Cryptographic Algorithms in Wearable Communications: An Empirical Analysis

Kristtopher Coelho, Danilo Damião, Guevara Noubir, Alex Borges, Michele Nogueira, and José Nacif

***Abstract*— In this letter, we assess the practical impact of lightweight block and stream cipher algorithms on power consumption and hardware resources for wearable devices that own low computational resources. Differently from the literature, we present an empirical and hardware-driven evaluation of the most representative encryption algorithms with regard to the requirements of wearable networks. We design and implement a cryptography library useful for wearable devices. Results confirm a strong correlation between the amount of logic/arithmetic operations, assembly instructions and power consumption for the two evaluated platforms, and they highlight the need to design encryption algorithms for wearable devices with high energy consumption efficiency, but strong security level similar to AES.**

***Index Terms*— Wearable devices, cryptographic algorithms, block cipher, stream cipher, and power consumption.**

I. Introduction

MARKET forecasts that worldwide shipments of wearable computing devices will reach 929 million in 2021, having as major drivers fitness and healthcare gadgets [1]. Wearable computing devices are smart electronic devices that can be incorporated into clothing, worn on the body, or implanted in the body, such as fitness trackers, smartwatches, and the “neural dust” implantable sensor. Wireless communication is essential for advancements in this field, once it allows the connection between devices in and around the human body, including low-rate devices like pedometers and high-rate devices like augmented-reality glasses. This communication relies on different standards such as those from the IEEE 802.15 family [2] or the next generation 5G cellular.

Given data sensitiveness, popularity and user-reliance on wearable devices, there has been an emergence of new and varied attack vectors targeting privacy intrusions, that so far cannot be addressed by classical techniques. In this letter, our goal lies in empirically evaluating the practical impact of the most representative lightweight cryptographic algorithms with regard to the requirements of wearable networks, such as high security and low computational resources, considering energy constraints from implantable and non-implantable devices.

Existing studies have investigated wearable network requirements either from a software perspective [3], [4] or by simulations and analytical models [5], [6]. Despite the importance of those studies, an empirical study complements them offering insights and knowledge, which can aid the design of more efficient and cost-effective solutions. Ours is the first to follow a hardware-driven and empirical evaluation, highlighting the impacts of the hardware specificity to cryptographic algorithms in wearable devices.

Our analysis targets symmetric cryptography, where the communicating wearable devices share the session key used to encryption. Particularly, we focus our investigations on two different classes of symmetric lightweight encryption algorithms, as block ciphers [7], and stream cipher. For our evaluation approach, we have designed and implemented a cryptography library useful for wireless wearable devices.[1] For power consumption measurements, we have designed an instrumentation circuit and integrated it in the evaluated platforms. Our analysis has focused on real-life, off-the-shelf wearable platforms which consider the transmission of data and other with greater processing power abstracting communication.

Our results confirm the strong correlation between the amount of logic/arithmetic operations required to encrypt data block or stream, and their respective power consumption [9]. Results indicate that SKIPJACK algorithm can be up to 18.76% more efficient among the evaluated algorithms in terms of power consumption, processing up to 32× fewer instructions. It also consumes up to ≈ 3.5× less ROM memory related to AES. Analyzing time vs. power consumption, the XTEA algorithm has a battery consumption almost 6× lower than AES. But, it is worth to highlight the high security level of AES, bringing us to the conclusion that it is necessary efforts to design encryption algorithms for wearable devices with high computational efficiency and high security level.

II. Lightweight Cryptographic Algorithms for Wearable Networks

A cryptosystem consists of a plaintext space $\mathcal{P}$, a ciphertext space $\mathcal{C}$, and a key space $\mathcal{K}$, an encryption algorithm $Enc : \mathcal{K} \times \mathcal{P} \rightarrow \mathcal{C}$, and a decryption algorithm $Dec : \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{P}$. For each $k \in \mathcal{K}$ and $p \in \mathcal{P}$, it is $Dec(Enc(p)_k)_k = p$. In the communication model by Shannon [10], a cryptosystem provides confidentiality to the information from an attacker. Hence, a sender and a receiver communicate by a public channel, where they exchange ciphertexts.

