Performance Evaluation of IoT Encryption Algorithms: Memory, Timing, and Energy

Sudip Maitra, Dylan Richards, Ahmed Abdelgawad, Kumar Yelamarthi
College of Science and Engineering
Central Michigan University
Mt Pleasant, MI 48859
k.yelamarthi@ieee.org

***Abstract*—Security in the Internet of Things is a crucial aspect and a lot of studies are focused on modular and scalable encryption algorithms. Resource constraints at the edge nodes of an IoT system require lightweight encryption algorithms. A comparative study of AES with and without hardware accelerators and XTEA is performed to analyze the performance of the algorithms in terms of memory, power and execution time and assess the feasibility of using XTEA in low resource embedded platforms. Although the hardware accelerated AES was fastest (0.5 ms) and consequently required the least amount of energy (0.01 mJ) out of the three, the execution time (1.25 ms) and energy consumed (0.024 mJ) by XTEA was comparatively close and can be a feasible encryption algorithm for low resource microcontrollers that do not have the resources to support AES implementation in software or lack a hardware accelerator. Software implementation of AES on 8-bit PIC architecture required 7538 bytes whereas XTEA required only 1184 bytes of program memory, leaving enough space for application firmware.**

Keywords—lightweight, block ciphers, XTEA, AES, IoT security, resource constraints, microcontroller.

I. Introduction

Internet of Things is a wireless network where sensors and actuators are connected by a wireless communication link to a central server. The "Thing" in IoT can be anything from toasters, thermostats, smart energy efficient houses, wearable electronics, and assistive technologies for the disabled and the elderly, medical & healthcare equipment, automotive applications such as vehicle-to-vehicle communication, structural health monitoring [1-2], environmental monitoring, agriculture, smart homes [3], industrial automation etc. Most smart objects in the vicinity are collecting and analyzing data, and acting upon the reached conclusion.

More devices are being connected with the internet as pervasive computing proliferates at a blistering pace to reduce human interaction, increase ease of use and improve efficiency. As the technology pervades different facets of our lives, it also brings its own share of challenges since all these sensors and smart devices are constantly collecting and sharing data about our surroundings, inherently putting our privacy at risk. Security risk varies across different applications but even the most benign data can be used for malicious purposes. As the system collects and exchanges sensitive, private data, ensuring security across all levels of the architecture is essential yet most IoT devices at least at the end nodes have significant security flaws. Most of the time IoT devices are small in memory size, have limited power and lack the computational capabilities to implement traditional encryption algorithms without burdening the restrained resources [4, 5]. Therefore, it is very necessary to employ a lightweight encryption algorithm to provide security and authentication to sensitive information while minimizing the encryption overhead in terms of computation, memory, time [6] and power [7].

Encryption provides security by converting the plaintext into ciphertext by means of a secret key as in Fig. 1. There are many encryption algorithms with varying security levels and resource requirements. All ciphers have two fundamental functions namely permutation which is a function that takes each object/letter/bit from the plaintext and transforms it with a defined function and mode of operation decides how the permutation function is going to be utilized to encrypt the whole plaintext. For the permutation function to be secure, it must provide randomness and should give different results for different keys. The mode of operation ensures that same letters don't have the same encrypted output and the encryption can't be broken by frequency analysis. The ciphertext should give away very little if nothing about the plaintext in terms of randomness and relation to other plaintexts in a meaningful way.

Encryption algorithms can be broadly classified into three different types, asymmetric encryption, symmetric encryption, and hash functions as presented in Fig. 2. In asymmetric encryption algorithms, different keys are used to encrypt and decrypt data. These algorithms use a pair of key, the receiver's public key is used for encryption & the receiver uses their private key to decrypt the data. These algorithms provide authentication and encryption. Hash functions take arbitrary bit length inputs and output a pseudorandom fixed bit length strings which are called message digests. Identical input message generates identical output digest from a hash function. Changing a single bit in the input bit stream gives completely different, random output digest. Popular Hash Functions such as SHA - 1, SHA - 256, MD5 are used in digital signage, pseudo-random number generators, password security, message authentication etc.

