Investigating and Analyzing the Light-weight ciphers for Wireless Sensor Networks

DEVESH C. JINWALA[1]
DHIREN R. PATEL[1]
KANKAR S. DASGUPTA[2]

[1]Department of Computer Engineering
S. V. National Institute of Technology,
Surat - 395 007, INDIA
[1](dcjinwala,dhiren29p)@gmail.com
[2]Space Applications Centre,
Indian Space Research Organization,
Ahmedabad - 380 015, INDIA
[2]ksdasgupta@yahoo.com

Abstract. The Wireless Sensor Networks (WSNs) are characterized by the severe constraints in the computational, storage and energy resources. Though there has been significant improvement in the available computational resources due to the proliferation of the next-generation sensor nodes, the energy and storage resources of sensor nodes are still limited. As the sensor nodes are often deployed in ubiquitous and pervasive environments, it is necessary to ensure communications security in a WSN. However, the use of the security protocols adds to the associated overhead. Therefore, ensuring communications security in a WSN is a challenge. We believe that since the core component of any security protocol is the cipher used therein; the overhead due to a security protocol can be largely reduced, by employing essentially a lightweight cipher. But at the same time, the cipher so employed must ensure appropriate levels of security with standard key-sizes. Currently, Skipjack (80-bit key-size) is the cipher of choice, for the software based security protocols like TinySec, SenSec and MiniSec, used in WSNs. However, with the help of exhaustive simulation, experiments and analysis, in this paper, we show that our optimized implementation of the Corrected Block Tiny Encryption Algorithm (XXTEA) (128-bit key-size), can be a good alternative to Skipjack; at least in the resource constrained WSN environments.

Keywords: Wireless Sensor Networks, Link Layer Security, Block Ciphers, Encryption, Authentication.

(Received November 30, 2008 / Accepted March 10, 2009)

1 Introduction

The Wireless Sensor Networks (WSNs) consist of sensor nodes collaborating with each other with three major operations viz. *sensing, processing* and *communicating*. The sensor nodes are miniature devices characterized by *lower* computational, storage and energy resources. In addition, due to wireless and the *data-centric* multihop communication paradigm followed by the WSNs (and the consequent necessity for the security protocols to operate at link layer), ensuring the communication security therein becomes a critical issue [33], [13].

Designing a link layer security protocol (LLSP) for WSNs is a precarious issue: the nature of the security protocol to be employed inherently entails greater overhead, whereas the underlying platform inherently faces the scarcity of resources. Hence, it becomes essential to investigate critically, the level and the type of security

offered by the LLSP employed.

Now, whether a security protocol is designed to operate at the link layer or otherwise, the cipher used therein is indeed a very vital component. In fact, the confidence in a security protocol is largely derived from that in the cipher. At the same time, the total overhead associated with the use of such a protocol is largely due to the *intrinsic* design of the cipher and due to the cipher parameters viz. *the key-size, block-size and the number of rounds*. Hence, it is essential to ensure that a lightweight cipher, operating without any compromise on the above cipher parameters (beginning with the key-size), is employed in WSNs.

Here, we attempt to do so. We have selected the lightweight unencumbered ciphers of the Tiny Encryption Algorithm (TEA) family viz. the Tiny Encryption Algorithm (TEA) [55], the Block TEA (XTEA) [44], the corrected Block TEA (XXTEA) [54] as the ciphers of our choice. We compare these ciphers with the Skipjack cipher used in TinySec [32]link layer security base platform. Skipjack cipher is the default unencumbered cipher of choice in the link layer security protocols like TinySec [32], SenSec [39], MiniSec [41], SPINS [12].

We have also implemented our own optimized version of the Corrected Block TEA (XXTEA) cipher, to further reduce the associated overhead and demonstrate the improvement in the performance.

As per our results, the optimized XXTEA cipher definitely can be a good choice for those WSN applications, which do not need the stringent security demands that comes with the AES cipher AES [31].

The long-term implications of our research exercise indeed are far reaching, than our results primarily show. The spectrum of applications for which the WSNs are deployed is very wide - ranging from *high security requirements oriented* Defense Applications to the *authentication only* demanding Environmental Monitoring and Control applications (e.g. monitoring the rainwater to forecast the flood) [29].

Therefore, it is essential that the link layer security protocol employed for a WSN is designed to be configurable with respect to the overhead associated. We discuss the preliminary design of such Configurable LLSP in [30].

The rest of the paper is organized as follows: In the next section we explain how the communication paradigm in WSNs is different from that in the conventional networks and survey the lightweight block ciphers. In section 3, we discuss the existing link layer security architectures implemented as open source software and their design requirements. In section 4 we discuss various peer attempts at evaluating the block cipher to show that our implementation of XXTEA in TinySec platform is indeed unique. Then, in section 5, we elaborate on our methodology for evaluation and the tools as well as the test application used. In section 6, we discuss the simulation results and analyze them. We conclude with the major impact of our exercise in the last section, including the suggestions for future research.

2 Theoretical Background and Motivation

In this section, we justify as to why it is necessary to investigate the communications security in the WSNs in general and the block cipher to be employed therein, in particular.

The WSNs follow the *data-centric* multihop communication paradigm instead of the conventional route-centric multihop communication. The data-centric multihop communication relies on the *in-network* processing of the data. In-network processing is on-the-fly processing (e.g. summarization, duplicate elimination or aggregation) of the data, during its transmission to the base station. The principal advantage of such in-network processing is the overall reduced communication costs.

In-network processing and the data-centric multihop communication necessitate that the data packets be investigated by the intermediate elements too, to decide their further course of communication. Therefore, the conventional security protocols like SSH, SSL [53], IPSec [5] cannot be directly used in WSNs. Hence, it is necessary to devise the security support at the link layer.

The link layer security protocols must offer the security attributes viz. data confidentiality (mechanism: *data encryption*), data integrity and data-origin authentication (mechanism: *keyed hash functions*) and data freshness with protection against replay attacks (mechanism: *an appropriate anti-replay algorithm*).

