Optimizing the Block Cipher Resource Overhead at the Link Layer Security Framework in the Wireless Sensor Networks

Devesh C. Jinwala, Dhiren R. Patel and Kankar S. Dasgupta,

***Abstract*—The security requirements in Wireless Sensor Networks (WSNs) and the mechanisms to support the requirements, demand a critical examination. Therefore, the security protocols employed in WSNs should be so designed, as to yield the optimum performance. The efficiency of the block cipher is, one of the important factors in leveraging the performance of any security protocol.**

In this paper, therefore, we focus on the issue of optimizing the security vs. performance tradeoff in the security protocols in WSNs. As part of the exercise, we evaluate the storage requirements of the block ciphers viz. the Advanced Encryption Standard (AES) cipher Rijndael, the Corrected Block Tiny Encryption Algorithm (XXTEA) using the Output Codebook Block (OCB) mode. We compare our results with the Skipjack cipher in Cipher Block Chaining (CBC) mode.

Our results clearly show the light-weight cipher XXTEA, as the optimal cipher and the Output Codebook Mode as the optimal mode of operation for the link layer security protocols. To the best of our knowledge, ours is the first experimental evaluation of the AES cipher Rijndael, the corrected block Tiny Encryption Algorithm (XXTEA) and the OCB mode in the link layer security architecture for WSNs.

***Index Terms*—authentication, block ciphers, encryption, link layer security, wireless sensor networks**

I. Introduction

Typical wireless sensor networks comprise of the wireless sensor nodes logically interconnected to each other, to realize some vital functionality. Wireless sensor nodes are characterized by severe constraints in power, computational resources, memory, and bandwidth and have small physical size with low power consumption [1].

The communication paradigm in WSNs is *data-centric* multi-hop communication, instead of *route-centric* multi-hop communication, as in case of conventional networks. The data-centric multi-hop communication is characterized by the *in-network* processing. In-network processing involves aggregation, summarization or duplicate elimination in the data collected from different sensor nodes. Since the processing of the data is done on-the-fly, while being transmitted to the base station; the overall communication costs are reduced [2]. Due to the multi-hop communication and the in-network processing demanding applications, the conventional end-to-end security mechanisms are not feasible for the WSN [3]. Hence, the use of the standard *end-to-end* security protocols like SSH, SSL [4] or IPSec [5] in WSN environment is rejected. Instead, appropriate link layer security architecture, with low associated overhead is required.

There are a number of research attempts that aim to do so. The notable ones are TinySec [3], SenSec[6] and MiniSec[7]. These link layer security protocols have an open-ended design so as to enable the use of any block ciphers with appropriate mode of operation.

Also, the range of applications for which the WSNs can be used is very wide. Hence, in order to optimize the security-levels-desired vs. resource-consumption trade-off, the link layer security protocol employed must be *configurable* with respect to (a) the actual cipher and the mode of operation to be employed and, (b) the security attributes desired i.e. encryption, message authentication or replay protection.

We believe that the efficiency of the block cipher is one of the important factors in leveraging the performance of the link layer protocol. Even though the Skipjack (80-bit cipher key with 64-bit block size) [8] is the default block cipher used by TinySec, Sensec and MiniSec; we have attempted to carefully investigate the applicability of

- the Advanced Encryption Standard (AES) block cipher Rijndael (128-bit cipher key with 128-bit block size) [9] and
- the light-weight cipher Corrected Block Tiny Encryption Algorithm (XXTEA) (128-bit cipher key with 64-bit block size) [10]
- the Offset Codebook Mode (OCB) [11] as against the Cipher Block Chaining (CBC) [12] mode as the desired block cipher mode of operation.

In this paper, therefore, we present our experimental results in implementation of the XXTEA cipher in the OCB mode and that of the AES cipher. We use the Skipjack cipher wired in CBC mode, as the baseline, for comparing our evaluation.

To the best of our knowledge, ours is the first attempt in implementing and benchmarking the storage requirements of the XXTEA and the AES ciphers in the CBC and the OCB mode, in the TinySec link layer security protocol.

Manuscript received April 12, 2008. This work was supported in part by the Ministry of Human Resource Department, Govt of India Grants.

Devesh C. Jinwala is with the National Institute of Technology, Ichchhanath, Surat, Gujarat, India, 395007. Phone: 91-261-2259765, fax: 91-261-2228394; e-mail: dcjinwala@gmail.com).

