COMPARISON OF AUTHENTICATION SCHEMES FOR WIRELESS SENSOR NETWORKS AS APPLIED TO SECURE DATA AGGREGATION

RELATORE: Ch.mo Prof. Michele Zorzi

CORRELATORI: Dott. Ing. Paolo Casari, Ing. Paolo A. Castellani

LAUREANDO: Matteo Canale

A.A. 2009-2010

Comparison of authentication schemes for wireless sensor networks as applied to secure data aggregation

Laureando: *Matteo Canale*

Relatore: Ch.mo Prof. Michele Zorzi

Laurea in Ingegneria delle Telecomunicazioni

Anno Accademico 2009/2010

9 Marzo 2010

Table of contents

Chapter 1

Introduction

Thanks to the recent advances of hardware manufacturing technology and efficient software algorithms, the deployment of a network composed by hundreds or thousands of small, low-cost wireless sensors has nowadays become possible. These sensor nodes are resource-constrained devices, capable of performing simple computations, measuring environmental data and communicating with neighbouring units. As a wireless sensor network (WSN) is typically composed of a considerable amount of nodes, these devices has to be small and inexpensive; furthermore, in order to maximise the network lifetime, they have to ensure a low power consumption.

Several applications has been proposed for WSNs, ranging from environmental monitoring to military applications, from medical care to the management of household electrical devices, etc. Therefore, many different requirements emerge, depending on the specific application and on the context where the network has to be deployed. However, because communication is the most expensive process in terms of energy consumption for a sensor node (the authors of [2] claim that transmissions take up to the 70% of the overall network energy consumption), a common goal for WSN applications is to reduce as much the amount of data to be transmitted as possible. These data typically consist of sensor readings that have to be collected and transmitted towards the gathering point by means of multihop communication. A possible approach for reducing the amount of information to be transferred to the upper levels in the network hierarchy is that of performing *in-network data aggregation*, i.e., processing data in order to reduce their size. Data aggregation dramatically decreases the radio energy consumption and, therefore, is a fundamental component for many WSN applications. It is performed by means of an aggregation function and/or data compression and its basic idea is that of trading communication complexity for computational overhead, since simple local computations are typically less expensive than radio transmissions in terms of energy consumption.

The wireless medium, however, is intrinsically unsafe, as it is accessible by all users in the transmission range, even if not authorised. Furthermore, since a wireless sensor network can be possibly deployed in a hostile environment (let us think, for example, of a military application), where sensor nodes can be somehow

compromised by an adversary, a certain level of security has to be ensured, to the network in general and, specifically, also to data aggregation protocols. Typical security requirements include data confidentiality (i.e., the attacker should not be able to leak information about the message contents), data integrity (i.e., accidental or malicious modifications to sent packets have to be detected) and authenticity (i.e., the receiver should be able to verify that received messages were actually sent by the claimed entities).
Unfortunately, while trying to embed security primitives in a data aggregation protocol, several security challenges arise. In first place, it should be observed that, basically, data aggregation and security have contrasting goals: while the first tries to minimise the amount of transmitted data, the latter adds a non-negligible computational and communication overhead in order to ensure the verification of some security properties.
Secondly, the in-network processing performed by a data aggregation protocol makes it even harder, if not impossible, to ensure a strict, global verification of the aforementioned security requirements, as the nodes that perform the aggregation need to process the received data. Therefore, since no node except the final gathering point can be considered trustworthy, it is necessary to address the problem of ensuring security also in the aggregation process.
Last but not least, while embedding security on sensor nodes, their severe limitations in terms of computational capabilities and energy resources should be taken into account. Traditional, strong security primitives, in fact, are not applicable on most sensor nodes. Therefore, energy efficient and computationally simpler algorithms have to be adopted, sometimes by sacrificing some security in order to make the implementation feasible on sensor nodes.
Summarising, data aggregation protocols should be designed jointly with security protocols, with the purpose of yielding a good trade off between the overall protocol complexity and the ensured level of security, while keeping energy consumption within acceptable boundaries.

