Message Authentication for Wireless Sensor Networks

MOISES SALINAS R., GINA GALLEGOS G., GONZALO DUCHEN S.
Graduate School, ESIME Culhuacan
Instituto Politecnico Nacional
Av. Sta. Ana No. 1000, 04430, D.F.
MEXICO
msrosales@acm.org, http://calmecac.esimecu.ipn.mx

Abstract: - This article describes an authentication protocol based on pairings for a wireless sensor network considering the execution cost. The proposed protocol provides a node authentication process based on identity, as well as a message authentication process using keyed hash functions. The evaluation of the protocol's execution cost is focused on message authentication operations in order to provide a notion of the network power consumption intensive message exchange between authenticated nodes.

Key-Words: - WSN, Authentication, Identity based encryption, Pairings, Message authentication, Cryptography

1 Introduction

Wireless Sensor Networks (WSN) represent a very interesting field for application designers, the number of applications for WSN have spread over the last two years, varying from simple environmental monitoring systems to critical in situ military surveillance systems or high precision patient monitoring systems for medical support [1].

A WSN is integrated by a set of interconnected sensor nodes. Each node is a small device 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 for the node. This configuration makes power-saving a strict design principle for any WSN component.

There is a group of applications with high data security requirements in terms of secure transmission, secure storage and secure network managing. Examples of such kind of applications are the ones mentioned above. In order to provide such security requirements it is necessary to involve cryptographic services, such as encryption, data integrity and authentication as well as key exchange.

However, a sensor node is a constrained device, any security addition to it functionality requires solving a design tradeoff between security and efficiency. In one hand symmetric cryptography offers low complexity in algorithms and small data pieces to manipulate and store. At this point, 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 propose the work of Du [2] and Li [3] among others.

In the other hand, asymmetric or public key cryptography (PKC) offers high flexibility and scalability through key agreement protocols and authentication mechanisms, but it represents also high demanding processing due to mathematical operations involved. Although PKC has been traditionally used successfully on desktop and high edge computing environments where resources do not represent a strict limitation, several constrains are imposed for its application in WSN; i.e. elliptic curve cryptography (ECC) is preferred [4] over RSA or ElGamal algorithms, and small size keys are preferred even decreasing security level offered by larger keys [5]. Also PKC approaches require key authentication mechanisms, such as digital certificates, and it would be convenient to avoid those ones, by exploring alternative mechanisms.

Identity based cryptography (IBC) is an idea originally proposed by Shamir in 1984 [6], but first implemented by Boneh [7]. IBC allows users to derive public keys from identity or other simple strings, demanding private keys have to be cleared by a trusted authority, enabling to users to avoid public key authenticity verification. Most common IBC implementations are based on bilinear functions or pairings.

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 [8].
It is our research interest to evaluate how suitable are pairing based applications over sensor networks in order to explore its characteristics and possibly combining those with other symmetric key algorithms that have a well known efficiency. In this work an authentication protocol for WSN is proposed combining IBC techniques for nodes authentication and key agreement, with keyed hashing functions for message authentication. Then cost estimation has been performed in order to determine how the protocol would impact to sensor operational life and the whole sensor network operation.

The rest of the document is organized as follows. Section 2 provides a general description of the authentication protocol and the considerations observed in order to evaluate its viability into a WSN. In section 3 a cost evaluation approach for our protocol is introduced and resulting cost estimations obtained are discussed. Finally section 4 presents our conclusions and draft further work for this investigation.

2 Protocol Description
2.1 General assumptions
The authentication protocol uses the Identity bases encryption scheme [7] 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 is an elliptic curve $E(K); y^2 + ay = x^3 + bx^2 + cx + d$, where $E$ defined over a finite field $K=GF(2m)$ and $a, b, c, d \in K$.
- Each point in the elliptic curve is denoted with capital letters, like $P$, and the scalar multiplication of such points is denoted by $[a]P$. Finally, the group of points of the elliptic curve is denoted by $G_1$.
- There is a bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$ defined over a group of points on the elliptic curve: $e(P, P) = (g)$ with $g \in GF(2m)^k$.

