Performance Evaluation Of Encryption Algorithm For Wireless Sensor Networks

Soufiene Ben Othman
UR PRINCE, ISITC, Hammam Sousse
University of Sousse, Tunisia
Email: ben_oth_soufiene@yahoo.fr

Abdelbasset Trad, Habib Youssef
UR PRINCE, ISITC, Hammam Sousse
University of Sousse, Tunisia
Email: trad.abdelbasset@gmail.com

Abstract— **With the widespread growth in applications for resource limited Wireless Sensor Networks (WSN), the need for reliable and efficient security mechanisms for them has increased manifold but its implementation is a non-trivial task. Limitations in processing speed, battery power, bandwidth and memory constrain the applicability of existing cryptography Algorithms for WSNs. Several security mechanisms, such as TinySec, have been introduced to address the need for security in WSNs. The cost of security, however, still mostly remains an unknown variable. To provide a better understanding of this cost we have studied three encryption algorithms, AES, RC5 and RC6. We have measured and compared their memory and energy consumption on Mica2 sensor motes. The results of our experiments provide insight into the suitability of different security algorithms for use in WSN environments and could be used by WSN designers to construct the security architecture of their systems in a way that both satisfies the requirements of the application and reasonably uses the constrained sensor resources.**

Keywords-component; Wireless sensor network; security; Energy consumption analysis

I. INTRODUCTION

Wireless Sensor Networks (WSN) has emerged as an important new area in wireless technology. A wireless sensor network [1] is a distributed system interacting with physical environment. It consists of motes equipped with task-specific sensors to measure the surrounding environment, e.g., temperature, movement, etc. It provides solutions to many challenging problems such as wildlife, battlefield, wildfire, or building safety monitoring. A key component in a WSN is the sensor mote, which contains (a) a simple microprocessor, (b) application-specific sensors, and (c) a wireless transceiver. Each sensor mote is typically powered by batteries, making energy consumption an issue.

Security is vital aspect in WSN applications. The implementation of security policies is a complex and challenging issue because of resource constrained nodes. Short transmission distances reduce some of the security threats, but there are risks, for example, related to spoofing, message altering and replaying, and flooding and wormhole attacks [6]. It is important therefore to consider security solutions that guarantee data authenticity, freshness, replay protection, integrity and confidentiality.

For secure communication in WSNs, efficient cryptographic algorithm suitable for WSNs environment is required. It is ideal to choose the most efficient cryptographic algorithm in all aspects; operation speed, storage and power consumption. However, since each cryptographic algorithm applied in WSNs has distinguished advantages, it is important to choose cryptographic algorithm suitable for each environment WSNs are exploited.

The data encryption algorithms used in WSNs are generally divided into three major categories: symmetric-key algorithms, asymmetric-key algorithms, and hash algorithms. A number of papers, [1–2], have investigated using asymmetric-key algorithms in WSNs. However, the results they present reveal that despite the use of energyefficient techniques, such as elliptic curve cryptography or dedicated cryptography coprocessors, asymmetric-key algorithms consume more energy than symmetric-key algorithms. Hash functions, on the other hand, are typically used for verifying the integrity of the exchanged messages and may increase the transmission cost [3,4].

Law et al. present a benchmark on cryptographic algorithms for wireless sensor networks [5]. They consider security properties, storage and energy-efficiency of a set of candidates : Skipjack [6], RC5 [7], RC6, Rijndael, Twofish, MISTY1, KASUMI and Camellia. In [5], the result of analysis provides us criterion of selecting cryptographic algorithm suitable for WSNs. In an environment where security is important, since sensor nodes store many keying material to establish pairwise key after deployment, memory efficient cryptographic algorithm are required. In an environment where availability of network is important, since sensor nodes that use up battery are no longer available in network, energy-efficient cryptographic algorithm has to be used.

This work's contributions are two fold. We have analyzed AES, RC5 and RC6 in terms of their energy consumption and memory requirements. We have studied how the different algorithm parameters (e.g. key size) influence the energy consumption.

The rest of the paper is structured as follows: Section 2 presents the wireless sensor networks. Section 3 describes the related work. Section 4 describes of studied block ciphers our analytical models of cryptographic algorithms complexity in terms of primitive operations. Performance results and evaluation of cryptographic algorithms are presented in Section 5. Finally, Section 6 ends up with final considerations and future works.

