Efficient Message Authentication Protocol for WSN

MOISES SALINAS ROSALES, GINA GALLEGOS GARCIA, GONZALO DUCHEN SANCHEZ
Graduate School, ESIME Culhuacan
Instituto Politecnico Nacional
Av. Santa Ana Num. 1000, 04430, Mexico City.
MEXICO
msrosales@acm.org, gganig@gmail.com, gduchen@ieee.org, http://calmecac.esimecu.ipn.mx

Abstract: - This paper describes a solution for nodes and message authentication problems in wireless sensor networks, this solution allows effectively avoiding node-impersonation and messaging falsification among the WSN nodes. The resulting protocol address authentication at two level using identity based cryptography and message authentication codes with SHA-1, for node and message authentication respectively. An implementation of the message authentication process into a TinyOS-based node is presented; also power consumption measurements obtained are discussed. Based in experimental results we show that message authentication process is suitable in terms of power consumption.

Key-Words: - WSN, Authentication, MAC codes, Cryptography, Pairings, TinyOS.

1 Introduction

Wireless Sensor Networks (WSNs) belong to a particular case of ad-hoc networks, sharing their main characteristics as flexible routing and latency management strategies among others. However WSNs add a challenging requirement for their operation: extra-low power consumption. This requirement impacts on all design areas of WSN components: hardware, OS, application software and communication protocols.

Nowadays WSNs represent a very interesting field for application designers. The applications for WSNs have spread over the last five years, varying from simple environmental monitoring systems to critical in situ military surveillance systems or even high precision patient monitoring systems for health support [1][2][3].

A WSN is composed by a set of interconnected sensor nodes. Sensor nodes are small devices, commonly consisting on a microcontroller, a short range radio unit and one or more transducers acting as sensors. A sensor node commonly is equipped with a power source like batteries that represents the only power source available. This configuration makes power-saving a strict design principle for any WSN.

Among current applications, there is a group with high data security requirements. Such requirements are in terms of secure transmission, secure storage and secure network management, just like above listed applications [4]. In most cases, it is necessary to involve cryptographic services, such as confidentiality, data integrity and authenticity, in order to satisfy those security requirements.

However, due to a sensor node is a constrained device, any security addition to its functionality requires solving design tradeoffs between security and efficiency, even cryptographic services. About cryptographic services, each WSN application will have different requirements, however there are a set of common elements to all applications that arise a set of common basic security requirements. Those requirements are associated with the usage of the wireless channel.

Confidentiality seems to be most comprehensive service, because it can minimize the traffic capture effect during a communication among two nodes, avoiding information eavesdropping. Nowadays, most radio units for nodes are based on industrial technologies such as Bluetooth or ZigBee, this makes possible to relay confidentiality service to radio units, commonly equipped with industrial level hardware encryption, featuring AES or similar algorithms. However, in order to operate symmetric key ciphers, there is a symmetric key to be agreed. This job can be done running key agreement protocols over the air.

Authenticity is another mandatory service for a secure WSN, because message falsification and message modification as well as node impersonation are problems that need to be solved. Message authentication is required in order to prevent that malicious agents be able to inject messages to the network without be detected.

Authenticity can be established using authentication techniques, such ones can be based

on symmetric o asymmetric key cryptography. But because the limitations of symmetric techniques, related to the small storage capacity on nodes and its limited scalability to establish huge WSN, the usage of asymmetric cryptography is the starting point for this research.

This work describes an authentication protocol that allows to WSN nodes to be authenticated each other, as well to authenticate exchanged messages between them. Nodes authentication make use of Identity Based Cryptography (IBC) in order to establish new authenticated links between nodes; message authentication is obtained applying message authentication codes for each message.

This document is organized as follows: section 2 presents related works found in the literature. Section 3 describes specific authenticity requirements for WSN. In Section 4 this work approach is described as well as the involved cryptographic techniques. Section 5 presents the proposed protocol, while section 6 shows the obtained results in terms of efficiency of the protocol, particularly during message authentication stage. Finally in section 7 present the conclusions and possible further works.

2 Related Works

Besides there are several proposals for implementing authenticity services on WSN, the most are based on symmetric techniques and only a few ones involve public key cryptography indirectly. Symmetric cryptography offers low complexity in algorithms and small data pieces to manipulate and store. At this point, as above mentioned, flexibility and scalability appears to be the main drawbacks for these techniques. This approach has been explored in several proposals for implement security services using symmetric encryption, keyed and un-keyed hash functions, and pre-distribution key techniques as proposed in the work of Du [5] and Li [6] among others.

