Comparison of Authenticated-Encryption Schemes in Wireless Sensor Networks

Marcos A. Simplicio Jr.*, Bruno T. de Oliveira*, Paulo S. L. M. Barreto*,
Cintia B. Margi*, Tereza C. M. B. Carvalho* and Mats Naslund†
*Escola Politecnica – University of São Paulo, Brazil
Email: {mjunior,btrevizan,cbmargi,pbarreto,carvalho}@larc.usp.br
†Ericsson Research – Stockholm, Sweden. Email: mats.naslund@ericsson.com

***Abstract*—Security is an important concern in any modern network. This also applies to Wireless Sensor Networks (WSNs), especially those used in applications that monitor sensitive information (e.g., health care applications). However, the highly constrained nature of sensors impose a difficult challenge: their reduced availability of memory, processing power and energy hinders the deployment of many modern cryptographic algorithms considered secure. For this reason, the choice of the most memory-, processing- and energy-efficient security solutions is of vital importance in WSNs. To date, several authors have developed extensive analyses comparing different encryption algorithms and key management schemes, while very little attention has been given to message authentication mechanisms. In this paper, we address this issues by identifying Authenticated Encryption with Associated Data (AEAD) schemes suitable for WSNs and by evaluating their features and performance on TelosB sensor nodes. As a result of this analysis, we identify the recommended choices depending on the characteristics of the target network.**

***Index Terms*—benchmark, Authenticated Encryption with Associated Data (AEAD), wireless sensor networks**

I. Introduction

Wireless Sensor Network (WSNs) can be seen as a especial type of ad-hoc network composed by a large number of tiny, cheap and highly resource constrained sensor nodes, known as motes [3]. The sensors are distributed in the area of interest, and can then gather and process data from the environment (e.g., mechanical, thermal, biological, chemical, and optical readings), enabling applications such as environment and habitat monitoring, support for logistics, health care, emergency response, as well as military operations [4].

Motes are typically battery-powered, which has motivated considerable research efforts on the development of energy-aware protocols, such as data link layer protocols (for a survey, see [41]). In general, one of the main goals driving the design of these schemes is to optimize network communications in order to save energy, and thus extend the network's lifetime.

On the other hand, security is often (and sadly) considered at the very last step in the design of WSNs. Actually, most WSN deployments do not even consider security among their requirements because the execution and energy overheads it adds to the system is seen as an undesirable "extra cost" in such constrained environments. However, in WSN-based applications that monitor sensitive information, it is essential to prevent eavesdropping, which is typically obtained by means of encryption algorithms (e.g., symmetric ciphers). Even when the information acquired is not confidential, it is still necessary to ensure data integrity and authenticity by means of message authentication mechanisms, since the acceptance of invalid data (generated either by natural causes or with malicious purposes) could lead to mistaken actions and severe consequences. Finally, given that such algorithms depend on the existence of secret keys for their functioning, applications need also to handle these keys' distribution.

TABLE I
Hardware specification of some motes.

	Processor	*Code Memory*	*RAM*	*Bandwidth*
MICAz [11]	7.3 MHz	128 KiB	4 KiB	250 Kbps
Mica2 [10]	7.3 MHz	128 KiB	4 KiB	38.4 Kbps
FireFly [27]	7.3 MHz	128 KiB	8 KiB	250 Kbps
TelosB [12]	8 MHz	48 KiB	10 KiB	250 Kbps

To date, many security-oriented architectures have been proposed for WSNs, such as TinySec [19], Sensec [23] and MiniSec [26]. In spite of these advances, a main challenge in the security field is that the low resource availability inherent to WSNs still imposes several limitations on the type of cryptographic algorithms that can be effectively deployed in such environments. As shown in Table I, motes usually have 48-128 KiB of code memory, 0.5-10 KiB of data memory (RAM) and are equipped with 8- or 16-bit processors operating at 4-16 MHz; their bandwidth is also small, ranging from 10 to 250 Kbps. Moreover, messages exchanged between nodes are frequently small, a typical packet being between 30 and 60 bytes in length [9]. Finally, a mote constantly operating in active mode is expected to run out of batteries in about 72 hours [43].

It is a well-known fact that transmission in WSNs consumes more energy than computation – 1 bit transmitted may require the power equivalent to executing 800-1000 instructions [19]. Nonetheless, once the communication is already fully optimized, identifying and optimizing resource consuming tasks becomes the next natural step, and cryptographic algorithms usually play a crucial role in this context due to their expected complexity. Indeed, this is the motivation behind several extensive analyses available in the literature. Most of these studies have been focused on the efficiency of symmetric ciphers [8], [16], [21], [24], hash functions [8], [40] and asymmetric algorithms [14], [15] on constrained platforms.

However, and despite the fact that most security architectures rely on message authentication algorithms, only recently some attention has been given to this subject: in [6], Bauer *et al.* evaluate the suitability of some schemes for Authenticated-Encryption with Associated Data (AEAD), which provide authentication of confidential and non-confidential data altogether. Using Atmel's *AVR Studio* to simulate MICAz [11] sensors, their conclusion is that CCFB+H [25] is the best choice for the three different packet formats considered: 1-, 17- and 29- byte messages having an 8-byte header and a 4-byte authentication tag. On the other hand, it remains unclear how the results from [6] generalize to a wider range of scenarios.

Aiming to close this gap, in this paper we present a more comprehensive study on prominent AEAD schemes for WSNs, namely CCFB+H, EAX [7], GCM [30], LETTERSOUP [37] and OCB [20]. We develop both a theoretical analysis – comparing the design characteristics of each algorithm – and an experimental evaluation in real motes – measuring a broad range of metrics (energy consumption, execution time, code size and RAM occupation) for different message, header and tag lengths. Our results show that, even though CCFB+H is indeed an interesting solution in some situations, it would actually provide a lower performance than its counterparts in many significant scenarios. In this manner, this study should help in the choice of the most suitable AEAD algorithm depending on the target application.