978-1-4673-1166-3/12/$31.00 ©2012 IEEE

II. WIRELESS SENSOR NETWORK

A wireless sensor network (WSN) consists of a large number of tiny sensor nodes deployed over a geographical area also referred as sensing field; each node is a low-power device that integrates computing, wireless communication and sensing abilities [8][9]. Nodes organize themselves in clusters and networks and cooperate to perform an assigned monitoring (and/or control) task without any human intervention at scales (Both spatial and temporal) and resolutions that are difficult, if not impossible, to achieve with traditional techniques. Sensor nodes are thus able to sense physical environmental information (e.g. temperature, humidity, vibration, acceleration or whatever required), process locally the acquired data both at unit and cluster level, and send the outcome or aggregated features- to the cluster and/or one or more collection points, named sinks or base stations (Figure 1). Nodes organize themselves in clusters and networks and cooperate to perform an assigned monitoring (and/or control) task without any human intervention at scales (both spatial and temporal) and resolutions that are difficult, if not impossible, to achieve with traditional techniques.

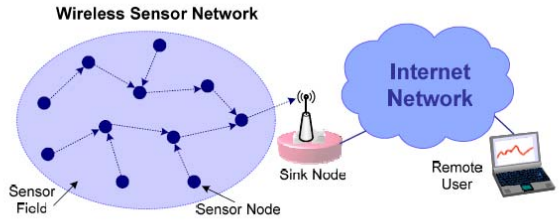

Figure1 .A typical sensor network architecture.

A WSN can thus be viewed as an intelligent distributed measurement technology adequate for many different monitoring and control contexts. In recent years, the number of sensor network deployments for real-life applications e.g., environmental monitoring [10][11], agriculture [11], production and delivery [12], military [10], structure monitoring [13] and medical applications [14] has rapidly grown with a trend expected to further increase in the incoming years [15].

However, energy consumption still remains one of the main obstacles to the diffusion of this technology, especially in application scenarios where a long network lifetime and a high quality of service are required. In fact, nodes are generally powered by batteries which have limited capacity and, often, can neither be replaced nor recharged due to environmental constraints. Despite the fact that energy scavenging mechanisms can be adopted to recharge batteries, e.g. through solar panels, piezoelectric or acoustic transducers (the interested reader can refer to [16]); energy is a limited resource and must be used judiciously. Hence, efficient energy management strategies must be devised at sensor nodes (and then at cluster and network level) to prolong the network lifetime as much as possible.

A. *Constraints in Wireless Sensor Networks*

A wireless sensor network consists of a large number of sensor nodes which are inherently resource-constrained. These nodes have limited processing capability, very low storage capacity, and constrained communication bandwidth. These limitations are due to limited energy and physical size of the sensor nodes. Due to these constraints, it is difficult to directly employ the conventional security mechanisms in WSNs. In order to optimize the conventional security algorithms for WSNs, it is necessary to be aware about the constraints of sensor nodes [14]. Some of the major constraints of a WSN are listed below.

- *Memory limitations:*

A sensor is a tiny device with only a small amount of memory and storage space. Memory is a sensor node usually includes flash memory and RAM. Flash memory is used for storing downloaded application code and RAM is used for storing application programs, sensor data, and intermediate results of computations. There is usually not enough space to run complicated algorithms after loading the OS and application code. In the SmartDust project, for example, TinyOS consumes about 4K bytes of instructions, leaving only 4500 bytes for security and applications [17]. A common sensor type- TelosB- has a 16-bit, 8 MHz RISC CPU with only 10K RAM, 48K program memory, and 1024K flash storage [13]. The current security algorithms are therefore, infeasible in these sensors [12].

- *Energy constraints:*

Energy is the biggest constraint for a WSN. In general, energy consumption in sensor nodes can be categorized in three parts: (i) energy for the sensor transducer, (ii) energy for communication among sensor nodes, and (iii) energy for microprocessor computation. The study in [13] found that each bit transmitted in WSNs consumes about as much power as executing 800 to 1000 instructions. Thus, communication is more costly than computation in WSNs. Any message expansion caused by security mechanisms comes at a significant cost. Further, higher security levels in WSNs usually correspond to more energy consumption for cryptographic functions. Thus, WSNs could be divided into different security levels depending on energy cost [12, 16].