2.2 Protocol execution
The protocol consists on the seven stages described in the following paragraphs.

Setup: This stage is proposed to be executed by the WSN manager acting as TA using its own facilities in order to minimize the nodes power consumption during this stage. Given a security parameter $k \in \mathbb{Z}^+$, the TA will proceed as follows:

1. Generates two groups $G_1, G_2$ with prime order $q$ satisfying the bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$.
2. Chooses a random generator on $(P) \in G_1$.
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 $L: \{0,1\}^m$ for some $m$ in accordance to maximal network size expected.
5. Chooses three cryptographic hash functions $H_1: \{0,1\}^n \rightarrow G_1^*$, $H_2: G_2 \rightarrow \{0,1\}^n$ 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 $ID \in L$ and their corresponding cryptographic keys. Then the TA runs the following procedure for each node:

1. Computes the node public key $Q_{ID} = H_1(ID) \in G_1^*$.
2. Obtains the node private key for $ID$ as $d_{ID} = [s]Q_{ID}$.
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 can be powered on and be deployed into field to start the network operation. During network operation each nodes will sense the channel for detect other nodes in order to establish a neighbourhood. When authenticated communication is required, then the next stage is started.

Node discovery: Once a single 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. $ID_S$ will try to establish an authenticated link by performing the following dialog:

1. $ID_S$ generates message $s = hello(ID_S, TS)$ containing the origin node identifier as well as a timestamp mark.
2. $ID_S$ sends $s$ to $ID_A$.
3. Once $ID_A$ 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 $ID_A$ received the request, it will generate a challenge message to verify that $ID_S$ belongs to the WSN. It would be verified only in case that the supplicant holds a TA-
cleared private key. To generate and send the challenge \( ID_A \) will proceed as follows:

1. Computes \( Q_{ID_A} = H_3(ID_A) \)
2. Selects randomly \( r \in \mathbb{Z}_q^* \) and computes \( v' = H_2 \left( \left( \hat{\epsilon}(Q_{ID_A}, P_{pub}) \right)^r \right) \)
3. Selects \( k \leftarrow \{0, 1\}^i \)
4. Computes \( v = k \oplus v' \) and \( u = rP \)
5. Sends the pair \( Ch = (u, v, ID_A) \) as challenge to \( ID_S \).

**Response generation:** Once the challenge \( Ch = (u, v, ID_A) \) is received, \( ID_S \) will proceed as follows to generate a valid response:

1. Computes \( k' = v \oplus H_2 \left( \hat{\epsilon}(d_{ID_S}, u) \right) \)
2. Sends \( \tau = H_3(k', ID_S) \) as response ticket to \( ID_A \).

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

**Challenge verification:** Once \( \tau \) is received, \( ID_A \) 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 \( ID_A - ID_S \).
Elsewhere do nothing.

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

1. The message \( m \) is sent by node \( ID_i \) to node \( ID_j \) accompanied with its MAC code \( H_3(k_{l-i}, m) \).
2. Once \( ID_j \) received \( m \) it validates its authenticity verifying 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 \( k_{l-i} \) exists in both \( list_{ID_j} \) and \( list_{ID_i} \). Elsewhere both nodes have to run this protocol in order to arrange a link-key.

### 2.3 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.

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 proceed to estimate the cost for message authentication considering the amount of MCU cycles consumed during its processing.

### 3 Cost Evaluation

#### 3.1 Involved operations

In order to determine how processing and storage needs are distributed among the protocol execution, the list of the cryptographic primitives executed by each node was collected and showed in Table 1.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Authenticator operations per node</th>
<th>Supplicant operations per node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Discovery</td>
<td>Ch. Gen. 1 rnd, 1 hash, 1 p, 1 ee, 1 me, 1 xor</td>
<td>Ch. Resp. 2 MAC, 1 p, 1 xor</td>
</tr>
<tr>
<td>Ch. Resp.</td>
<td>Resp. Verifying 1 MAC</td>
<td>Message Auth 1 MAC</td>
</tr>
</tbody>
</table>

Table 1 Distribution of Execution of cryptographic operations.

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.

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

Those both functions have to be evaluated on the amount of resources consumed, but message authentication deserves special attention due its frequent execution that is expected during intensive message exchange between once authenticated nodes. Then we proceeded to establish the processing and storage needs involved in the evaluation of MAC codes by two nodes \( ID_i \) and \( ID_j \).