In symmetric encryption also known as secret key cryptography, one secret key is used for both encryption and decryption. They can be further classified into two primary types of Stream Ciphers and Block Ciphers. In stream ciphers

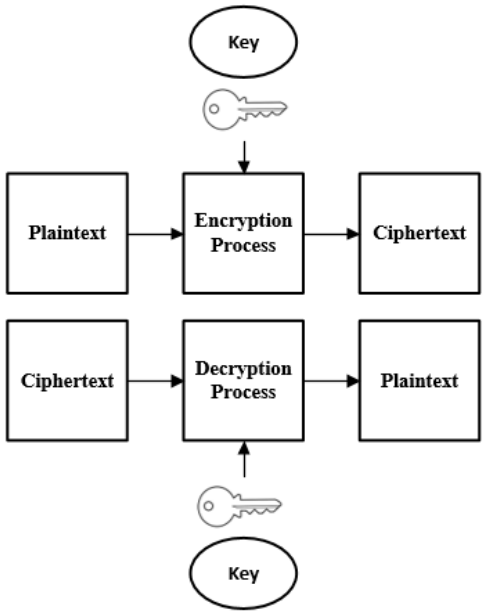

Fig 1. Encryption and Decryption Process

978-1-5386-7713-1/19/$31.00 ©2019 IEEE

each bit is combined with the corresponding key stream, whereas in block ciphers a number of bits are grouped together and combining operation with the key is performed on the whole block. Lightweight block ciphers are a type of ciphers which require very minimum resources in terms of code space, execution time, RAM space and execution time which make them ideal for providing security to resource-constrained embedded platforms in IoT applications. This research explores feasible encryption algorithms that are both lightweight and easy to implement, namely the Advanced Encryption System (AES) and eXtended Tiny Encryption Algorithm (XTEA) [8, 9] were compared. AES is optimized for resource-constrained hardware, where the operations are byte oriented, so it works efficiently in 8-bit, 16-bits and 32-bit architectures. Although AES-256 provides ample security to sensitive data, it takes a toll on the low resource microcontrollers. There are some embedded platforms in the market such as MSP432 by Texas Instruments and Microchip PIC microcontrollers which have integrated AES accelerator which improves the efficiency and performance of the algorithm but most 8-bit microcontrollers do not have this peripheral feature. For simple IoT applications such as acquiring ambient data, 16 and 32-bit processors would be unconscionable in terms of computational power, power consumption and waste of extended resources which is why this paper evaluates the efficiency of XTEA for simple applications against hardware AES so that low resource 8-bit microcontrollers can be used in suitable applications without compromising performance and security.

II. Related Works

Zhang et al. [10] assess various security schemes and estimation of power consumption by WSN protocols. The WSN nodes feed the data to the base station (BS) and processes this data if it requires processing, and delivers it to the Internet and displays the data to the end user. Security is needed at every junction of the network to make it robust and secure. So even the WSN nodes with small resources need security which introduces a message and computational overhead. This goes against the objective of making WSN nodes energy efficient and low power. Many schemes and protocols have been suggested for an energy efficient, robust, reliable WSN but most of them do not provide the security which is very much needed.

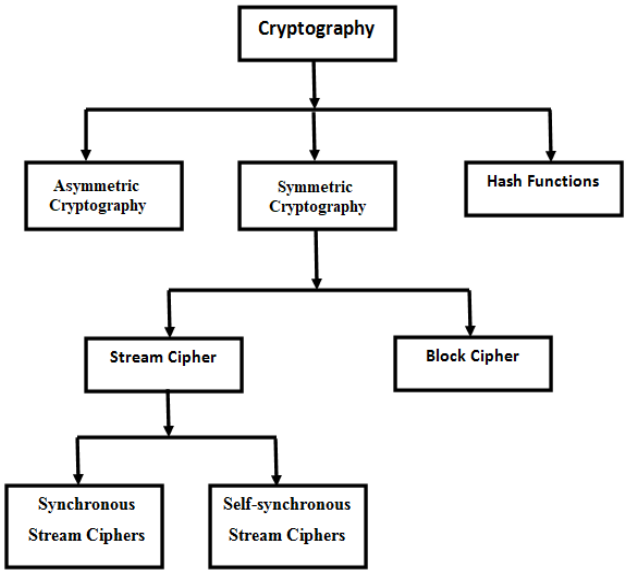

Fig 2. Classification of encryption algorithms

Xiao et al. simulated encryption and decryption (AES algorithm) overhead in microseconds [11]. They discuss different routing protocols, characteristics, limitations, vulnerabilities, and countermeasures. They also suggest a new, improved, efficient routing protocol. This proposed routing protocol is implemented and the empirical measurements have been compared with the traditional mathematical models.

Barahtian et al. implemented and evaluated block ciphers on 8-bit, 16-bit and 32-bit microcontrollers focusing on widely used algorithm tinyAES and also others which are suitable for embedded platforms such as Simon and Speck family of block ciphers, comparing the performance and energy efficiency of each algorithm [12]. Many algorithms designed for computers were reevaluated to efficiently operatre on IoT devices. Cryptographic algorithms on these devices are essential for providing authentication, protect against tapping or verifying data integrity. They have implemented different algorithms across different hardware platforms for better comprehensiveness.

