Efficient Access Control for Wireless Sensor Data

Alessandro Sorniotti[1,2], Refik Molva[2], Laurent Gomez[1], Christophe Trefois[3], and Annett Laube[2]

[1] SAP Research
805, Docteur Maurice Donat, 06250 Mougins, France
Email: first.last@sap.com
[2] Institut Eurécom
2229, Route des Crêtes, 06560 Valbonne, France
Email: first.last@eurecom.fr
[3] EPFL
CH-1025 Lausanne, Switzerland
Email: christophe.trefois@a3.epfl.ch

Abstract. Although very developed in many sectors (databases, filesystems), access control schemes are still somewhat elusive when it comes to wireless sensor networks. However, it is clear that many WSN systems – such as healthcare and automotive ones – need a controlled access to data that sensor nodes produce, given its high sensitivity. Enforcing access control in wireless sensor networks is a particularly difficult task due to the limited computational capacity of wireless sensor nodes. In this paper we present a full-fledged access control scheme for wireless sensor data. We enforce access control through data encryption, thus embedding access control in sensor data units. We also propose a lightweight key generation mechanism, based on cryptographic hash functions, that allows for hierarchical key derivation. The suggested protocol only relies on simple operations, does not require interactions between nodes and data consumers and has minimal storage requirements.

1 Introduction

Wireless Sensor Networks (WSNs) can be seen - in a raw approximation - as sources of input, delivering data from the real world into the digital world. Low cost often arises as a requirement to justify the adoption of WSN technology for particular business scenarios: this results in sensors often being simple devices, with limited computing and transmitting power and limited battery capacity too.

In many WSN scenarios, nodes are required to sense a vast range of different data types: in an elderly-care scenario for instance, ambient sensors sensing room occupancy, temperature, motion, activity and sound are used together with body sensors sensing blood pressure, HRV, galvanic skin response, SpO2, blood glucose rate and so forth [1].

In scenarios such as the healthcare one, the sensed data is often highly sensitive. Moreover, the sensed data often has very different levels of sensitivity: the mere information on the room occupancy of a hospital is not highly sensitive, whereas the ECG of a given patient is indeed very private information, since it could possibly reveal information about the health status of the person.

There can also be several consumers of wireless sensor data, belonging to an heterogeneous population, and having intrinsically different data access rights: within a healthcare scenario, patients, social workers, nurses, relatives, generic physicians and specialists naturally form a hierarchy of entities that are interested in the data delivered by a healthcare WSN. Data consumers can be therefore conveniently organized in hierarchies. Low levels in the hierarchy can just access data with low level of sensitivity whilst higher levels can also access more sensitive data.

A problem of hierarchical access control therefore clearly arises. The solution to such a problem is additionally complicated by the resource limitation of some WSN installations. In this paper we present a hierarchical access control scheme for wireless sensor data. Access control is enforced using cryptography: sensors encrypt data prior to its transmission, thus embedding access control right at the source. Thanks to the key generation mechanism, multiple consumers, with different access rights, converge on the same decryption key if their privileges are sufficient. The presented protocol achieves two very desirable goals for WSNs: it does not use complex operations and it does not require any interaction between the different nodes and the different data consumers.

The rest of the paper is organized as follows. Section 2 states the problem. Section 3 presents the state of the art in related areas. Our access control scheme is presented in Section 4. Section 5 and 6 analyse the security of the scheme. Section 7 gives conclusions.

2 Problem Statement and Approach

Sensor nodes produce, on a broadcast medium, highly diverse data, which is often very sensitive. Sensors listeners may be numerous, diverse and have different access rights to sensor data. The problem of multiple-resources/multiple-accesses is usually solved using access control. Under a standard access control scenario, entities that wish to benefit from the produced information, have to authenticate themselves, receive a credential, produce the credential to the data source and receive a specialized stream of information that contains just the information the requester received an authorization for. Many solutions exist for this problem [2], however most of them are unsuitable for WSN scenarios, given the technological constraints of the nodes. In addition, nodes produce data in real time, hence the generation of multiple streams is difficult.

Our solution relies on cryptography: right from its production, data is encrypted, and therefore its access is intrinsically restricted. This way, sensors can encrypt data and publish it regardless of the present consumers: the knowledge of the cryptographic key used to encrypt data, belonging to a given level, allows

proper decryption – and therefore access – to data belonging to that level. Conversely, it is impossible to access encrypted data for consumers who do not have the proper decryption key.

To satisfy the hierarchical requirement, the idea is to map each distinct sensor data type to an *authorization level*. Data, whose disclosure does not rise high privacy issues, is mapped to low *authorization levels*. Similarly, highly private data will be mapped to high *authorization levels*. The resulting mapping expresses the security preferences of a central access control policy point. The hierarchy of authorization levels is then mapped to keys in a hierarchical structure, whereby low-level keys can be derived from high-level ones.

We model the hierarchy of authorization levels (al) as a tree. al_0 is the root of the tree, and represents the highest level. $\mathcal{C}(al_i)$ represents the set of children nodes of a given level al_i. $\mathcal{P}(al_i)$ represents the parent node of a given level al_i, with $\mathcal{P}(al_0) = \emptyset$. We restrict the admissible hierarchies to the ones which can be modeled by trees such that $\forall\ al_i,\ \exists!\, al_k\ :\ al_k = \mathcal{P}(al_i)$, i.e. where nodes have just a single parent. We assume that a user who receives access rights to al_i is able to derive access rights to $\mathcal{C}(al_i)$, $\mathcal{C}(\mathcal{C}(al_i))$ and so forth, while the converse (i.e. deriving access rights for $\mathcal{P}(al_i)$) is not allowed. In section 4 we will detail how we implement such mapping between authorization level tree and cryptographic keys.

The adoption of encryption as a way to enforce access control reduces the problem of granting, denying and revoking access rights to a problem of key management. We assume the presence of a central access control manager (ACM) which – after evaluation of data consumers' (from now on also referred to as users) credentials – takes care of granting, denying and revoking access rights. Granting a user to a given authorization level means giving her the key to decrypt all data units mapped to that level and to descendant ones. Denying access simply implies not providing the decryption key(s). Finally, revocation of access rights is based on rekeying: changing the keys used at a given point, forces data consumers to re-contact the ACM in order to receive the new keys. Consumers whose access rights have been revoked do not receive the new keys, which accomplishes the revocation. This approach achieves the desirable property of no specific interactions between data producers (the sensor nodes) and data consumers, other than data publishing.

3 Related Work

The seminal work of Akl and Taylor [3] first proposes a solution for data access control based on cryptography. Access controlled resources (data), users and cryptographic keys are mapped to a hierarchy of classes, represented by a directed acyclic graph. Data belonging to a given class is encrypted with the key associated to that class. The key generation scheme uses the homomorphic properties of modular exponentiation. It assures that a user, who is given the decrypting key of a class, can generate the keys of that class' descendants, and therefore access data mapped to descendant classes as well. On the contrary,

the inverse – generating the key of a parent class – is unfeasible. However, the expensive operations used in the scheme (modular exponentiation) make this scheme unsuitable for a WSN environment.

In [4], Chien proposes a much lighter key generation scheme, based on one way hash functions instead of modular exponentiation. In addition, the author places a time bound on keys, introducing time periods: during each time period, a new key for each class of data is derived. However, this scheme suffers from a few drawbacks: first of all it requires tamper resistant devices, in order to store secret material used to derive keys. Second, similarly to Akl's scheme, it is impossible to revoke a user's access right to a lower class in the hierarchy. Finally, in [5], Yi showed an attack where, despite the tamper resistance requirement, a coalition of three user can access some secret class keys that they should not know according to Chien's scheme.

In [6], Tzeng proposes a time-bounded key assignment scheme for hierarchies. The computation of the keys however, involves particularly expensive Lucas function computation. This scheme is not suitable for resource-constrained WSN nodes due to the particularly expensive operations required for the computation of keys.

In [7], Shehab et al. propose a mechanism to generate and distribute hierarchical keys. The mechanism presented in the paper, based on hashing as well, is somewhat more refined then the one we present in this paper, allowing for instance arbitrary hierarchies. Although efficient and very well suited for WSN, this scheme has no time bound on keys, and therefore it is not ready to represent a fully flourished access control solution.

In [8], and [9], Atallah and colleagues propose a general and efficient scheme to incorporate time bounds in existing management scheme. In addition, they show how to create a full-fledged hierarchical access control scheme with time capabilities. The scheme is elegant and efficient, relies just on one way hash functions, but – seen from a WSN viewpoint – requires a too elevated amount of public information in order to allow for efficient key derivation.

4 The Scheme

In Section 2 we presented an approach to sensor data access control based on data encryption with a hierarchical key structure, which allows for key derivation of children authorization levels. In this Section we first introduce the cryptographic primitives used to implement the scheme, then the various mechanisms used in the scheme and finally we wrap-up showing how all the pieces come together to form an access control system.

4.1 Preliminary definitions

Let $h : \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^n$ be a message authentication code (MAC) based on a one-way hash function (OWHF) f (such as RIPEMD-160 or SHA-1). We recall that OWHFs require *preimage resistance* (given a value $F \in \{0,1\}^n$,

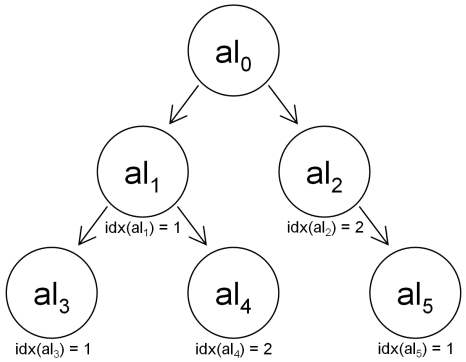

Fig. 1. Authorization level structure

it is computationally infeasible to find x such that $f(x) = F$) and *2nd-preimage resistance* (given x, it is computationally infeasible to find x' such that $f(x) = f(x')$). h takes as input an n-bit secret value and an arbitrarily long string and produces a pseudo-random n-bit string that strongly depends on both the secret value and the string. An example of such function is the well-known HMAC [10].