The present thesis mainly focuses on the problem of data authentication provided to a WSN data aggregation protocol, though an extensive overview on secure data aggregation protocols is provided. Three different authentication algorithms were implemented and analysed, together with their application in a suitable secure data aggregation protocol.

The work is organised as follows. Chapter 2 introduces the reader to the data aggregation problem and to the related security requirements; furthermore, the relationship between the two is analysed. In chapter 3, a survey on the current state of the art in secure aggregation protocols is presented, together with a comparison among the considered proposals. Chapter 4 investigates authentication and its relationship with data aggregation. Three authentication schemes, namely CBCMAC-RC5, HMAC-SHA1 and HMAC-MD5, are described and a protocol, ESAWN, which addresses the problem of ensuring end-to-end authenticity, is analysed. Chapter 5 presents the results obtained from the implementation on real sensor nodes (the chosen platform are TelosB motes) of the three aforementioned authentication schemes and jointly evaluates the ESAWN

protocol performance. Finally, chapter 6 draws the conclusions on the presented work and considers possible future developments.

Chapter 2

Data aggregation and the need for security

Wireless sensor networks are typically composed of a large number of sensor nodes, each one collecting application specific information from the environment. This information usually needs to be transferred to a central gathering point via multihop communication. There it can be analysed and processed for further use. Because communication is very expensive in terms of energy consumption, it is desirable to minimise it in order to enhance network lifetime, which is indeed a fundamental parameter in a WSN.
Data aggregation moves exactly towards this direction. The basic idea is to jointly process distinct sensor readings by means of an aggregation function and/or of data compression, in order to reduce the amount of information that has to be transmitted towards the next hop. A hierarchical network structure has to be defined as well, so that routing paths are established and aggregation points are distributed on the way to the gathering point, i.e., the base station (BS, from now on).

While addressing the problem of reducing communication, however, data aggregation poses severe security problems, related both to the underlying network protocols (routing through compromised nodes) and to the need of in-network data processing (which usually does not allow to simultaneously achieve end-to-end confidentiality, authenticity, etc., as data needs to be processed at intermediate nodes).

In this chapter, data aggregation and security requirements are explored together with the difficulties in making them coexist in a secure data aggregation protocol.

2.1 Data aggregation

Note: in this section an extensive reference to the exhaustive surveying work presented in [13] is made.

As widely anticipated, data aggregation is a fundamental feature for a WSN, since it allows a considerable reduction of the communication complexity in exchange for a reasonable computational overhead at sensor nodes. In a WSN, in fact, local computations usually consume much less energy than data transmissions; the goal of data aggregation is precisely that of finding a suitable trade off between communication and computational complexity, in order to limit the overall energy consumption.

According to [13], the *in-network aggregation* process is defined as follows:

Definition 1 (In-network aggregation)**.** *In-network aggregation is the global process of gathering and routing information through a multihop network, processing data at intermediate nodes with the objective of reducing resource consumption (in particular, energy), thereby increasing network lifetime.*

Moreover, two different approaches exist, namely:

- *In-network aggregation with size reduction* - the data received by an aggregator node from its children is processed in such a way to reduce its size; for example, if an aggregator node receives two temperature measurements from its two children, it could compute their average and forward the result to the upper-level aggregator, thus sending a single packet instead of two.
- *In-network aggregation without size reduction* - the aggregator node merges packets coming from different sources into a single packet, without performing data processing. This allows a reduction of the overhead, while data size is not reduced.

According to [13], in-network aggregation techniques are based upon three fundamental components: suitable *networking protocols*, effective *aggregation functions* and efficient ways of *representing data*. A brief description of each component is provided in the remainder of this section, while a deeper analysis can be found in [13].

2.1.1 Routing protocols

In order to perform data aggregation, a specific routing protocol has to be designed. In fact, while traditional routing protocols aim at finding the shortest path towards the destination (according to a specified metric), a routing protocol suitable for data aggregation is expected to choose paths that ensure optimal data aggregation of semantically similar data, with the final goal of minimising energy consumption.
This forwarding paradigm is referred to as *data-centric routing*. Using such an approach implies that the metrics used for defining routing paths must consider variables like the position of most suitable aggregation points, data type, overall energy efficiency, etc.

Another specific issue that needs to be taken into account while designing a data aggregation protocol is the *timing strategy*, i.e. some form of synchronisation

among nodes. According to the specific application, one of the following policies could be chosen:

- *periodic simple aggregation* - aggregation operations are triggered by a prefixed timer; therefore, when the timer is triggered, each aggregator node aggregates data received in the current time-slot and forwards the result to the designated node(s).
- *periodic per-hop aggregation* - aggregation operations are performed as soon as a node hears from all of its children. A time-out is used to prevent protocol freezing in case some children's packets are lost.
- *periodic per-hop adjusted aggregation* - the approach is identical to that of *periodic per-hop aggregation* except that time-outs are adjusted for each node, depending on its position in the network structure.

Finally, a routing protocol is characterised by the underlying network structure. In the following the main approaches suitable for in-network aggregation are presented. A detailed description of the main routing protocols based on the considered approaches can be found in [13].

Tree-based approaches

Tree structures are the most widely used in data aggregation protocols, because of their simplicity and efficiency. The topology is that of a spanning tree rooted at the Base Station (BS), i.e. the gathering point demanded to perform processing for further data use. A possible tree topology is represented in figure 2.1(a).

Typically, the BS queries for a specific subset of sensor readings; the request is spread along the tree. Successively, sensor nodes answer by performing data aggregation along the aggregation tree.

Tree-topologies have a major drawback: if a packet is lost at some level of the aggregation tree, all information related to the corresponding sub-tree is lost as well. This is a direct consequence of the hierarchical structure imposed by the tree-based approach and makes it unsuitable for highly dynamic networks, where link/node failures are frequent. On the other hand, however, as the authors of [13] claim, "these approaches are particularly suitable for designing optimal aggregation functions and performing efficient energy management", as confirmed by several studies [14, 15, 16]. Examples of tree-based protocols can be found in [17, 18, 19].

Cluster-based approaches

Cluster-based schemes [20, 21, 22, 23] are very similar to tree-based ones, as they are founded on a hierarchical organisation of the network. According to this approach, though, nodes are subdivided into clusters and for each cluster a *cluster-head* is defined. Cluster-heads are responsible for data aggregation of their

cluster and have to forward the result to the BS. An example of cluster-based topology is represented in figure 2.1(b).

Advantages and drawbacks of this type of scheme are essentially the same of tree-based solutions.

Multipath approaches

A different approach, whose aim is to overcome the major drawback of classical hierarchical approaches (i.e. their unsuitability for networks with a high rate of link failures), is pursued by multipath schemes. The basic idea is that of sending data over multiple paths, exploiting the broadcast nature of the wireless medium. This approach enhances the protocol resiliency against packet loss, while introducing a significant overhead. Representative of this class are the protocols proposed in [24, 25, 26].

Hybrid approaches

Finally, an hybrid approach between tree/cluster-based and multipath schemes can be pursued. In particular, the data aggregation structure can be adjusted according to specific network conditions and to some target performance figures. A proposal that applies this paradigm is described in [27].

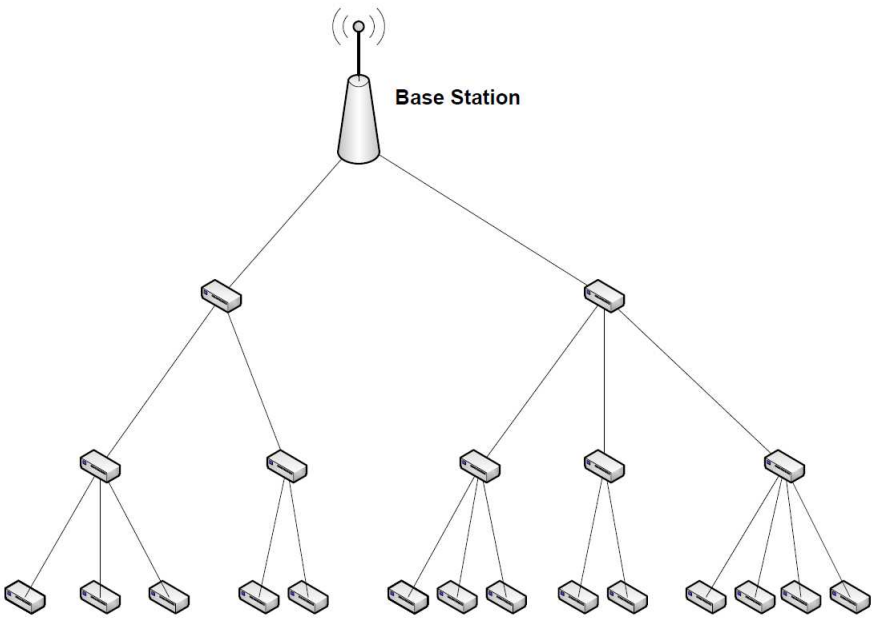
2.1.2 Aggregation functions

In order to perform in-network aggregation, an effective aggregation function is needed, so that data coming from different sources can be combined at the aggregator node. By "effective" we mean that these functions need to take into account the scarce resource availability of sensor nodes and must cope with their severe limitations.

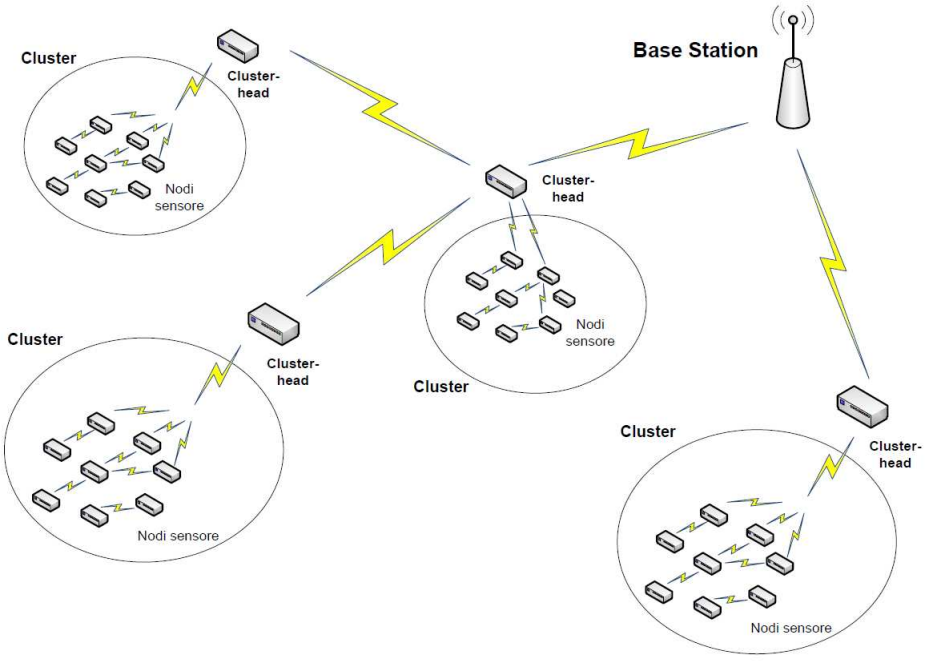
Many types of aggregation functions exist, depending on the specific sensor application. Typical examples are functions such as mean, median, quantile, max and min. As a deeper analysis is not functional to this thesis, only a coarse classification is provided in the remainder of this section. Specifically, we can distinguish:

- *Lossy Vs. Lossless aggregation functions.*
  Data can be aggregated either by losing some of the original information content (lossy aggregation) or by preserving all of it (lossless aggregation). Lossless aggregation ensures a complete recovery of all single sensor readings at the BS, while the lossy approach does not allow a perfect reconstruction.

- *Duplicate sensitive Vs. Duplicate insensitive aggregation functions.*
  Whether an aggregator node receives multiple sensor data with the same content, he could take into account the redundant information or simply discard it. In the first case, the aggregation is duplicate sensitive, while in the second one it is not. Major representatives of the two approaches



(a) Tree-based.



(b) Cluster-based.

Figure 2.1: Examples of network topologies.

are the *mean* and the *median* functions, which are respectively duplicate sensitive and duplicate insensitive aggregation functions.

2.1.3 Data representation

Finally, a proper data aggregation protocol should use an efficient way of representing data, in order to comply with the limited storage capabilities of sensor nodes and to reduce the number of bits to be transmitted over the wireless medium (thus improving network lifetime).
Ideally, data representation should be adjusted according to the specific type of data to be processed and to application requirements. An interesting proposal that moves towards this direction is presented in [28]: it exploits source coding techniques to optimally represent and compress data, based on information about its correlation.

As a conclusion to this section, we highlight that the three previously described components should be jointly designed and implemented in order to ensure optimal performance even though they were presented separately. Furthermore, they should be tightly coupled with the requirements of the specific application of the considered WSN.

2.2 Security in a Wireless Sensor Network

As widely anticipated in Chapter 1, security is a paramount requirement in Wireless Sensor Networks, especially for certain applications. Furthermore WSNs have many characteristics that make them extremely vulnerable to malicious attacks, especially when they are deployed in hostile environments.

Firstly, the wireless medium is, by definition, open to everyone; an attacker can easily gain access to transmitted packets and take part to the communication, simply by tuning his radio device on the correct frequency. He could also jam the transmission, preventing the network from working correctly.

Secondly, the elements that compose a WSN are cheap, resource-constrained sensor nodes, with limited computational power, reduced storage capabilities and scarce energy resources. Therefore it is not feasible to implement traditional strong security algorithms, as they would be too demanding for sensor nodes; on the other hand, though, a weakened security protocol cannot face a strong adversary. Hence a suitable trade off between performances and security is needed, i.e. efficient algorithms should ensure adequate protection against malicious attacks.

Thirdly, a WSN can scale up to thousands of sensor nodes and, thereby, security protocols should be flexible and scalable as well. e Last but not least, we remark that most WSNs protocols, including data aggregation ones, usually do not consider potential security threats at the design stage. On the contrary, security mechanisms are added later as stand-alone modules. This is an outstanding limitation, as security should be taken into account from the very beginning

stages of WSNs protocol design.

To summarise, security is a critical issue for a WSN and poses several challenges while designing a protocol. In the following sections, security requirements, attack types and an adversarial model are presented. As a conclusion, the problem of designing a secure data aggregation protocol is introduced.

2.2.1 Security Requirements

Several properties could be demanded to a secure protocol, according to the specific application. The major security requirements ones are listed below.

Data confidentiality

Data confidentiality ensures that secrecy of sensed data is never disclosed to unauthorised parties. This means that the information transmitted by each sensor node is readable only by those nodes to which that information is addressed.

A brief introduction to cryptographic primitives for data confidentiality.
Confidentiality can be achieved through cryptographic algorithms, which can be distinguished mainly into two categories: symmetric-key crypto (SKC) algorithms and public-key crypto (PKC) algorithms. In the following, a brief introduction to these two categories of crypto algorithms is given; the main features are described, while a detailed description is beyond the purpose of this thesis.

In first place we define as:

- **encryption process** an algorithm that, given an encryption key, transforms the *plaintext* data received as input into a bit-stream, called *ciphertext* that is unreadable by all the parties who do not know the encryption key.
- **decryption process** the inverse of the encryption process, that is the algorithm that, given a decryption key (matched with the encryption one), transforms the received encrypted bit-stream (ciphertext) into a plaintext message.