#### 3.2 Cost estimation of a MAC code

A MAC code is a cryptographic primitive build over other primitives such as hash functions or block
ciphers. There is a wide variety of MAC codes that can be suitable for implementing the proposed protocol. In this paper, for estimation purposes, we selected the HMAC MAC code [9] that is based on a cryptographic hash function. HMAC is 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. The construction of the HMAC code is described in the following equation

\[ \text{HMAC} = H(k \oplus \text{opad} \Vert H(k \oplus \text{ipad} \Vert m)) \]

Where \( H \) denotes a cryptographic hash function, \( k \) is the key, and \( \text{opad} \) and \( \text{ipad} \) are both two byte strings corresponding to 0x56 and 0x36 values respectively.

From simple observation it can be established that the evaluation of HMAC is composed by two evaluations of XORs and two evaluations of the underlying hash function. Here, we assume to use the SHA-1 hash function that produces a hash value of 160 bits, short enough for a WSN environment. In this way we are able to obtain a closer estimation of the message authentication cost.

Another interesting observation is that HMAC is defined upon the assumption that at least four hash evaluations has to be done, or an hash pre-calculating alternative can be followed in order to reduce to two hash invocation during message authentication. Using this alternative requires the user key to pre-combined with the \( \text{ipad} \) and \( \text{opad} \) 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. In order to take this alternative we will require a slight modification to the link-key list 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 \( \text{ipad} \) and \( k \) XOR \( \text{opad} \) respectively, those ones that have to be obtained once the link-key has been agreed.

### 3.3 Cost estimation of SHA-1

The SHA-1 hash function was published in 1998 as a Federal Information Processing Standard by the US government for hashing purposes [10]. It is able to process messages in 512-bit blocks and to generate a 160-bit output.

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 other 64 words calculated as expansion of the original message.

In order to obtain an indicator of how the SHA1 hash function would perform over an specific platform, we selected a target platform within the available platforms for WSN: a TELOS B node [11] from Xbow; it runs a 16 bits TI MSP430 microcontroller with a 4 Mhz. clock.

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.

About arithmetical additions modulo \( 2^{32} \) executed during the iterated block, it requires to perform the addition for the low end 16-bit word, and then do 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 other arithmetic operations are required by SHA-1.

In summary, the estimated operations resulted from mapping SHA-1 hash function to the available operations into a MSP430 MCU are showed in the Table 2.

<table>
<thead>
<tr>
<th>MSP430 Instr.</th>
<th>Ops.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROTL</td>
<td>1254</td>
</tr>
<tr>
<td>ROTL</td>
<td>480</td>
</tr>
<tr>
<td>AND</td>
<td>334</td>
</tr>
<tr>
<td>XOR</td>
<td>376</td>
</tr>
<tr>
<td>NOT</td>
<td>40</td>
</tr>
<tr>
<td>ADD</td>
<td>1041</td>
</tr>
<tr>
<td>ASG</td>
<td>715</td>
</tr>
<tr>
<td>CMP</td>
<td>96</td>
</tr>
<tr>
<td>INC</td>
<td>96</td>
</tr>
</tbody>
</table>

Table 2 Operations for SHA-1 evaluation on MSP430.

From the estimations for the SHA-1, it is possible to estimate the evaluation cost for HMAC code, just it is necessary to apply a factor 2 (remembering that HMAC involves at least 2 calls to SHA-1), we observed that each MAC code will consume approximately 8,864 instructions on the
MCU, excluding data loading and storing. From this data and considering an average of 4 cycles per instruction it would take 35,456 cycles per MAC that would take 8.864 milliseconds just for processing.

These results can be considered as simple indicator of how much overhead could represent hash calculation compared with a specific application running on the sensor node.

4 Conclusions and further work
The MAC codes evaluation 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 are required to obtain a complete evaluation of the authentication protocol proposed.

The results presented resume the cost associated for message authentication in a static fashion view, where only a one node scenario was observed. Then the used approach can considered as estimation that reflects only local effects over separated nodes. According to this, other interesting issue to be addressed in further work is to obtain estimation methods about the dynamic behavior of the network under the execution of our authentication protocol, reflecting how topology changes impact on the network efficiency.

Due presented cost estimations were obtained using estimations for a specific hardware, the estimation would be adjusted for a general case.

Finally, it is important to remark that any estimated performance is useful like a general indicator of performance, in contrast to a functional implementation for the protocol that would be exposed to miscoding, compiling and language overhead.

References: