01 Dec 2008

Implementation and Analysis of Practical Algorithm for Data Security

Willi Ballenthin

F. Kacani

Julia Albath

Sanjay Kumar Madria
Missouri University of Science and Technology, madrias@mst.edu

Recommended Citation

W. Ballenthin et al., "Implementation and Analysis of Practical Algorithm for Data Security," *Proceedings of the Fourth International Conference on Wireless Communication and Sensor Networks, 2008*, Institute of Electrical and Electronics Engineers (IEEE), Dec 2008.
The definitive version is available at https://doi.org/10.1109/WCSN.2008.4772690

Implementation and Analysis of Practical Algorithm for Data Security

Willi Ballenthin, Foti Kacani, Julia Albath and Sanjay Madria
Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65401
madrias@mst.edu

In this paper, we present a complete implementation of the Practical Algorithm for Data Security (PADS) proposed by Albath et al., an end-to-end security scheme employing symmetric key encryption. The implementation takes full advantage of the modular design of the TinyOS environment. The simplicity of the algorithm allows for efficient implementation in hardware, a requirement for resource constrained devices. The protocol adds only four bytes of data per packet, on par with industry standards. Simulation and empirical results of the scheme are also provided. The analysis shows that the Practical Algorithm for Data Security is superior to standard security schemes.

I. INTRODUCTION

A wireless sensor network (WSN) is a system of independent devices able to collaborate via radio on a set of common tasks that often includes sensing local environmental variables. Sensors, or motes, are provided with a limited computing capacity in addition to a radio communication stack. A network of such nodes is commonly used to measure variables such as temperature, barometric pressure, sunlight intensity, acoustic noise, seismic activity, and local acceleration. The Mica2 sensors [1] developed by Crossbow Technology is an extendable sensing device often employed in WSNs. It features an 8MHz processor along with 128 kilobytes of program memory and 4 kilobytes of random access memory. Its radio stack is well suited for local two way communication and supports a bandwidth of nearly 40 kilobits per second (see table 1). The authors in [2] enumerate a number of WSN applications such as terrain surveillance, troop monitoring, forest fire or flood detection, sructural monitoring (i.e. bridges), habitat monitoring [3], tele-monitoring of physiological data of patients, tracking of doctors, nurses, and patients in a care center among others.

Power efficiency is a prime design consideration both at the hardware level (processor, radio, memory usage) and software level (instruction count, memory footprint, radio utilization). The network lifetime, being the life expectancy of the network as a whole, is used as a network power indicator. Network lifetime maximization involves several aspects such as efficient routing and data aggregation. Hence, researchers are looking into scavenging or harvesting available energy from the environment [4].

Routing [5,6] is one of the main components of WSN, as in any type of network. Routing algorithms need to be computationally efficient and power aware in order to meet the network constraints. Often, however, selecting the shortest path does not result in reduced network lifetime. Hence, routing algorithms which account for sub-optimal paths need to be considered [7]. In-network aggregation, the fusion of data from different sources promises to increase network lifetime [8]. As shown in Figure 1, data from the lower levels is combined at the higher levels. This approach results in reduced overhead of packet exchanges throughout the network while ultimately communicating the same effective information.

A well designed sensor network is built with long term goals in mind. Often times a limited opportunity exists in which to deploy any sort of network and the initial setup must be maintained throughout measurements. For example, a network deployed on the seafloor by a research vessel is not easily modified, yet may be expected to collect data for each season. Resiliency and durability in hostile environments is often accomplished by over-saturating the test bed with sensors. A WSN must be a self organizing structure, so as the topology of the network changes, connections remain wherever possible. As motes begin to fail, others are expected to step up and fill in. Similarly, some devices may be programmed to wake up late in the life of a network in order to extend its life. An ideal implementation might take into account of battery power and expected lifetime of each node to maximize dependability.