Dhiren R. Patel is with the National Institute of Technology, Ichchhanath, Surat, Gujarat, India, 395007. Phone: 91-261-2201595, fax: 91-261-2228394; e-mail: dhiren29p@gmail.com)

Kankar S. Dasgupta is with the Space Applications Centre, Indian Space Research Organization, Ambavadi, Ahmedabad (e-mail: ksdasgupta@yahoo.com)

We believe that the actual cipher to use and the specific mode of operation to be employed, must be arrived at only after looking at the specific security demands of the application under consideration – rather than by following any abstract model.

Therefore, for the resource constrained WSN environment, we believe this implementation and evaluation exercise will be useful in arriving at the choice of the block cipher, in tune with the available resources and the type of the security desired.

The rest of the paper is organized as follows: in section II, we present the necessary background on link layer protocols and an overview of the related work in the area. In section III, we briefly describe the characteristics of the Skipjack, AES, XXTEA ciphers, and the OCB mode of operation used by us. In section IV, we describe our methodology of evaluation and the experimental setup used. In section V, we present the significance of the results obtained, whereas we conclude in section VI with the future work aimed.

II. BACKGROUND AND RELATED WORK

A. Existing Link Layer Security Architectures

In this section, we briefly discuss the characteristics of the existing link layer security architectures.

TinySec proposed in [3] is designed for the Berkeley Mica Motes. TinySec employs link layer encryption with Skipjack as the default cipher with Cipher Block Chaining (CBC) mode and CBC-MAC (Message Authentication Code) [13] as the authentication mechanism. In TinySec, the authors also optionally evaluate the performance of the block cipher RC5 [14]. The performance overhead with security enabled therein is within 10% of the same without security attributes enabled [3].

The authors of TinySec exploit the advantage of implementing link layer security in software by providing minimal *configurable* security attributes. The configurable security allows different modes of operation viz. (a) support for encryption and authentication, both (b) support for only message authentication (provided by default) or, (c) disabling the security support altogether.

Tieyan Li *et al,* [6] propose an alternate link layer security architecture viz. SenSec that draws upon its basic design from TinySec, but offers encryption as well as authentication by default. Thus, it does not support the configurable link layer security.

Neither TinySec nor SenSec offer replay protection, relegating it to be handled at the application level.

Luk Mark *et al* [7] propose another alternate architecture viz. MiniSec that is designed for the Telos motes [15]. MiniSec uses a different approach in that it offers two operating modes, one tailored for single source communication, whereas the other, for multi-source broadcast communication. It offers all the basic desired link layer security properties viz. data encryption, message integrity and replay protection.

The IEEE 802.15.4 specification specifies a new class of wireless radios and protocols targeted at low power devices, wireless personal area networks (WPANs), and sensor nodes [16]. Unlike wireless local area networks (WLANs), connections effected via WPANs involve little or no infrastructure. This feature allows small, power-efficient, inexpensive solutions to be implemented for a wide range of devices. One of the protocols confirming to the IEEE 802.15.4 standard, is the ZigBee protocol [17]. ZigBee is a specification, targeted at RF applications that require a low data rate, long battery life, and secure networking. But, the use of ZigBee protocol involves appropriate licensing and membership of the ZigBee Consortium.

B. Existing Evaluations of the block ciphers & modes

In this section, we discuss other attempts at evaluating the block ciphers and their modes of operation and emphasize the distinction of our work, here.

In general, the block ciphers used for evaluation in WSN environment are viz. RC5 [14], Skipjack [8], Rijndael [9], Twofish [18], KASUMI [19], Camellia [20] TEA [21].

There have been many benchmarks and evaluation of the block ciphers for the WSNs as surveyed here. But none of them focus specifically on the security at the link layer framework.

Law *et al* in [22], presents a detailed evaluation of the block ciphers viz. Skipjack, RC5, RC6, MISTY1, Rijndael, Twofish, KASUMI, and Camellia. The evaluation is based on security properties, storage and energy efficiency of the ciphers. The results prescribe Skipjack (low security at low memory), MISTY1 (higher security at low memory) and Rijndael (highest speed but higher memory) as the most suitable ciphers depending upon the availability of memory and the required level of security.