Regarding to asymmetric techniques, one of the more interesting is Du's work. He established that it is possible to use public key algorithms such as digital signatures; however public key authentication has to be solved jet. In [7] Du proposed to use a Merkle Tree as a mechanism to construct an authentication path for public keys between all nodes in the WSN using hash values. Actually Du also proposes to divide a global WSN tree into regional trees not so taller in order to speed up the lookup into the tree. The main drawback of this proposal is that each node has to exchange almost the authentication path with any other node in order to be authenticated; this makes costly the execution of the protocol, considering the associates channel usage cost.

Other related works are the classification for PKC alternative proposed by Gaubatz [8], and the TinyOS implementation of RSA proposed by Watro [9]. Uhsadel proposed a scalar multiplier for 8 bits platforms in [10] and Gura proposed an efficient Elliptic Curve Cryptography (ECC) implementation in [11]. Finally Piotrowski provided in [12] a power consumption profile for PKC primitives on WSN.

Recently However Oliveira et al. proposed in [13] a key distribution method that allows to two nodes to agree a common key. Oliveira's approach explodes the use of IBC to accomplish the key exchange.

3 Authenticity in WSN

Wireless Sensor Networks are composed by hundreds of even thousands of sensor nodes. In real applications, all nodes can be managed by a single entity or by a set of such ones. Prior to deployment all nodes are considered to be able to an *inhouse* setup, setting network parameters, as well as security parameters. In this way, all nodes prepared to deployment at the same time can be considered to belonging the same brood.

However, as the time pass over, the WSN will be reflect some changes as node destruction, node's battery exhausting, new alert and sensing patterns, and finally new sensing granularity requirements. All such cases would require to increase the number of nodes on the network, even for replace defeated nodes or to increase network coverage. Adding new nodes to the WSN involve preparing a new brood of nodes and executing their deployment.

Once the new nodes have been deployed, those ones have to be connected to the already established WSN. The procedure to establish such connection requires so assure that only *authentic nodes* will be allowed, avoiding any other entity to interfere on WSN operation. Then, after the fresh nodes are connected, all messages, coming-from and received-at, have to be assured to be *authentic* in order to prevent attacks by message injection o message alteration.

Upon this point, the following security requirements for WSN can be established:

1. Once deployed, each node has to be authenticated against the WSN before any other action is allowed.
2. Once authenticated, each node has to be able to verify any received message's authenticity, that is, that effectively was sent by the claimed

source. Any message not satisfying this condition has to be discarded and sender's information added can be used to form a black list.

In the next section an approach to satisfy such requirements and the techniques involved are discussed.

4 Our approach

In order to address security requirements for authenticity in WSN, the proposed protocol explodes the combination of asymmetric and symmetric key techniques in order to address each authentication process and minimizing the amount of power consumption.

Node authenticity is established using a challenge-response protocol, where two nodes are supposed to share a secret if and only if the belong to the WSN, so they can be considered as authentic. In this scenario two nodes are considered to interact, one node already belongs to the WSN and is labeled as *authenticator*, the other node is requesting to be connected to the networks and has *supplicant* label.

The challenge-response protocol is constructed using IBC primitives, specifically Identity Based Encryption (IBE) proposed by Boneh and Franklin in [14]. The main reason to use an identity based protocol is that, as proposed by Shamir, the verification of the public key is avoided by substituting such value by public identity-related information, such as network address or serial number for sensor nodes. During the challenge-response protocol, a secret value is agreed, and it is proposed to be used as a symmetric key for message authentication purposes.

Once two nodes have been authenticated each other, they are capable to authenticate any message exchanged between them by applying a Message Authentication Code (MAC) over the message and using the agreed secret value as symmetric key.

In the following subsections, a brief introduction to those main techniques is presented.

4.1 Identity Based Cryptography

Due PKC approach require key authentication mechanisms, such as digital certificates, it would be convenient to avoid those ones, by exploring alternative mechanisms.

IBC is an idea originally proposed by Shamir in 1984 [15], but first implemented by Franklin and Boneh. IBC allows users to derive public keys from identity or other simple strings, demanding private keys have to be *cleared* by a trusted authority, and enabling to users to avoid public key authenticity verification. Most common IBC implementations are based on bilinear functions or pairings. The most important characteristic of this construction is the bilinearity property, expressed as $\hat{e}([a]P, Q) = \hat{e}(P, [a]Q) = \hat{e}(P, Q)^a$, where $\hat{e}$ is a bilinear function.

Bilinear Pairings are rational functions defined over a field of functions [16]. In cryptography the most common used arrangement is set of rational functions defined over the field of functions over an elliptic curve E defined over a finite field [17]. Commonly used pairing functions are Wëil pairing, Tate pairing, and recently Ate and Eta-t pairings [18].

Besides pairings evaluation demands high amounts of processing quotes, recently some improvements have been developed achieving considerable savings on processing, enabling in such way the use of pairings usage in resource constrained environments like WSN [19].