We briefly discuss the existing link layer security protocols in Section 3.

2.1 The Block Ciphers for Link Layer Protocols

The core component of any security protocol is no doubt, the cipher employed therein. The practical cipher so employed only ensures computational security and does not guarantee the unconditional security (except for the one-time pad) [51].

Fundamentally, the computational security of a cipher can be ensured by (1) using a key-size sufficient enough to prevent the brute-force attacks and (2) an attack-proof design. The former is a (tangible) logarithmic measure of the fastest known computationally

feasible attack on the cipher. The latter is a tangible measure of the largest effort required to orchestrate a brute-force attack on the cipher.

Now, in order to ensure that the cipher is secure at least against the brute-force attack, it is essential to ensure minimally that it uses a state-of-the-art key-size. As such, higher the key-size, higher would be cipher's resistance to the brute-force attacks and more would be the confidence in the overall security of the cipher.

Can we then deduce that we should use a cipher with the maximal possible key-size defined in its configuration? Well, there are two issues to be pondered upon, as follows: (a) as per [36], merely using a cipher with higher key-sizes does not ensure the essential cryptographic strength. Thus, having a cipher with a standard key-size is necessary but not sufficient condition (b) with the increase in the permissible key-size of a cipher, the associated overhead also increases.

We centre our discussion only on (b) here assuming that unless any cryptanalytic weaknesses have been discovered, a cipher must be having intrinsic cryptographic strength and it is sufficient to ascertain proper key-sizes with respect to the associated overhead.

Now, employing a key-size that is longer than even the state-of-the-art key-sizes is not a problem in conventional networks, because the resulting overhead is easily absorbed with the abundance of the resources. However, in the resource constrained WSN environments, the key-sizes have to be necessarily optimum, to ensure good level of security and at the same time, be comparable to the existing standards.

In fact, in case of sensor nodes, the resource limitations today, even with the proliferation of the next generation motes; are far greater than those in the conventional PCs. This fact can be observed from the huge difference in the specifications of Berkeley Mica2 [3] motes (released first in 2004), that of the Crossbow Iris motes [2] released in 2007 and the of a typical desktop PC. Hence, when selecting a security protocol for the WSNs, it is essential to carefully arrive at the cipher and the key-size to be employed therein.

Currently, all the link layer security architectures employ the 80-bit key-sized Skipjack [6] as the default cipher of choice. However, as we show in section III, that it is necessary to employ a 128-bit key-size cipher to ensure better levels of security. The current Advanced Encryption Standard (AES) cipher viz. AES indeed supports 128-bit key. Hence, it is tempting to accept the AES as the cipher of choice, even in the WSNs.

However, the following three arguments forewarn against doing so:

- First, AES had not been designed with a focus only on the resource-constrained environments but with that on standardization and long-term security. Indeed, the feasible implementations of the AES on even Smart Cards demand a crypto processor (an accelerator for cryptographic operations) as in [49].
- The energy and computational resource constraints in the WSNs are much more stringent than those even in the embedded systems.
- Besides AES, it is essential to investigate whether any other 128-bit key-sized lightweight cipher can be used in WSN environments at lesser overhead or not.

Along with the above, one aspect that needs clarification is that merely having a cryptographically secure cipher may not be sufficient. The overall strength of a cipher is decided by the *intrinsic strength* of the cipher determined by its design, by the *key-size* employed and by the *extrinsic strength* of the cipher determined by the cryptosystem that uses the cipher i.e. by the actual cipher implementation. One of such parameter is the resistance of the cipher implementation to the side-channel attacks [14]. However, we do not focus on the extrinsic strength of the cipher implementation here. We believe that the peripheral weaknesses of a cryptosystem come into play later; more essential it is, to ensure the intrinsic cryptographic strength of a cipher.

3 Link Layer Security Architectures

In this section, we investigate and discuss various issues related to the design, implementation and practical usages of the link layer security architectures.

3.1 Goals

The security attributes desired from a link layer security solution for WSNs are as follows: (a) Confidentiality: Confidentiality of the data is achieved by encrypting it using a symmetric key block cipher. The data encryption can be achieved with a shared key k and the functions *Ek(m)* and *Dk(m)* for encryption and decryption respectively. As mentioned earlier, in order to ensure the cryptographic strength of the cipher, it is essential to ascertain that the cipher employs a standard key length. Lenstra in [37], attempts to quantify the security of a cryptosystem with respect to its key length in terms of the trust in Data Encryption Standard [20]. Since, DES after being introduced in 1977, was reviewed every five years, it was trusted at least till 1982. Lenstra proposes that the security of DES

in 1982 be treated as the *base security* to calculate the security margin *y* of the cipher. Assuming *k* as the key-size required to carry out the best known attack against a cipher *A*, the number of years till the cipher *A* can be considered secure i.e. the *security margin y* of the cipher *A* can be defined as

$$y = 1982 + (k - 56)\frac{30}{23}$$

If *y* is known i.e. the year up to which the cipher is required to be secure, the minimum required *k* can be found out. If calculated in this manner, the 80-bit key-sized cipher can be considered as safe till 2013 and a 128-bit key-size cipher, till 2076.

However, Lenstra gave this hypothesis in 2001. The recent conservative prescription by ECRYPT [4] dictates a minimal of 128-bits keys for any cipher. In the Table-I, we show the prescribed key-sizes by ECRYPT, NIST [9], NSA [11], and RFC3766 [45].

Thus, we deduce that ensuring the design of a cipher to be attack-proof is a sufficiency condition for security of the cryptosystem, but employing key-sizes equal to 128-bits, is a necessary condition.