- *Unreliable communication:*

Unreliable communication is another serious threat to sensor security. Normally, the packet-based routing of sensor networks is based on connectionless protocols and thus inherently unreliable. Packets may get damaged due to channel errors or may get dropped at highly congested nodes. Furthermore, the unreliable wireless communication channel may also lead to damaged or corrupted packets. Higher error rate also mandates robust error handling schemes to be implemented leading to higher overhead. In certain situation even if the channel is reliable, the communication may not be so. This is due to the broadcast nature of wireless communication, as the packets may collide in transit and may need retransmission [10].

- *Mistakes Higher latency in communication:*

In a WSN, multi-hop routing, network congestion and processing in the intermediate nodes may lead to higher latency in packet transmission. This makes synchronization very difficult to achieve. The synchronization issues may sometimes be very critical in security as some security mechanisms may rely on critical event reports and cryptographic key distribution [16].

- *Unattended operation of networks:*

In most cases, the nodes in a WSN are deployed in remote regions and are left unattended. The likelihood that a sensor encounters a physical attack in such an environment is therefore, very high. Remote management of a WSN makes it virtually impossible to detect physical tampering. This makes security in WSNs a particularly difficult task.

B. *Security Goals and Challenges*

In order to defend against some attacks, many security mechanisms have been proposed. The goal of each one is to achieve some or all of the following security goals [18]:

- Data Confidentiality :

Confidentiality means keeping information secret from unauthorized parties. A sensornetwork should not leak sensor readings to neighboring networks. In many applications (e.g. key distribution) nodes communicate highly sensitive data. The standard approach for keeping sensitive data secret is to encrypt the data with a secret key that only intended receivers possess, hence achieving confidentiality. Since public-key cryptography is too expensive to be used in the resource constrained sensor networks, most of the proposed protocols use symmetric key encryption methods. The creators of TinySec [18] argue that cipher block chaining (CBC) is the most appropriate encryption scheme for sensor networks. They found RC5 and Skipjack to be most appropriate for software implementation on embedded microcontrollers. The default block cipher in TinySec is Skipjack. SPINS uses RC6 as its cipher.

- Data Authenticity :

In a sensor network, an adversary can easily inject messages, so the receiver needs to make sure that the data used in any decision making process originates from the correct source. Data authentication prevents unauthorized parties from participating in the network and legitimate nodes should be able to detect messages from unauthorized nodes and reject them. In the twoparty communication case, data authentication can be achieved through a purely symmetric mechanism: The sender and the receiver share a secret key to compute a message authentication code (MAC) of all communicated data. When a message with a correct MAC arrives, the receiver knows that it must have been sent by the sender. However, authentication for broadcast messages requires stronger trust assumptions on the network nodes. The creators of SPINS [19] contend that if one sender wants to send authentic data to mutually untrusted receivers, using a symmetric MAC is insecure since any one of the receivers know the MAC key, and hence could impersonate the sender and forge messages to other receivers. SPINS constructs authenticated broadcast from symmetric primitives, but introduces asymmetry with delayed key disclosure and one-way function key chains. LEAP [20] uses a globally shared symmetric key for broadcast messages to the whole group. However, since the group key is shared among all the nodes in the network, an efficient rekeying mechanism is defined for updating this key after a compromised node is revoked. This means that LEAP has also defined an efficient mechanism to verify whether a node has been compromised.

- Data Integrity :

Data integrity ensures the receiver that the received data is not altered in transit by an adversary. Note that Data Authentication can provide Data Integrity also.

- Data Freshness:

Data freshness implies that the data is recent, and it ensures that an adversary has not replayed old messages. A common defense (used by SNEP [19]) is to include a monotonically increasing counter with every message and reject messages with old counter values. With this policy, every recipient must maintain a table of the last value from every sender it receives. However, for RAM constrained sensor nodes, this defense becomes problematic for even modestly sized networks. Assuming nodes devote only a small fraction of their RAM for this neighbor table, an adversary replaying broadcast messages from many different senders can fill up the table. At this point, the recipient has one of two options: ignore any messages from senders not in its neighbor table, or purge entries from the table. Neither is acceptable; the first creates a DoS attack and the second permits replay attacks. In [21], the authors contend that protection against the replay of data packets should be provided at the application layer and not by a secure routing protocol as only the application can fully and accurately detect the replay of data packets (as opposed to retransmissions ,for example). In [18], the authors reason that by using information about the network's topology and communication patterns, the application and routing layers can properly and efficiently manage a limited amount of memory devoted to replay detection. In [19], the authors have identified two types of freshness: weak freshness, which provides partial message ordering, but carries no delay information, and strong freshness, which provides a total order on a request response pair, and allows for delay estimation. Weak freshness is required by sensor measurements, while strong freshness is useful for time synchronization within the network.