However, (a) this work does not consider the OCB block cipher mode of operation (b) as against the recommendation of these results, RC5 has been reported to be having higher speed than AES in [23] (c) the evaluation of the ciphers in [22] is not done within any link layer architecture and (d) no attempt has been made to optimize the cipher code – instead, simply the openSSL [24] versions of the ciphers are employed.

In [25], Großshädl Johann *et al* attempt at energy evaluation of the software implementations of the block ciphers. The authors have considered the ciphers RC6 [26], Rijndael, Serpent [27], Twofish [18] and XTEA [28]. The have used the simulation for the StrongARM SA-1100 processor that is used principally in embedded systems like cell phones and PDAs. The authors also claim to use the optimized "lightweight" implementations of the ciphers that restrict the runtime memory usage to 1 KB.

However, this evaluation does not consider (a) the overhead due to the operating system support or due to the link layer security protocol used (b) the actual deployment of the code on the sensor nodes or any typical WSN platform [29].

In [30] Guimarães Germano *et al* discuss another attempt at evaluating the security mechanisms in WSNs. The authors carry out a number of measurements like (a) the impact of packet overhead on energy consumption (b) the impact of different ciphers on the CPU and memory usage (c) the impact of security layer (including cipher) on the message/network throughput, on the network latency and on the energy consumption (using the PowerTOSSIM simulator).

The authors evaluate the ciphers viz. TEA, Skipjack and RC5. They use the TinySec platform with the Mica2 motes using Atemega128L processor at 7.3728 MHz with 128 KB flash (program memory) and 4 KB of system RAM (Data memory) and Chipcon CC1000 radio. But, in this evaluation, (a) neither a specific cipher is prescribed as a winner (b) nor various other important ciphers like the AES Rijndael and XXTEA are considered for evaluation.

In [31], Luo Xiaohua *et al* evaluate the performance of ciphers viz. SEAL [32], RC4 [33], RC5, TEA by implementation on the Mica2 motes. The evaluation makes the a surprising claim that RC5 is not suited for the WSNs.

In [34], Ganesan Prasanth *et al* attempt on analyzing and modeling the encryption overhead by estimating the execution time and memory occupancy for the encryption as well as message digest algorithms viz. RC4, IDEA[35], RC5, MD5[36], and SHA1[37] on various hardware platforms viz. Atmega 103, Atmega 128, Mitsubishi M16C/10, Intel StrongARM SA-110, Intel XScale PXA250 and SUN UltraSPARC II processors. Thus, the algorithms like the AES Rijndael, XXTEA, Skipjack are not considered.

Thus, none of these evaluations consider the evaluation of (a) OCB block cipher mode of operation (b) the corrected Block TEA cipher and (c) the AES Rijndael cipher on link layer architecture, as we attempted to do, here.

III. The Block Ciphers and the Modes Examined

We have selected the AES Rijndael and XXTEA ciphers for evaluating their performance against the TinySec default cipher Skipjack [8].

Skipjack cipher uses 80-bit key with a 64-bit block size and 32 rounds of an unbalanced Feistel network. It was a classified cipher designed to be used in the Clipper chip and implemented in hardware. But, the cipher was declassified in 1998 with an aim to replace then standard cipher viz. the DES. The best cryptanalytic attack against the cipher was carried out on 31 of the 32 rounds of the cipher, employing differential cryptanalysis [38]. We are using Skipjack for evaluation as a baseline, since it is the cipher of choice, in all existing software based link layer security architectures.

We believe that the size of the cipher key is an indicative measure of the strength of the *computational security* of the cipher. At the minimum, the cipher key size must be enough, so as to prevent the brute force attack against the cipher. With the rapid advancement in technology, the conventional key size of 80-bits is longer sufficient. As per the claims of RSA Security Labs, 80-bit keys would become crackable by 2010 [39]. Hence, it is essential to move towards ciphers with 128-bit cipher key sizes.

Our selection of the Corrected Block Tiny Encryption Algorithm is based on using a 128-bit key size cipher. XXTEA is a simple lightweight cipher, proposed by David Wheeler and Roger Needham of Cambridge University in 1998 [10]. The cipher was proposed to improve upon its predecessor cipher XTEA [28]. XXTEA is an unbalanced Feistel network cipher with 128-bit cipher key with at least 64-bit block size, employing 32 cycles. Because of its simplicity in design, we believe XXTEA is appropriate cipher for the resource constrained WSN environments.