(b) Data integrity: Data integrity ensures that the data is not altered fraudulently, during its transmission from the source to the destination. Typically, the unkeyed hash functions can be employed to ensure data integrity. Along with the data integrity, another related essential attribute is source (data origin) authentication. To achieve both, the data integrity as well as data origin authentication, a keyed hash function known as a Message Authentication Code (MAC) algorithm is employed. A message authentication code (MAC) algorithm is typically a family of functions *hk(k, x)* that takes a secret key and the message as its input and generates a MAC-value, which is sent along with the message as a proof of its integrity. MAC must be hard to forge without the secret key [16]. For the WSNs, the number of bits used for MAC-value is generally 4-8 bytes.

(c) Replay Protection: In a replay attack, an adversary stores a data packet without authorization and then retransmits the same, to trick the receiver into accepting it as a genuine packet. The replay attacks can be prevented, therefore by employing, in general, an anti-replay window that stores the arrival information of up to *w* consecutive sequence numbers. Any packet that has a sequence number lesser than the highest sequence number in the received set of packets, would be treated as a replayed packet and silently discarded. Otherwise, it would be accepted as a genuine packet and the sequence number in the set of received packets be advanced.

(d) Message freshness: The message freshness is an indication of whether the data packet received by the receiver is indeed not stale. The freshness parameter captures the gap between the transmission of data from the source and its delivery to the recipient. Message replay protection is a special case of ensuring message freshness.

(e) Availability: The security properties to be satisfied must not prevent the message from being available in time to the recipient. For any security protocol, availability of the data is always a prime concern.

(f) Low Overhead: Low overhead concerns with the overall resource expenditure in ensuring the security properties. In the resource constrained WSN environment, it is essential to ensure that the overhead is at the minimum to optimize the ratio of gain due to security versus the expenditure entailed.

3.2 Implementations of Link Layer Framework

Our cipher implementation is done in TinySec [32]. TinySec uses three different modes of operations, providing either (a) no security support or (b) support for message authentication only (based on Cipher Block Chaining Message Authentication Code (CBCMAC) [16] or (c) support for (b) and message confidentiality via encryption in Cipher Block Chaining (CBC) [15] mode. As mentioned earlier, TinySec employs 80-bit key-sized Skipjack, as the default block cipher. The authors of TinySec also discuss the experimental evaluation of the patented 128-bit key-sized and 128-bit block-sized RC5 block cipher.

We have used TinySec for our experimentation, because of it being open-source and it following a modular plug-in oriented design (to enable further experimentations with the block cipher and modes). We must emphasize that our results are *independent* of the link layer protocol implementation i.e. we expect similar results when using MiniSec or any other protocol.

4 Block Ciphers and their Evaluations

Amidst the presence of a standard cipher like AES, there have been numerous attempts in the pre-WSN period, to design lightweight simple ciphers for the embedded environment. Hence, the applicability of such ciphers in the WSN environment without compromising any security - under the available resources - must be evaluated. We discuss the characteristics of the cipher used by us for evaluation in the next section.

Table 1: Cipher Key-sizes

$Organization/Method$	$DesirableSKCkey-sizeinbits$	$DesirablePKCkey-sizeinbits$	$Securetill$
$ECRYPT$	128	3248	2029
$NIST$	128	3072	2030
NSA	128	-	-
$RFC3766$	128	3253	-

4.1 Characteristics of the ciphers

Skipjack is the 80-bit key size of block cipher, which has the 64-bit block-size and 32 unbalanced Feistel rounds. The best cryptanalytic attack against the cipher was the differential cryptanalysis, carried out on 31 of the 32 rounds of the cipher [18].

The TEA cipher uses 64-bit block cipher with 128-bit key-size and 64 rounds of operation. It is a short Feistel iteration cipher with no preset tables, nor any explicit key mixing routines. The cipher makes alternate use of XOR and ADD to provide non-linearity.

Various cryptographic attacks have been reported on TEA. These include slide attacks by Biryukov and Wagner in [19], equivalent keys in [34], distinguishing attack in [25], [26] and susceptibility to differential cryptanalysis in [43], [8].

The XTEA was proposed by Needham et al. in [44] to overcome the limitations of TEA, principally *the related key attack*. However, XTEA is more vulnerable to differential and truncated differential attacks as per [28], [43], and [35].

The Corrected Block TEA cipher viz. XXTEA was proposed by Wheeler et al. in [54] as an improvement over another simple cipher viz. Block TEA [44] that was published by them in the same paper as the XTEA [44]. XXTEA also uses 128-bit key-sizes with a Feistel network design and variable 6 to 32 rounds depending on the block size. No major published cryptanalytic weaknesses of this cipher are known.

We have selected the TEA family of cipher not only because of the 128-bit key-size employed therein, but also because of the simplicity, minimal key-setup involved and non-encumbrance of the ciphers. We believe that because of its simplistic design, these ciphers ought to be appropriate for the resource constrained WSNs. In the next section, we relate our work with the peer attempts at investigating the block ciphers in the WSN environment.

4.2 Contemporary Evaluations of Block Ciphers

The block ciphers that have been employed in general, for the evaluation in WSN environments are viz. RC5 [47], Skipjack, AES, Twofish [50], Kasumi [42], Camellia [42] and TEA. We survey the attempts at evaluating these ciphers now:

In [46], Soren Rinne et. al. present the performance analysis of the block ciphers viz. TEA, XTEA, SEA [21], AES, HIGHT [8], and DES on the 8-bit AVR Microcontroller architecture [10]. The authors propose that if the memory is highly critical constraint, then the ciphers like TEA, XTEA can be considered as good options.

Shuang Liu in [40] discusses the implementation of TEA algorithm on the Berkeley Motes platform. The authors show that with respect to the execution time requirements, TEA is a favorable cipher in the sensor networks.

Großshädl Johann et al in [23] attempt at the energy evaluation of the software implementations of the block ciphers. In their paper, the authors compare the performance, the energy consumption, the run-time memory requirements and the code sizes of the block ciphers viz. RC6 [48], AES, Serpent [17], Twofish and XTEA. The base platform used for their exercise is the Strong ARM SA-1100 processor (used principally in embedded systems like cell phones and PDAs).