- Robustness and Survivability :

The sensor network should be robust against various security attacks, and if an attack succeeds, its impact should be minimized. The compromise of a single node should not break the security of the entire network.

C. *Types of Attacks on WSNs*

Wireless networks are vulnerable to security attacks due to the broadcast nature of the transmission medium. Furthermore, wireless sensor networks have an additional vulnerability

because nodes are often placed in a hostile or dangerous environment where they are not physically protected.

- Passive Information Gathering

An intruder with an appropriately powerful receiver and well designed antenna can easily pick off the data stream. Interception of the messages containing the physical locations of sensor nodes allows an attacker to locate the nodes and destroy them. Besides the locations of sensor nodes, an adversary can observe the application specific content of messages including message IDs, timestamps and other fields. To minimize the threats of passive information gathering, strong encryption techniques needs to be used.

- Subversion of a Node

A particular sensor might be captured, and information stored on it (such as the key) might be obtained by an adversary. If a node has been compromised then how to exclude that node, and that node only, from the sensor network is at issue (LEAP [22] defines an efficient way to do so).

- False Node and malicious data

An intruder might add a node to the system that feeds false data or prevents the passage of true data. Such messages also consume the scarce energy resources of the nodes. This type of attack is called "sleep deprivation torture" in [23]. Insertion of malicious code is one of the most dangerous attacks that can occur. Malicious code injected in the network could spread to all nodes, potentially destroying the whole network, or even worse, taking over the network on behalf of an adversary. A seized sensor network can either send false observations about the environment to a legitimate user or send observations about the monitored area to a malicious user. By spoofing, altering, or replaying routing information, adversaries may be able to create routing loops, attract or repel network traffic, extend or shorten source routes, generate false error messages, partition the network, increase end-to-end latency, etc.

Strong authentication techniques can prevent an adversary from impersonating as a valid node in the sensor network.

- The Sybil attack

In a Sybil attack [24], a single node presents multiple identities to other nodes in the network. They pose a significant threat to geographic routing protocols, where location aware routing requires nodes to exchange coordinate information with their neighbors to efficiently route geographically addressed packets. Authentication and encryption techniques can prevent an outsider to launch a Sybil attack on the sensor network. However, an insider cannot be prevented from participating in the network, but (s)he should only be able to do so using the identities of the nodes (s)he has compromised. Using globally shared key allows an insider to masquerade as any (possibly even nonexistent) node. Public key cryptography can prevent such an insider attack, but it is too expensive to be used in the resource constrained sensor networks. One solution is to have every node share a unique symmetric key with a trusted base station. Two nodes can then use a Needham- Schroeder like protocol to verify each other's identity and establish a shared key. A pair of neighboring nodes can use the resulting key to implement an authenticated, encrypted link between them. An example of a protocol which uses such a scheme is LEAP [22], which supports the establishment of four types of keys.

- Sinkhole attacks

In a sinkhole attack, the adversary's goal is to lure nearly all the traffic from a particular area through a compromised node, creating a metaphorical sinkhole with the adversary at the center. Sinkhole attacks typically work by making a compromised node look especially attractive to surrounding nodes with respect to the routing algorithm. For instance, an adversary could spoof or replay an advertisement for an extremely high quality route to a base station. Due to either the real or imagined high quality route through the compromised node, it is likely each neighboring node of the adversary will forward packets destined for a base station through the adversary, and also propagate the attractiveness of the route to its neighbors. Effectively, the adversary creates a large "*sphere of influence*" [25], attracting all traffic destined for a base station from nodes several hops away from the compromised node.

- Wormholes

In the wormhole attack [26], an adversary tunnels messages received in one part of the network over a low latency link and replays them in a different part. The simplest instance of this attack is a single node situated between two other nodes forwarding messages between the two of them. However, wormhole attacks more commonly involve two distant malicious nodes colluding to understate their distance from each other by relaying packets along an out-of-bound channel available only to the attacker.

An adversary situated close to a base station may be able to completely disrupt routing by creating a well-placed wormhole. An adversary could convince nodes who would normally be multiple hops from a base station that they are only one or two hops away via the wormhole. This can create a sinkhole: since the adversary on the other side of the wormhole can artificially provide a highquality route to the base station, potentially all traffic in the surrounding area will be drawn through her if alternate routes are significantly less attractive.