Rijndael, in accordance with the requirements for the Advanced Encryption Standard, is a block cipher with variable 128/192/256-bit key size, the variable 128/192/256-bit block size and variable 10, 12 or 14 rounds. We have selected the AES Rijndael cipher in the configuration of 128/128/10 i.e. using 128-bi cipher key, 128-bit block size and 10 rounds. The Rijndael cipher follows the substitution permutation network structure. We have considered Rijndael for our evaluation because it is the current symmetric key cipher standard.

Finally, we have selected the Output Codebook Mode (OCB) because it combines encryption as well as message authentication in a single pass. The OCB mode was first proposed by Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz, in [12].

OCB scheme integrates the message authentication code (MAC) into the operation of a block cipher. The principal advantage of OCB is overall lower computational and storage costs because it avoids the need to use two different systems viz. a computation of MAC for authentication and a block cipher encryption for privacy.

IV. Experimental Setup & Methodology of Evaluation

We employ a two-step process in the methodology for the evaluation

(a) first, we simulated the performance of the ciphers and modes in the link layer architecture TinySec in TinyOS [40] environment. TinySec is tightly coupled with the TinyOS execution environment with the nesC language [41] as the language of implementation. We have used TOSSIM [42] as the WSN simulator.

(b) next, we deployed the application under consideration on the Mica2 motes with the configuration viz. Atemega128L processor at 7.3728 MHz with 128 KB flash (program memory) and 4 KB of system RAM (Data memory) and Chipcon CC1000 radio.

As compared to Mica2 motes, the *next generation* motes like Intel iMote [43] and Crossbow Iris motes [44] are indeed having higher computational and storage power. But we believe that our evaluation that is carried out on more stringent environment of Mica2 motes, can always be true in more resource-rich environments.

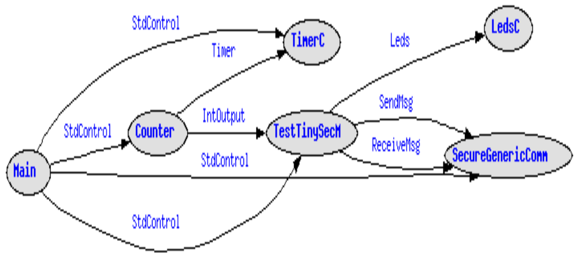

Fig. 4.1 TestTinySec application in TinySec

The basic components of this graph are (a) nodes representing the nesC components (b) the labeled arrows representing the interfaces. The outgoing arrow from a component denotes that the component is using the interface labeled on the flow, whereas an incoming arrow into a component denotes that the component implements the labeled interface. All the components have a .nc extension but we do not use it for simplicity.

As shown in the Fig. 4.1, the *TestTinySecM* module is the main component implementing the application. It uses the TinyOS interfaces SendMsg and ReceiveMsg. These interfaces are implemented by the component SecureGenericComm that is responsible for sending and receiving the secure messages over the radio.

TestTinySecM implements a counter that is incremented on firing of the timer. The counter value modified by the component Counter, is further passed by TesTinySecM through the SendMsg interface for onward transmission over the radio, to the component SecureGenericComm. Also, when the message is sent, the Leds interface is used to toggle the LED on the mote. When the message is transmitted by a mote, the LED is turned green whereas, when the message is received by a mote, the LED is turned red.

The entire communication takes place with the security attributes enabled in TinySec. In Fig. 4.2, we show the partial call-graph showing the security components of the TinySec that come into play, during the execution.

TinySec has been designed to be modular with respect to the selection of the block cipher and the modes of operation. But, as shown in Fig. 4.2, the component SkipJackM that implements the Skipjack cipher and the component CBCModeM that wires Skipjack cipher in CBC mode; are the default cipher and mode of operation. The authentication support is implemented by the TinySec designers in the component CBCMAC. Thus, SkipJackM, CBCMAC and CBCModeM components are not implemented by us. We use them for comparing our components XXTEAM, AESM and OCBM.

We modify the configuration files of TinySec to use the ciphers Rijndael and XXTEA and the OCB as the mode of operation. In Fig. 4.3, we show the partial snapshot of the TestTinySec call-graph with XXTEA cipher in OCB mode.