Guimarães Germano et al in [24] present an evaluation of the security mechanisms in WSNs. The authors attempt principally at evaluating the energy/power consumption, memory requirements, and throughput of the ciphers RC5, RC6, TEA, Skipjack and DES. All these evaluations have been carried out on the TinySec platform with the Mica2 mote. However, the authors do not consider the performance of XXTEA cipher (and neither its optimization) as we attempt to do here.

Even otherwise, we believe that our work substantially differs from all of the above attempts because we focus primarily on the performance of the cipher in the link layer security architecture for the WSNs and on the optimization of the XXTEA implementation.

5 Our Evaluation

5.1 The Target Platform

We use the MICA2 motes platform for implementation and simulation. It uses the low power ATmega128 mi-

crocontroller [10]. The CC1000 transceiver used by MICA2 is also designed for very low power and very low voltage wireless applications.

5.2 Implementation Tools

The experimental platform used is TinyOS [27], [7] with nesC [22] as the implementation language and TOSSIM [38] and AVRORA [52] as the simulators.

TinyOS is a small event-driven, component based operating system designed specifically for supporting the concurrency intensive operations required by networked sensors. The programming language used for TinyOS is nesC - a language for programming structured component-based applications.

TOSSIM WSN simulator provides a scalable simulation environment for sensor networks based on TinyOS. TOSSIM is used popularly for testing, debugging, and analyzing TinyOS applications. We have deliberately used TinyOS 1.1.1x because the support for TinySec has been integrated with this version, so as to enable simulation of security features also through TOSSIM.

Although TOSSIM captures TinyOS behavior at very low level, it does not model the power consumption for the motes. This is because it does not model the CPU execution time, and thus, cannot provide accurate information for calculating the CPU energy consumption. Therefore, we are using AVRORA [52] to measure the CPU cycles and power consumption for a particular node. AVRORA runs actual Mica2 code and is an emulator. Avrora runs code in an instruction-by-instruction fashion.

The principal functional module of TinySec is the file `TinySecM.nc`. `TinySecM.nc` nesC module (a) initializes the cipher context required and the routines for computing and verifying the Message Authentication Code (MAC) (b) calls appropriate routines to encrypt, decrypt the data and (c) calls appropriate routines to compute and verify the MAC.

As part of its functionality, it calls the default cipher component SkipJackM.nc. In our experimentation, we replaced SkipJackM.nc with our own appropriate cipher components viz. `TEAM.nc` (for TEA), `XTEAM.nc` (for XTEA), `XXTEAM.nc` (for XXTEA) and `OXXTEAM.nc` (for our own optimized XXTEA implementation).

5.3 Evaluation Methodology

For evaluation, we employed a two-step process:

- First, we used the definition and the C implementations of the TEA, XTEA and XXTEA ciphers as in [55], [44], and [54] and tested them. Next, we implemented these ciphers in nesC and plugged them in the TinySec library. Then we executed our test application and evaluated the performance of each cipher against that of the SkipJack cipher.

- Next, we deployed the application under consideration on the Mica2 motes and evaluated the performance.

As compared to Mica2 motes, the next generation motes like Intel iMote [1] and Crossbow Iris motes [2] are indeed having higher computational and storage power. However, we believe that our evaluation, carried out on more stringent environment of Mica2 motes, can always be applicable in more resource-rich environments.

As the test application, we have used the application `TestTinySecM` that comes bundled with the TinySec installation. The call-graph of this application is as shown in the Fig. 1. As can be seen from the figure, the `TestTinySecM` module is the main component implementing the application. `TestTinySecM` implements a counter that is incremented on firing of the timer. `TestTinySecM` further passes the counter value modified by the component `Counter`, through the `SendMsg` interface for onward transmission over the radio, to the component `SecureGenericComm`.

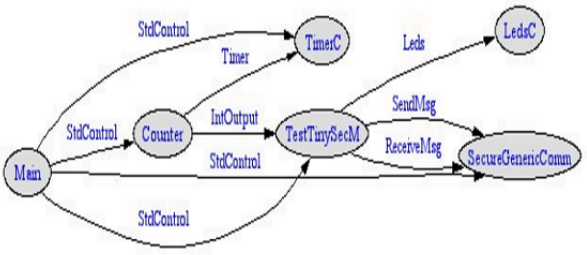

Figure 1: TestTinySec Application in TinySec

In addition, when the message is sent, the `Leds` interface is used to toggle the LED on the mote. The algorithm for the application is shown below. The entire communication takes place with the security attributes enabled in TinySec.

```
1. counter = timer
2. while (counter == fired) {
3. if (Send(Data Packet)) then LED=Green
4. else if (Receive(Data Packet))
5. then LED=Red
}
```

In Fig. 2, we show the partial call-graph showing the default security components of the TinySec that come into play, during the execution. As can be seen, we obtained this call-graph for the default cipher Skipjack. TinySec uses the cipher Skipjack (`SkipJackM`) in

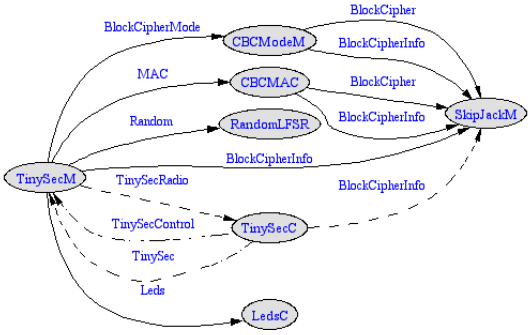

Figure 2: TestTinySec in Default Configuration

the CBC mode as the default mode of operation (`CBCModeM`) with the CBC-MAC as the default message authentication code algorithm (`CBCMAC`). Thus, `SkipJackM`, `CBCMAC` and `CBCModeM` components are not implemented by us. We use them for comparing our components `TEAM`, `XTEAM`, `XXTEAM` and `OXXTEAM`. In Fig. 3, we show the partial snapshot of the TestTinySec call- graph with XXTEA cipher in the CBC mode. We modified the TinySec configuration files to execute