III. RELATED WORK

There are several papers that have studied the cost of security algorithms and operation modes. Ganesan et al. [27] measure the energy consumption with respect to different encryption and hash algorithms so that designers can easily predict the performance of their system. They study three different encryption algorithms (RC4, IDEA, and RC5) and two message digest algorithms (MD5 and SHA1) and measure the computational overhead they cause on 6 different platforms (Amega 103, Atmega 128, M16C/10, SA-1110, PXA250, and UltraSparc2). However, they do not study MSP430, which is the MCU used in TelosB.

Law et al. [28] analyze the influence of block ciphers on WSNs and compare the MCU cycles and memory used by encryption algorithms. They have studied multiple algorithms (Skipjack, RC5, RC6, MISTY1, Rijndael, Twofish, KASUMI, and Camellia). They have also measured the memory usage of

the standard modes of operation (CBC, CFB, OFB, and CTR) but have not evaluated their energy consumption. In addition, they do not specify the platform they use in their experiments. Guimaraes et al. [29] evaluate the energy cost of security algorithms (TEA, Skipjack, and RC5) on Mica2 using TinySec. Their results show the impact of security on the MCU and memory usage for a single mote. However, they use Mica2 motes with a CC1000 radio. In addition, according to their results, Skipjack consumes more energy than RC5.
Jinwala et al. [30] have implemented AES and XXTEA for Mica2. They have included these algorithms into TinySec and compared their performance to the default TinySec cipher, Skipjack. The experiments were performed using the Avrora simulator, which provides the CPU cycles, throughput, and energy consumption. The authors have concluded that XXTEA with an OCB mode of operation is the optimal security combination for WSNs.

As we mentioned earlier, none of the previous work has studied the impact of the different algorithm parameters (e.g. key size) on the energy consumption of AES and RC5. In addition, although hardware implemented AES-128 is expected to be extremely secure, its energy consumption has not been measured. Further, no previous work has evaluated the energy and memory requirements of MAC algorithms.

Most of the previous papers, except [32] and [32], have used simulation for their evaluations. We, however, believe that directly measuring the current through the motes and the latency caused by the security algorithms provides more realistic data. In addition, since different papers use different sensor platforms and usually study different encryption algorithms, it is very hard to compare the results they provide.

Chang et al. [31], [32] use a setup similar to ours to measure the energy consumption introduced by hash functions and symmetric-key algorithms on Mica2 motes with a CC1000 radio and Ember sensors with an EM2420 radio. They use a PicoScope 3206 oscilloscope to sample the voltage drop across two registers and, based on the measured data, speculate about the energy consumption. They study a different subset of symmetric key algorithms. In addition, since they run their experiments on Mica2 motes, the results they provide cannot be used as a reference for MicaZ or TelosB motes.

One of the first requirements for providing a security mechanism is establishing the cryptographic keys to be used by the encryption algorithms. Due to the limited resources and the need for scalability in WSNs, they key establishment protocols used in other fields are not suitable for WSN environments. To address this problem, a lot of work has been done to develop and evaluate specialized key establishment protocols [33], [34], [35].

IV. STUDIED BLOCK CIPHERS

In this section we give a more detailed overview of the cryptographic algorithms we have studied. All of the algorithms are blocks ciphers that have been used outside the WSNs community for many years. Block ciphers are known to be the most efficient in terms of energy consumption and latency when compared to other security algorithms. This is the reason why they are preferred for use in WSN environments.

A. RC5 encryption scheme

RC5 is a block cipher with variable parameters: block size (32 bits, 64 bits, and 128 bits), key size and encryption rounds, and it can be expressed as RC5— w/r/b. It was designed by Ron Rivest and analysed by the RSA laboratory.[7] RC5 algorithm is a very compact algorithm with its data processing using only the general operation for the common microprocessors, such as modular addition, XOR, and cyclic shift. In RC5 two w-bits registers are used to store the plaintext input, and the ciphertext is stored in the same registers. Because it is a symmetric encryption algorithm, the decryption is the inverse of encryption. From what this algorithm shows us, RC5 makes the input data depend on the number of cyclic shifts in order to achieve the capacity that the number of the cyclic shifts cannot be predicted. With the features of low storage space, fast speed, and variable number of rounds and key length, RC5 is widely used in WSNs. With in-depth theoretical analysis, we find that RC5 has some security risks because of its intrinsic weak diffusion.[7] Concrete algorithm can be found in Ref. [6].