As the experiment is scaled up from 8 bit to 32-bit architecture, very little improvement is observed on larger architectures since tinyAES extensively uses single byte operations. Simon & Speck provides high performance across a wide range of devices. Simon is optimized for hardware implementation and Speck for software. They are both built on ARX (Add-Rotate-XOR) philosophy. The authors admit that the comparison would be fair if the same implementation was tested on each platform, but since the instruction sets of these microcontrollers are not cross-platform compatible it is not possible. The reasoning behind this is that most architectures nowadays have large program memories but all are limited by the amount of power they can use. The energy required to execute a single instruction can be obtained from the microcontroller's datasheet. From there it is fairly straightforward to find out the effect on battery life. The 32-bit architectures benefit from the algorithms' extensive use of 32-bit data. So encryption algorithm needs to be considered closely with the whole system architecture. But this inherently implies lack of modularity of algorithms across a wide range of platforms.

Noura et al. in [13] point out that the traditional cryptographic algorithms use a static structure which requires several rounds of computation - adding significant overhead on execution time and computational resources. Author further claim that the problem is compounded when the content type is multimedia since the algorithms have strict QoS requirements. They offer a dynamic cipher structure which minimizes the rounds to a single one without compromising randomness and security. They also demonstrated the robustness of the proposed cipher scheme with various tests. In [14], Biryukov et al. give a list of algorithms and best cryptanalysis against them. The author split lightweight cryptography into two groups namely, ultra-lightweight cryptography and ubiquitous cryptography. The authors provide design constraints that make an algorithm lightweight in terms of hardware and software. They also provide a benchmark for assessing good performance on a given platform. In [15] Dinu et al. carried out a comprehensive study on the performance of different ciphers across a wide variety of platforms. Author implemented a testbed for evaluating the ciphers based on security level and impact on low resource devices. Kotel et al. evaluate software implementation of different lightweight block ciphers using FELICS benchmarking framework in [7] across 8-bit and 16-bit architectures.

III. PROPOSED METHODOLOGY

The proposed testbed consists of two microcontrollers (8-bit and 32-bit), their respective software and programming tools to implement the aforementioned block ciphers. The 8-bit PIC microcontroller is chosen for evaluating XTEA and software AES in low resource embedded platforms. PIC18F27K40 has 128 Kbytes [16] of program memory which is sufficient to implement software AES algorithm combined with eXtreme Low-Power (XLP) technology making it energy efficient and thus ideal for IoT applications. However, most microcontrollers do not have such large program space and software AES cannot be implemented in microcontrollers that lack sufficient program memory such as PIC18F24K42 which has only 16 Kbytes of program memory [17]. The 32-bit MSP432 microcontroller has AES hardware accelerator and enables the microcontroller to perform complex computation without consuming precious program memory and is efficient in terms of time and power. AES can be implemented in software across different architectures (8-bit, 16-bit, and 32-bit) but it is not as efficient as hardware accelerated AES. A software implementation of XTEA was done on all platforms including the 8-bit PIC18F27K40 which does not have hardware accelerator for AES. From the proposed testbed the cycles, power, and code size, RAM space needed to encrypt and decrypt sample data were measured. The execution time was measured in terms of clock cycles using the oscilloscope as seen from the block diagram in Fig. 3 and experimental setup in Fig. 4. The internal timers inside the microcontrollers were used to record the number of cycles while encrypting and decrypting

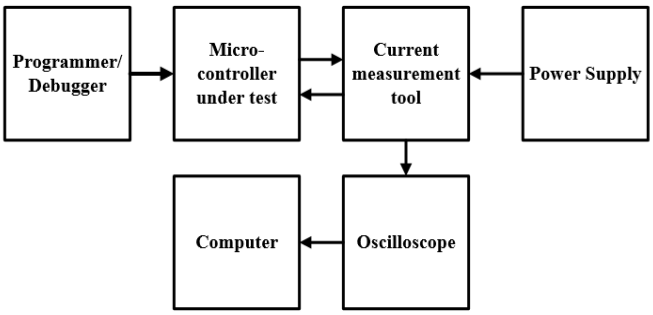

Fig 3. Block diagram of the testbed

sample data. To further verify the results, the execution time was also be measured with an oscilloscope.

Efficiency can be measured in terms of latency, clock cycles, and power consumption. Software efficiency can be quantified by considering the number of cycles it takes to encode and decode particular sets of data. As most of the sensor nodes in an IoT application have limited resourced and often times feed sensor data to the cloud in real time, the encryption process should not add too much of overhead. Most of the sensor nodes are either battery powered or have stringent restrictions on power. So making sure that the encryption process does not increase the power consumption is essential.