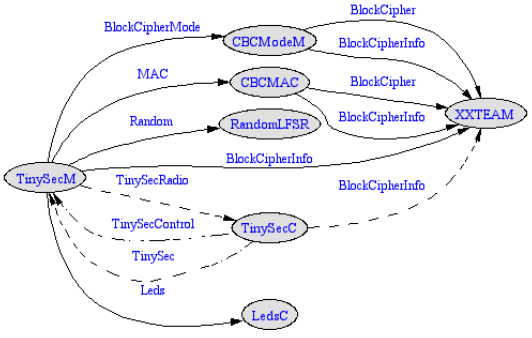

Figure 3: TestTinySec with XXTEA in CBC Mode

the TestTinySec application using all the combinations of cipher in the CBC mode of operation. We also attempted to *optimize* the basic XXTEA operation. For the computations of the XXTEA cipher functions, we use *three specific optimization techniques* in implementation to improve the performance.

First, we use the pre-computed tables for the following sub-computations in various rounds of the cipher function to speed up the operations.

$sum = sum + DELTA$
$e = sum >> 2\&3$
$key[cnt\&3^e]$

These computations are dependent only on the values of DELTA and the `cnt` (i.e. the round number) - that are all constants. Therefore, instead of computing the values at the time of the encryption or decryption, it is beneficial to *pre-compute* and *pre-store* the same in the form of a table and merely *lookup* the values in the able at the run time. Again, these values can be loaded at the *program load time* and *only once*; therefore, to save the overhead at the run time, we use *static const qualifier* to define, compute and pre-store these tables.

Secondly, we exploit the benefit of *inline templates* indirectly, with the use of *predefined macro computations*. The *Inline templates* are a mechanism for directly inserting pre-computed code into an executable. Typically, this approach is used to obtain the better performance for a given function, or to implement an algorithm in a specific way. In general, as such, the compiler is responsible for doing the optimizations in the given high-level code. However, the computations of the XXTEA cipher functions in each round are really *algorithm specific* and hence even a highly optimizing compiler cannot do anything to optimize such code. Therefore, we resort to the use of a *macro definition*. The advantage of a macro definition is not only in the improved readability but also in faster execution; because the macro generates in-line code, avoiding the overhead of a function calls. We also define the macros for both the encryption and decryption routines that call the MX functions repeatedly.

Thirdly, when handling the XXTEA cipher, it is essential to do mutual conversion between the *cipher state* and *the long data type* when doing the cipher operations. We used the inline assembly code for copying a four byte character buffer to a long data type and long data type to a four byte character buffer.

Executing the TestTinySec application, we obtained the results for the memory overhead, CPU cycles and the power consumption for the block ciphers. As we demonstrate in the next section, these optimizations help us in being able to significantly improve the performance of the cipher in the resources constrained sensor nodes.

5.4 Overhead Measurement

For resource overhead measurements, to obtain the simulation results on the desktop, we executed the application in the `dbg` mode with the `crypto` flag set (i.e. `DBG=crypto` in TOSSIM). The `crypto` flag in the TOSSIM simulation allows monitoring and evaluating the cryptographic operations, defined in the application.

For deployment of the application on the actual Mica2 motes, we compiled the application with Mica2 option. By default, the nesC compilation in TinyOS outputs the ROM and RAM requirements of the compiled application for the target motes. Thus, we determine the storage requirements for the simulation and the actual Mica2 motes with the help of the nesC compiler itself, using appropriate options.

To measure the CPU cycles we simulated the TestTinySec application with our implemented block cipher and then converted the Mica2 executable file into ELF file format to emulate it in the AVRORA. We simulated the application for *2 nodes* and for *5 seconds*. The throughput was computed assuming the CPU being clocked at 8 MHz.

We also used the AVRORA simulator for measuring the energy consumption for every node. We ran the energy simulation for *45 nodes* and for *100 seconds*. The results of our experimentation are discussed in the next section.

6 Performance Results and Analysis

6.1 Memory Usage

The result of memory requirements (for RAM and ROM) for every cipher for Mica2, is shown in Fig. 4. We can observe that the entire TEA family of ciphers requires *lesser* overall storage as compared to the Skipjack cipher. To be precise, looking at the total available RAM of 4KB in Mica2 motes, the ciphers Skipjack, XXTEA and our own Optimized XXTEA (XXTEAO) consume 20.31%, 15.23% and 15.23% of the available RAM respectively. Similarly, with the available ROM of

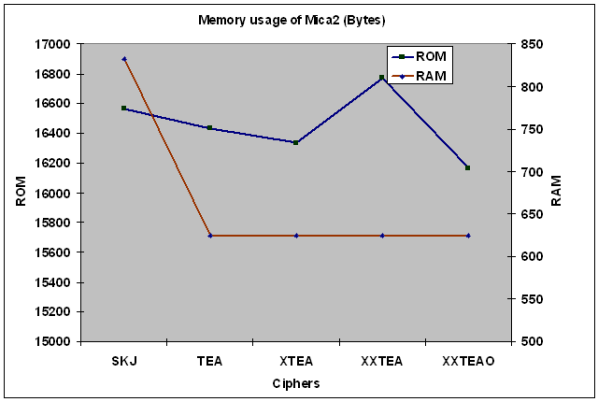

Figure 4: Storage Requirements of the TEA ciphers

128 KB in the Mica2 motes, the ciphers Skipjack, XXTEA and XXTEAO require 12.64%, 12.80% and 12.33% of the available ROM respectively.

Two vital aspects to be emphasized here when analyzing these results, are as follows: (a) We observed that the default implementation of XXTEA from [54]required marginally higher ROM (2%) as compared to Skipjack. Hence, we reimplemented the cipher optimizing it to ensure the lesser overhead. Thus, the XXTEO implementation consumes 2.42% lesser ROM relative to Skipjack - at the same time offering higher security levels due to its 128-bit key-size. (b)In addition, RAM is more critical component in the resource starved sensor nodes. The RAM size in the newer generation motes has not increased as much as (with the improvements in the overall technology) the ROM sizes. Hence, the saving in RAM in the entire TEA family of ciphers, as compared to Skipjack, is significant and vital.

In Fig. 5, we show the results of our evaluation of the CPU cycles consumed for encryption, decryption as well as keys setup operations for the ciphers under consideration. We observed that the Skipjack requires lower number of CPU cycles, as compared to those required by TEA family of ciphers for encryption and decryption. However, for *keys setup*, the Skipjack cipher requires the highest number of CPU cycles. This

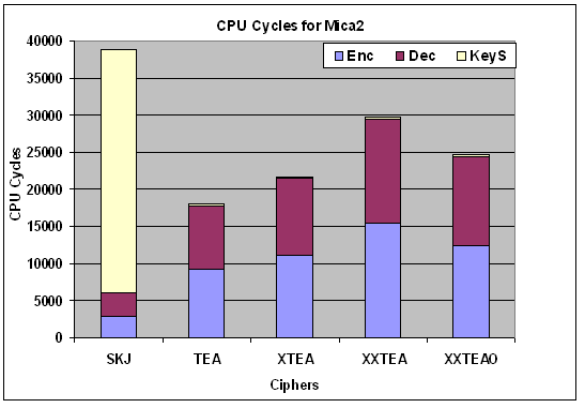

Figure 5: CPU Cycles for security related operations

means that, relative to Skipjack, TEA performs the best with a significant 53.33% lesser total CPU cycles required for encryption, decryption and keys setup. The corresponding figures for XXTEA and XXTEAO are: 23.12% and 33.36% lesser total CPU cycles required relative to Skipjack. Thus, the tradeoff between *the security level* and *CPU overhead* is best achieved in our own implementation of XXTEA viz. OXXTEA.

6.2 Throughput

We considered the parameters viz. the block-size of ciphers, the CPU cycles and CPU clock at 8 MHz for the throughput evaluation. As can be observed from Fig. 6, Skipjack has the highest throughput for encryption and decryption, but it has the lowest throughput for keys setup. The overall performance of Skipjack *degrades* due to this, when compared to the ciphers of TEA family. From the performance results it can be observed that Skipjack is obviously unsuitable for smaller data stream oriented encryption and decryption operations (likely to be found in typical WSN applications). Hence, TEA family of ciphers is more suitable for the WSN applications.

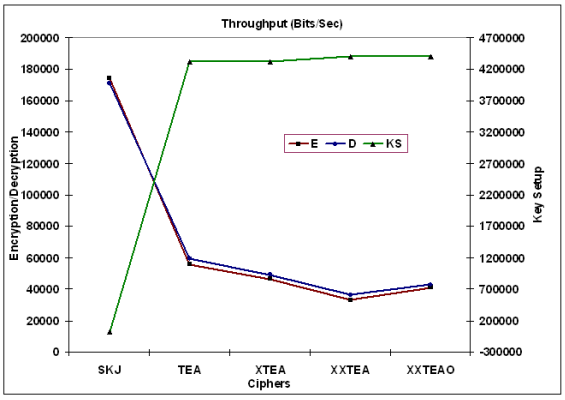

Figure 6: Throughput in bits/seconds

6.3 Energy Consumption

Lastly, in Fig. 7, we show the results of the energy consumption of the CPU and Radio components for every cipher. The energy required for the Radio operations for all the ciphers is almost the same. However, very significantly, the results show that XXTEA and the XXTEAO implementations require only 6.27% and 5.03% more energy than Skipjack. Looking to the fact that the energy is a very critical resource in the WSN nodes, and considering the 128-bit key-size in XXTEA, as compared to the 80-bits in Skipjack, the corresponding increase in energy demand in XXTEA is not significant - at less than 10%.

7 Conclusion

From the experimental results and the analysis, we primarily conclude that the XXTEA cipher is a suitable cipher for the WSN environment. Although we found the Skipjack cipher to be faster and energy efficient for the

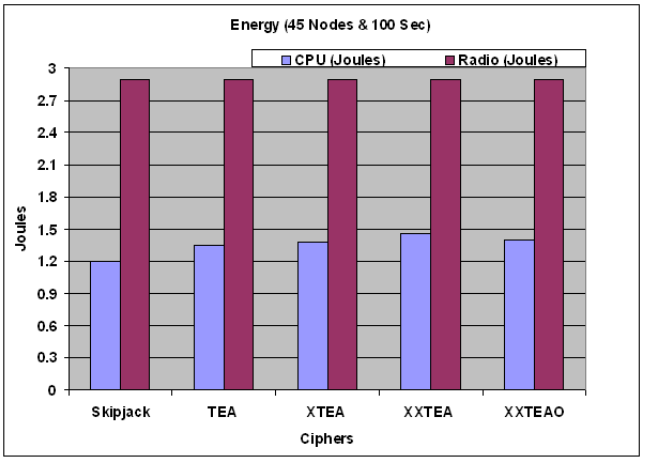

Figure 7: Energy evaluation of ciphers

decryption and encryption operations, the key expansion in Skipjack consumes higher energy. Thus, overall energy requirements in Skipjack are higher. Over an above, it is desirable to have security due to 128-bits key-size, which Skipjack does not offer. The fact that the block ciphers of TEA family have no key expansion, makes these ciphers highly suited for application domains where short messages are encrypted, e.g. in sensor networks.

We again emphasize that the significance of our exercise extends beyond the statistical results that we depict. Our results clearly justify the need for a configurable link layer security architecture - wherein not only the security attributes desired, but the selection of the cipher too, can be based on the nature of the application. The performance gained as a result of employing a light-weight cipher, that provides the level of security sufficient for the application under consideration, is indeed significant for the resource starved sensor nodes. Our current research work is focused on implementing the same. Also, in order to emphasize further confidence in XXTEA cipher, we are investigating the cryptanalysis of the XXTEA cipher.

8 ACKNOWLEDGMENT

We thank all those anonymous reviewers who have devoted significant time and efforts to point out the corrections and improvements to bring this paper to the shape it is in, now.

References

[1] Crossbow iMote2 Data Sheet. http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/IMote2_Datasheet.pdf.

[2] Crossbow IRIS motes. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_Datasheet.pdf.

[3] Crossbow Mica2 Motes. http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.

[4] ECRYPT yearly report on Algorithms and Key Lengths Rev 1.1. http://www.ecrypt.eu.org/documents/D.SPA.28-1.1.pdf.

[5] IPSec: Requests For Comments, RFC 2401, RFC 2402, RFC 2406, RFC 2408. http://www.ietf.org/rfc/rfc240n.txt.

[6] Skipjack - a representative of a family of encryption algorithms as part of the NSA suite of algorithms. http://csrc.nist.gov/cryptval/des.htm.

[7] TinyOS Tutorial Lessons. http://www.tinyos.net.

[8] HIGHT: A New Block Cipher suitable for Low-Resource Device. In *CHES*, pages 46–59, 2006.

[9] NIST Recommendation for Key Management: Special Publication 800-57 Part 1. http://csrc.nist.gov/groups/ST/toolkit/key_management.html, 2007.

[10] ATMEL - 8-bit AVR Microcontroller with 128K Bytes In-System programmable Flash, Atmega 128. http://www.atmel.com/atmel/acrobat/doc2467.pdf, 2008.

[11] NSA Fact Sheet Suite B Cryptography. www.nsa.gov/ia/programs/suiteb_cryptography/index.shtm, 2008.

[12] Adrian, P., Robert, S., D., T. J., Victor, W., and E., C. D. SPINS: Security protocols for sensor networks. *Wirel. Netw.*, 8(5):521–534, 2002.

[13] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[14] Bar-El, H. Introduction to Side-Channel Attacks. http://www.hbarel.com/publications/Introduction_To_Side_Channel_Attacks.pdf.

[15] Bellare, M., Desai, A., Jokipii, E., and Rogaway, P. A Concrete Security Treatment of Symmetric Encryption. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 394, Washington, DC, USA, 1997. IEEE Computer Society.

[16] Bellare, M., Kilian, J., and Rogaway, P. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer System Sciences.*, 61(3):362–399, 2000.

[17] Biham, E., Anderson, R., and Knudsen, L. Serpent: A New Block Cipher Proposal. In *Fast Software Encryption 1998*, pages 222–238. Springer-Verlag, 1998.

[18] Biham, E., Biryukov, A., and Shamir, A. Cryptanalysis of Skipjack reduced to 31 rounds using Impossible Differentials. *Journal of Cryptology*, 18(4):291–311, 2005.

[19] Biryukov, A. and Wagner, D. Slide Attacks. In *Proceedings of the 1999 Fast Software Encryption Conference*, pages 245–259, Berlin Heidelberg, 1999. Springer-Verlag.

[20] Coppersmith, D. The Data Encryption Standard (DES) and its strength against attacks. *IBM Journal of Research and Development*, 38(3):243–250, 1994.

[21] François-Xavier, S., Piret, G., Gershenfeld, N., and Quisquater, J.-j. SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In *Smart Card Research and Applications, Proceedings of CARDIS 2006, LNCS*, pages 222–236. Springer-Verlag, 2006.

[22] Gay, D., Welsh, M., Levis, P., Brewer, E., Behren, R. v., and Culler, D. The nesC language: A Holistic Approach to Networked Embedded Systems. In *In Proceedings of Programming Language Design and Implementation (PLDI*, pages 1–11, 2003.

[23] Großshädl, J., Tillich, S., Rechberger, C., Hofmann, M., and Medwed, M. Energy evaluation of software implementations of block ciphers under memory constraints. In *DATE '07: Proceedings of the conference on Design, automation and test*

in Europe, pages 1110–1115, San Jose, CA, USA, 2007. EDA Consortium.

[24] Guimaraes, G., Souto, E., Sadok, D., and Kelner, J. Evaluation of Security Mechanisms in Wireless Sensor Networks. In *ICW '05: Proceedings of the 2005 Systems Communications*, pages 428–433, Washington, DC, USA, 2005. IEEE Computer Society.

[25] Hernandez, J. and Isasi, P. Finding efficient distinguishers for cryptographic mappings, with an application to the block cipher TEA. In *Proceedings of the 2003 Congress on Evolutionary Computation*, volume 3, pages 2189–2193, 2004.

[26] Hernandez, J. and Isasi, P. New results on the genetic cryptanalysis of TEA and reduced-round versions of XTEA. *New Generartion Computing*, 23(3):233–243, 2005.

[27] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. E., and Pister, K. S. J. System Architecture Directions for Networked Sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[28] Hong, S., Hong, D., Ko, Y., Chang, D., Lee, W., and Lee, S. Differential Cryptanalysis of TEA and XTEA. In *ICISC: Lecture Notes in Computer Science*, pages 402–417, Berlin, Heidelberg, 2004. Springer-Verlag.

[29] Jinwala, D., Patel, D., and Dasgupta, K. A Security Attributes Driven Taxonomy of Wireless Sensor Applications. In *Proceedings of the (Indian Nuclear Society and University of Applied Sciences, Germany sponsored) International conference on Sensors and Related Networks (SENNET 07)*, pages 313–319, INDIA, 2007. Allied Publishers.

[30] Jinwala, D., Patel, D., and Dasgupta, K. Configurable Link Layer Security Architecture for Wireless Sensor Networks. In *Proceedings of IAENG sponsored International Conference on Wireless Networks*, pages 776–780, Hong Kong, China, 2008. World Congress on Engineering.