The remainder of this document is organized as follows. Section II discusses the usage of AEAD algorithms in the context of WSNs, further motivating our research. Section III describes and analyzes the features of the AEAD schemes covered in this document. Section IV details the benchmark methodology adopted, while section V presents the results obtained. Section VI provides some recommendations depending on the characteristics of the target application and platform. We present our final conclusions in section VII.

II. AUTHENTICATED-ENCRYPTION AND WSNS

Several applications require the deployment of confidentiality and message authentication as security services. Indeed, these are exactly the services offered by most WSN-oriented security architectures, such as TinySec [19], SenSec [23], and Minisec [26]. One important reason for this fact is that confidentiality alone cannot assure a secure communication: without access to the secret key K, attackers are unable to access the content of an encrypted message $C = E_K(M)$; however, they still can flip some bits of C, turning it into C', in such a manner that its receiver will believe that $M' = E_K^{-1}(C')$, and not M, was the message originally sent. Indeed, verifying data authenticity in the whole data path can help saving energy in WSNs, since this allows invalid data to be discarded swiftly, avoiding unnecessary communications [42]. On the other hand, the cryptographic mechanism used for this task needs to be efficient, or its repeated execution on the different nodes would itself become a source of energy waste, facilitating Denial-of-Service attacks [18]. Hence, adopting efficient solutions is essential even in very simple networks in which the nodes basically forward data to a base station. *Authenticated-Encryption (AE)* schemes provide these security services in a seamless manner, combining the operations of a cipher and of a message authentication algorithm under a single key. They can be classified according to how the encryption and authentication operations are bound together. When the solution makes two passes through the data (one aimed at providing privacy and the other, authenticating it), it is called a *Two-Pass* scheme; when single pass is made, we have a *One-Pass* scheme. One-Pass schemes are potentially faster than Two-Pass solutions, but the existence of patents covering the usage of the formers hinders their broad adoption.

Moreover, many applications require a cryptographic solution that allows the authentication of non-encrypted data. This is the case when one desires to authenticate not only the packet's confidential payload but also some plaintext associated to it (e.g., the TCP/IP information, which shall remain unencrypted for packet routing). An AE solution that supports the authentication of messages consisting of both plaintext and ciphertext in this manner is called an *Authenticated Encryption with Associated Data (AEAD)* scheme, where the "associated data" (also called a *header*) refers to the portion of the message that is transmitted as plaintext. Most modern AE schemes – including the solutions hereby considered – are also AEAD schemes, and rely on some Message Authentication Code (MAC) when authenticating the message's header.

AEAD schemes take as arguments an Initialization Vector (IV), together with the message M and header H to be authenticated. Internally, M is encrypted with secret key K, using an underlying cipher under some operation mode, yielding ciphertext C; either M or C is then authenticated together with H, resulting in the authentication tag T (which can be truncated to a suitable length). The output of the algorithm is $C \parallel T$, which can be decrypted and verified using the same key K applied during the encryption process. The authentication of the ciphertext instead of the plaintext is usually preferred, since it allows the receiver to check the authentication tag without having to decrypt the whole message, thus discarding invalid messages earlier. Additionally, like in many block cipher operation modes, the IVs of AEAD schemes usually must be non-repeating (i.e., nonces) in order to prevent attacks. An AEAD scheme is said to provide minimal expansion when we have $|M| = |C|$; this is a common requirement of such solutions, especially when they are used in scenarios such as WSNs, where the cost of transmitting extra bits is very high.

III. LITERATURE SURVEY

In the following, we describe the four two-pass schemes (namely, EAX, GCM, LETTERSOUP and CCFB+H) and the one-pass scheme (OCB) that are subject of our analysis. Since all surveyed solutions display provable security assuming a secure underlying block cipher, the description is focused on performance, implementation and flexibility issues, while some practical security considerations are left for section III-G.

A. EAX

EAX [7] is a strictly sequential two-pass AEAD scheme. Data encryption with this solution is performed using the Counter Mode of operation (CTR) [28], which provides minimal expansion in a straightforward manner. Authentication is provided by CMAC^t, a variant of CMAC [29] where a constant block t is prepended to its input. Different values of t are used for handling the IV, the ciphertext C and the header H, and the results from each process are XORed together for yielding the authentication tag T. The initial counter value of the CTR mode is computed from the IV, also using the CMAC^t algorithm; this allows EAX to process IVs of arbitrary length, but also implicates the need of non-repeating IVs.

Upon reception of a packet authenticated with EAX, the receiver can verify its authenticity prior to its decryption, since T can be computed directly from C. Furthermore, the algorithm does not require the implementation of the cipher's decryption algorithm, since only the encryption process is used by CMAC^t and by the CTR mode. Finally, this scheme is flexible enough to allow the header and the message to be processed in any desired order.

The motivation for including EAX in this survey is that this AEAD scheme is a NIST recommendation that displays a fairly simple design, introducing little memory overhead when a regular CMAC implementation is available. Indeed, this simplicity greatly facilitates implementation, and allows interesting optimizations depending on the amount of memory available for pre-computing some key-dependent constants.