Using h, we can efficiently associate values to each authorization level, so that the derivation process suits the hierarchical requirement of the scheme. We then use these values to derive the keys used in the scheme. Let us label each of the direct children nodes of a parent node with an incremental index, 1 for the leftmost child, 2 for the next one and so forth. We refer to the index of a generic node al_i as $idx(al_i)$. Then, the values $\mathcal{V}(al_i)$ associated to each level can be computed as

$$\mathcal{V}(al_i) = \begin{cases} V_0, & \text{if } i = 0; \\ h(\mathcal{V}(\mathcal{P}(al_i)), idx(al_i)), & \text{if } i \neq 0; \end{cases}$$

where V_0 is a given initial value, whose generation mechanism will be detailed later on in this Section. For example, with reference to the hierarchy in Figure 1,

$$\begin{aligned} \mathcal{V}(al_4) &= h(\mathcal{V}(\mathcal{P}(al_4)), idx(al_4)) \\ &= h(\mathcal{V}(al_1), 2) \\ &= h(h(\mathcal{V}(\mathcal{P}(al_1)), idx(al_1)), 2) \\ &= h(h(\mathcal{V}(al_0), 1), 2) \\ &= h(h(V_0, 1), 2) \end{aligned}$$

It is straightforward to see how it is easy to derive values for descendant levels from higher ones, whereas the converse is unfeasible thanks to the one-wayness of f.

In order to further explain the scheme, we introduce the encryption scheme used in the system, which is the well-known one-time-pad (OTP) scheme [11].

The efficiency of OTP makes it very suitable for WSN environment. As widely known in literature, one-time-pad is information-theoretically secure as long as the encryption key is never reused twice. We must keep in mind this requirement when we design the key generation mechanism.

4.2 Key Generation, Distribution and Derivation

Keys for a given authorization level al_i are derived from the values $\mathcal{V}(al_i)$. The use of OTP as encryption and decryption mechanism requires to have a different key for each sensor, since the encryption of data belonging to the same authorization level leads to key reuse, which opens the possibility of breaking the encryption scheme with statistical attacks. A sequence number is also needed to differentiate the keys used by the same sensor node to encrypt the different data units – belonging to the same authorization level – that are generated by the same sensor, for the same reason mentioned before.

Keys can therefore be computed as

$$K_{al_i,ID,seq} = h(\mathcal{V}(al_i), ID \| seq)$$

where ID is a univocal numeric identifier for each sensor, it can be easily derived from any univocal network layer address, and seq is a sequence number, locally maintained at each sensor node and incremented each time a new data unit is sent.

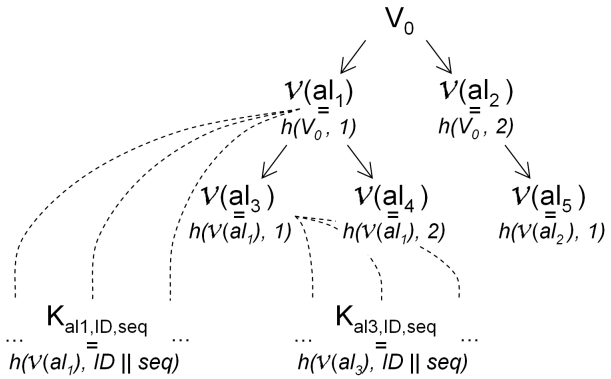

Fig. 2. Derivation of keys

When a user wants to access sensor data at a given authorization level al_l, she contacts the ACM, produces her credentials and she is either cleared or refused. We point out that these aspects of the protocol (e.g. details on how to obtain credentials, how to authenticate to the ACM, the policies in use and so forth) are out of the scope of this paper and therefore not addressed in this paper. If she is cleared she receives an access right value. From the latter she will be able

to derive all the keys to decrypt data, classified to authorization level l or to its descendants in the hierarchy.