Table 1: Mica Specs

Processor Speed	7.37 MHz
Program Flash Memory	128 Kilobytes
EEPROM Storage	4 Kilobytes
Radio Frequency	869 or 916 MHz
Maximum Data Rate	38.4 Kbaud
Maximum Transmission Range	500 feet
Processor Current Draw	Active 8 ma
	Sleep < 15uA
Radio Current Draw	Transmit 27 mA
	Receive 10 mA
	Sleep < 15uA

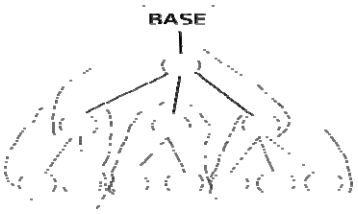

Figure 1: Simple network

Security is another important aspect and its development has enhanced widespread adoption of WSNs. The challenge is two-fold; on one hand, security in the sense of data integrity, confidentiality, and availability, needs to be provided; on the other hand, the power resources of the device and network as a whole are extremely limited. Data integrity is a security feature in which a message sent at one end of the communication channel is guaranteed to be the same message received – imagine that a malicious attacker may attempt to modify transmitted data, but each tampered packet is rejected at reception. If a message cannot be understood by an attacker then the message is said to be confidential. Here, the content of such messages are known only by the sender and the recipient. Finally, availability requires that a message can be transmitted anytime, anywhere, independent of a system. All the three aforementioned characteristics of data communication

978-1-4244-3328-5/08/$25.00 ©2008 IEEE

introduce a significant overhead into wireless sensor networks; the greater the overhead, the shorter the network lifetime.

1.1. BACKGROUND

Security paradigms fall under two major categories: public key based encryption and symmetric key encryption [9]. Symmetric key encryption is based on two principles: a strong cryptographic algorithm and a secret key shared for encryption and decryption. A strong cryptographic system implies that adversaries with knowledge of the cipher text and/or the plain text must not be able to deduce the algorithm being used or its' key. Symmetric key cryptosystems can be subdivided into two broad categories: block ciphers and stream ciphers [10]. A block cipher is a cipher in which the block of plaintext is treated as a whole to produce a block of cipher text of equal length. A stream cipher is similar to a block cipher; however the cryptographic algorithm is applied one bit or one byte at a time. The Vernam cipher is a classical encryption scheme, in which the key is bitwise XORed with the plaintext.

Security schemes can employ two different network approaches end to end encryption and hop by hop encryption as shown in Figure 2a & 2b). End to end security is shown in Figure 2a and performs just two securing functions: encryption of the plaintext at the sender and its decryption at the final destination. This protects sensitive data from sniffing by any intermediary node, trusted or no trusted. In order for this approach to be feasible, the header information such as the routing protocol type and the destination address should not be encrypted. This way, the packets can be directly forwarded to the next nodes.

Unlike an end to end scheme, a hop by hop protocol (as shown in Figure 2b) performs encryption and decryption for each hop along the route. From a security standpoint, hop by hop is more secure when the authenticity of nodes can be assumed. However, when physical security of nodes cannot be assured, end to end protocols might be a more appropriate.

TinySec [11] is a fully-implemented, link layer security architecture for WSN which is natively supported in TinyOS platform [12]. It is built upon existing primitives such as the Message Authentication Code (MAC), a cryptographically secure checksum appended at the end of a message. It also utilizes initialization vectors (IV) with periodic updates for semantic security—in other words, encrypting the same plain text two times yields two different cipher texts. TinySec provides both researchers and application developers with an extensible research platform whose impact on bandwidth, latency, and power is limited. The encryption protocol used in TinySec is a cipher block chaining (CBC) scheme. CBC was designed to be used with IVs to provide semantic security [9]; to produce the first block of cipher text, the IV is XORed with the first block of plaintext, while for the decryption process, and the IV is XORed with the first block of the output of the decryption. The block cipher used was RC5 and Skipjack. Enabling TinySec in TinyOS platforms can be done by setting a parameter during compile time. TinySec provides two options: an authentication-only mode, which is the default mode, and an encryption with authentication mode.

The authors in [13] present a Secure Network Encryption Protocol (SNEP). SNEP gives data integrity through data authentication, protection against replay messages, and data freshness. SNEP has communication overheads of 8 bytes per message. In addition, each end of the communication pair keeps track of a counter, used for identifying double packets. The implementation of the protocol is based on one single block cipher. The code space is reduced by utilizing the same functions for encryption and decryption; a one time pad, based on the counter state, is XORed with the plaintext or the cipher text. A message authentication code, computed per packet, is used to check for authentication and integrity of the message.

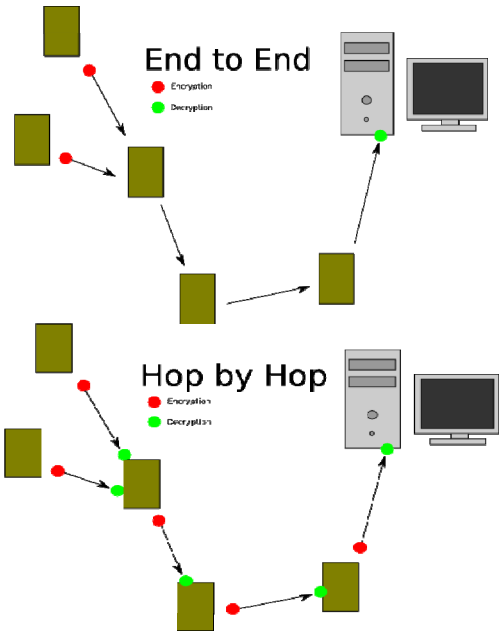

Figure 2a & 2b: Security approaches

1.2. PROBLEM DEFINITION

In communication systems, security remains one of the prime challenges, particularly as personal or sensitive information becomes more widely accessible. Traditional security approaches cannot be directly applied to WSN as the inherent constraints of sensors devices prohibit computationally intensive algorithms. New approaches, therefore, need to be implemented.

Furthermore, WSNs experience vastly different challenges than large-scale wired networks. Eavesdropping is as easy as turning on a radio receiver, while message packets are often implemented as extremely small and simple structures. The lives of soldiers on a battlefield depend on a wireless security that can combine real time accessibility with sufficient privacy. Tradeoffs between power consumption and complexity yield a great range of possible protocols, at times leaving appropriate choices obscured. Without a doubt security schemes optimized for wireless sensor networks have not been fully developed. Current techniques face weaknesses in certain situations, as aggregation or routing aspects prevent top efficiency. It is our belief that further exploration in this field can produce solutions better suited to modern sensor networks.

II. PRACTICAL ALGORITHM FOR DATA

SECURITY

In [14], Albath et al. propose the Practical Algorithm for Data Security (PADS), an end-to-end security scheme based on one time padding. The authors of the PADS technique suggest that the scheme is well suited to wireless sensor networks, and expect an implementation to provide a strong alternative to the industry standard TinySec in certain cases. This paper presents an implementation of the PADS algorithm, as well as empirical data gathered in a number of test environments. The following section provides an overview of the proposed scheme and a summary of the design methodology implemented.

2.1 OUR APPROACH

We provide an analysis of the scheme in order to demonstrate the fitness of this algorithm. At the core of the PADS algorithm is a one time pad (OTP) scheme organized between individual sensors and their respective base stations; the OTP provides both confidentiality and privacy required in many hostile environments.

A one time pad security scheme is a method of encrypting some plain text by combining it with a key, also known as a pad. The pad is generated randomly beforehand, and transforms each element of the plaintext independently. Because the plaintext is not transformed as a unit, as with a block cipher, any subset of the cipher text can yield no information about the plaintext. To an eavesdropper, a three letter cipher text may correspond to any conceivable three letter plaintext. In other words, once the plaintext has been encrypted with a truly random pad, no information besides its length can be determined. For this reason, a well one time executed application is considered perfectly secure.

A critical caveat remains, however, and is directly related to the true randomness of the pad key. Ideally, each pad key should not be used more than once. Repetitions or patterns present within the pad is a critical security flaw, and renders the scheme nearly useless. The proposed approach circumvents this issue by emulating random pad key generation as a function of MAC, data, and time.

The computations involved in encrypting and decrypting with a one time pad are considered relatively simple and can be achieved using bit level operations natively provided by modern processors. While industry standard protocols such as Diffie-Hellman and ElGamal [15] cryptography rely on exponentiation and the factoring large numbers, the application of a pad can be carried out though a byte by byte addition, or even bit by bit exclusive or(XOR) operation. It is this efficient, yet powerful, transformation that makes a onetime pad solution desirable for WSNs.

However, due to the secret key nature of a one time pad based system, and the fact that the pad must be at least the length of plaintext, the gains in computation are apparently lost in memory requirements. Both the remote sensor and base station must share the key, yet also remain functional as the network is scaled up or downwards. It would be infeasible to preload a wireless sensor with a pad long enough to encrypt all transmissions over a multiyear deployment. Furthermore, the requirements of the base station would grow at least linearly with the network size.

The PADS algorithm is proposed to solve this dilemma by requiring only an initialization vector (IV) at compile time, and diverting the key generation phase until just before transmission time. The OTP is then composed by feeding a time variant key and message dependent values into a pseudorandom number generator, and applying the resulting values. By optimally selecting the time variant key period based on message transmission frequency, an application can expect a high degree of randomness, and therefore security.

Despite the apparent advances in message encryption, the PADS protocol is unproven in wireless sensors. WSN performance depends on a number of factors beyond memory requirements, and small changes in protocols may chaotically affect network efficiency as information is routed from node to node. This paper aims to analyze performance, benefits, and tradeoffs of the PADS algorithm. The results will be compared with two other cases: when no security is present, and when another security protocol is enabled, namely TinySec. The first step involved implementing a portable PADS component in the spirit of the TinyOS operating system. This yielded a component that seamlessly interfaces between the application and routing layers. This design methodology encapsulates all the features of PADS in one component; no external knowledge of routing or the application is required for the PADS algorithm to function.

By providing a well designed implementation, subsequent analysis was able to focus on near real life applications and deployment. The focus of our simulation and empirical results is: memory footprint, packet overhead, network latency, power consumption and throughput.

III. DESCRIPTION OF THE FINAL DESIGN

This paper presents an implementation of the PADS algorithm developed by Albath, et al to be used in wireless sensor networks. The code was optimized to take advantage of the modularity of TinyOS architecture; it lives between the application and the routing layer, and performs encryption and decryption of data independent of the application or the routing protocols. In order to test the functionality of the PADS component, several test units were developed. This custom software modeled common network topologies and allowed for empirical data collection. With multimeter and stopwatch in hand, it was possible to compare the TinySec security suite to the PADS component with respect to power consumption, network latency and throughput.

3.1 SOFTWARE ARCHITECTURE

The TinyOS operating system is an open source event-driven application stack built for low power sensor networks. The core utilities were developed at the University of California-Berkeley and first released in the year 2000. Active development continues on a version 2 release, all of which is covered by the BSD license. Components are written in nesC—a variant of C for embedded systems and designed to be highly portable. Functionality is organized through a system of public interfaces and required structures among modules, encouraging rapid mixing-and-matching for custom deployments.

By imitating the interfaces required by radio-aware software, PADS was built as a transparent component sitting above the routing protocol. Data gathering applications are unaffected, as they continue to generate outgoing messages to a global interface. In unsecured networks, a routing system would pick up the message buffer and immediately execute the transmission. However with the PADS component activated –through the addition of four lines in a configuration file – the message buffer is first encrypted. This encrypted buffer is next passed to the routing scheme

as if no detour had taken place. Such a design allows for the rapid deployment of a PADS-based security protocol, as current projects require little-to-no modification.

The software layers of the three protocols under the microscope are graphically depicted in Figure 3. For the developer, the PADS component implements two major interfaces: Send and Receive. Encryption and decryption takes place within these events, as the one time pad cannot be calculated without an entire message payload. Per the published algorithm, the message authentication code is combined with a section of the time variant key to yield a stream of pseudorandom bits. This pad is conveniently the same length of the message payload itself, so both buffers are combined using the exclusive-or bit operation. Assuming the base station and remote mote are time synchronized, each party has enough information to recompose the pad.

This module also provides the Intercept interface, however it does nothing more than pass the message buffer to the application level. As each remote sensor has a unique time variant key coordinated with only the base station, it would be useless to attempt to decrypt messages in transit. Some components, however, may still be able to usefully act upon the message buffer. A secure hierarchical aggregation module may be implemented at near application level, and could depend on the Intercept event. For this reason, the PADS component provides the Intercept interface for flexibility and developers' convenience.

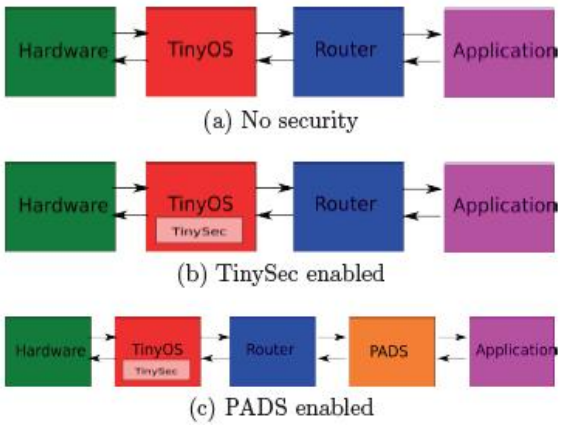

Figure 2: A network of 18 nodes with costs and bandwidth

The lower level interfaces SendMsg and ReceiveMsg are not provided. Both are typically used by a routing layer to direct message packets along specific paths or individual nodes. While this is an attractive functionality to provide, the PADS algorithm cannot be efficiently implemented as a hop-by-hop scheme. Doing so would require a large chunk of memory that could not scale well with network size. The PADS component expects to communicate with a single other node, the base station. For local inter-node communication, a separate parameterized interface of SendMsg should be included by the application developer.

IV. RESULTS AND DISCUSSION

After coding the PADS algorithm into a discrete TinyOS component, a number of experiments were performed to measure performance. Data was gathered concerning power consumption, network throughput, and latency. These three metrics are probably the most important aspects in describing a security protocol, and represent many of the tradeoffs involved. To quantify PADS performance, the same experiments on two additional protocols: a group with no security and a group with TinySec were conducted. In addition to this empirical data, we present theoretical and methodological differences among approaches.

4.1 MEMORY REQUIREMENTS

In terms of memory requirements, we expect the control group to perform most efficiently. Since the PADS component uses libraries provided by TinySec, it is reasonable to assume that the compiler will include much of the TinySec code in the PADS binary as well. This suggests that a PADS-enabled binary should require the greatest amount of resources. In order to test the memory requirements of the various protocols, a simple network application was designed and compiled. An Ad hoc On Demand Distance Vector (AODV) component was included to simulate a routing protocol and control interactions with the hardware. The results parsed from the output of the nesC compiler can be found in Table 2. Overall, the results support our hypothesis. As expected, the PADS component indeed had greater memory requirements than both the Control and TinySec groups.

Table 2: Memory utilization of three security protocols on Motes

Scheme	RAM (Bytes)	ROM (bytes)
MemTest no security	743	13770
MemTest with TinySec (authentication)	1002	22886
MemTest with PADS	1606	30370

4.2 PACKET STRUCTURE

The packet size affects the computational/processing and the radio operation. Since they account for the majority of the power consumption, there is a direct correlation between the packet size and the power consumption.

Figure 3 shows the packet structure of the groups of interest. Compared to the control group, the PADS scheme has an additional 4 bytes overhead, equal to the size of the MAC key used for security operations. Compared to TinySec, we note that there are virtually no differences between the paclet structures. There is, however, a semantic difference as to the location of MAC; while in TinySec the MAC is at the end of the message, in PADS the MAC is coupled to the data message of the packet itself. This difference stems directly from the inherently different approaches to security of the two protocols, end to end and hop by hop. For instance, since PADS is end to end, the MAC should be included in the data field of the packet. This method allows encapsulation of the data, which is later deciphered at the end receiver.

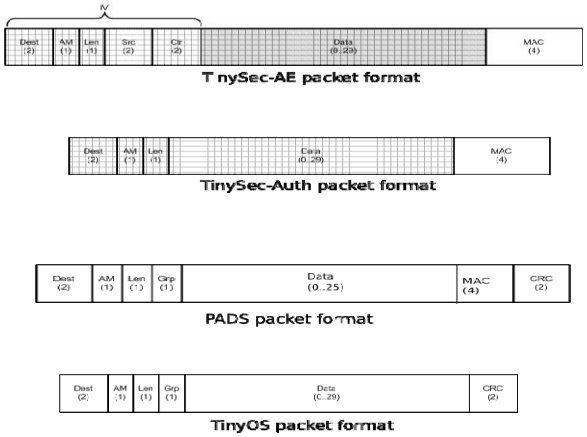

Figure 3: Packet structure of the three security schemes

4.3 LATENCY

To measure the latency of a reasonably sized WSN, it was necessary to design a routing level component for time synchronization. An accurate latency value—which is on the order of a few dozen milliseconds— becomes hard to achieve using multiple timers in a sensor network. Attempts to synchronize clocks across a WSN fall victim to the same latency as to be documented, while two way routing protocols are similarly difficult to implement. The network architecture designed in this paper simulates a multihop topology with a single timer—using just two nodes. A message is transmitted by a master node, which at the same moment begins a timer. The lower level routing protocol bounces this message back and forth between the master and slave node, imitating the process of passing transmissions along a path. After some even number of repetitions, the routing layer of the master node passes the message back up to the application layer, which again records a timestamp. By comparing the two timestamps with respect to the number of messages sent, one is able to determine the average network latency. Due to the flexibility of this design, it is easy to test various network sizes, for nothing but the number of bounces must be modified.

4.4 POWER

Applications were developed to realistically model network communications in a manner that also allowed for reasonable data collection. Power consumption was calculated both by the PowerTOSSIM [16] embedded systems simulator and by voltage monitoring of a distributed network. PowerTOSSIM is a utility provided by Harvard University to the TinyOS community that aims to accurately simulate discrete events across an arbitrary network topology. Extensive hardware data collection had taken place prior to PowerTOSSIM development, and yielded a detailed power model for the Mica2 sensor. The simulator takes advantage of the modular design of TOSSIM [17], the TinyOS simulator, to track events propagated within a simulation. By breaking down network events into atomic units, the simulator is able to determine the power costs of each computational cycle or hardware interrupt. Such a simulator made it possible to run thousands of tests for complicated network topologies of up to 75 nodes. Simulated networks of 10 or 20 nodes can be tested in greater than real time on a reasonably modern personal computer. However, as network complexity grows beyond 30 or so motes, the time requirements per simulation increase rapidly. While we could simulate 2 minutes in a network of 10 nodes in around 30 seconds, a network of 30 nodes could easily take 10 minutes. Time and computing constraints limited data precision of power consumption trials at networks greater than 25 sensors.

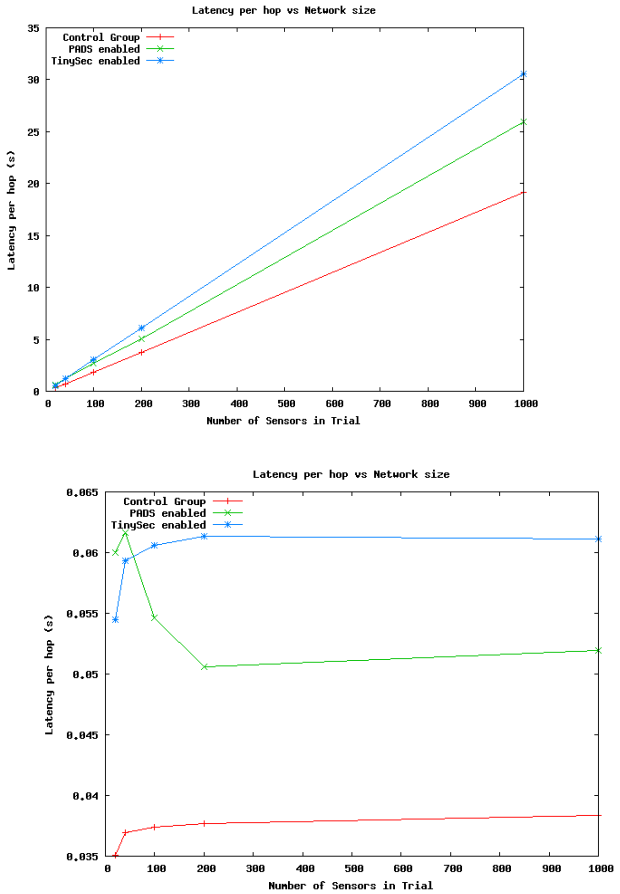

Figure 4. Latency Results (a) Latency overall and (b) Latency per hop

The final results for the central processing unit's (CPU) power consumption and total power consumption are depicted in Figure 5a, and Figure 5b, respectively. The graphs show minuscule difference of the three protocols. There is little we can say with certainty about this data. The authors of TinySec demonstrate a 10% increase in power consumption when their encryption algorithm is enabled. As the PADS component is computationally less complex than the ciphers used by TinySec, we would expect an increase in power consumption by less than 10%. However, to show with sufficient precision (a percent or two) the gradation among the groups, simulations would need to have been run continuously for 10-15 weeks. Standard error calculations suggest each data point is reasonably precise, yet not great enough for true differentiation among groups. A future endeavor with a more relaxed time constraints, or perhaps cluster computing facilities, could produce more enlightening results.