B. OCB

The *Offset CodeBook (OCB)* [20] algorithm, sometimes called OCB 2.0, is a fully parallelizable one-pass AEAD scheme based on a tweaked version of the ECB [28] encryption mode. During OCB's operation, each block of the plaintext is XORed with an IV- and key-dependent offset, encrypted, and then XORed again with that same offset. Only the last block is treated differently: its *length* is XORed with the corresponding offset, encrypted, and then XORed with this last portion of plaintext, preventing the undesirable expansion of the encrypted message. The (potentially padded) plaintext blocks are XORed together and with a final offset, and the result is then encrypted. The output of this operation is the authentication tag T if there is no header to be authenticated; otherwise, the header is processed using PMAC1 [34] and T is computed by XORing the outputs from both processes. Hence, message and header blocks can be computed in any desired order. However, the tag verification process cannot take place prior to decryption, since the tag computation depends on the plaintext data. Moreover, despite the great similarity between the encryption and decryption operations, the latter requires the implementation of E_K^{-1}. This requirement adds some hindrance when implementing OCB, but this task remains relatively simple thanks to a quite clear specification.

The additional overhead introduced by the authentication process in OCB – when compared to the encryption using some non-authenticated mode of operation – consists essentially in the computation of offsets and in the final encryption used for computing the tag; for large enough messages, this overhead is likely to remain very low, accounting for less than 0.1 encryption per block. Thus, even though the adoption of PMAC1 for handling the header leads to an overhead of one full encryption per block of the unencrypted data, OCB is expected to deliver a high performance and a low memory overhead, which motivated its inclusion among the schemes surveyed in this paper. Nonetheless, it is important to note that the algorithm is covered by patents in USA, while its royalty-free use is allowed in projects conforming to the GNU General Public License (GPL) [20].

C. GCM

The Galois/Counter Mode (GCM) [30] is a parallelizable two-pass AEAD scheme. It follows the Carter-Wegman design [31], internally adopting a universal hash-function (GHASH [30]) for authentication and the Counter Mode (CTR) for data encryption.

The specification of GCM has many minutiae, but its operation is essentially as follows: the message M is encrypted using the CTR mode initialized with an IV-dependent value; the resulting ciphertext C and the header H are concatenated with their lengths and then processed using GHASH initialized with a key-dependent hash sub-key; finally, the resulting hash value is XORed with the encryption of the initial counter value, yielding tag T with the same length as the cipher's block size. Such as in EAX, the tag can be verified prior to decryption; however (and unlike EAX), unless finite field exponentiation algorithms are available, the header must be completely processed before the authentication of the ciphertext part starts.

Due to the adoption of the CTR mode, GCM does not require the implementation of the cipher's decryption algorithm and provides minimal ciphertext expansion. Additionally, GCM accepts IVs of any length, since they can be processed using GHASH; there is, however, a significant shortcut for IVs of $(n-32)$ bits (considering a n-bit underlying block cipher): in this case, the IVs are simply padded. In any case, the IVs must be non-repeating for any given key.

The main advantage of the Carter-Wegman design is that it potentially reduces the amount of processing required for its execution. This happens when costly operations, such as finite field multiplications, are implemented with key-dependent lookup tables (LUTs); in these cases, the overhead per message block can be as low as 10%–25% of a cipher call. However, such pre-computed tables are usually large and change as frequently as the key does, i.e., they cannot be statically stored in scenarios where re-keying mechanisms are deployed. Therefore, although such optimizations allow GCM to achieve a high throughput in memory-abundant platforms, they are difficult to be used in memory-constrained platforms.

We decided to include GCM in our analysis because it is part of a NIST recommendation [30] and, thus, it is useful to evaluate its suitability on platforms where space for large

LUTs is hardly available. We note, however, that efficiently implementing GCM is not an easy task due to the number of details and potential pitfalls involved in its specification, such as the mixing of big-endian and little-endian operations and the fact that most of the proposed optimizations are mainly focused on powerful platforms.

D. LetterSoup

LETTERSOUP [37] is a parallelizable two-pass AEAD scheme that targets constrained platforms. It is based on the Linear Feedback Shift Register Counter (LFSRC) mode of operation for encryption and on MARVIN [37] for message authentication. One of the main interests of using MARVIN is that it follows the ALRED construction [13], meaning that each block of the message blocks are processed using a few unkeyed rounds of an underlying block cipher (the so called Square Complete Transform – SCT) instead of a full encryption as in CMAC or PMAC1. This can be 2 to 4 times slower than a Carter-Wegman solution using LUTs, but it does not require extra storage since it reuses part of the underlying block cipher's implementation. At the same time, the LFSRC mode provides minimal data expansion, does not require the implementation of E_K^{-1} for decryption, and, in the case of LETTERSOUP, benefits from MARVIN's choice of offsets (that are also generated using a LFSRC), thus saving processing, memory and bandwidth.

The algorithm's operation uses IV- and key-dependent offsets that are encrypted and then XORed with the message blocks, yielding the ciphertext; the blocks of the ciphertext are then authenticated using MARVIN and the same offsets used during the encryption process, while the header is processed using MARVIN with offsets that do not depend on the IV value; the results from both authentication processes are combined together before a final encryption, which finally generates the tag T. The message and the header can be processed in any desired order, and the tag can be verified before the decryption process takes place. The IVs used must be non-repeating.

Due to the adoption of the ALRED design, authenticating with LETTERSOUP requires about 0.25–0.4 encryptions per (message or header) block processed, which makes this algorithm an interesting subject for our evaluation. However, the main difficulty when implementing LETTERSOUP is also related to the structure of SCTs: since the underlying cipher is not treated as a black-box, the cipher algorithm itself must be adapted in order to obtain an SCT. Except for this issue, the design of LETTERSOUP is quite simple, leading to a reasonably low implementation effort.

E. CCFB+H

The CCFB+H scheme [25] is a strictly sequential two-pass AEAD scheme whose structure strongly resembles that of one-pass solutions. It provides minimal ciphertext expansion and does not need E_K^{-1} to be implemented.

The algorithm adopts a somewhat unusual combination of the Counter (CTR) and Cipher Feedback (CFB) modes of operation [28] for providing both encryption and authentication: for an n-bit underlying block cipher and a τ-bit tag, the input of the block cipher call is the concatenation of the previous $(n - \tau)$-bit ciphertext block and a τ-bit counter; likewise, $(n - \tau)$ bits of the cipher's output are XORed with the plaintext for encryption, while the remaining τ bits are used for computing the authentication tag. The initialization of CFB is performed using an $(n - \tau)$-bit non-repeating IV, which is beforehand XORed with the tag of the same size resulting from the CMAC computation of the header, if it is present. Hence, the header authentication must be finished before the private data can be processed. Moreover, the tag computation process requires the generation of the same keystream used for decrypting the message and, thus, verifying the tag prior to or after decryption involves approximately the same computational effort.

Note that the exact choice of the size of the tag has great impact on the scheme's efficiency: the larger the tag, the higher the number of block cipher invocations per authenticated block (e.g., for $\tau = n - 1$, the encryption+authentication of every single bit would require a full cipher invocation). In fact, this characteristic of CCFB+H also affects its implementation, since some optimization techniques (e.g., the manual unrolling of loops) depend on the size of internal buffers, which in turn depend on the exact tag length adopted; for this reason, in spite of its clear specification, implementing CCFB+H involves many trade-offs between flexibility and efficiency. Nonetheless, as discussed in [6], with the tag length typically adopted by TinySec (namely, 4 bytes), CCFB+H displays a better performance than many similar solutions, which is the main reason why the scheme is included in this study.

F. Security considerations

An attacker trying to create forgeries (i.e., valid message-tag pairs) using any of the discussed solutions is expected to succeed after approximately $2^{\tau-1}$ attempts. Therefore, the tag length τ is an important security parameter for all soltuions considered here, and should be chosen according to the presumed capabilities of the attackers.

Moreover, all AEAD schemes considered in this document require non-repeatable (albeit not necessarily unpredictable) IVs for a same key. The effects of IV repetition is especially disastrous when the encryption is performed using a stream mode of operation (e.g., CTR, LFSRC, and CFB): in EAX, GCM and LETTERSOUP, two messages M_1 and M_2 yield ciphertexts C_1 and C_2 in such a manner that $C_1 \oplus C_2 = M_1 \oplus M_2$; for CCFB+H, the same relationship exists for the first block of the message, while subsequent blocks display the same behavior only if all previous plaintext blocks are identical; finally, the same behavior is observed for the last blocks of OCB only if they have the same length, while the effect of IV repetition for all other blocks (encrypted under an ECB-like mode) is that identical plaintexts result in identical ciphertexts. In contrast, IV repetition in a CBC-like encryption mode allows attackers to gain a minimal amount

TABLE II
COMPARISON BETWEEN AEAD SCHEMES USING AN n-BIT (b-BYTE) UNDERLYING BLOCK CIPHER AND τ-BIT TAGS. THE MAC COST IS COMPUTED AS THE NUMBER OF EXTRA ENCRYPTIONS NEEDED FOR THE AUTHENTICATION PROCESS; THUS, IT DEPENDS ON THE $|H|/|M|$ RATIO FOR OCB, ON THE TYPE OF LUTS USED FOR GCM AND ON $\{|H|, |M|, \tau\}$ FOR CCFB+H.

	EAX	*OCB*	*GCM*	LETTERSOUP	*CCFB+H*
#Passes	Two	One	Two	Two	Two
Tag length (bits)	0 to n	0 to n	0 to n	0 to n	0 to $n - |IV|$
IV size (bits)	Any	n	Any (favored: $n - 32$)	n	0 to $(n - \tau)$
Requires non-repeating IV?	Yes	Yes	Yes	Yes	Yes
Encryption Mode	CTR	Tweaked ECB	CTR	LFSRC	CFB+CTR
Header Handling	CMAC	PMAC1	GHASH	MARVIN	CMAC
Input order	Free	Free	Partially	Free	Header First
Tag Verifiable Before Decryption?	Yes	No	Yes	Yes	No
MAC cost	1	≈ 0.1 to 1	≈ 0.1–0.25 to > 1	≈ 0.25–0.4	≈ 0.1 to > 1
Storage (blocks)	$O(1)$	$O(1)$	$O(1)$ to $O(2^8 b)$	$O(1)$	$O(1)$
Parallelizable?	No	Yes	Yes	Yes	No
Only E_K required?	Yes	No	Yes	Yes	Yes
Minimal Data Expansion?	Yes	Yes	Yes	Yes	Yes
Patents?	No	in USA	No	No	No

of information about the plaintexts, namely the length of their longest shared prefix, in blocks.

IV repetition can be avoided by using sufficiently large IVs, e.g., by taking their values from monotonically increasing counters having the same size as the AEAD's block cipher. However, this strategy must be adopted with care, since adding large IVs to each packet transmitted will inevitably increase energy consumption and thus reduce the sensors lifespan. There are, though, some techniques for addressing this issue. For example, instead of sending the whole IVs in every packet, both sender and receiver could keep a synchronized counter, incremented at the reception of a packet, and from which the IV value is taken. This strategy is adopted by SNEP (Secure Network Encryption Protocol) [33], in which the counter value corresponds to the whole IV and, thus, no IV is sent; it is also used by MiniSec, in which the packets include a few bits of the IV, thus facilitating resynchronization when packets are lost. It is also possible to reuse some of the packet header fields as part of the IVs, as done in TinySec and SenSec: both solutions reuse 4 bytes of the header for composing the IV. Finally, before the IV repetition occurs, re-keying mechanisms such as those proposed in [44] should be employed in order to update the nodes' keys.

G. Summary and Discussion

Table II summarizes the characteristics of the surveyed AEAD algorithms. We note that parallelizability is usually not considered an important feature in WSNs, although it may be of interest in scenarios having powerful nodes in the network, such as a multi-core base station processing many large packets resulting from data aggregation [32]).

IV. BENCHMARK METHODOLOGY

The testbed used in our benchmark is composed by Crossbow TelosB motes (see Table I), running TinyOS 2.0.2.

TinyOS [22] is a lightweight, event-driven operating system for sensor nodes that was originally developed as a research project at the University of California Berkeley. Nowadays it is maintained by an open-source community, and is many times considered the *de facto* standard for WSNs. TinyOS is based on the nesC programming language [17], a component-oriented extension of C, where the components in WSNs usually are abstractions of the mote's hardware modules (radio interface, sensors, LEDs, etc.). Its memory requirements depend on the libraries used by the specific application developed, but the base OS occupies about 400 bytes of code memory.

The compiler used is the GNU avr-gcc [5] with the `-Os` optimization option, which means "optimize for size". We note that this option provided better results in terms of both compactness and performance than its counterparts did.

A. Implementations

We implemented all the algorithms from scratch using the C language, and created a nesC wrapper module for calling the algorithms from TinyOS. We tested some variations on the coding strategy trying to identify memory- and speed-optimized constructions for constrained platforms, and also aiming to provide a fair comparison through similar optimizations, interfaces and coding style. Since faster algorithms usually lead to lower energy consumption, we focused on a more speed-oriented approach, including inline functions and pre-computing small constants whenever their impact on memory was not too significant. In this sense, key-dependent constants used in the algorithms setup phase (e.g., the constants used by the CMAC^t instances in EAX) are computed only once and then stored in RAM. The source code is available at www.larc.usp.br/~mjunior.

As underlying block cipher, we adopted CURUPIRA-2 [38] with 96-bit keys and on-the-fly key expansion [36]. This is a special-purpose 96-bit block cipher from the same family as AES, but tailored for constrained platforms. This choice has been motivated by the cipher's good performance and fairly reduced memory footprint in constrained platforms, even when both encryption and decryption processes need to be implemented [38, Section 5.2], as in the case of OCB. Nonetheless, we emphasize that the adoption of a different block cipher (e.g., AES) would not affect our comparative analysis, since this modification would impact the performance of every algorithm in a similar manner. As a final remark, we note that the static arrays used by the cipher (e.g., its S-Box) are stored in ROM in order to save RAM memory.

1) On the implementation of GCM: Even though GCM can be optimized by means of key-dependent LUTs, these are not adopted in our implementation for the following reasons. The slightest level of optimization proposed in [30] would double GCM's speed at the cost of 256 bytes of RAM[1]. However, GCM using such LUTs would use approximately twice more RAM than any other algorithm considered in this paper and would still be slower than them (see Tables III and IV), especially considering that a similar amount of memory could easily be used to accelerate the other algorithms (e.g., by pre-computing the underlying cipher's round keys, their performance gain would be $\approx 30\%$). With 4096 bytes of RAM, GCM could run four times faster than its LUT-free version [30]. Nevertheless, this more aggressive optimization would consume almost 50% of all memory available on TelosB (and 100% of the RAM in other popular platforms such as MICAz), and would still make GCM's speed only as fast as the other algorithms analyzed.

2) On the implementation of CCFB+H: Due to the implementation concerns discussed in section III-E, we developed different versions of CCFB+H for each tag length considered in this study. In order to differentiate these versions, we use CCFB+H$_\tau$ to denote CCFB+H with τ-byte tags.

B. Metrics and Methodology

The memory required by each individual algorithm in TinyOS was obtained from the compiler itself; for Contiki we used the *MSP430-size* and *MSP430-ram-usage* tools [39] for measuring, respectively, ROM and RAM usage. However, the measurement of their power consumption and execution time required a more careful approach, described in the following.

In order to obtain an accurate measurement of the energy consumption, we performed direct measurements on the motes, after turning off their radio. Instead of batteries, we use an Agilent E3631A power supply [1] configured to provide 3.0 V for the TelosB motes. An Agilent 34401A digital multimeter [2] is then employed to measure the system current flow as the different algorithms are executed, with reading rate of 60 Hz. The data measured in this manner is sent to a computer through a GPIB cable, and displayed in the LabView[2] interface, as illustrated in Figure 1. After we eliminated the influence of the current drained when the system is not executing the task we are interested in, I_{idle}, the energy consumption E was obtained as the time integration of the total current drained I multiplied by the (constant) voltage V used to power up the mote, i.e., $E = V \times \int_t (I - I_{idle})dt$.

When measuring the execution time, one would typically rely on the system's clock function, called just before a given operation starts and right after it finishes. However, during our experiments, we noticed that the intervals given in this manner by TinyOS were much lower than those observed in

[1]Key-dependent LUTs can be implemented in ROM instead of RAM only if the keys never change, which usually leads to lower security.

[2]http://www.ni.com/labview/

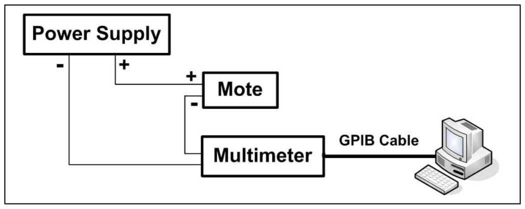

Fig. 1. Measurements setup.

TABLE III
MEMORY OCCUPATION OF AEAD ALGORITHMS ON TELOSB (IN BYTES).