For implementation we have used the size-optimized C-versions of AES in [45] and the XXTEA version in [10] and converted these versions into the nesC language.

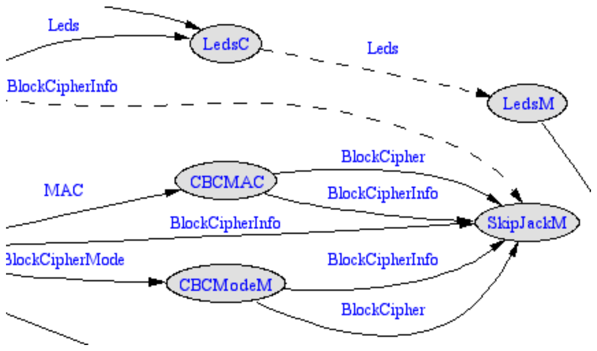

Fig 4.2 TinySec with Skipjack in CBC Mode

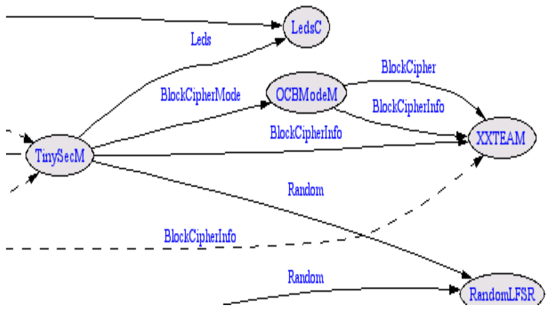

Fig 4.3 TinySec with XXTEA in OCB Mode

We subsequently modified the TinySec configuration files to execute the TestTinySec application using all the combinations of cipher and their modes of operation viz. Skipjack-CBC, Skipjack-OCB, XXTEA-CBC, XXTEA-OCB and AES-CBC.

We compared the openSSL version of AES with the version described in [45]. The openSSL version uses one 8-bit 256 entries S-box and four 32-bit 256 entries forward and reverse tables each, thus consuming a total static storage of 8.448 KB.

As compared to openSSL version, our nesC version of AES is size-optimized. The AES version in [45] uses dynamic computation of tables using only one 8-bit 256 entries S-box and one 32-bit 256 entries forward and reverse tables each, thus consuming a total static storage of 2.304 KB. The reduction is storage is 72% over the openSSL version.

Also, since AES is a 128-bit cipher as compared to the 64-bit Skipjack and XXTEA, we made appropriate logical changes in the TinySec files, for obtaining this support. For XXTEA and AES, we also changed the default tinyos-keyfile to enable the support for 128-bit cipher keys.

V. PERFORMANCE RESULTS

We present the results – evaluating only the storage requirements in all the tested configurations for the application - in the Table I and Table II. We show the simulation results in Table I while the storage requirements for the actual deployments on Mica2 motes, as we evaluated, are shown in Table II.

As we can observe from Table III, when using OCB mode with 64-bit ciphers, significant saving in storage is obtained. The Mica2 motes have 128 KB of program memory while 4 KB of data memory. Hence, approximately 16% conservation in storage, using OCB mode is definitely an advantage over the CBC mode.

TABLE I – TOSSIM SIMULATION

Sr No	Name of the Cipher and its configuration	ROM occupied in bytes	RAM occupied in bytes
1	Skipjack 64/80/32 – CBC	88064	1280880
2	Skipjack 64/80/32 – OCB	78336	1129888
3	XXTEA 64/128/32 – CBC	88064	1284880
4	XXTEA 64/128/32 – OCB	77824	1137888
5	AES 128/128/10 – CBC	94270	4070880

TABLE II – MICA2 IMPLEMENTATION

Sr No	Name of the Cipher and its configuration	ROM occupied in bytes	RAM occupied in bytes
1	Skipjack 64/80/32 – CBC	16754	840
2	Skipjack 64/80/32 – OCB	16650	704
3	XXTEA 64/128/32 – CBC	18142	856
4	XXTEA 64/128/32 – OCB	18038	704
5	AES 128/128/10 – CBC	25562	1332

TABLE III – IMPROVEMENT IN MEMORY UTILIZATION WITH OCB MODE OVER CBC MODE

Sr No	Details	Percentage in ROM	Percentage in RAM
1	Skipjack: MICA2 Implementation	0.62 %	16.19 %
2	XXTEA: MICA2 Implementation	0.57 %	17.75 %