We believe that the accuracy of our results can be improved significantly by increasing the virtual time of the experiment, the number of nodes, and the number of rounds of each of the experiments.

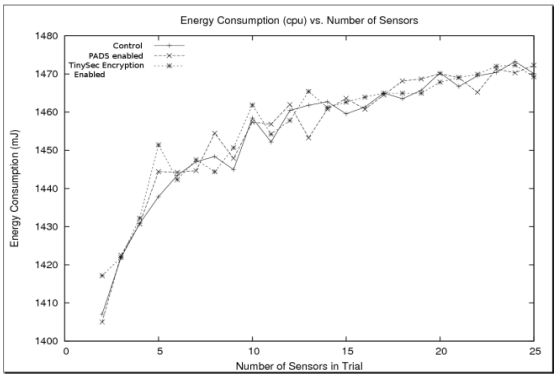

Figure 5a. CPU Energy

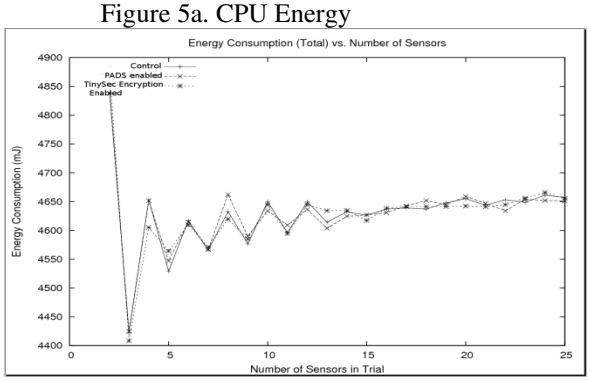

Figure 5b. Total Energy

V. CONCLUSIONS

In this paper we have presented simulation and empirical results of the Practical Algorithm for Data Security, a fully implemented TinyOS security scheme employing symmetric key encryption. Unlike other security protocols, PADS is implemented near the application layer providing end to end security. This feature allows PADS to perform better than standard approaches, particularly as the network size increases. The data we have gathered indicates that PADS is indeed suitable algorithm for data security in WSN. Specifically, PADS performs comparable to or better than TinySec with respect to latency and throughput, while maintaining power consumption rate similar to that of the industry standard.

REFERENCES

[1] I. Technology, "MICA2: Wireless Measurement System," Mica2 Datasheet. Available in: http://www. xbow. com/products/Product pdf _les/Wireless pdf/MICA2 Datasheet. pdf.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Computer Networks, vol. 38, no. 4, pp. 393-422, 2002.

[3] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pp. 88-97, 2002.

[4] J. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, and T. Tuan, Picoradics for wireless sensor networks: the next challenge in ultra-low-power design," Solid-State Circuits Conference, vol. 2, 2002.

[5] C. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," Proceedings of the conference on Communications architectures, protocols and applications, pp. 234-244, 1994.

[6] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing, in Mobile Computing Systems and Applications Proceedings.(WM-CSA'99), pp. 90-100, 1999.

[7] R. Shah and J. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," Wireless Communications and Networking Conference, WCNC2002, IEEE, vol. 1.

[8] L. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on, pp. 575-578, 2002.

[9] W. Stallings, Cryptography And Network Security: Principles and Practice, Prentice Hall, 2006.

[10] A. Menezes, Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publishers, 1993.

[11] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 162—175, 2004.

[12] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, et al., "TinyOS: An Operating System for Sensor Networks," Ambient Intelligence, 2005.

[13] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, "SPINS: Security Protocols for Sensor Networks," Wireless Networks, vol. 8, no. 5, pp. 521—534, 2002.

[14] J. Albath and S. Madria, "Practical algorithm for data security (PADS) in wireless sensor networks," Proceedings of the 6th ACM international workshop on Data engineering for wireless and mobile access, pp. 9-16, 2007.

[15] T. ELGAMAL, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE transactions on information theory, vol. 31, no. 4, pp. 469—472, 1985.

[16] V. Shnayder, M. Hempstead, B. Chen, G. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 188—200, 2004.

[17] P. Levis, N. Lee, M. Welsh, and D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems, (New York, NY, USA), pp. 126-137, ACM Press, 2003.