4.2 Message Authentication Codes

Message Authentication Codes are cryptographic constructions that are designed to identify manipulation and falsification on electronic messages. Although there are MAC constructions defined over symmetric cipher as modes of operation, the most known MAC codes are constructed using one-way hash cryptographic functions, just like SHA-1 [20] or MD5. Examples of former are HMAC, NMAC and UMAC.

One way hash functions are also known as Modification Detection Codes, but are commonly called hash functions providing a very efficient integrity verification method.

A message authentication code uses a secret key k which is known for two entities that communicate a message of arbitrary length m, the code gives a MAC output value MAC = H_k(m). The MAC value generated by the issuer protects the integrity and its authenticity in a message, enabling the receiving entity to recalculated MAC value through the secret key, to verify any change in content message, as well as the source of the sender indicated.

The construction of HMAC was proposed by proposed by Bellare et al. in [21], it uses the notation H for the iterated hash function initialized with its value fixed usual IV. The HMAC function works with m entries of arbitrary length and uses a single secret key k of length l which is included as part of the message in a block b, the HMAC of the method can be defined as:

$$HMAC_k(m) = H((k \oplus opad)||H((k \oplus ipad)||m))$$

Where k is the padded key with zeros to a full block length b iterated the hash function, | | represents the concatenation of the information, $\oplus$ is the logical operation XOR, The *ipad* string is consisting of the 0x36 hexadecimal number (00110110 in binary) because each hexadecimal character is formed by 4 bits, this value is repeated as many times as necessary to fill a block of b bits, in the case of MD5 and SHA-1, the blocks are 512 bits, then the hexadecimal value must be repeated 64 times, another string *opad* is defined similarly using the hexadecimal number 0x5C (01011100 in binary).

Additional processes for this function are not made in the external structure of the hash function, just calling the function without any modification. The results of the second hash function are directly dependent on the outcome of the first, thus the dependence of the processes and security of the algorithm.

HMAC is also defined as an IETF keyed hash function under the RFC 2104 and is considered to be secure against attacks under the assumption of that the underlying hash functions is secure.

5 Authentication Protocol

The proposed authentication protocol follows the previously outlined approach. In the subsection 5.1 some general parameters are explained. Subsection 5.2 provides a detailed description of the protocol itself, and in subsection 5.3 some consideration for its evaluation are discussed.

5.1 Protocol Parameters

The authentication protocol uses the Identity Based Encryption scheme assuming that the network manager would be able to act as Trusted Authority (TA).

The following considerations have been adopted as part of the environment under the protocol will operate:

- There exists a binary finite field *K = GF(2m).*
- There exists an elliptic curve

$$E(K): y^2 + ay = x^3 + bx^2 + cx + d,$$

where E is defined over a finite field with $a, b, c, d \in K$.

- Each point in the elliptic curve is denoted with capital letters, like *P*.
- $[a]P$ denotes the scalar multiplication of point the *P* by *a* times.
- There exists an additive group consisting of points over the elliptic curve, denoted by G_1.
- There exists a field extension $GF(2m)^k$ of order *k* constructed over K
- There exists a bilinear pairing function $e: G_1 \times G_1 \rightarrow G_2$ that is defined over a group of points on the elliptic curve: $\hat{e}(P, P) = \langle g \rangle$ *with* $g \in GF(2m)^k$, $k \in \mathbb{Z}^+$ is a security parameter according to the embedding degree of the elliptic curve.

5.2 Protocol Description

The protocol consists on the stages described in the following paragraphs.

Setup: This stage is to be executed by the WSN manager acting as a TA, using its own facilities for processing in order to minimize the nodes power consumption. During this protocol stage the TA proceeds as follows:

1. Generates two groups G_1 and G_2 with prime order q satisfying the bilinear pairing $\hat{e}: G_1 \times G_1 \rightarrow G_2$.
2. Chooses a random generator point $\langle P \rangle \in G_1$.
3. Selects randomly TA's master key, $s \in \mathbb{Z}_q^*$ and set TA's public key $P_{pub} = [s]P$.
4. Selects a suitable space for identity labels $\mathcal{L}: \{0,1\}^m$ for some value *m* in accordance to maximal network size expected.
5. Chooses three cryptographic hash functions $H_1: \{0,1\}^* \rightarrow G_1$, $H_2: G_2 \rightarrow \{0,1\}^n$ for some n , and a keyed hash function $H_3: \{0,1\}^n \times \{0,1\}^m \rightarrow \{0,1\}^h$ for a small *h* value, i.e. 160 or 128.

Key extraction: During this stage all nodes will be assigned with an identity label $ID \in \mathcal{L}$ and their corresponding cryptographic keys. In order to accomplish that, the TA runs the following procedure for each node:

1. Computes the node public key $Q_{ID} = H_{1(ID)} \in \mathbb{G}_1^*$.
2. Obtains the node private key for *ID* by evaluating $d_{ID} = [s]Q_{ID}$.
3. Sets an empty link-key list denoted by $list_{ID}$ with capacity to store $\{0,1\}^n \times \{0,1\}^m$ pairs.
4. Loads the node with values $(ID, d_{ID}, Q_{ID}, list_{ID})$.