TABLE IV – PENALTY IN MEMORY UTILIZATION USING AES CIPHER

Sr No	Increased Memory Requirement as compared to Cipher	Percentage in ROM	Percentage in RAM
1	Skipjack in CBC mode: MICA2 Implementation	52.57 %	58.57 %
2	XXTEA in CBC mode: MICA2 Implementation	40.89 %	55.60 %

From table IV, we state that using the AES cipher required approximately 50% of increased memory resources. The performance penalty incurred in higher storage requirements should be evaluated against the higher gain in the confidence in security levels, with the use of the current standard cipher.

VI. CONCLUSION AND FUTURE WORK

From the experimental observations, we conclude that (a) using the 128-bit key-sized XXTEA cipher is an attractive option as compared to the 80-bit Skipjack cipher because the increase in storage cost is negligible as compared to the increase in the level of security due to the application of 128-bit key (b) using the OCB mode of operation indeed provides considerable less overhead in storage (c) it is indeed possible to use the AES Rijndael as a block cipher in the link layer architecture implemented in software, too. Although the memory requirements in the AES implementation are still higher than those in the XXTEA and SkipJack, but with the gain in terms of the increased and proven security of a standard cipher with higher key size, it is indeed an attractive option.

We further intend to expand the horizon of this evaluation by doing energy and speed analysis of the same and anticipate that the results derived then, can further strengthen our claims here.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for giving suggestions to refine the contents of this paper.

REFERENCES

[1] Wireless Sensor Networks: Getting Started Guide: Crossbow Technology Incorporated – http://www.crossbow.com
[2] Jeffery Undercoffer, Sasikanth Avancha, Anupam Joshi, John Pinkston, “Security in Wireless Sensor Networks”, *Security Research Symposium, CADIP*, 2002
[3] Chris Karlof, Naveen Sastry, David Wagner, “TinySec: Link Layer Encryption for Tiny Devices”, *ACM Conference on Embedded Networked Sensor Systems*, 2004.
[4] Secure Socket Layer - http://www.openssl.org
[5] IPSec: Requests for Comments viz. RFC 2401, RFC 2402, RFC 2406, RFC 2408 http://www.ietf.org/rfc/rfc240*n*.txt
[6] Tieyan Li, Hongjun Wu, Xinkai Wang, Feng Bao, “SenSec Design, I2R Sensor Network Flagship Project”; Technical Report TR v1.0
[7] Mark Luk, GhitaMezzour, Adrian Perrig, Virgil Gligor, “MiniSec: A Secure Sensor Network Communication Architecture”, *ACM International Conference on Information Processing in Sensor Networks*, April 2007
[8] Skipjack - a representative of a family of encryption algorithms as part of the NSA suite of algorithms; http://csrc.nist.gov/cryptval/des.htm
[9] J.Daemen, V.Rijmen, “AES Proposal: Rijndael”, http://www.esat.kuleuven.ac.be/˜rijmen/rijndael/rijndaeldocV2.zip
[10] David J. Wheeler and Roger M. Needham, “XXTEA: Correction to XTEA” – http://www.cl.cam.ac.uk/ftp/users/djw3/**xxtea**.ps
[11] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, “A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation” *Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97)*, 1997.
[12] Phillip Rogaway, Mihir Bellare, John Black, “OCB: A block-cipher mode of operation for efficient authenticated encryption”, *ACM Transactions on Information and System Security (TISSEC)*, Volume 6, Issue 3, pp.365-403, August 2003
[13] Mihir Bellare, Joe Kilian, Phillip Rogaway, “The security of the cipher block chaining message authentication code”, *Journal of Computer and System Sciences*, Vol 61 Isssue 3, pp.:362-399, December 2000.
[14] Rivest R; “The RC5 Encryption Algorithm”, *Proceedings of the Second International Workshop on Fast Software Encryption*, 1994.
[15] Telos motes – http://www.moteiv.com
[16] IEEE 802.15.4: IEEE Standard for Information technology, Telecommunications and information, exchange between systems, Local and metropolitan area networks specific requirements, IEEE Computer Society, September 2006
[17] ZigBee Alliance. ZigBee specification, Technical Report Document 053474r06, Version 1.0, ZigBee Alliance, June 2005.
[18] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson, “Twofish: A 128-Bit Block Cipher”, http://www.schneier.com/paper-twofish-paper.pdf, June 1998
[19] Specification of the 3GPP Confidentiality and Integrity Algorithms Document 2: KASUMI Specification. ETSI/SAGE Spécification Version: 1.0, Dec 1999. URL http://downloads.securityfocus.com/library/3GTS35.202.pdf
[20] M. Matsui and T. Tokita; “MISTY, KASUMI and Camellia Cipher Algorithm. Mitsubishi Electric ADVANCE (Cryptography Edition)”, Dec 2000. URL http://global.mitsubishielectric.com/pdf/advance/vol100/vol100_complete.pdf
[21] David Wheeler, Roger Needham, “TEA, a tiny encryption algorithm”, *Fast Software Encryption: Second International Workshop, volume 1008 of Lecture Notes in Computer Science*, Leuven, Belgium, 1994.
[22] Law, Y.W., Doumen, J., Hartel, P, “Survey and benchmark block ciphers for wireless sensor networks”; *ACM Transactions on Sensor Networks*, 2006
[23] J Deng, R Han, S. Mishra, “A performance evaluation of intrusion tolerant routing in wireless sensor networks”, *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks*; (IPSN 03), 2003.
[24] openSSL – http://www.openSSL.org
[25] Johann Großshädl, Stefan Tillich, Christian Rechberger, Michael Hofman, Marcel Medwed, “Energy Evaluation of Software Implementations of Block Ciphers under Memory Constriants”, *Proceedings of the 10th Conference to Desgin, Automation and Test in Europe*, (DATE) 2007.
[26] RC6 cipher - http://people.csail.mit.edu/rivest/Rc6.pdf
[27] Serpent page - http://www.cl.cam.ac.uk/~rja14/serpent.html Technical report, Computer Laboratory, University of Cambridge, October 1998