B. RC6 encryption scheme

RC6 is a new block cipher submitted to NIST for consideration as the new Advanced Encryption Standard (AES). The design of RC6 began with a consideration of RC5 as a potential candidate for an AES submission. Modifications were then made to meet the AES requirements, to increase security, and to improve performance.[21] The inner loop, however, is based on the same "half-round" found in RC5. RC6 is a further development based on RC5 by using the quadratic function. Function $f(x) = x\,(2x+1)$ is used to enhance the diffusion rate, so RC6 can increase the security with less loops than RC5. At the same time, RC6 handles 128-bits input/output blocks with the variable key size and the number of rounds, so users can flexibly set the parameter of RC6 to meet the future growth and market demands. Concrete algorithm is seen in Ref. [36].

C. AES encryption scheme

AES is an iterative block cipher and designed by Joan Daeman and Vincent Rijmen in Belgium in response to the AES. It is composed mainly of nonlinear components, linear components, and round keys, and though it employs an iterative structure, is does not have a Feistel network structure but an SP structure instead. The AES specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. [6, 31] The AES encryption scheme was designed to handle additional block sizes and key lengths. However they are not adopted in this standard. AES uses the SP (substitute permutation) network structure with the 128-bits blocks and three optional key lengths of 128-bits, 192- bits, and 256-bits. Round r depends on the key length. If key length is 128-bits, the r = 10; if the key length is 192-bits, the r = 12; if the key length is 256-bits, the r = 14 [6].

V. PERFORMANCE EVALUATION OF THE ALGORITHMES

This section presents a comparative performance and energy consumption analysis of this algorithms has been investigated by comparing with alternative popular algorithms. We have selected three crucial parameters to compare each algorithm's performance namely; memory requirements, execution time, and energy efficiency.

A. *Memory Efficiency* :

Memory usually includes flash memory(ROM) and RAM. Flash memory is classified into programming flash memory and data flash memory. Programming flash memory is used to store downloaded application programming code. Data flash memory stores temporary or sensing data. RAM is used for program execution. Because memory in a sensor node is not only limited but also require energy to retain or store data, efficient usage of memory is important.

Table. 1 describes amount of needed memory of RC5, RC6 and AES. RC5 requires a little memory in both ROM and RAM. But AES has poor memory efficiency. The memory efficiency of RC6 is better than that of AES but worse than that of RC5.

Table 1 : Memory requirements of each algorithm

Encryption algorithm	RAM (Kb)	ROM (Kb)
AES	2.14	9
RC5	0.15	3
RC6	0.05	4.45

In RC5, in contrast to the above algorithms, the number of rounds does not depend on the key size. Therefore, RC5's security strength is determined by both the key size and the number of rounds.

B. *Operation Time* :

Operation speed is also an important factor when evaluating performance. After estimating operation time by repeatedly executing encryption and decryption process, we calculate the average of estimated value.

We have measured the execution times of key setup, encryption, and decryption and calculated the corresponding energy consumption of AES-128, AES-192, and AES- 256. The results are shown in Table 2. As we can see, all three phases key setup, encryption, and decryption are influenced by the key size. The longer the key, the longer it takes for the phases to execute. Note that for all three algorithms AES-128, AES-192, and AES- 256, the encryption time is about 43% of the decryption time. This is a reasonable ratio since the decryption phase is more complicated and requires more computations.

The number of rounds has a proportional effect on the security of RC5 [35]. To see how the number of rounds influences the energy consumption we have executed RC5 with 16 rounds. We have also used different values for the key size. We have used the same key sizes as in AES – 128, 192, and 256 bits, we have measured the time necessary to perform the key setup, encryption, and decryption phases and calculated the operation time. Since RC5 and RC6 are usually structured with simple operation such as XOR, bitwise rotation, and addition. Table 2 show the operation time of cryptographic algorithms. We discover they have very similar operation time.

Table 2 : Operation time requirements of each algorithm

Key size (bits)		Key setup (ms)	Encryption (ms)	Decryption (ms)
AES	128	2.44	1.53	3.52
	192	2.68	1.82	4.25
	256	3.01	2.11	4.98
RC5	128	2.33	2.01	2.47
	192	2.62	2.30	3.20
	256	2.92	2.59	3.93
RC6	128	4.50	4.33	5.95
	192	4.79	4.62	6.68
	256	5.08	4.91	7.41

C. *Energy Efficiency* :

The energy consumed by a processor during the execution of a piece of software, such as a block cipher, corresponds to the product of the average power dissipation and the total running time. The former depends on a number of factors including supply voltage, clock frequency, and the average current drawn by the processor while executing individual instructions of the program code. The computational complexity of an algorithm translates directly to its energy consumption. Assuming the energy per CPU cycle is fixed, by measuring the number of CPU cycle executed per byte of plaintext processed, we get the amount of energy consumed per byte. We estimate CPU cycle by using PowerTOSSIM, which is extension of TOSSIM, an event driven simulation environment for TinyOS applications. PowerTOSSIM provides accurate estimation of power consumption for a range of applications and scales to support very large simulation. The energy consumed by a processor during the execution of a piece of software, such as a block cipher, corresponds to the product of the average power dissipation and the total running time.

Table 3 : Energy Efficiency requirements of each algorithm

Key size (bits)		Key setup (μJ)	Encryption (μJ)	Decryption (μJ)
AES	128	62.32	39.08	89.90
	192	68.45	46.48	108.55
	256	76.88	53.89	116.19
RC5	128	71.51	34.22	33.71
	192	75.34	34.22	33.71
	256	76.10	34.22	33.71
RC6	128	90.35	40.10	58.95
	192	96.48	40.10	58.95
	256	104.91	40.10	58.95

Table 3 presents the power consumption of AES, RC5 and RC6 when each cryptographic algorithm is executed on Mica2 mote.

As we see the table, AES is the most energy-efficient cryptographic algorithm. RC5 is better than RC6 in the aspect of power consumption. Although RC5 presents relatively well performance, it does not provide enough security.

These results, however, could change depending on the mode of operation that is used. Some modes allow decryption to be performed without using a decryption block, which can significantly speed up the execution. Although several papers, such as [28] and [30], have included operation modes in their experiments, few of them have focused on the modes themselves.

VI. CONCLUSION

The performance evaluation of cryptographic algorithms is vital for the safe and efficient development of cryptosystem in devices with low computational power. The approach presented in this paper and its respective tools allow efficiency estimation of the algorithms in resource constrained devices. We evaluated the performance and energyefficiency of the block ciphers AES, RC5 and RC6 cryptographic algorithms.
We conducted our evaluation with "lightweight" software implementations optimized for small code size and low memory footprint.
One future research is to explore adaptive cryptographic mechanisms to optimize energy consumption by varying cipher parameters with timely acquisition of resource-context in WSN environment. The adaptability of the security system will improve sensor nodes battery's lifetime.

REFERENCES

[1] Guido Bertoni, Luca Breveglieri, Matteo Venturi, Power aware design of an elliptic curve coprocessor for 8 bit platforms, in: PERCOMW 2006, pp. 337.

[2] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, Sheueling Chang Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: PERCOM 2005, pp. 324–328.

[3] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, Niraj K. Jha, A study of the energy consumption characteristics of cryptographic algorithms and security protocols, in: IEEE TMC 2005, pp. 128–143.

[4] Chih-Chun Chang, Sead Muftic, David J. Nagel, Measurement of energy costs of security in wireless sensor nodes, in: ICCCN 2007, pp. 95–102.

[5] Y. W. Law, J. M. Doumen, and P. H. Hartel. "Benchmarking block ciphers for wireless sensor networks (extended abstract)". In 1st IEEE Int. Conf. onMobile Adhoc and Sensor Systems (MASS), page electronic edition, Fort Lauderdale, Florida, Oct 2004. IEEE Computer Society Press, Los Alamitos, California.

[6] "SkipJack and KEA algorithm Specifications". National Institute of Standards and Technology, Mai. 1998.

[7] R. L. Rivest, "The RC5 Encryption Algorithm", in Proc. 1994 LeuvenWorkshop on Fast Software Encryption, 1995, pp. 86.96.

[8] I.F.Akyildiz, W. Su, Y. Sankarasubramaniam E. Capirci, «Wireless Sensor Networks: a Survey », Computer Networks, Vol 38, N. 4, March 2002.

[9] Romer, K.; Mattern, F., "The design space of wireless sensor networks," Wireless Communications, IEEE, vol.11, no.6, pp. 54-61, Dec. 2004.

[10] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, « Wireless sensor networks for habitat monitoring, » Proc. ACM international workshop on Wireless sensor networks and applications, pp. 88–97, 2002.

[11] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, Monitoring volcanic eruptions with a wireless sensor network, Wireless Sensor Networks, 2005. Proceeedings, pp. 108–120 ;

[12] K.B. Lee; M.E. Reichardt, "Open standards for homeland security sensor networks," IEEE Magazine on Instrumentation & Measurement, vol.8, no.5, pp. 14-21, Dec. 2005

[13] H. Baldus, K. Klabunde, and G. Muesch, ―Reliable Set-Up of Medical Body-Sensor Networks,‖ in Proc. EWSN 2004 Berlin, Germany, Jan. 2004.

[14] Alippi, C.; Galperti, C., "An Adaptive System for Optimal Solar Energy Harvesting in Wireless Sensor Network Nodes," Circuits and Systems I: Regular Papers, IEEE Transactions on, vol.55, no.6, pp.1742-1750, July 2008.

[15] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck, and M.B. Srivastava, "On communication security in wireless ad-hoc sensor networks", In Proceedings of 11th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE‟02), 2002, pp. 139-144.

[16] D.W. Carman, P.S. Krus, and B.J. Matt, "Constraints and approaches for distributed sensor network security", Technical Report 00-010, NAI Labs, Network Associates Inc., Glenwood, MD, 2009.

[17] S. Lindsey, C. Raghavendra, K.M. Sivalingam, Data gathering algorithms in sensor networks using energy metrics, IEEE Trans. Parallel Distrib. Sys. 13 (9) (2002) 924–935.

[18] Chris Karlof, Naveen Sastry, David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. ACM SenSys 2004, November 3-5, 2004.

[19] Yang Xiao,(Eds.) Wireless Sensor Network Security: A Survey. Security in Distributed, Grid, and Pervasive Computing, Auerbach Publications, CRC Press, 2006

[20] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In Mobile Computing and Networking, pages 263–270, 1999.

[21] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In Proceedings of the 6th annual international conference on Mobile computing and networking, pages 243–254. ACM Press, 2000.

[22] Sencun Zhu, Sanjeev Setia, Sushil Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. In The Proceedings of the 10th ACM conference on Computer and communications security, 2003.

[23] F. Stajano, R. Anderson. "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks", 3rd AT&T Software Symposium, Middletown, NJ, October 1999.

[24] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In The Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), 2002.

[25] Chris Karlof David Wagner. In Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures.

[26] Y.C. Hu, A. Perrig, and D. B. Johnson, "Wormhole detection in wireless ad hoc networks," Department of Computer Science, Rice University, Tech. Rep. TR01-384, June 2002.

[27] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, Mihail Sichitiu, Analyzing and modeling encryption overhead for sensor network nodes, in: WSNA 2003, pp. 151–159.

[28] Niels Ferguson, Richard Schroeppel, Doug Whiting, A simple algebraic representation of Rijndael, Selected Areas in Cryptography, LNCS 2001, pp. 103–111.

[29] G. Guimaraes, E. Souto, D. Sadok, J. Kelner, Evaluation of security mechanisms in wireless sensor networks, in: Systems Communications Proceedings 2005, pp. 428–433.

[30] Gaurav Jolly, Mustafa C. Kusçu, Pallavi Kokate, Mohamed Younis, A low-energy key management protocol for wireless sensor networks, in: ISCC 2003, pp. 335–340.

[31] Chih-Chun Chang, Sead Muftic, David J. Nagel, Measurement of energy costs of security in wireless sensor nodes, in: ICCCN 2007, pp.95–102.

[32] Yee Wei Law, Jeroen Doumen, Pieter Hartel, Survey and benchmark of block ciphers for wireless sensor networks, ACM Transactions on Sensor Networks (2006) 65–93.

[33] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J.D. Tygar, SPINS: security protocols for sensor networks, Wireless Networks (2001) 189–199.

[34] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, Sheueling Chang Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: PERCOM 2005, pp. 324–328.

[35] B.S. Kaliski Jr., Y.L. Yin, On the security of the RC5 encryption algorithm, RSA Laboratories, September 2006, TR-602, Version 1.0.

[36] Guido B, Luca B, Israel K Paolo M and Vincenzo P 2003 14th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'03) (The Hague: The Netherlands) p. 423