Once *Key extraction* stage had completed, all nodes are ready to be powered on and be deployed into field to start the network operation.

During its operation into the network, each node will sense the channel for detect other nodes in order to establish a neighbourhood.

Node discovery: Once a non-WSN node detects a reachable link to an unknown neighbour, it will adopt a supplicant role, denoted here by ID_S.

The counterpart will be managed within an authenticator role ID_A under the consideration that it would have to a valid link the whole network.

The nodes ID_S and ID_A will try to establish an authenticated link by performing the following message exchange:

1. Node ID_S generates a message $s = hello(ID_S, TS)$ containing the origin node identifier as well as a timestamp mark.
2. Node ID_S sends *s* to node ID_A.
3. Once node ID_A receives *s*, it will validate the *TS*. If *TS* is within a pre-specified *t* threshold, it will accept and process the request. The *TS* timestamp will prevent abuse and DoS attacks from a hostile node.

Challenge generation: Once node ID_A has received the request, it will generate a challenge message to verify that the node ID_S belongs to the WSN. The verification will be successful only in case that the supplicant holds a *TA-cleared* private key. To generate and send the challenge, node ID_A will proceed as follows:

1. Computes $Q_{ID_S} = H_{1(ID_S)}$
2. Selects randomly a value $r \in \mathbb{Z}_q^*$ and computes $v' = H_2\left(\left(\hat{e}\left(Q_{ID_S}, P_{pub}\right)\right)^r\right)$
3. Selects $k \overset{\$}{\leftarrow} \{0,1\}^l$
4. Computes $v = k \oplus v'$ and $u = [r]P$
5. Sends the pair $Ch = (u, v, ID_A)$ as challenge to ID_S.

Response generation: With the challenge $Ch = (u, v, ID_A)$ received, the supplicant node ID_S will proceed as follows to generate a valid response:

1. Computes $k' = v \oplus H_2\left(\hat{e}\left(d_{ID_S}, u\right)\right)$
2. Sends $\tau = H_3\left(k', ID_S\right)$ as response ticket to the node ID_A.
3. Saves k as a link key to be used to authenticate messages with the node ID_A, adding the pair (k, ID_A) to its list $list_{ID_S}$

Due to bilinear properties of pairing, it is required that the node ID_S holds a valid d_{ID_S} corresponding to the value of its public key.

Challenge verification: Once τ is received, the node ID_A will verify it validating the following condition:

1. If $\tau = H_3(k, ID_S)$ then:
   a. Add the pair (k, ID_S) to its list $list_{ID_A}$
   b. Then k is established as link-key for the link $ID_A - ID_S$.
2. Elsewhere do nothing or collect information from the supplicant node for generate a black list.

Up to this point, the two nodes have been authenticated itself and are ready to exchange messages over an authenticated link. From the stage of Node discovery until the stage of challenge verification, the protocol will be executed by a node once with each neighbour node, the rest of the protocol is related to message authentication, which will be the most recurrent operation among the network operation.

Message authentication: From this point any message exchanged between two nodes ID_i and ID_j can be authenticated on a hash basis:

1. When message m_k is sent by node ID_i to node ID_j , it is accompanied with its MAC code $H_3\left(k_{i-j}, m\right)$. So the data exchanged are $(H_3\left(k_{i-j}, m_k\right), m_k)$.
2. Once ID_j receives m_k, it validates its authenticity verifying the MAC code. If both MAC codes coincide then ID_j accept the message, elsewhere it will reject the message.

The only condition is that a value k_{i-j} exists in both $list_{ID_i}$ and $list_{ID_j}$. Elsewhere both nodes have to run this protocol from Node Discovery stage in order to arrange a link-key.

5.3 Considerations for viability evaluation

The viability of the above protocol involves several aspects to consider. Firstly the efficiency is determined by the operational cost for the protocol and it can be established according to the amount of memory, processing and transmission needs.

From the observation of the protocol, it is possible to distinguish between the two processes that have to be executed by the WSN nodes. The first one, node authentication, has be executed only one time per each pair of nodes in the neighborhood. The second one, message authentication, will be required to be executed for each message transmitting and receiving. This situation has to be observed carefully in order to accurately determine the cost of the protocol and then its efficiency. Definitively, a MAC code routine will be executed more frequently that a pairing routine, no matter that the cost of last would be several times the cost of the first one.

Other aspects to consider about viability concerns to security, flexibility, scalability, and

interoperability among others, but in the mean time all those were out of the scope of this work and have to be considered for further analysis.