Algorithm	*Code*	*RAM*	*Algorithm*	*Code*	*RAM*
CCFB+H$_4$	3856	204	LETTERSOUP	3682	218
CCFB+H$_8$	4020	208	OCB	5120	216
EAX	4528	252	CURUPIRA-2	1784	50
GCM	3862	220			

the multimeter; for this reason, we consider only the measurements obtained with the latter. This unexpected behavior of TinyOS is probably caused by its process scheduling, which does not necessarily lead to a clock reading right after a call to the clock function. On the other hand, we did not observe such discrepancy when measuring the algorithm's execution time in Contiki: the time results obtained with Energest are compatible with those observed with the multimeter. We note, though, that Energest's accuracy in terms of time did not apply to the energy: a comparison between the values measured with this tool and the multimeter shows differences that ranges from -30% to 130%; this fact, together with the unavailability of energy profiling tools on TinyOS, motivated the approach described above for measuring the algorithms' energy consumption in both operating systems.

V. EXPERIMENTAL RESULTS

This section presents and discusses our benchmark results.

A. Memory Occupation

We start our evaluation with Table III, which displays the code size and the amount of RAM used by each algorithm. This table shows that, using a similar amount of RAM, LETTERSOUP is the most compact algorithm, followed closely by GCM and CCFB+H$_4$. OCB, on the other hand, has the largest memory footprint mainly due to the need of implementing cipher's decryption routine, which does not happen with the other solutions. Finally, EAX displays the highest RAM usage due to the larger number of pre-computed constants used by the algorithm.

B. Performance and Energy Consumption

Table IV shows the execution time and energy consumed by the AEAD algorithms, considering both their initialization and the encrypted-authentication of messages having typical sizes, (i.e., up to 60 bytes of confidential data [9]). We note that the costs of authenticating messages whose length is $nb + 1$ to $nb + b$ bytes are very similar for any $n \geqslant 0$, where b stands for the cipher's block size; hence the lengths shown in the Table IV are enough for the comparison.

The analysis of this table shows the cost of initializing each algorithm is quite similar, with a small advantage for CCFB+H

TABLE IV
PERFORMANCE OF AEAD ALGORITHMS ON TELOSB FOR DIFFERENT MESSAGE LENGTHS.

		Execution Time (ms)						*Energy Consumed (μJ)*					
		[Standard Deviation $\leqslant$ 0.1]						*[Standard Deviation $\leqslant$ 0.02]*					
		Setup	*12*	*24*	*36*	*48*	*60*	*Setup*	*12*	*24*	*36*	*48*	*60*
$\|Header\| = 0$	CCFB+H_4	1.4	3.4	4.5	6.7	7.8	9.8	6.8	18.8	25.1	37.5	43.9	56.4
	CCFB+H_8	1.5	4.6	7.7	11.0	13.9	17.0	6.9	24.8	43.2	61.8	80.1	98.6
	EAX	2.4	4.5	6.7	8.8	10.9	13.0	13.0	25.4	37.9	50.3	62.4	74.4
	GCM	1.2	15.0	21.9	29.2	32.2	38.5	6.4	81.9	122.5	163.2	179.7	215.7
	LETTERSOUP	1.7	5.1	7.0	8.8	10.7	12.6	8.7	29.1	40.3	51.7	62.9	74.2
	OCB	2.0	5.0	6.6	8.3	9.8	11.3	10.1	28.0	37.6	47.4	57.2	66.8
$\|Header\| = 12$	CCFB+H_4	1.4	5.5	6.6	8.8	9.8	11.9	6.8	31.1	37.5	50.1	56.4	68.7
	CCFB+H_8	1.5	6.6	9.7	13.1	16.0	19.3	6.9	37.1	55.6	74.1	92.6	110.9
	EAX	2.4	5.7	7.8	9.9	12.1	14.2	13.0	32.1	44.7	57.2	69.3	81.5
	GCM	1.2	20.9	28.1	35.6	37.5	43.8	6.4	116.1	156.7	197.5	209.4	245.5
	LETTERSOUP	1.7	5.9	7.8	9.7	11.6	13.5	8.7	34.2	45.5	56.9	68.0	79.2
	OCB	2.0	6.6	8.3	9.8	11.3	12.9	10.1	37.6	47.2	56.8	66.6	76.3
$\|Header\| = 24$	CCFB+H_4	1.4	6.6	7.7	9.8	10.9	13.0	6.8	37.5	43.9	56.5	62.7	75.3
	CCFB+H_8	1.5	7.6	10.8	14.2	17.1	20.2	6.9	43.7	61.8	80.8	98.9	117.3
	EAX	2.4	6.8	8.9	11.0	13.2	15.3	13.0	38.5	51.0	63.5	75.6	87.8
	GCM	1.2	27.0	33.8	41.4	42.8	49.1	6.4	150.8	189.3	232.1	239.3	275.6
	LETTERSOUP	1.7	6.4	8.3	10.2	12.0	13.9	8.7	37.0	48.3	59.7	70.7	82.0
	OCB	2.0	8.1	9.7	11.3	12.8	14.4	10.1	46.6	56.3	66.1	75.8	85.5

and GCM. However, given that the algorithms initialization need to be performed only once for each key, this small difference between them should not be too significant in scenarios where the authentication keys are not changed often, such as in WSNs where each sensor communicates with a same neighbor most of times.

Turning our attention to the cost of processing messages, it is easy to see that CCFB+H is indeed a very efficient solution for 4-byte tags, as concluded in [6]. However, the same does not apply in scenarios with 8-byte tags, for which CCFB+H becomes less appealing than EAX, OCB and LETTERSOUP. The relative advantage of the latter three solutions depends on the size of the associated data $|H|$ to be authenticated. For $|H| = 0$ EAX is more efficient when $|M| = 1$ block, while OCB is better for a larger $|M|$; when $|H| \leqslant 1$ block, EAX e OCB are the most efficient solutions for $0 \leqslant |M| \leqslant 2$ and $|M| > 2$ (in blocks), respectively, while LETTERSOUP presents itself as an intermediary between these two AEADs; finally, for $|H| \geqslant 2$ blocks, LETTERSOUP displays the best performance. In contrast, GCM is up to 4 times more resource-demanding than the other AEADs analyzed.

VI. DISCUSSION AND RECOMMENDATIONS

According to our results, CCFB+H is the most attractive AEAD scheme for scenarios having short tags, usually smaller than half a block: the algorithm is uncovered by patents, takes good advantage of the underlying block cipher (which leads to a small memory footprint), and adds a reduced impact in terms of energy consumption. EAX, OCB and LETTERSOUP, on the other hand, are recommended in scenarios having stricter security requirements, i.e., demanding larger tags. The choice between them depends on the average size of the confidential and associated data to be authenticated: LETTERSOUP is very compact, and is more adequate for processing packets with reasonably large headers (bigger than two blocks), while scenarios with smaller headers would benefit from the adoption of the less compact EAX and OCB schemes for messages smaller

TABLE V
RECOMMENDATION OF AEAD ALGORITHMS DEPENDING ON THE AVERAGE DATA SIZE (IN BLOCKS) AND SECURITY LEVEL REQUIRED.

$\rightarrow$ data size	$\|H\| \leqslant 2$		$\|H\| > 2$
$\downarrow$ security	$\|M\| < 3$	$\|M\| \geqslant 3$	any $\|M\|$
$\tau < 1/2$	CCFB+H	CCFB+H	CCFB+H
$\tau \geqslant 1/2$	EAX	OCB	LETTERSOUP

and larger than three blocks, respectively. We emphasize, however, that due to the non-negligible overhead associated to the switching between transmit, receive and sleep radio states, the transmission of tiny messages (as opposed to consolidated, larger messages) in WSNs is usually discouraged [35]. For this reason, solutions adapted to larger messages are likely the better choice in a larger number of applications.

Finally, some deployment scenarios may have less common requirements that also influence in the decision for the most suitable algorithm. One case is when the algorithm's underlying block cipher is already implemented in an inflexible manner and must be treated as a black-box, which is usually the case when the sensor node is equipped with a hardware encryption module; in this scenario, computing SCTs in software is likely less efficient than performing full encryptions in hardware, at least to typical message lengths (i.e., shorter than 60 bytes [9]), leading to a lower efficiency of LETTERSOUP when compared to EAX or OCB even for large bulks of associated data. Another aspect refers to the existence of patents: applications that cannot afford the use of patented algorithms may require the adoption of EAX or LETTERSOUP in cases where OCB would deliver a better performance.

Table V summarizes this discussion: assuming that enough memory is available, it links the different AEAD schemes with their recommended application scenarios, which depend on the security level desired and on the average size of the data to be authenticated.

VII. CONCLUSIONS

The deployment of message authentication mechanisms in Wireless Sensor Networks (WSNs) is essential to prevent the insertion of fake data into the system. However, and in spite of this importance, there are to date few studies covering the suitability of existing algorithms for such applications. In this paper, aiming to cover this gap, we provide theoretical and practical comparisons between some relevant AEAD schemes in representative scenarios.

As a result of this analysis, we make several recommendations based on the characteristics of the target network and available hardware, showing that the suitability of CCFB+H, OCB, EAX and LETTERSOUP depends on the adopted size of the authentication tags (which relates to system's required security level) and also on the average size of the confidential and associated data in the packets to be authenticated.

ACKNOWLEDGMENTS

This work was supported by the Innovation Center, Ericsson Telecomunicações S.A., Brazil. It was also partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under grant 303163/2009-7 and by the São Paulo Research Foundation (FAPESP) under grant 2010/02909-8.

REFERENCES

[1] Agilent, *E363xA Series Programmable DC Power Supplies – Data Sheet*, 2004, http://www.home.agilent.com/.

[2] ——, *Agilent 34401A Multimeter – Uncompromising Performance for Benchtop and System Testing*, 2007, http://www.home.agilent.com/.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, March 2002.

[4] A. Alemdar and M. Ibnkahla, "Wireless sensor networks: Applications and challenges," in *Proc. of ISSPA'07*. IEEE, 2007, pp. 1–6.

[5] Atmel, "AVR32 GNU toolchain 2.1.6," http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4118, 2009.

[6] G. Bauer, P. Potisk, and S. Tillich, "Comparing block cipher modes of operation on MICAz sensor nodes," in *Proc. of PDP'09*. IEEE Computer Society, 2009, pp. 371–378.

[7] M. Bellare, P. Rogaway, and D. Wagner, "The EAX mode of operation: A two-pass authenticated-encryption scheme optimized for simplicity and efficiency," in *FSE'04*. Springer, 2004, pp. 389–407.

[8] K. Choi and J.-I. Song, "Investigation of feasible cryptographic algorithms for wireless sensor network," in *Proc. of ICACT'06*, vol. 2. IEEE, 2006, pp. 1379–1381.

[9] C. Cordeiro and D. Agrawal, *Ad Hoc & Sensor Networks: Theory And Applications*. River Edge, USA: World Scientific Publishing Co., 2006.

[10] Crossbow, "MICA2 datasheet," http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf, 2008.

[11] ——, "MICAz datasheet," http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf, 2008.

[12] ——, "TelosB datasheet," http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf, 2008.

[13] J. Daemen and V. Rijmen, "A new MAC construction ALRED and a specific instance ALPHA-MAC," in *FSE*, 2005, pp. 1–17.

[14] B. Doyle, S. Bell, A. Smeaton, K. McCusker, and N. O'Connor, "Security considerations and key negotiation techniques for power constrained sensor networks," *The Computer Journal*, vol. 49, pp. 443–453, 2006.

[15] B. Driessen, A. Poschmann, and C. Paar, "Comparison of innovative signature algorithms for WSNs," in *WiSec'08*. ACM, 2008, pp. 30–35.

[16] N. Fournel, M. Minier, and S. Ubéda, "Survey and benchmark of stream ciphers for wireless sensor networks," in *Proc. of WISTP'07*, ser. LNCS, vol. 4462. Springer, 2007, pp. 202–214.

[17] D. Gay, M. Welsh, P. Levis, E. Brewer, R. von Behren, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," in *Proc. of ACM PLDI'03*. ACM Press, 2003, pp. 1–11.

[18] F. Hu and N. Sharma, "Security considerations in ad hoc sensor networks," *Ad Hoc Networks*, vol. 3, no. 1, pp. 69–89, 2005.

[19] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *SenSys'2004*. ACM, 2004, pp. 162–175.

[20] T. Krovetz and P. Rogaway, "Internet draft: The OCB authenticated-encryption algorithm," http://www.cs.ucdavis.edu/~rogaway/papers/ocb-id.htm, March 2005.

[21] Y. W. Law, J. Doumen, and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 1, pp. 65–93, 2006.

[22] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, *TinyOS: An operating system for wireless sensor networks*. Springer-Verlag, 2004.

[23] T. Li, H. Wu, X. Wang, and F. Bao, "SenSec design," InfoComm Security Department, Tech. Rep., February 2005.

[24] W. Liu, R. Luo, and H. Yang, "Cryptography overhead evaluation and analysis for wireless sensor networks," *Communications and Mobile Computing, International Conference on*, vol. 3, pp. 496–501, 2009.

[25] S. Lucks, "Two-pass authenticated encryption faster than generic composition," in *Fast Software Encryption*. Springer, 2005, pp. 284–298.

[26] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: A secure sensor network communication architecture," in *Proc. of IPSN'07*. ACM Press, 2007, pp. 479–488.

[27] Nano-RK, "FireFly 2.2 datasheet," http://www.nanork.org/wiki/FireFly, 2007.

[28] NIST, *SP 800-38A – Recommendations for Block Cipher Modes of Operation, Methods and Techniques*, National Institute of Standards and Technology, December 2001.

[29] NIST, *SP 800-38B Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*, National Institute of Standards and Technology, May 2005.

[30] ——, *(SP 800-38D) Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, National Institute of Standards and Technology, November 2007.

[31] M. N.Wegman and J. L. Carter, "New hash functions and their use in authentication and set equality," *JCSS*, vol. 22, pp. 265–279, 1981.

[32] M. Patel, S. Venkatesan, and D. Weiner, "Role assignment for data aggregation in wireless sensor networks," in *Proc. of AINAW'07*, vol. 2, May 2007, pp. 390–395.

[33] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: security protocols for sensor networks," in *Proc. of MOBICOM'01*. ACM Press, 2001, pp. 189–199.

[34] P. Rogaway, "Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC," in *Advances in Cryptology - Asiacrypt'04*, ser. LNCS, vol. 3329. Springer-Verlag, 2004, pp. 16–31.

[35] A. Ruzzelli, "Wireless sensor networks: Enhancing performance through integration of mac and routing protocols," Ph.D. dissertation, University College Dublin - School of Computer Science and Informatics, 2008.

[36] M. Simplicio, "CURUPIRA-2: C version for 8-bit platforms (96-bit keys only)," http://www.larc.usp.br/~mjunior/en/downloads/index.html, 2009.

[37] M. Simplicio, P. Barbuda, P. Barreto, T. Carvalho, and C. Margi, "The Marvin message authentication code and the LetterSoup authenticated encryption scheme," *Security and Communication Networks*, vol. 2, pp. 165–180, 2009.

[38] M. Simplicio, P. Barreto, T. Carvalho, C. Margi, and M. Näslund, "The CURUPIRA-2 block cipher for constrained platforms: Specification and benchmarking," in *Proc. of Int. Workshop on Privacy in Location-Based Applications (PiLBA'08/ESORICS'08)*, vol. 397. CEUR-WS, 2008.

[39] S. Underwood, "Mspgcc," http://mspgcc.sourceforge.net/, 2009.

[40] R. Venugopalan, P. Ganesan, P. Peddabachagari, A. Dean, F. Mueller, and M. Sichitiu, "Encryption overhead in embedded systems and sensor network nodes: modeling and analysis," in *Proc. of CASES'03*. ACM, 2003, pp. 188–197.

[41] B. Yahya and J. Ben-Othman, "Towards a classification of energy aware MAC protocols for wireless sensor networks," *Wirel. Commun. Mob. Comput.*, vol. 9, no. 12, pp. 1572–1607, 2009.

[42] L. Yao, Z. Yu, T. Zhang, and F. Gao, "Dynamic window based multihop authentication for WSN," in *Proc. of the 17th ACM conference on Computer and communications security*, 2010, pp. 744–746.

[43] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi, "Hardware design experiences in ZebraNet," in *SenSys'04*. New York, NY, USA: ACM, 2004, pp. 227–238.

[44] W. Zhang, s. Zhu, and G. Cao, "Pre-distribution and local collaboration-based group re-keying for wireless sensor networks," *Ad Hoc Networks*, vol. 7, no. 6, pp. 1229–1242, 2009.