To measure the time required to encrypt data, one of the output pins of the microcontroller was driven high at the start of the encryption process and driven low at the end providing a pulse with equal to the encryption time which was measured with an oscilloscope. Keysight's InfiniiVision MSO-X 3054T Oscilloscope was used to conduct all the measurements. The same process was taken to measure the timing of the decryption process. The power was measured with the help of a current measurement tool called Real-Time Current Monitor [18]. This tool has a precise current sensing resistor and amplification stages to measure in line low dynamic currents. The amplification stages are fast enough to capture instantaneous data which is essential as the encryption and decryption time is in the order of tens of milliseconds. The measured values were exported to the computer with the help of the USB connectivity of the Current Measurement Tool. The instantaneous current values during encryption and decryption were multiplied with the reference power supply unit Tektronix PWS 2323 to get the power. The voltage of the power supply was measured at the device under test end with Tektronix DMM 4020 precision benchtop multimeter to compensate for the wire losses.

The respective timing data was used to obtain the energy used for the encryption and decryption process. The code size was acquired from the hex file generated by the compiler. The RAM space was obtained from the compiler. The optimization feature in the compilers was disabled to cancel out any interferences caused by the compiler optimization. All the microcontrollers were run at the same speed to negate effects of the CPU clock speed on the encryption/ decryption timing. The data obtained from the experiments were presented in the following sections.

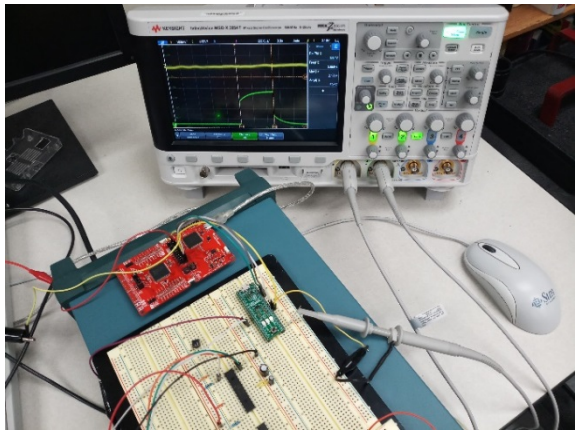

Fig 4. Experimental setup

IV. ALGORITHM IMPLEMENTATION

A. AES Implementation

AES is the widely used symmetric key encryption algorithm. It was originally called "Rijndael Cipher". Structurally AES is not a Feistel Network but a Substitution-Permutation Network. AES encryption converts data into ciphertext and decryption converts the ciphertext back into plaintext. The decryption algorithm differs from the encryption although similar steps are used to encrypt and decrypt data. The whole encryption process can be divided into four functional blocks. AES works with 128-bit data blocks, and if the data is longer than 128 bits, it is broken up into 16 bytes blocks. Each block is encrypted independently. If the message is not exactly divisible by 16, the remaining empty bits are padded with zeros. AES requires some initialization steps and then some recurring rounds as in Fig. 5. Initially, key expansion and the initial round are performed followed by a series of encryption rounds, using expanded keys from the key expansion step. After defined number rounds, the ciphertext is passed through one final round. The number of rounds increase (10, 12, 14) along with key size (128, 192, 256 bits). The keys used for each round is generated from the original key in the key expansion step. So different keys generated from the original key is used for every single round.

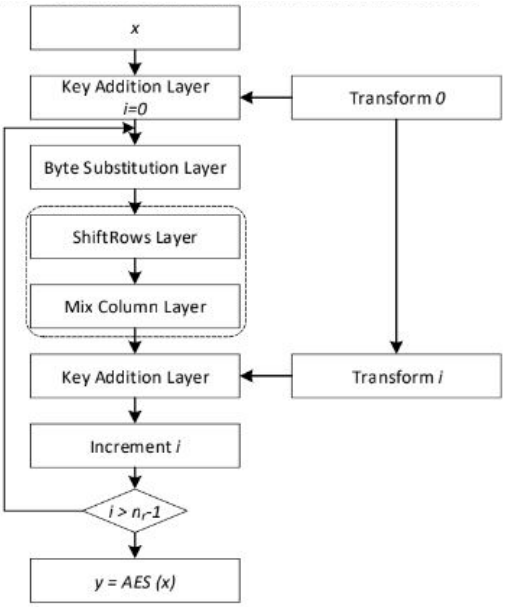

Fig 5. Functional flowchart of AES algorithm

AES encryption process can be divided into four functional blocks namely key expansion, initial round, recurring rounds, and final round. For 128 bit key, 44-word keys are generated in key generation step. In case of the 192-bit key we would need 52 words and for a 256-bit key, 60 words. The first four words are XORed with the input matrix before the round functions begin. The rest 40 words are used for the 10 subsequent rounds, 4 words in each round. The 128-bit key is arranged in a 4x4 matrix in such a way that the first 4 bytes of the key are put into the first column of the matrix as seen in Fig. 6.

