.

[15] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. H. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In P. Paillier and I. Verbauwhede, editors, Cryptographic Hardware and Embedded Systems — CHES 2007, volume 4727 of Lecture Notes in Computer Science, pages 450–466. Springer Verlag, 2007.

[16] M. R. Albrecht, B. Driessen, E. B. Kavun, G. Leander, C. Paar, and T. Yalçin. Block ciphers – Focus on the linear layer (feat. PRIDE). In J. A. Garay and R. Gennaro, editors, Advances in Cryptology — CRYPTO 2014, volume 8616 of Lecture Notes in Computer Science, pages 57–76. Springer Verlag, 2014.

[17] J. Borghoff, A. Canteaut, T. Güneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, and T. Yalçin. PRINCE – A low-latency block cipher for pervasive computing applications. In X. Wang and K. Sako, editors, Advances in Cryptology — ASIACRYPT 2012, volume 7658 of Lecture Notes in Computer Science, pages 208–225. Springer Verlag, 2012.

[18] R. L. Rivest. The RC5 encryption algorithm. In B. Preneel, editor, Fast Software Encryption — FSE '94, volume 1008 of Lecture Notes in Computer Science, pages 86–96. Springer Verlag, 1995.

[19] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede. RECTANGLE: A bit-slice lightweight block cipher suitable for multiple platforms. Science China Information Sciences, 58(12):1–15, Dec. 2015.

[20] A. Baysal and S. Sahin. RoadRunneR: A small and fast bitslice block cipher for low cost 8-bit processors. In T. Güneysu, G. Leander, and A. Moradi, editors, Lightweight Cryptography for Security and Privacy — LightSec 2015, volume 9542 of Lecture Notes in Computer Science, pages 58–76. Springer Verlag, 2016.

[21] V. Grosso, G. Leurent, F.-X. Standaert, and K. Varici. LS-designs: Bitslice encryption for efficient masked software implementations. In C. Cid and C. Rechberger, editors, Fast Software Encryption — FSE 2014, volume 8540 of Lecture Notes in Computer Science, pages 18–37. Springer Verlag, 2015.

[22] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.

[23] D. Dinu, L. Perrin, A. Udovenko, V. Velichkov, J. Großschädl, and A. Biryukov. Design strategies for ARX with provable bounds: Sparx and LAX. In J. H. Cheon and T. Takagi, editors, Advances in Cryptology — ASIACRYPT 2016, volume 10031 of Lecture Notes in Computer Science, pages 484–513. Springer Verlag, 2016.

[24] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013.

[25] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. TWINE: A lightweight, versatile block cipher. In G. Leander and F.-X. Standaert, editors, Proceedings of the 1st ECRYPT Workshop on Lightweight Cryptography (LC 2011), pages 146–169, 2011.

[26] Arduino website http://www.arduino.cc.

[27] Will G Hopkins. 2016. A New View of Statistics. http://sportsci.org/resource/stats/ [accessed on February 2019]