Symmetric key cryptography assumes a secure channel used by the communicating parties to establish a secret session key k, not accessible to the adversary. Given p, k, and the cryptosystem, the sender can construct the ciphertext c and send it to the receiver. The receiver can reconstruct the plaintext p, given c, k, and the cryptosystem. Symmetric key cryptography is relevant for wearable networks, that devices have severe resource constraints (e.g., energy, memory, and

Manuscript received June 19, 2019; revised July 17, 2019; accepted August 19, 2019. Date of publication August 27, 2019; date of current version November 11, 2019. This material was based upon work partially supported by CAPES, CNPq, FAPEMIG, the Rede Nacional de Pesquisa under grant No. 99/2017, and the National Science Foundation under Grant No. 1740907. The associate editor coordinating the review of this letter and approving it for publication was D. Ciuonzo. *(Corresponding author: Michele Nogueira.)*

K. Coelho, D. Damião, and J. Nacif are with the Science and Technology Institute, Federal University of Viçosa, Viçosa 35690-000, Brazil.

G. Noubir is with the College of Computer and Information Science, Northeastern University, Boston, MA 02120 USA.

A. Borges is with the Computer Science Department, Federal University of Juiz de Fora, Juiz de Fora 36036-900, Brazil.

M. Nogueira is with the Department of Informatics, Federal University of Paraná, Curitiba 81531-980, Brazil (e-mail: michele@inf.ufpr.br).

Digital Object Identifier 10.1109/LCOMM.2019.2937782

[1] https://github.com/UFV-Alumni/lib_crypto

processing), and applications demand for low response time. The attacker's main goal lies in recovering p or k and, according to Kerckhoff's principle, an attacker knows the specification of the cryptosystem and has access to the ciphertext c.

While in the last decades the progress in the security cryptographic primitives was based in modeling [11], this letter focuses on power consumption analysis of established block and stream ciphers. A block cipher is a cryptosystem with $gf(p^n)$, where gf denotes the Galois field in order $n \in \mathcal{Z}+$ and plaintext $p \in \mathcal{P}$. For each key k, the encryption function $Enc(p)_k$ is a permutation. In the most general case, the $\mathcal{K}$ corresponds to the set of permutations of size $2^n!$, where a single k lies in a table of size 2^n. The use of a subset of permutations is reasonable by generating a small key. To encrypt messages longer than the block size, we use a mode of operation, such as Cipher Block Chaining, and integrity protection, such as Galois Counter Mode [11].

A stream cipher encrypts binary digits of a plaintext one at time. It follows an internal state $x \in \mathcal{X}$, an update function $L : \mathcal{X} \rightarrow \mathcal{X}$, and an output function $f : \mathcal{X} \rightarrow \mathcal{Z}$, where $\mathcal{Z}$ is called the keystream alphabet. An output $z \in \mathcal{Z}$ is produced at time t, according to $zt = f(xt)$, where $xt = Lt(x)$ and x is the initial state. The stream of outputs $z_0, z_1, \ldots$ is called the keystream. Each output symbol is combined to the corresponding plaintext symbol to produce a ciphertext symbol.

We have analyzed recent literature on wearable cryptographic algorithms [4], [5], [12]. We have chosen the algorithms based on power and processing restrictions imposed by wearable devices (e.g., energy limitations of implantable and non-implantable devices). XTEA, XXTEA, SKIPJACK, RC2, and AES are block ciphers; whereas RC4 is a stream cipher. These encryption algorithms provide a security level that can handle thresholds related to low-resource, minimal area, low-memory and low-power, being well-known as "light" algorithms. We have initially considered other algorithms, e.g., KSEED, TWOFISH, and CAST5, but, they have shown to be impractical for the current wearable device architecture due to the excessive memory use, reported from MSP430 GCC.

The eXtension to TEA (XTEA) and the Corrected Block TEA (XXTEA) encryption algorithms employ a 128-bit key and blocks of 64-bits. XTEA operates in 64 rounds and XXTEA has a variable number of rounds. In both, permutations follow simple operations, *e.g.*, addition, shifting and XOR. For key recovery, the best attack reported on XTEA was a related-key differential attack on 26 out of 64 rounds. The cryptanalysis of XXTEA describes a successful chosen plaintext attack with 2^{59} plain-ciphertext pairs [4].