To understand the key derivation mechanism, we refer to Figure 2. Let us assume that a given user is cleared to authorization level al_1. Then she will receive the value $\mathcal{V}(al_1)$. From that value she can easily derive the keys $K_{al_1,ID,seq}$ for all seq and ID in one step computing $h(\mathcal{V}(al_1), ID \| seq)$. She can also easily compute all the keys for authorization levels that are descendants in the hierarchy. For instance, she can compute the keys $K_{al_3,ID,seq}$ for all seq and ID in two steps, first computing $\mathcal{V}(al_3) = h(\mathcal{V}(al_1), 1)$ and then computing the key as $h(\mathcal{V}(al_3), ID \,\|\, seq)$. The same process can be applied to compute any key which is a descendant of the granted one. In general, when a user is cleared to authorization level i, she then receives $\mathcal{V}(al_i)$ from the ACM.

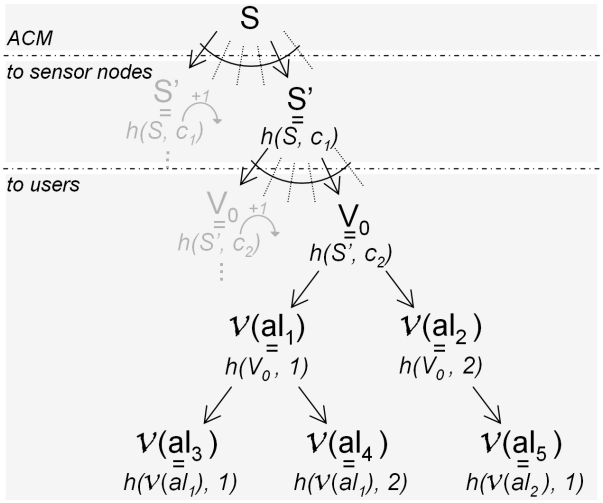

Fig. 3. Complete key generation scheme

Figure 3 shows a complete picture of the key generation scheme, which also encompasses the generation of the value V_0. In order to do so, let us introduce two counters, c_1 and c_2, and a secret value S. Both counters are initialized to one. S and c_1 are used to compute S' as $h(S, c_1)$. S' is updated as c_1 increases. Similarly, S' and c_2 are used to finally compute V_0 as $h(S', c_2)$. V_0 too is updated as c_2 increases.

The counter c_2 is incremented each time the need for revocation arises: each time a user grant needs to be revoked, c_2 is increased and V_0 is updated. From the updated value, a new set of keys is generated, using the technique introduced earlier on in this section. Previous keys are therefore no longer used for encryption and consequently, keys that were previously used for decryption cannot be used any longer; hence all the access rights are revoked altogether.

The remaining problem is how to distribute keys to sensor nodes. A simple approach could be the predistribution of the secret value S to each sensor node prior to the deployment of the network. This solution allows a node to generate all the keys of the system. Although very practical, this solution suffer from a security exposure, since by compromising a single node, an attacker would be able to decrypt all data. This problem could be solved by requiring nodes to be tamper-resistant; however this requirement clashes with the economic constraints of sensor nodes. On the other hand, distributing the updated values of V_0 to sensor nodes as access grants are revoked, would result in excessive transmission overhead.

As a good tradeoff, nodes are instead given the intermediate values of S' (see Figure 3). This way, we allow sensors to update the value of V_0 across multiple revocation phases, yet, if ever a node is compromised, the ACM simply increments c_1 and computes a new S', which is transmitted to non-compromised sensors using a reliable, confidential, authenticated broadcast scheme [12][13].

4.3 Putting it All Together

At system startup, the ACM assigns a random secret value to S and sets c_1 and c_2 to one. Then it creates a mapping of each of the data types sensed by sensors into authorization levels, and installs this mapping onto each sensor. Finally, the value $S' = h(S, c_1)$ is broadcasted to each sensor on a secure channel, along with the initial value of c_2.

Each sensor has a numeric identifier ID that univocally identifies it in the sensor population. Each sensor also has a counter seq which is initialized to zero. Upon sensing a value v, a sensor derives its associated authorization level, say, al_x, generates the proper encryption key $K_{al_x,ID,seq}$ using its ID and the current value of seq and broadcasts

$$\{K_{al_x,ID,seq} \oplus v,\ al_x,\ ID,\ seq,\ c_2\} \tag{1}$$

It then increments seq and sets itself ready to sense another value.

When a user joins the system, it contacts the ACM and performs the authentication/authorization process, at the end of which she might be cleared to an authorization level. Let us assume that the granted level is al_g. She is then given $\mathcal{V}(al_g)$, computed with the value of c_1 and c_2 currently in use. She is also given the value c_2.

Upon receiving sensor data as in 1, a user first checks if the received c_2 is equal to the held one. If so, she then checks if al_x is equal to, or a child authorization level of, the granted level al_g ($al_g \preceq al_x$). If so, then she uses the key derivation procedure to compute the key and decrypt the value. If c_2 is different, she contacts again the ACM to refresh its grant. Finally, if $al_g \not\preceq al_x$, she simply does not have sufficient privileges to access data belonging to that class.

If the ACM needs to revoke the access rights of a user, it broadcasts – through an authenticated and reliable channel – a command to force each sensor to increment c_2 and update V_0 accordingly. Keys are therefore re-computed from the

new value of V_0: this results in an immediate access rights revocation for all users. Users are therefore forced to re-contact the ACM to get the new access right values $\mathcal{V}(al_i)$. Naturally, a user whose grants are to be revoked will not succeed in this operation, thus effectively having its access rights revoked. Thanks to this feature, the proposed scheme can handle situations in which a sudden escalation of access rights is required. A user can be granted access to a very high authorization level (even the root of the tree if needed), only to be revoked the exceptional permissions later on when this is not required anymore, so that the original privileges can be easily reinstated.

Our scheme also handles the case of compromised sensor nodes. We assume that a compromised node is eventually detected: different detection strategies exist in literature [14, 15]. If a node is compromised, the ACM increments c_1 and c_2, and transfers to all non-compromised sensors, on a secure channel, the updated value $S' = h(S, c_1)$: this step can be performed through a secure broadcast scheme[4]. Object of the broadcast will only be non-compromised nodes: the compromised nodes that are detected, are pruned from the communication infrastructure and do not receive the new value. The ACM also sends the command to force each sensor to increment c_2. Sensors then update V_0 using the new value of S' freshly received from the ACM, and the incremented c_2. Thus, the compromised sensor(s) holds an outdated V_0, whose exposure to an attacker is not a concern. We point out that – from a user's perspective – this process is undistinguishable from a normal revocation: users just witness an incremented value of c_2 and are therefore forced to re-contact the ACM to get new access right values.

5 Scheme Analysis

In this section we firstly evaluate the security of the scheme. In order to properly accomplish this task, we need to bear in mind that the ultimate objective of the scheme is to allow sensor data access control right from sensor data production. Consequently, we will focus our analysis only on the soundness of the access control mechanism. Other orthogonal security aspects are not touched upon in this paper, as there is plenty of active research addressing them. The security requirement that we will investigate are therefore only the security of the key derivation process and the security of the encryption scheme used.

To tackle the security analysis, we need to state a few assumptions: first of all, we assume that a corrupted or malfunctioning node will eventually be detected and singled out. This is a common assumption, and it represents an active area of research (for instance see [14, 15]). Last, we assume that there is a way to distribute key updates secretly and on an authenticated channel, and there is a way to distribute authenticated messages commanding nodes to update their counters. Also these last two assumptions are reasonable, as the area of authenticated broadcast and secure communications in WSN is very prolific

[4] For a comprehensive view on the state of the art in the domain, the reader can refer to [16]

and has already presented viable solutions to these problems (e.g. [16]). With this in mind, we can move on to some considerations about the security of our access control approach.

First of all it is clear that an outsider, with only publicly known information available, cannot generate a valid key, since every key depends on the secret value S. S is never disclosed, and therefore an outsider has no chance to get it and use it to generate a valid key.

Let us now focus on the security of the key derivation scheme. Our requirement is that once the ACM has distributed access control grants (in the form of valid $\mathcal{V}(\cdot)$ values) according to an access control decision based on system policies, users cannot bypass this decision. Stating this more formally, any coalition of users cannot escalate their privileges or, similarly, for every coalition, the highest attainable class is the highest granted one. This claim easily follows from the choice of a secure message authentication code such as [10] which does not allow existential forgery under chosen-plaintext attacks. This property translates into the impossibility – from $\mathcal{V}(al_i)$ – to forge $\forall\ \mathcal{V}(al_j) : al_j \in \mathcal{C}(\mathcal{P}(al_i))$ i.e. the value of any of the direct siblings of al_i. The one wayness of f in turn, protects us from the derivation of the value of the parent node $\mathcal{V}(\mathcal{P}(al_i))$ from $\mathcal{V}(al_i)$. It is straightforward to see how these two assurances together lead to the impossibility of privilege escalation.

Public storage	mapping
Required storage on sensor nodes	seq, ID, $c2$, S'
Required storage on users	$\mathcal{V}(al_i)$, c_2
Key derivation	$O(n)$ hash operations
Encryption/Decryption	1 hash + 1 xor
c_2 **rekeying**	update message
c_1 **and** c_2 **rekeying**	update message + S'

Table 1. Considerations on the performances of the scheme

Let us now analyse the security of the encryption scheme. It is well known that the one-time pad algorithm assures semantic security if the encryption keys are randomly chosen and never re-used. We have shown that no two sensor data can be encrypted with the same key. Therefore, since keys are never reused, the only possibility for a statistical attack is some correlation between different keys. However, with the choice of a strong hash function for f, all the keys are pseudo-random and the correlation between them is so small as to discourage any statistical attack. In addition, it is possible to reduce the encryption scheme to $d \oplus h(K, \mathit{counter})$ where d is the cleartext value and K is a secret value. If we model h as a random oracle [17], it is easy to see that the encryption scheme is equivalent to the well known CTR encryption scheme, introduced in [18], which exhibits strong security properties. In the next Section we are going to investigate this aspect in more details.

It is clear that, the encryption scheme used in the system being a symmetric one, malicious users can inject encrypted data in the system and have honest consumers decrypt it as if it were a legitimate data unit. However the data origin authentication problem is not addressed here since this is an access control scheme that deals just with confidentiality and authorization.

As for the performances of the scheme, we can see in Table I that the proposed scheme achieves remarkable results. The required public storage only amounts to the mapping of data types to authorization levels, which is a data structure representing – for instance – the tree in Figure 1; sensor nodes have to store two counters (c_1 and c_2), their numeric identifier and the current value S', whereas users are just required to store one counter (c_1) and the access right value $\mathcal{V}(al_i)$ for a given granted class ac_i. Looking at the number of operations, we can see that to derive a key, users and nodes require an average of $O(n)$ hash computations, where n is the depth of the tree of the authorization levels. We underline that this operation is just performed once to derive $\mathcal{V}(al_j)$ from the access right value $\mathcal{V}(al_i)$, $\forall al_j \ : \ al_i \prec al_j$. After the derivation, encryption and decryption are performed with one single hash evaluation to derive the key and a single xor operation to perform OTP. Finally, the two different types of rekeying, the one that involves the increment of c_1 (in turn, of both c_1 and c_2), just requires one message to order sensor nodes to increment c_1 (in turn, one message to order sensor nodes to increment c_1 plus the secure broadcast of the updated S').

6 Randomness Evaluation

In Section 5 we mentioned that the proposed scheme is semantically secure, provided that keys are randomly chosen and never reused, so that the correlation between them is so small as to discourage any statistical attacks on the ciphertext. The randomness is dependent on both the choice of the hash function and the way it is used in the construction of the keys. In our current implementation, we are using RIPEMD-160 [19] as hash function for which there are no known attacks at the time of this writing. In this Section we will evaluate the randomness of the keys using the NIST SP 800-22b test battery.

The NIST (National Institute of Standards and Technology) Test Suite SP 800-22b (detailed in [20]) is a statistical package consisting of 15 tests. These tests evaluate the randomness of a given sequence of keys. The NIST organization provides an ANSI C implementation of that test battery which can be freely downloaded from [21]. DIEHARD [22] is another free of charge package that offers a variety of statistical tests. However, many of the tests are based on the NIST 800-22b package and so we choose NIST to perform the testing.

6.1 Parameter Choices

For each test, the NIST test suite extracts a probability, called the *p-value*. The *p-value* is a summary of the strength of the analyzed sequence against a perfectly random one. A *p-value* of 0 suggests that the sequence is completely

non-random. We denote α as the confidence level which typically ranges from 0.001 to 0.01 [20]. Let Seq be the sequence of bits, then for each test we have

$$Randomness(Seq) = \begin{cases} True \ , \ if \, p - value > \alpha \\ False \ , \ if \, p - value < \alpha \end{cases}$$

NIST recommends running the tests on 1000 sequences of keys with each sequence having at least 10^6 bits in order to determine if a key stream is random or not. We generated 10^7 keys of 160 bits each. The resulting 1.6 Gib file was then used as input for the 15 tests. Each test was run on 1000 sequences of 10^5 keys.

The significance level α was set to 0.01, which is the suggested level when dealing with cryptography [20, 4.3.f]. This means that if a $p\text{-}value > 0.01$ the sequence is accepted as random with a confidence of 99%. Similarly, if a $p\text{-}value < 0.01$, the sequence is non-random with confidence of 99%.

Figure 4 shows a list of the 15 tests of the NIST Suite with a short description of each. For more details about the tests, the reader can refer to [20].

Test Name	N°	Short Description
Frequency	1	Analyze the proportion of 0 and 1's in the sequence.
Block Frequency	2	Analyze the proportion of 0 and 1's within M-bit blocks.
Cumulative Sums	3	Maximal excursion (from 0) of the random walk defined by a cumulative sum in the sequence.
Runs	4	Total number of 0 and 1 runs in the sequence, where a run is an uninterrupted sequence of identical bits.
Longest Run of Ones	5	Longest run of 1's within M-bit blocks.
Rank	6	Rank of disjoint sub-matrices of the entire sequence.
Discrete Fourier Transform	7	Peak heights in the discrete Fast Fourier Transform to detect periodic features.
Non-Periodic Template Matching	8	Number of occurrences of pre-defined non-periodic patterns.
Overlapping Template Matching	9	Number of occurrences of pre-defined runs of 1's.
Universal Statistical	10	Number of bits between matching patterns to determine compressibility.
Approximate Entropy	11	Frequency of all overlapping m-bit pattern.
Random Excursions	12	Number of cycles having exactly K visits in a cumulative sum random walk.
Random Excursions Variant	13	The number of times that a particular state occurs in a cumulative sum random walk.
Serial	14	Frequency of all overlapping m-bit pattern in the entire sequence.
Linear Complexity	15	The length of a generating feedback register to determine if the sequence is complex enough to be random.

Fig. 4. List of Tests and their Description

6.2 Results

There are two suggested approaches to interpret the results of the NIST Test Suite. The first one deals with the examination of the proportion of sequences that pass the test while the second one focuses on the uniformity of the *p-values*. The two approaches are shortly discussed below.

Examination of Proportion of Passing Sequences The final analysis report generated by the suite contains a value called proportion for each test. The proportion is the number of sequences that passed (e.g. with p-value $> 0.01 = \alpha$) divided by the total number of sequences tested. In other words, the proportion is the percentage of passed tests.

NIST specifies a range of acceptable proportions determined by using the confidence level defined as $\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}$, where $\hat{p} = 1 - \alpha$ and m is the sample size (e.g. 1000).

In our case, we used $m = 1000$ sequences and each sequence had 1.6 Mib bits. Using the formula for the confidence level above, our range of acceptable proportions is from 0.9805 to 0.9994. Figure 5 shows the proportion for each test. Since the proportion for each test lies within the range we computed, we can accept the sequence as a random bit sequence.

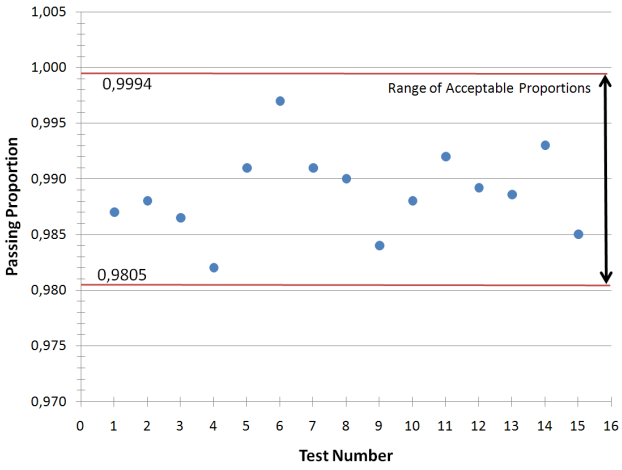

Fig. 5. Passing Proportion of the 15 NIST Tests

Examination of the Uniformity of p-values The second approach presented by NIST is the analysis of the uniformity of the *p-values*. If the *p-values* form

a uniform distribution, then we accept the sequence as random. Uniformity can be examined by performing a χ^2 (Chi-Square) test on the p-values, with

$$\chi^2 = \sum_{i=1}^{10} \frac{(C_i - \frac{m}{10})^2}{\frac{m}{10}}$$

where C_i is the number of p-values in the sub-interval $[\frac{i-1}{10}, \frac{i}{10})$, i $= 1 \ldots 10$ and m is the sample size (the number of sequences tested, e.g. 1000). Finally, a new p_Γ-value is computed from the original p-value as

$$p_\Gamma - value = igamc(\frac{\chi^2}{2}, \frac{9}{2})$$

where igamc is the Incomplete Gamma Function as defined in [20, Section 5.3.3]. In conclusion, the *p-values* form a uniform distribution if $p_\Gamma > 0.0001$ and do not otherwise.

Figure 6 shows the p_Γ-values for the tests we conducted. From the table we see that all test passed the uniformity condition and thus the sequence is random.

Test N°	1	2	3	4	5	6	7	8
$p_\Gamma - value$	0.371	0.417	0.717	0.798	0.684	0.371	Skipped	0.493
Conclusion	**Pass**	**Pass**	**Pass**	**Pass**	**Pass**	**Pass**	**NA**	**Pass**
Test N°	9	10	11	12	13	14	15	
$p_\Gamma - value$	0.469	0.987	0.209	0.642	0.640	0.332	Error	
Conclusion	**Pass**	**Pass**	**Pass**	**Pass**	**Pass**	**Pass**	**NA**	

Fig. 6. Uniformity Distribution of p-values

We omitted the Fast Fourier Transform test from the examination of uniformity as the results were not usable. Furthermore, in [23], the authors state that the results of the FFT test are degrading when the number of sequences increases. In particular, if the number of sequences is bigger than 10000, then any Pseudo-Random Number Generator fails the uniformity test. For that reason, we decided not to include this test in the uniformity evaluation.

The evaluation of uniformity for the Linear Complexity test raises an exception when we run it. At this point we are still investigating the cause of this as we are not aware of the reasons behind the failing of this test. In order not to bias our results by using erroneous values, we decided to remove the Linear Complexity test from the Uniformity Evaluation.