In the next section we present results about efficiency of the protocol in terms of estimations and experimental measurements.

6 Results on Protocol Efficiency

The protocol for authentication has to be evaluated in terms of its efficiency in order to establish how its use would impact on node's power consumption.

This evaluation on the protocol is focused exclusively into the execution of the MAC routine and its power consumption on the node. This is because its usage frequency is greater that the pairing routing routine. However, a further evaluation of the pairing routine would raise data for comparison and to obtain a complete perception of the related costs.

The first step into the evaluation has been to establish some parameters like the MAC code and hash function to be used, as well the target platform where the protocol will be executed.

For practical reasons, HMAC code and SHA-1 hash function have been selected. The target platform is a TelosB sensor node manufactured by CrossBow Technology and featured with a 16-bit MSP430 microcontroller, a ZigBee radio unit and a set of transducers for light, humidity and temperature measurement.

The rest of this section describes the operations involved during protocol execution, estimations of cost and measurements obtained doing experimentations with the mentioned sensor node.

6.1 Involved operations

In order to determine how processing and storage requirements are distributed among the protocol, the list of the cryptographic primitives executed by each node was collected and showed in Table 1, where the following notation is used:

- R stands for random number generation
- H stands for hash or MAC evaluation
- P stands for pairing evaluation
- E stands for modular exponentiation over the field extension.
- M stands for scalar multiplication
- X stands for bitwise xor computation.

As showed in Table 1, the authenticator node has processing needs for one random number generation, two MAC code evaluations, one pairing evaluation, one exponentiation over the field extension and one scalar multiplication over a curve point. In counterpart the supplicant node requires only three MAC code evaluations, one pairing evaluation and one XOR that can be ignored.

As previously discussed, the protocol involves two functions: node authentication and message authentication. A node pair authentication involves generation and response of challenge, requiring two evaluations of pairings that are known to be quite expensive due to operate on the field extension. The other function, message authentication, requires only the evaluation of two MAC codes; those are commonly considered a cheap primitive.

Stage	Cryptographic operations per node	
	Authenticator	Supplicant
Ch. Gen.	1 R, 1 H, 1 P, 1 E, 1 M, 1X	
Ch. Resp.		2 H, 1 P, 1 X
Resp. Verifying	1 H	
Message Auth	1 H	1 H

Table 1 Distribution of cryptographic operations among the protocol.

Those both functions have to be evaluated on the amount of resources consumed, but as previously established message authentication deserves special attention due its frequently execution that is expected during intensive message exchange. In the following section we proceeded to establish the processing and storage needs associated to the evaluation of MAC codes into two nodes ID_i and ID_j.

6.2 Cost estimation for HMAC evaluation

In this paper, for estimation purposes, the HMAC has been selected. It is based on a cryptographic hash function.

From simple observation of the construction defined in subsection 4.2, it is possible to establish that the evaluation of HMAC is composed by two evaluations of XORs and two evaluations of the underlying hash function. As mentioned above, SHA-1 is assumed to be used as the hash function. It produces a hash value of 160 bits, short enough for a WSN environment. In this way is possible to obtain a closer estimation of the message authentication process cost.

Another interesting observation is that HMAC is defined upon the assumption that at least four hash evaluations has to be done, or a hash pre-calculating alternative can be followed in order to reduce to two

hash routine invocations during message processing. Using this alternative requires the node key to be pre-combined with the *ipad* and *opad* padding strings, and then storing the resulted values carefully as the key itself.

This alternative assumes some pre-processing to be done during the challenge verification and key agreement stage. Following this alternative a slight modification to the link-key list is required in order to use a $\{0,1\}^n \times \{0,1\}^n \times \{0,1\}^m$ triple instead the pair denoted in the step 4 of key extraction stage. The two first values of the triple now should correspond to the values k XOR *ipad* and k XOR *opad* respectively, those ones that have to be obtained once the link-key has been agreed.

Due the HMAC operations are defined in terms of SHA-1 function, it is required to obtain cost estimation for the hash evaluation.

The main structure of SHA-1 is a block of logic and arithmetic modulo 2^{32} operations on 32-bits variables. This main block is invoked 80 times from the main loop of the function while the 512-bit expanded message is processed. The expanded message consists on the original 16 32-bit words plus 64 words calculated as expansion of the original message.

Considering that the selected platform is able to process 16-bit operations, we can estimate the cost of evaluating SHA-1 as follows.

Basically bit-logical and bit-shifting operations do not require extra computations more than the double of operations defined by SHA-1, remain that it is necessary to map 32-bit values to 16-bit variables.

Considering arithmetical additions modulo 2^{32} executed during the iterated block, it requires performing the addition for the low end 16-bit word, and then doing it for the high end reusing the carrier resulting from the low end addition. Then the result of the simple addition has to be reduced modulo 2^{32}. This reduction requires an additional subtraction, in case that the carrier bit becomes high. So one modular addition involves (in average) 1.5 32-bit simple additions, which are translated to 3 16-bit additions each one. No extra arithmetic operations are required by SHA-1. However other operations as bit shifting and logical evaluations are also required for the hash evaluation.

After analysis of involved operations, an estimation resulted from mapping SHA-1 hash function to the available operations into a MSP430 MCU are showed in the Table 2 and the following notation is used to represent assembler mnemonics:

- ROTL: left word rotation,
- ROTL: right word rotation,
- AND: bitwise logical and
- XOR: bitwise logical xor
- NOT: bitwise logical negation
- ADD: arithmetic addition
- ASG: assignation operation
- CMP: comparison
- INC: arithmetic incremental

All operations denote 16-bits operations.

MSP430 Instr.	*Ops.*
ROTL	1254
ROTL	480
AND	334
XOR	376
NOT	40
ADD	1041
ASG	715
CMP	96
INC	96

Table 2 Operations for SHA-1 evaluation on MSP430.

From the estimations for the SHA-1 evaluation cost, it is possible to estimate the evaluation cost for HMAC code. Applying a factor of 2 (remembering that HMAC involves at least 2 calls to SHA-1), it can be observed that each MAC code will consume approximately 8,864 instructions on the MCU, excluding data loading and storing. The 8,864 operations must be affected by a tentative load and store factor of 4, this would lead the estimation to a closer result once the loading and storing operations are included. From this data and considering an average of 8 cycles per instruction it would take 283,648 cycles that would take 70.912 milliseconds per MAC, according with a simple mapping from operations to assembler instruction for the target platform.

In the next two sections, the implementation of HMAC is showed and the experimental measurements using the node are reported and discussed.

6.3 HMAC Implementation in TinyOS.

For experimental purposes, the component of the protocol related with MAC codes evaluation for messages authentication has been implemented in software as a TinyOS [22] component and then compiled and transferred to the node.

The implementation of HMAC consists on a component of TinyOS that is defined by two calls to another component that evaluates SHA-1 hash function.

As a required part within MAC methods, the SHA-1 algorithm was implemented for iterating the hash function *F(x)* used to calculate the MAC value. SHA-1 creates an image of 160 bits on a message of up to 2^{64} bits, however in this work a message with size minor than 448 bits has been defined considering that messages exchanged between nodes will be so small.

The process for calculating the hash value using the SHA-1 algorithm uses 32 bit data, while TelosB can process data of 16 bits; this requires an adaptation to the code in nesC that is carried out by the compiler.

The developed code for SHA-1 requires the implementation of two components:

- ShaToolsC: Containing the tools that the algorithm needs to calculate the Hash value. Fig. 1 shows a module containing the code ShaToolsC about commands defined in the interface ShaTools.

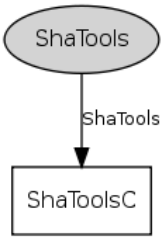

Fig.1 ShaTools component

- Sha1C: contains the procedure for calculating the hash value. This component makes use of the interface ShaTools, which is an access point that contains the commands ShaToolsC. Figure 2 shows the module Sha1C, whose commands are defined in the interface Comp_sha1, which in turn uses the command interface using ShaTools to link ShaToolsAppC component.

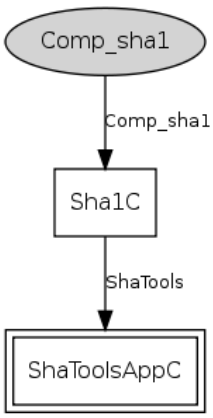

Fig. 2 Sha-1 component

Once the SHA-1 function has been implemented, the implementation of the HMAC is possible. The usage of the call *boot()* into the code is to associate the boot process of the node with the call the HMAC routine during the initializing of the node. This call makes a double call to the iterated hash function *F(x)*, assuming the secret key needed to be included in a block of 512 bits is loaded in the initialization vector, instead to be part of the message. The general structure of HMAC is showed in Figure 3.

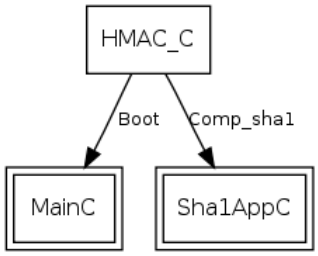

Fig. 3 HMAC component

6.4 Experimental Results

Once the HMAC component has been implemented, and compiled into TinyOS, the resulting system image can be transferred into the node in order to observe the real behavior of the component and determine operational costs.

During coding, the HMAC component has been loaded with a fixed message to be used for mac value calculation; the message used was the corresponding to the string "abc". No matter the message length is extremely short the SHA-1 function completes it with a conventional padding that normalizes the message to a regular up to 448-bit message.

A testbed was prepared for measuring sensor behavior during the experimentation, collecting processing time for the HMAC evaluation, current consumer drawn and finally power consumption. The testbed consist on the circuit showed in figure 4 and the following specifications:

- Power source for 2.8 volts.
- Digital oscilloscope
- Digital multimeter.
- 10 ohms resistor

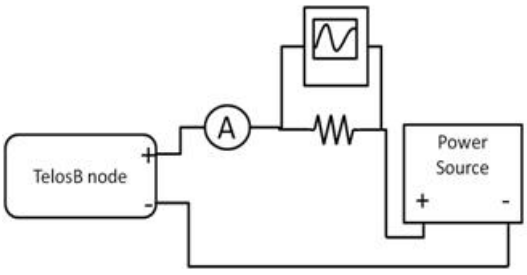

Fig. 4 Testbed used for measure power consumption

In order to measure the current drawn during the HMAC evaluation, the component was modified to run a infinite loop calling the calculus routine. During this experiment, the multimeter sensed a current on the circuit of 1.624 mA. differing substantially from the current drawn of 10.3 µA. sensed during an idle period.

Another experiment conduced was to measure the voltage drop at the resistor during the HMAC evaluation. For this test, the component was modified to call the calculus routine by intervals of 10 seconds, time enough for sampling the voltage drop using the oscilloscope. The output sampled in this experiment is showed in the Figure 5.

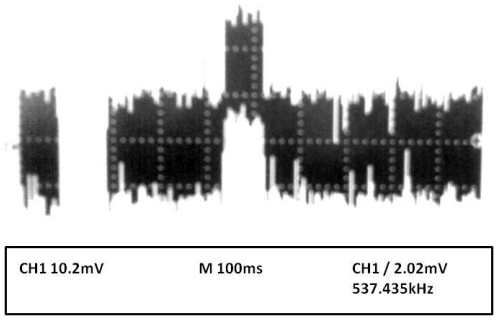

Fig. 5 Voltage drop at resistor.

From observation from the equipment, the voltage drop was about 30 mV at the resistor, corresponding to the increase of current in the circuit consumed by the node during HMAC evaluation.

Once that voltage and current measurement have been established, finally it is possible to obtain the power consumption for HMAC corresponding to the message authentication process. The results are showed in Table 3.

	Idle period	HMAC evaluation
Current drawn	10.3 µA.	1.624 mA.
Source Voltage	2.8 V	2.8 V.
Voltage drop	10.6 mV.	40.7 mV.
Power consumed	28.84 µW.	4.547 mW.

An additional measurement collected was an approximation of the time consumed for HMAC evaluation, it can be observed in Figure 6, where the horizontal resolution is setup to 50 milliseconds per division, and the approximate duration of duty cycle is about 80 milliseconds. This measurement is close to the estimation reported in section 6.2.

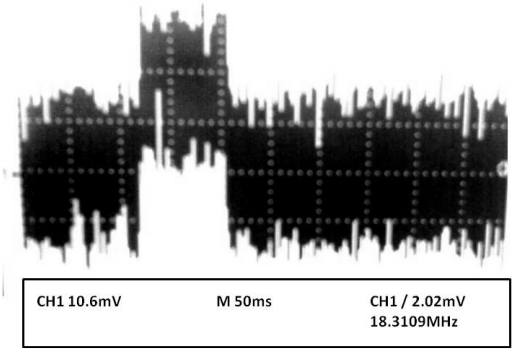

Fig. 6 Duty cycle duration.

According with Piotrowski [12], the power consumption during radio transmitting at -25dBm is 25.5 mW while receiving is 56.4 mW. Using this information as reference for comparing the measurements obtained in our experiments, the message authentication cost represents an overhead of 8.06% for the receiver node, and 17.83% for the transmitter node. This means that with this HMAC implementation, the overhead cost is about 12.5% in average. This makes clear the need to work in improvements for reduces the component power consumption, but the current results are promising.

7 Conclusions and further work

Traditionally the MAC codes evaluation are considered to have a small impact over power consumption on a sensor node, and it that can be traduced to an efficient operation for the proposed protocol when long-term links are operated between nodes. However, the studies of estimation for node authentication, as well as the experimental measurements reported a small over cost for the nodes that suggest their impact is about 10% but not more.

Estimations obtained in section 6.2 correspond closely to the measurement obtained during experimentation; this represent that the followed technique analysis is valid and it would be desirable to apply it in order to corroborate is effectiveness.

Results presented describe only the message authentication stage of the protocol that effectively corresponds to the more frequent invoked operation during the network operational life, however the evaluation of the node authentication relates stages is required in order to obtain a full caption of the impact of the protocol to a WSN operation.