The SKIPJACK algorithm is a 32-round cipher which applies two distinct rules labeled as A and B. These rules are applied interleaved as A, B, A, B per 8 rounds. Permutations comprise of shifts and Feistel's, which use 32 of the 64 bits from the secret key per permutation. Despite the controversy around SKIPJACK design, cryptanalysis point out a resistance for attacks of 2^{48}, using at least 2^{34} plaintexts [13]. As SKIPJACK, RC2 works on 64-bit blocks and allows a variable key size. It follows the key expansion and encryption steps. Key expansion can extend any key size, in the range of 1 to 128 bytes, up to a 128-byte key. Encryption performs permutations based on a substitution table. Estimates to retrieve a secret key are proportional to the effort for analyzing about 2^{4r} (for $r = 16$) chosen plaintexts [14].

The Advanced Encryption Standard (AES) algorithm has become the primary choice for various security services due to its strong defense against known attacks. The best known attacks against AES are slightly faster than brute-force and require $2^{126.2}$ operations to recover an AES-128 key. In [15], the authors presented an optimized version of AES for devices with low computational capacity and memory resources, while still providing low power consumption.

RC4 is a stream cipher and it comprises of a Key Scheduling Algorithm (KSA) and a Pseudo-Random Generation Algorithm (PRGA). KSA transforms a random key in an initial permutation, whereas PRGA uses this initial permutation to generate a pseudo-random output sequence. Cryptographic transformations applied by the algorithm are linear and simple, using permutations and sums of integer values. However, the secure use of RC4 is non-trivial as experienced with Wi-Fi WEP. The recovery requires a complex process of about 2^{13} algorithm operations for 256-bit key [16].

III. Experiments and Methodology

In this letter, the experiments rely on two platforms: (i) wearable devices from the Shimmer platform, model 2R and (ii) a Teensy™ 3.2 microcontroller. The Shimmer devices are equipped with a MSP430 F1611 microcontroller, 16-bit RISC architecture. Each wearable device contains 48KB flash memory and 10KB RAM. These devices sense vital signs and movements from users by accelerometers, magnetometers, and gyroscope, and transmit them to a coordinator device (*e.g.*, a smartphone) through wireless communication. These low-power wireless devices run TinyOS, a Real-Time Operating System (RTOS). The Teensy platform is equipped with an ARM® Cortex®-M4 of 72 MHz CPU and 32-bit architecture. This device also contains a 256KB flash memory and 64KB RAM memory. For Teensy, the algorithms were implemented in C language and deployed using the Arduino interface.

We measure power consumption in different states (*i.e.*, idle, and run). At a glance, we set up the devices to the desired state and continuously monitor it. The devices are automatically placed in a low-power mode when the task queue is empty (*idle* state). Hence, we are able to measure the device power consumption in this state. Finally, to analyze the wearable on the run state, we set up the device to continuously perform a cryptography task — on 64-bits data blocks — using one of the aforementioned cryptography algorithms on both platforms, *run* state). In the Shimmer platform, *run* state, we consider the cost of encrypted data transmission. Finally, unless we tell otherwise, at each state we perform 2,000 samples and present mean confidence interval of 95%.

We have designed and assembled a circuit for power consumption measurement adapted from [8]. The circuit comprises of a low-cost data acquisition board (DAQ - ADALM1000) connected to a wearable, a $0.10\ \Omega$ resistor, and a computer (Figure 1). We use Active Learning Interface for Circuits and Electronics (ALICE) software to acquire voltage measurements from both terminals of the resistor which are connected to channels CH_A and CH_B of the DAQ. The voltage can be easily transformed to current following the law of Ohm, $V = R \times I$, since the resistance value is known. To make comparisons, we calculate the power consumption by multiplying the current to the voltage. Then, power consumption follows: $P = ((\text{CH\_A} - \text{CH\_B})/0.10) * V$ mW.

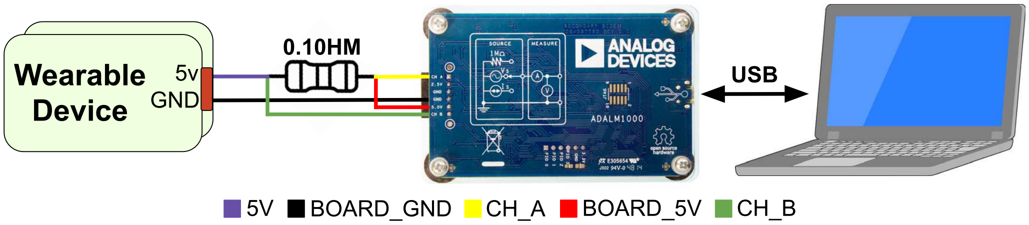


Fig. 1. Power consumption measurement.

DAQ delivers a maximum sampling rate of 100 ksps (kilosamples per second). Therefore, we calculate power consumption, mean, and the total consumption time for the algorithms in each analyzed state of wearable devices. Also, the computational complexity of the algorithms is of great relevance because power consumption bottlenecks occur during data processing and transmission. Hence, we also consider the size of machine code, when it represents a large share of the hardware resource consumption.

We also count the number of Assembly instructions using the Godbolt online compiler and a manual process known as Table Test. The Godbolt compiler converts programs from several languages into Assembly code. For the experiment, we use the MSP430 GCC compiler version 5.3.0 for Shimmer platform and AVR GCC version 4.6.4 for Teensy platform, both without optimization directives. Then, we convert the code to Assembly code. Next, using the Table Test, we have counted the final number of Assembly instructions.

Similarly, we also analyze the main operations in each cryptographic algorithm. The considered operations are *shift left, shift right, and, or, not, xor, sum, subtraction*, and *multiplication*. We enumerate all these logical and arithmetic operations when we want to confirm if the number of operations can be directly correlated with the final performance and power consumption of each algorithm implementation [9]. Also, since wearable devices are severely constrained in computational resources, and implantable devices have hard limitations for replacement, we analyze the amount of memory the implementation of each algorithm requires. We derive this information to memory consumption (ROM and RAM separately) of each cryptography algorithm using MSPGCC compiler for Shimmer platform and AVR GCC for Teensy platform [5]. To ensure equivalence between measurements, we disregard the overhead produced by TinyOS on the Shimmer platform. Hence, we can assert that the presented data refers exactly to each algorithm.

IV. Results

Power consumption is one of the critical factors in the design and development of wearable networks for both high-end and low-end embedded devices. Thus, a comprehensive power efficiency analysis, considering all possible factors is of great relevance. A Power State Machine (PSM) represents the possible states of a device, and a transition between two states means power cost and delay. Low power states have a longer delay between transitions for *run* states. The transition time is presented in [17]. The time for other transitions is insignificant and it is not represented in PSM.

Figure 2 represents the PSM of the evaluated devices. In the *idle* state, the employed platforms run automatically under low energy consumption, being attractive because they manage themselves the different levels of suspension and interruptions, which makes easier for the developer. The figure also presents the average power consumption for each cryptographic algorithm and evaluated state and the transition time between states. Thus, we highlight the SKIPJACK algorithm, that improves energy efficiency in 18% compared to AES.

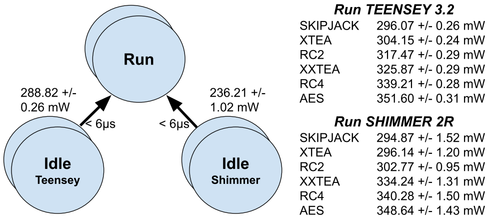


Fig. 2. Wearable device power state machine (PSM).

TABLE I

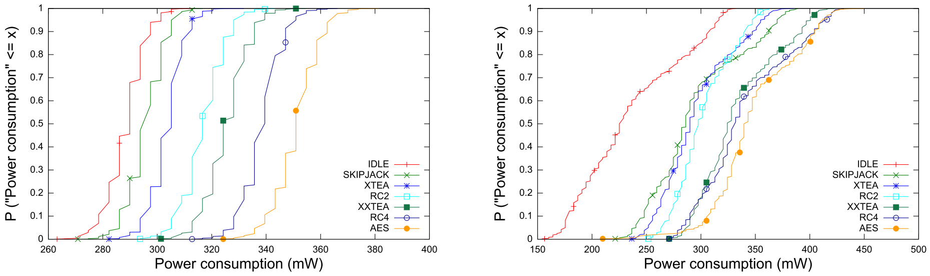
Computational Complexity vs. Memory Consumption.

ALGORITHM	COMPLEXITY	MEMORY CONSUMPTION (BYTES)			
		Shimmer 2R		Teensy 3.2	
		ROM	RAM	ROM	RAM
SKIPJACK	$O(1)$	6,834	608	13,892	4,584
XTEA	$O(1)$	6,772	612	13,360	4,620
RC2	$O(1)$	6,786	726	14,028	4,828
XXTEA	$O(n)$	7,064	604	13,456	4,556
RC4	$O(n)$	6,994	604	13,348	4,556
AES	$O(1)$	24,068	1,978	14,048	4,812

The *run* state asymptotically dominates energy consumption. The analysis of power consumption for the Shimmer platform includes cryptographic processing and radio data transmission. With the Teensey platform, we have excluded the transmission operation and we can observe a similar behavior to the energy consumption of the cryptographic algorithms. Figure 3 shows the behavior of the evaluated algorithms in both platforms through the Cumulative Distribution Functions (CDFs). Figure 3a and 3b illustrate the results for the Teensy Shimmer platforms.

The computational cost of logical and arithmetic operations has a direct effect on processing time and wearable device power consumption. Table I shows the number of operations for each evaluated algorithm and their respective complexity. The count is relative to the encryption function, once the wearable device performs this function, but not decryption. Thus, power consumption has a direct correlation with the number of operations. Another correspondence observed is the proportionality of ROM/RAM occupancy between the algorithms, $\approx$ 11%. In addition to finding a ROM memory consumption about $\approx 3.5\times$ higher of AES in relation to SKIPJACK, considering the Shimmer platform.

Table II displays information about the amount of logical/arithmetic operations and assembly instructions performed by each cryptographic algorithm. This allows us to draw a direct correlation between these parameters and energy consumption. Hence, we observe that the SKIPJACK algorithm performs fewer operations and, thus, fewer instructions



(a) Power consumption per state Teensy 3.2

(b) Power consumption per state Shimmer 2R

Fig. 3. Power consumption in milliwatts on the *run* state.

TABLE II

LOGIC/ARITHMETIC OPERATIONS VS. ASSEMBLY INSTRUCTIONS.

ALGORITHM	#LOGICAL/ ARITHMETC OPERATIONS	NUMBER OF ROUNDS	Shimmer 2R		Teensy 3.2	
			MAIN LOOP	TOTAL #INSTR.	MAIN LOOP	TOTAL #INSTR.
SKIPJACK	496	32	665	760	1,680	1,908
XTEA	576	32	1,184	1,206	4,256	4,329
RC2	804	16	1,550	1,645	6,832	7,075
XXTEA	1,490	12	5,748	5,776	15,936	16,034
RC4	1,992	8	400	10,677	1,088	30,703
AES	2,704	9	21,636	24,117	48,087	54,552

TABLE III

BATTERY LIFE EXPECTANCY.

ALGORITHM	TIME (S)	AVERAGE CONSUMPITION (mA)	BATTERY LIFE (HH:MM)
SLEEP MODE	——	0.0011	——
SKIPJACK	33.00	58.974	40:55
XTEA	12.93	59.228	67:00
RC2	14.62	60.554	62:58
XXTEA	39.64	66.848	33:24
RC4	59.47	68.056	24:17
AES	138.00	69.728	11:34

(≈ 32×), requiring less hardware performance and less energy, particularly, when compared to AES.

Taking as a basis the cryptanalysis presented in Section II, we analyze the tradeoff between power consumption and the security level for each algorithm. SKIPJACK and AES are the two extremes. SKIPJACK is the most power efficient; whereas AES has the highest power consumption. However, AES presents the highest security level.

We could also predict the battery lifetime for the devices, as shown in Table III. We consider an internal battery of 450 mA in the Shimmer platform and a demanding scenario, in which the device performs a data transmission per minute. It is estimated that the device can respond uninterruptedly for up to 67 hours using XTEA as a cryptographic algorithm. This means that the choice of the algorithm can directly influence up to ≈ 5.9× the battery lifetime.