[31] Joan, D. and Vincent, R. *The Design of Rijndael*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[32] Karlof, C., Sastry, N., and Wagner, D. Tinysec: a link layer security architecture for wireless sensor networks. In *n: SenSys '04: Proceedings of the 2nd international conference on Embedded Networked Sensor Systems*, volume 1008, pages 162–175, ACM, New York, USA, 2004.

[33] Karlof, C. and Wagner, D. Secure routing in wireless sensor networks. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2002.

[34] Kelsey, J., Schneier, B., and Wagner, D. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In *ICICS '97: Proceedings of the First International Conference on Information and Communication Security*, pages 233–246, London, UK, 1997. Springer-Verlag.

[35] Ko, Y., Hong, S., Lee, W., Lee, S., and Kang, J.-S. Related Key Differential Attacks on 27 Rounds of XTEA and Full-Round GOST. In *FSE*, pages 299–316, 2004.

[36] Lenstra, A. K. Selecting Cryptographic Key Sizes. *Journal of Cryptlogy: The Journal of International Association for Cryptographic Research*, 14(4):255–293, 2001.

[37] Lenstra, A. L. *Handbook of Information Security*, chapter Key Lengths. Wiley, 2004.

[38] Levis, P., Lee, N., Welsh, M., and Culler, D. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM.

[39] Li, T., Wu, H., Wang, X., and Bao, F. SenSec Design, I2R Sensor Network Flagship Project. Technical report, Infocomm Security Department, Institute for InfoComm Research, Singapore, 2005.

[40] Liu, S., Gavrylyako, O. V., and Bradford, P. G. Implementing the TEA algorithm on sensors. In *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference*, pages 64–69, New York, NY, USA, 2004. ACM.

[41] Mark, L., Ghita, M., Adrian, P., and Virgil, G. MiniSec: a secure sensor network communication architecture. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 479–488, New York, NY, USA, 2007. ACM.

[42] Matsui, M. and Tokita, T. MISTY, KASUMI and Camellia Cipher Algorithm. *Mitsubishi Electric ADVANCE (Cryptography Edition)*, 2000.

[43] Moon, D., Hwang, K., Lee, W., Lee, S., and Lim, J. Impossible Differential Cryptanalysis of Reduced Round XTEA and TEA. In *FSE '02: Revised Papers from the 9th International Workshop on Fast Software Encryption*, pages 49–60, London, UK, 2002. Springer-Verlag.

[44] Needham, R. M. and Wheeler, D. Tea extensions. Technical report, Computer Laboratory, University of Cambridge, 1997.

[45] Orman, H. and Hoffman, P. Determining Strengths for Public Keys used for Exchanging Symmetric Keys, RFC 3766. *RFC Editor*, 2004.

[46] Rinne, S., Eisenbarth, T., and Paar, C. Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers. In *ECRYPT Workshop SPEED - Software Performance Enhancement for Encryption and Decryption*, Amsterdam, 2007.

[47] Rivest, R. L. The RC5 Encryption Algorithm. Technical report, MIT Laboratory for Computer Science, Cambridge, Massachusetts, March 1997.

[48] Rivest, R. L., Robshaw, M. J. B., and Yin, Y. L. RC6 as the AES. In *AES Candidate Conference*, pages 337–342, 2000.

[49] Sano, F., Koike, M., Kawamura, S.-i., and Shiba, M. Performance evaluation of AES Finalists on the High-End Smart Card. In *Proceedings of the 3rd AES Candidate Conference*, pages 82–93, 2000.

[50] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and Ferguson, N. *The Twofish encryption algorithm: a 128-bit block cipher*. John Wiley & Sons, Inc., New York, NY, USA, 1999.

[51] Shannon, C. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28:656–715, 1949.

[52] Titzer, B. L., Lee, D. K., and Palsberg, J. Avrora: scalable sensor network simulation with precise timing. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, page 67, Piscataway, NJ, USA, 2005. IEEE Press.

[53] Viega, J., Chandra, P., and M, M. *Network Security with OpenSSL*. O'Reilly and Associates, Inc., 2002.

[54] Wheeler, D. and Needham, R. M. XXTEA: Corrections to XTEA. Technical report, Computer Laboratory, University of Cambridge, 1998.

[55] Wheeler, D. J. and Needham, R. M. Tea:a tiny encryption algorithm. In *Proceedings of the Fast Software Encryption:Second International Workshop, LNCS Book Series*, volume 1008, Leuven, Belgium, 1994.

Devesh C. Jinwala was born on 3rd July 1964. He has a Master's degree in Electrical Engineering from the Maharaja Sayajirao University of Baroda, India with specialization in Microprocessor Systems and Applications. He is employed as an Assistant Professor in Computer Engineering with Sardar Vallabhbhai National Institute of Technology, Surat (India) since 1991. He is currently working on Configurable Link layer Security Protocols for Wireless Sensor Networks. His major areas of interest are Information Security Issues in Resource Constrained Environment, Algorithms & Computational Complexity and Software Engineering.

Dhiren R. Patel was born on 29th July 1966. He has a Master's degree in Computer Science & Engineering from IIT Kanpur, India and Ph D in Computer Engineering from the South Gujarat University (REC Surat), India. He is employed as a Professor of Computer Engineering at NIT Surat, India. His major areas of interest are Information/Network Security, Web Engineering and Ubiquitous Architectures. Apart from his numerous International publications and distinguished talks, He has also authored a book *Information Security: Theory & Practice* published by Prentice Hall of India in 2008.

Kankar S. Dasgupta was born on 14th September 1951. He has a Masters Degree in Computer Science and Engineering from Jadavpur University, Kolkata and Doctorate in Electrical Engineering from the Indian Institute of Technology, Bombay. He is currently the Director of DECU at the Indian Space Research Organization, Ahmedabad.