Cost reported were obtained in a static fashion view, where only a *one node* scenario was considered, reflecting only local effects. Another interesting issue to be addressed is to obtain estimation methods about the dynamic behavior of the network.

Finally, it is important to recall that the results reported have been obtained using specific software and hardware components, so there would be minor variations among other platforms that have to be estimated in order to use such results as a general case.

References:

[1] A. Perrig, J. Stankovic, and F. Wagner, Security in Wireless Sensor Networks, *Communications on The ACM*, Vol. 47 No. 6, pp. 53-57, 2004

[2] Z. Bojkovic, B. Bakmaz, A Survey On Wireless Sensor Networks Deployment, *WSEAS Transactions on Communications*, Vol. 7, No. 12, pp. 1172-1181, 2008.

[3] T. Shih., W. Chang, Hierarchical Localization Strategy For Wireless Sensor Networks, *WSEAS Transactions on Computers,* Vol. 7, No. 8, pp. 1260-1269, 2008.

[4] P. Huang, The Investigation Of The Elliptic CurveCryptology Applies To The New Generation Protocol, *Transactions on Computers,* Vol. 7, No. 6, pp. 694-703, 2008

[5] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, A Pairwise Key Predistribution Scheme for Wireless Sensor Networks, *ACM Transactions on Information and Systems Security*, Vol. 8, No. 2, 2005, pp. 228-258.

[6] G. Li, H. Ling, T. Znati, and W. Wu, A robust on-Demand Path-Key Establishment Framework via Random Key Predistribution for Wireless Sesor Networks, *EURASIP Journal on wireless Communications and Networking*, Vol. 2006, ArtID 91304, pp. 1-10, 2006.

[7] W. Du, R. Wang, and P. Ning, An efficient scheme for authenticating public keys in sensor networks. *In Proc. of the 6th ACM international Symposium on Mobile Ad Hoc Networking and Computing*, pp. 58-67, 2005.

[8] G. Gaubatz, J.P. Kaps, and B. Sunar, Public Key Cryptography in Sensor Networks—Revisited, *in Lecture Notes on Computer Science,* Vol. 3313, pp. 2-18, 2005.

[9] R. Watro, D. Kong, S. Cuti,, C. Gardiner, C. Lynn, and P. Kruus, TinyPK: securing sensor networks with public key technology, *In Proc. 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks,* pp 59-64, 2004.

[10] L. Uhsadel, A. Poschmann, and C. Paar, Enabling Full-Size Public-Key algorithms on 8-bit Sensor Nodes, *in Proceedings of 4th European Workshop ESAS*, LNCS 4572, pp. 73-86, 2007.

[11] N. Gura, A. Patel, A. Wander, H. Eberle, and S. Chang-Shantz, *Comparing elliptic curve cryptography and RSA on 8-bit CPUs*, *in Proceedings of CHES'2004: Workshop on Cryptographic Hardware and Embedded Systems*, pp. 925-943, 2004.

[12] S. Peter, P. Langendorfer, and K. Piotrowski, Public key cryptography empowered smart dust is affordable, *International Journal of Sensor Networks*, Vol. 4, No. 1/2, pp. 130-143, 2008.

[13] L.B. Oliveira, M. Scott, J. Lopez, R. Dahab, TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, *In Proc. of International Conference Networked Sensing Systems INSS2008*, pp. 173-180, 2008.

[14] D. Boneh and M. Franklin, Identity Based Encryption from the Weil Pairing, *SIAM J. of Computing*, Vol. 32, No. 3, pp. 586-615, 2003.

[15] A. Shamir, Identity-Based Cryptosystems and Signature Schemes, *in Proceedings of CRYPTO'84 Conference*, pp. 47-53, 1984.

[16] L.C. Washington, Elliptic Curves: Number Theory and Cryptography, Chapman & Hall, 2003.

[17] N. Bardis, N. Doukas, K. Ntaikos, A New Approach Of Secret Key Management Lifecycle For Military Applications, *Transactions on Computers,* Vol. 7, No. 12, pp. 2011-2021, 2008.

[18] F. Hess, N. P. Smart, F. Vercauteren, The Eta Pairing Revisited, *IEEE Transactions on Information Theory*, Vol. 52 No. 10, pp. 4595-4602, 2006.

[19] P. S. Barreto, S. D. Galbraith, C. Héigeartaigh, and M. Scott, Efficient pairing computation on supersingular Abelian varieties, *Designs, Codes and Cryptography*, Vol. 42, No. 3, pp. 239-271, 2007.

[20] National Institute of Standards and Technology, *FIPS 180-2*, 2002.

[21] H. Krawczyk, M. Bellare, and R. Canetti, HMAC: Keyed-Hashing for Message Authentication, *Internet RFC 2104*, 1997.

[22] TinyOS Website. http://www.tinyos.net