These results seem to confirm that using hash functions lead to pseudo-random key generators. Furthermore, we showed that the way keys are computed (e.g. by incrementing a counter for each key) seems to preserve pseudo-randomness and could be compared to other pseudo-random number generators (PNRGs) [24] such as Blum Blum Shub [25] or Fortuna [26].

Beside generating random key streams, the hash function needs to be collision resistant. In Cryptography, a brute force collision search has a time and memory complexity of $\sqrt{2^n}$ where n is the hash size [24]. This means that for a hash of 160 bits, the complexity is $\sqrt{2^{160}} = 2^{80} = 1.20892582 \times 10^{24}$. Thus, if we suppose that the computation of a hash takes 1 [ns], then a brute force attack requires 38 million years to be successful. This means that if no statistical attacks are known, the hash function can be considered collision resistant.

7 Conclusion

We have presented a hierarchical access control scheme for wireless sensor networks. The scheme relies upon data encryption in order to protect data access from the moment of its production. A lightweight key derivation protocol – solely based on the computation of message authentication codes (MACs) – achieves hierarchical derivation of keys: users having sufficient, yet possibly different, access rights, can derive the same decryption key. The protocol supports easy revocation of access rights through rekeying, which can be performed seamlessly at each sensor. The intervention of the access control module is just required upon detection of a compromised node.

As future work, we intend to analyze if commutative hash accumulators can be used to solve the problem addressed in this paper, and if so, what are the advantages of disadvantages of such solution. Indeed hash accumulators seem perfectly suited for the task: if one can easily accumulate values related to authorization levels and epochs, the resulting hash value could then be a perfect key associated to a particular level and epoch. In addition, if the operation is semi-commutative, then the key derivation process becomes extremely easy and elegant. Nonetheless, this approach brings disadvantages, notably the expensive operations required for accumulation of values. We deem the subject worthy of future research.

References

1. Wasp Consortium. D6.2-II Elderly Care Application: In-depth scenarios and use cases. http://www.wasp-project.org/, 2007.
2. William Tolone, Gail-Joon Ahn, Tanusree Pai, and Seng-Phil Hong. Access control in collaborative systems. *ACM Comput. Surv.*, 37(1):29–41, 2005.
3. Selim G. Akl and Peter D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Comput. Syst.*, 1(3):239–248, 1983.
4. Member-Hung-Yu Chien. Efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1301–1304, 2004.
5. Xun Yi. Security of chien's efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1298–1299, 2005.
6. W. G. Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):182–188, 2002.

7. Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Efficient hierarchical key generation and key diffusion for sensor networks. In *Second Annual IEEE Communications Society Conference on Sensor and AdHoc Communications and Networks*, 2005.
8. Mikhail J. Atallah, Marina Blanton, and Keith B. Frikken. Incorporating temporal capabilities in existing key management schemes. In *ESORICS*, pages 515–530, 2007.
9. Mikhail J. Atallah, Marina Blanton, and Keith B. Frikken. Incorporating temporal capabilities in existing key management schemes. Cryptology ePrint Archive, Report 2007/245, 2007.
10. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *CRYPTO*, pages 1–15, 1996.
11. J-O. Mauborgne and G. Vernam. One time pad scheme. *at http://en.wikipedia.org/wiki/One-time_pad*.
12. Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, pages 480–491, 1993.
13. Shang-Ming Chang, Shiuhpyng Shieh, Warren W. Lin, and Chih-Ming Hsieh. An efficient broadcast authentication scheme in wireless sensor networks. In *ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 311–320, New York, NY, USA, 2006. ACM.
14. Mary Mathews, Min Song, Sachin Shetty, and Rick McKenzie. Detecting compromised nodes in wireless sensor networks. In *SNPD '07: Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, pages 273–278, Washington, DC, USA, 2007. IEEE Computer Society.
15. Tao Li, Min Song, and Mansoor Alam. Compromised sensor nodes detection: A quantitative approach. *icdcsw*, 0:352–357, 2008.
16. Adrian Perrig and J. D. Tygar. *Secure Broadcast Communication in Wired and Wireless Networks*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.
17. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
18. Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, page 394, Washington, DC, USA, 1997. IEEE Computer Society.
19. H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. In *Fast Software Encryption*, pages 71–82, 1996.
20. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST special publication 800-22, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 2001.
21. NIST 800-22b. Download documentation and software for the nist 800-22b special publication.
22. G. Masaglia. The marsaglia random number cdrom including the diehard battery of tests of randomness, 1995.
23. S. Kim, K. Umeno, and A. Hasegawa. Corrections of the NIST statistical test suite for randomness, 2004.
24. A. J. Menezes, S. A. Vanstone, and P. C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

25. L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.
26. N. Ferguson and B. Schneier. *Practical Cryptography*. J. Wiley & Sons, Inc., New York, NY, USA, 2003.