B. XTEA Implementation

The algorithm XTEA is the improved version of TEA, proposed by David Wheeler and Roger Needham in 1997. It is a 64 round Feistel cipher with a block size of 64 bits and a key size of 128 bits. The key schedule is simple and keys for the rounds are scheduled dynamically so it reduces impact on memory which is ideal for microcontrollers with low memory. The code comprises of operations such as shifts, additions and XORs as presented in Fig. 7.

The source code is written in C for easy of compilation and execution on microcontrollers. The number of rounds can be changed which can updated for different levels of security. The rounds are identical and uses different multiples of magic constant. It requires no initialization steps and does not use S-boxes like AES which again reduces memory requirement. The 64 bit data block is split into two 32 bit variables and operations such as rotation, addition and XOR is used to mix the data with the key for each round. The decryption simply inverses the operations occurred during encryption.

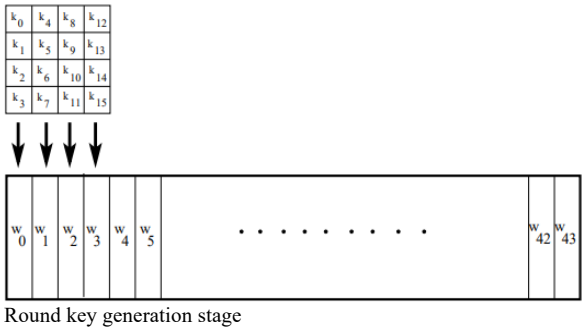

Fig 6. Round key generation stage

V. Results & Discussion

From the data obtained from the experiment, it is evident that AES software implementation is by far the most resources hungry encryption in terms of execution time, code space and energy. Table I and II displays the code size in bytes, required RAM space in bytes, execution time in milliseconds, power in milliwatts, and energy in millijoules for AES and XTEA on MSP432 and PIC18F27K40.

As it can be seen from Fig 8, AES without hardware accelerator took the most amount of time as encrypting and decrypting the data requires several computationally demanding operations such as rotate, XOR, multiplication

TABLE I. Comparisons for 8-b and 32-b Architectures with AES256

	PIC18F27K40 8 bit	**MSP432P401RIRGC 32 bit**	
		SW	***HW***
Code Size (bytes)[a]	7538	16251	1258
RAM (bytes)	1105	3508	576
Time (ms)	75.62	281.5	0.5
Power (mW)	19.23	21.8	21.8
Energy (mJ)	1.45	6.1	0.0109
Compiler	XC8	C2000 -CGT	

[a.] Results for C implementation

TABLE II. Comparisons for 8-b and 32-b Architectures with XTEA

	PIC18F27K40 8 bit	**MSP432P401RIRGC 32 bit**
Code Size (bytes)[b]	1184	1946
RAM (bytes)	99	556
Time (ms)	1.25	1.196
Power (mW)	19.23	21.8
Energy (mJ)	0.024	0.026
Compiler	XC8	C2000 -CGT

[b.]Results for C implementation

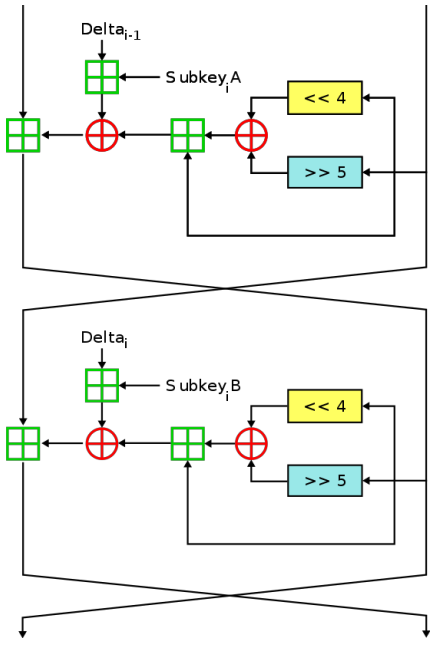

Fig 7. One cycle of the XTEA Algorithm

and substitution from lookup tables. The hardware accelerated AES took the least amount of time, over 500 times less than that of the software implementation of AES on the 32-bit MSP432P401RIRGC because it is implemented in hardware and isn't restricted by the sequential completion of the CPU. The advantage of hardware accelerator is also apparent in terms of memory. The software AES on PIC takes 7538 bytes of code space and 1105 bytes of RAM. Application developers cannot afford to dedicate so much of program memory and RAM space just for the encryption process. Many 8-bit microcontrollers do not even have the memory capacity for AES to be implemented in software. It can be seen from Table 3 that the implementation of XTEA on the 8-bit PIC was more efficient in terms of RAM usage as it only required 99 bytes compared to the implementation on 32-bit MSP432 which required 556 bytes. The temporal effect is also seen in terms of energy as AES without hardware accelerator took more energy than the other two as seen in Fig. 9.

The 8-bit PIC18F27K40 doesn't have the AES hardware accelerator so AES was only implemented in software and it takes around 60 times more cycles than XTEA as seen from Fig 10. This is also reflected in Fig. 11 as the energy consumed by XTEA is $1/60^{th}$ of the software AES in PIC18F27K40. Fig 12 and 13 depicts the comparison of AES implemented in MSP432P401RIRGC with hardware accelerator and XTEA implemented on PIC18F27K40 in terms of execution time and energy. Although the XTEA took longer to execute and as a result spent more energy, it is very close to the execution time and consumed energy by the MSP432P401RIRGC while running hardware accelerated AES.

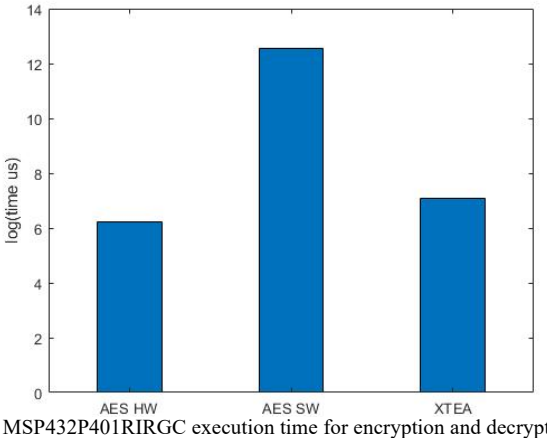

Fig 8: MSP432P401RIRGC execution time for encryption and decryption

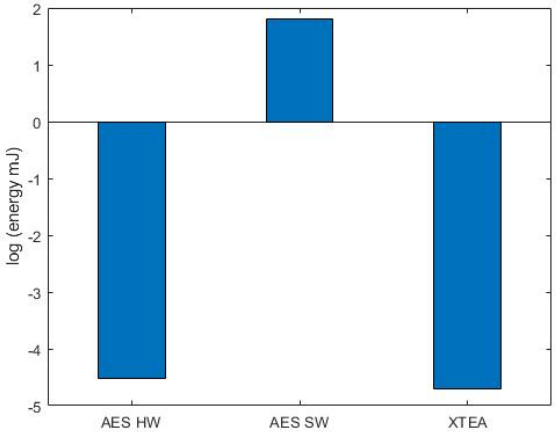

Fig 9: MSP432P401RIRGC energy required for encryption and decryption

As seen from Table 2 and 3, XTEA requires less code space, ram than software AES and is almost on par with hardware accelerated AES in terms of execution time and energy. Although from our experiment it is clear that the hardware accelerated AES was most efficient, most 8-bit microcontrollers do not have this peripheral feature. As XTEA comes really close to the performance of hardware accelerated AES, it can be easily implemented on the 8-bit microcontroller which does not have hardware accelerator built in. XTEA uses comparatively less memory and time so it can be used to provide a minimum level of security to 8-bit microcontrollers at the edge nodes which do not have dedicated crypto-engines to use hardware AES. Microcontroller that do have hardware AES accelerator can use the crypto engine which is the most efficient in terms of power, code space, and memory usage among the three schemes tested in the experiment and leave ample space for the application firmware. Optimizing the security required by the specific application with the resources is a challenging aspect of this study as one needs to consider the temporal relevance of the data being protected for the encryption algorithm to be feasible in an application. Cryptanalysis against the encryption algorithms can yield quantitative insight into the security level facilitated by the algorithms which can help determine the suitable encryption approach for a specific application provided that the temporal relevance of the data can be estimated.

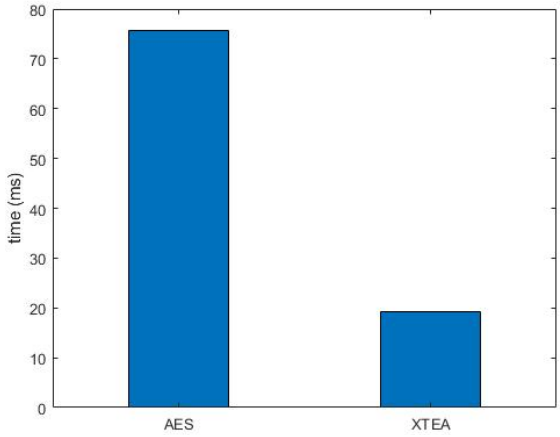

Fig 10: PIC18F27K40 execution time for encryption and decryption

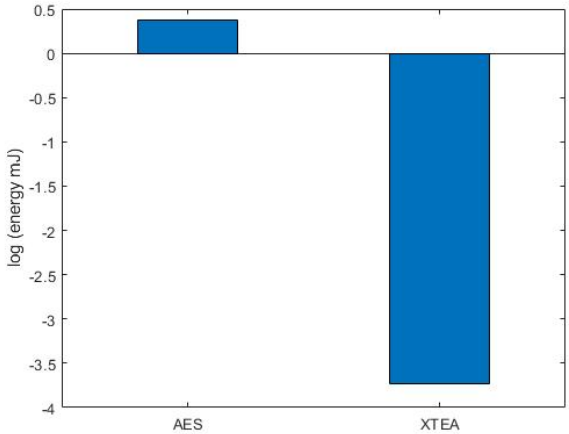

Fig 11: PIC18F27K40 required energy for encryption and decryption

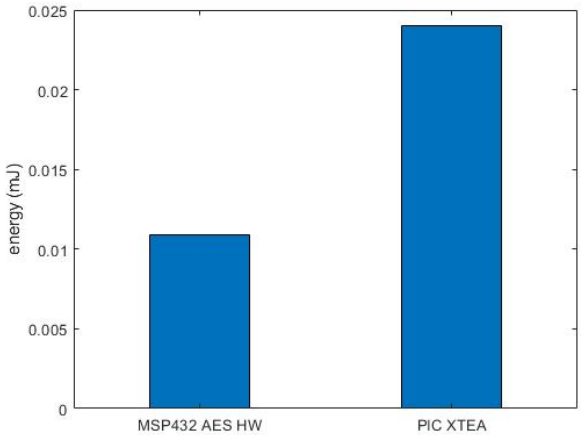

Fig 12: Comparison of AES Hardware accelerated MSP432P401RIRGC and PIC18F27K40 running XTEA in terms of time

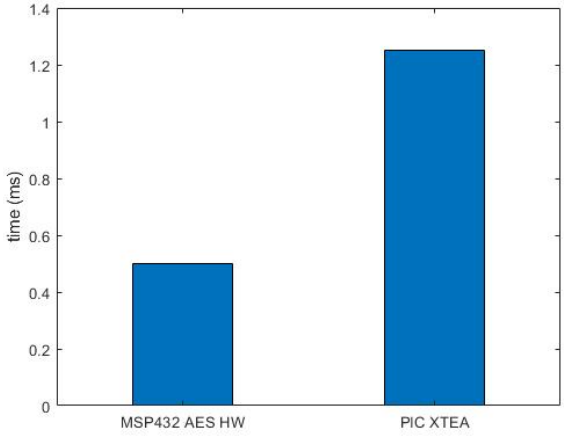

Fig 13: Comparison of AES Hardware accelerated MSP432P401RIRGC and PIC18F27K40 running XTEA in terms of energy

VI. Conclusion

Security is a crucial part of the Internet of Things but edge node devices in an IoT architecture often have constraint resources such as memory and power. Implementing software AES in these embedded platforms is not feasible. As our experiment displayed, the performance in terms of energy efficiency and execution time of XTEA on a microcontroller without a dedicated hardware accelerator for AES is almost on par with that of a device which has a crypto engine. As seen from the results the XTEA was around 60 times more efficient in terms of power compared to software implementation of AES on 8-bit PIC microcontroller while only taking $1/7^{th}$ of the program memory. From analyzing the performance of AES and XTEA on different architectures, it is clear the encryption algorithm needs to be considered closely with the whole system architecture. Microcontrollers which have sufficient memory and power at its expense can implement AES in software with room for application firmware but in most applications the edge nodes have restrained resources, making software implementation of AES infeasible. Some 16-bit and 32-bit microcontrollers have dedicated crypto engines leaving ample space for the application firmware but very few 8-bit microcontrollers offer this peripheral feature. Without any encryption scheme these microcontrollers are susceptible to malicious attacks. Although XTEA may not be on par with AES in terms of providing security, it is better to have no security at all and implementing XTEA on these 8-bit microcontrollers without crypto engines provides a level of security which may be sufficient for certain applications with narrow window of temporal relevance. This paper assesses the time and power requirement of the algorithms and our future work will focus on testing these algorithms in terms of security implementing different attack models to observe the robustness of each algorithm.

References

[1]. M. A. Mahmud, K. Bates, T. Wood, A. Abdelgawad and K. Yelamarthi, "A complete Internet of Things (IoT) platform for Structural Health Monitoring (SHM)," *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, Singapore, 2018, pp. 275-279.

[2]. K. Yelamarthi, A. Abdelgawad, A. Khattab, "IoT-Based Health Monitoring System for Active and Assisted Living," *Smart Objects and Technologies for Social Good: Second International Conference,* pp.11-20, Dec 2016.

[3]. C. Coelho, D. Coelho, and M. Wolf, "An IoT smart home architecture for long-term care of people with special needs," *2015 IEEE 2nd World Forum on Internet of Things,* pp. 626-627, 2015.

[4]. M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating Critical Security Issues of the IoT World: Present and Future Challenges," in *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2483-2495, Aug. 2018.

[5]. M. Botta, M. Simek, and N. Mitton, "Comparison of hardware and software-based encryption for secure communication in wireless sensor networks," *36th International Conference on Telecommunications and Signal Processing*, Rome, pp. 6-10. 2013.

[6]. D. Richards, A. Abdelgawad, K. Yelamarthi "How Does Encryption Influence Timing in IoT" *IEEE Global Conference on Internet of Things,* Dec. 2018.

[7]. S. Kotel, F. Sbiaa, M. Zeghid, M. Machhout, A. Baganne, and R. Tourki, "Performance evaluation and design considerations of lightweight block cipher for low-cost embedded devices," *IEEE/ACS 13th International Conference of Computer Systems and Applications*, pp. 1-7, 2016.

[8]. R. M. Needham and D. J. Wheeler. Tea extensions. Technical report, University of Cambridge, 1997.

[9]. J. Grossschadl, S. Tillich, C. Rechberger, M. Hofmann and M. Medwed, "Energy Evaluation of Software Implementations of Block Ciphers under Memory Constraints," *Design, Automation & Test in Europe Conference & Exhibition,* pp. 1-6, 2007.

[10]. F. Zhang, "On the Security and Energy Consumption Estimation of Wireless Sensor Network Protocols" (Doctoral dissertation), 2012.

[11]. Y. Xiao, H. Chen, B. Sun, R. Wang, S. Sethi, "MAC Security and Security Overhead Analysis in the IEEE 802.15.4 Wireless Sensor Networks", *EURASIP Journal on Wireless Communications and Networking*, Vol. 2006(2), Springer, April 2006, pp. 1-12.

[12]. O. Barahtian, M. Cuciuc, L. Petcana, C. Leordeanu, and V. Cristea, "Evaluation of Lightweight Block Ciphers for Embedded Systems," *Innovative Security Solutions for Information Technology and Communications Lecture Notes in Computer Science*, pp. 49–58, 2015.

[13]. H. Noura, A. Chehab, L. Sleem, M. Noura, R. Couturier, and M. M. Mansour, "One round cipher algorithm for multimedia IoT devices," *Multimedia Tools and Applications*, vol. 77, no. 14, pp. 18383–18413, 2018.

[14]. "State of the Art in Lightweight Symmetric Cryptography." [Online]. Available: https://eprint.iacr.org/2017/511.pdf.

[15]. D. Dinu, Y. L. Corre, D. Khovratovich, L. Perrin, J. G. schadl, A. Biryukov, "Triathlon of lightweight block ciphers for the internet of things", *Cryptology ePrint Archive Report 2015/209*, 2015, [online] Available: http://eprint.iacr.org/.

[16]. "PIC18F27K40," *PIC18F27K40 - Microcontrollers and Processors - Microcontrollers and Processors*. [Online]. Available: https://www.microchip.com/wwwproducts/en/PIC18F27K40. [Accessed: 20-Dec-2018].

[17]. "PIC18F24K42," *PIC18F24K42 - 8-bit Microcontrollers - Microcontrollers and Processors*. [Online]. Available: https://www.microchip.com/wwwproducts/en/PIC18F24K42. [Accessed: 20-Dec-2018].

[18]. Real-Time Current Monitor with USB," *ee-quipment.com*. [Online]. Available: https://www.ee-quipment.com/products/real-time-current-monitor-with-usb. [Accessed: 20-Dec-2018].