V. CONCLUSION

We have investigated block and stream ciphers for end-to-end wearable devices secure communications. We have performed a hardware-driven power consumption evaluation under two platforms. The SKIPJACK algorithm exhibits the best performance for power consumption and the second least memory usage. The XTEA algorithm presents the longest battery lifetime. Differently from AES, SKIPJACK and XTEA have potential vulnerabilities pointed out in the literature. Hence, despite the computational and energetic efficiency of SKIPJACK and XTEA for the evaluated wearable devices, AES still presents a high security level, leading us to the conclusion that there is still a need to design encryption algorithms for wearable devices with high energy consumption efficiency and security level similar to AES.

REFERENCES

[1] C. C. Cheung, A. D. Krahn, and J. G. Andrade, “The emerging role of wearable technologies in detection of arrhythmia,” *Can. J. Cardiol.*, vol. 34, no. 8, pp. 1083–1087, 2018.

[2] Y. Li, Z. Chi, X. Liu, and T. Zhu, “Passive-ZigBee: Enabling ZigBee communication in IoT networks with 1000X+ less power consumption,” in *Proc. 16th ACM Conf. Embedded Netw. Senor Syst.*, Nov. 2018, pp. 159–171.

[3] S. Kerckhof, F. Durvaux, C. Hocquet, D. Bol, and F. X. Standaert, “Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint,” in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2012, pp. 390–407.

[4] S. Sallam and B. D. Beheshti, “A survey on lightweight cryptographic algorithms,” in *Proc. IEEE Region Conf.*, Oct. 2018, pp. 1784–1789.

[5] M. Cazorla, K. Marquet, and M. Minier, “Survey and benchmark of lightweight block ciphers for wireless sensor networks,” in *Proc. Int. Conf. Secur. Cryptogr. (SECRYPT)*, 2013, pp. 1–6.

[6] M. El Azhari, N. El Moussaid, A. Toumanari, and R. Latif, “Equalized energy consumption in wireless body area networks for a prolonged network lifetime,” *Wireless Commun. Mobile Comput.*, vol. 2017, Dec. 2017, Art. no. 4157858.

[7] B. J. Mohd, T. Hayajneh, and A. V. Vasilakos, “A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues,” *J. Netw. Comput. Appl.*, vol. 58, pp. 73–93, Dec. 2015.

[8] T. Bessa, C. Gull, P. Quintão, M. Frank, J. Nacif, and F. M. Q. Pereira, “JetsonLEAP: A framework to measure power on a heterogeneous system-on-a-chip device,” *Sci. Comput. Program.*, vol. 173, pp. 21–36, Mar. 2019.

[9] B. J. Mohd and T. Hayajneh, “Lightweight block ciphers for IoT: Energy optimization and survivability techniques,” *IEEE Access*, vol. 6, pp. 35966–35978, 2018.

[10] C. E. Shannon, “Communication theory of secrecy systems,” *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, 1949.

[11] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2014.

[12] M. Dener, “Comparison of encryption algorithms in wireless sensor networks,” in *Proc. ITM Web Conf.*, vol. 22, 2018, pp. 1087–1097.

[13] E. Biham, A. Biryukov, and A. Shamir, “Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials,” in *Proc. Int. Conf. Theory Appl. J. Cryptograph. Techn.*, 1999, vol. 18, no. 4, pp. 12–23.

[14] L. R. Knudsen, V. Rijmen, R. L. Rivest, and M. B. J. Robshaw, “On the design and security of RC$_2$,” in *Proc. Int. Workshop Fast Softw. Encryption*, 1998, pp. 206–221.

[15] Y. A. Nasser, M. A. Bazzoun, and S. Abdul-Nabi, “AES algorithm implementation for a simple low cost portable 8-bit microcontroller,” in *Proc. 6th Int. Conf. Digit. Inf. Process. Commun. (ICDIPC)*, 2016, pp. 203–207.

[16] J. Son, E. Ko, U. B. Boyanapalli, D. Kim, Y. Kim, and M. Kang, “Fast and accurate machine learning-based malware detection via RC4 ciphertext analysis,” in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, 2019, pp. 159–163.

[17] M. Goraczko, J. Liu, D. Lymberopoulos, S. Matic, B. Priyantha, and F. Zhao, “Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems,” in *Proc. 45th ACM/IEEE Des. Automat. Conf.*, Jun. 2008, pp. 191–196.