[28] Roger Needham and David Wheeler; Tea extensions; Technical report, Computer Laboratory, University of Cambridge, 1997

[29] Jason Hill, Mike Horton, Raplh Kling, L Krishnamurthy, “The Platforms Enabling Wireless Sensor Networks”, *Communications of ACM;* June 04.

[30] Germano Guimarães, Eduardo Souto, Dajamel Sadok, Judith Kelner, “Evaluation of Security Mechanisms in Wireless Sensor Networks”, *Proceedings of the 2005 Systems Communications (ICW’05)*, 2005

[31] Xiaohua Luo, Kougen Zheng, Yunhe Pan, Zhaohui Wu, “Encryption algorithms comparison for wireless networked sensors”, *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2004

[32] Phillip Rogaway, Don Coppersmith, “SEAL Software-Optimized Encryption Algorithm”; Journal of Cryptology, 1997

[33] RC4 page http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html

[34] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, Mihail Sichitiu, “Analyzing and Modeling Encryption Overhead for Sensor Network Nodes”, *WSNA*, 2003

[35] Lai, Xuejia, and James Massey, "A Proposal for a New Block Encryption Standard" *Proceedings of the Advances in Cryptology-EUROCRYPT '90*, 1992

[36] R. L. Rivest, “The MD5 Message-Digest Algorithm”, RFC 1321; 1992

[37] RFC 3174 on SHA1 - http://tools.ietf.org/html/rfc3174

[38] Biham, E., Biryukov, A., Shamir, A., “Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials”, *EUROCRYPT*, 1999

[39] RSA Laboratories – http://www.rsa.com/rsalabs

[40] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister, “System Architecture Directions for Networked Sensors” *ASPLOS*, 2000

[41] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, David Culler, “The nesC Language: A Holistic Approach to Networked Embedded Systems”, *Proceedings of the ACM SIGPLAN: The Conference on Programming Language Design & Implementation*, 2003.

[42] Philip Levis and Nelson Lee and Matt Welsh and David Culler, “ TOSSIM: accurate and scalable simulation of entire TinyOS applications”, *Proceedings of the 1st international conference on Embedded networked sensor systems*; *SenSys’03*, 2003

[43] Intel’s Imote - http://www.xbow.com/Products

[44] IRIS motes - http://www.xbow.com/Products/wproductsoverview.aspx

[45] AES smaller version; http://hostap.epitest.fi/wpa_supplicant/devel/aes_8c.html