SKC algorithms assume that the two nodes involved in the transmission , e.g. S_i and S_j, share a common secret key, e.g. K_{ij}. This key enables both the encryption and the decryption process, namely:

- *SKC Encryption* - $E : K \times P \mapsto C$, $c = E(K_{ij}, p)$,
- *SKC Decryption* - $D : K \times C \mapsto P$, $p = D(K_{ij}, c)$,

being K the set of all the possible keys, P the set of all possible plaintexts, C the set of possible ciphertexts, $p \in P$ the chosen plaintext and $c \in C$ the corresponding ciphertext.
Widely used SKC algorithms are AES [3], RC5 [4], DES [6].

Conversely, PKC algorithms use a key pair (K_{publ},K_{priv}), i.e. respectively the public and the private key. As the names suggest, K_{publ} is publicly known, while K_{priv} is known only to the authorized addressee of the message. More specifically, using the public key, every node can encrypt data that can be decrypted only knowing the corresponding private secret key:

- *PKC Encryption* - $E : K \times P \mapsto C$, $c = E(K_{publ}, p)$,
- *PKC Decryption* - $D : K \times C \mapsto P$, $p = D(K_{priv}, c)$,

where K, P, C, p and c have the same meaning as before.
Many PKC schemes are based on the presumed hardness of a mathematical problems, such as factoring the product of two large primes or computing discrete logarithms. Assuming that these problems are not feasible to be solved, the security of these schemes is proven. Well regarded PKC algorithms are RSA [7], the ElGamal algorithm [9], the Diffie-Hellman key exchange protocol [10] and various Elliptic Curve (ECC) techniques [8]. It should be noticed that ECC algorithms grant a considerable improvement in terms of complexity reduction over traditional PKC algorithms: in order to achieve the a comparable level of security of a PKC system based on RSA with a public key of size 1024 bits, a PKC system based on ECC just needs a public key of 160 bits.

PKC-based crypto-systems have a considerable advantage over SKC schemes in the key management process. SKC schemes require each pair of the communication end-points to be equipped with the same secret key; therefore, if secure communication needs to be established between every couple of end-points in a network of n nodes, the overall amount of pairwise-key required would be equal to $\frac{n(n-1)}{2}$. Conversely, PKC schemes would need just n public keys, thus leading to a considerably lower overhead. On the other hand, SKC algorithms are much less computationally intensive with respect to PKC ones; this advantage is typically from 2 to 3 orders of magnitudes. This means, of course, that PKC is much more demanding in terms of resources and energy consumption: that is the main reason why PKC is hardly feasible in Wireless Sensor Networks, at least when the designer is dealing with low-end motes, such as MicaZ and TelosB.

Authentication

Since sensors use a shared wireless medium to communicate, authentication mechanisms are needed to detect maliciously injected or spoofed packets, in order to ensure that data is actually provided by the expected party. The lack of authenticated communication could lead to several problems, such as node impersonation, Sybil attack (please refer to §2.2.2 for further details) and so on.
Referring to data aggregation, in the absence of authentication mechanisms an attacker could alter the aggregation result providing the aggregator node with fake data, claiming to be a legal sensor node while actually he is not; or, even worse, the aggregator node itself could cheat on the aggregation, providing the upper level aggregator or the sink with forged aggregated data. There are several mechanisms that allow message and source authentication: the most used are the so-called Message Authentication Codes (MACs). Widely used MACs are the CBCMAC (Cipher Block Chaining MAC) and the HMAC (Hashed MAC), which are based respectively on block ciphers (such as AES, RC5, Skipjack, etc) and on hash functions (such as MD5, SHA-1, RIPEMD, etc.); these schemes will be analysed in detail in chapter 4.

Data Integrity

This property ensures that the content of a message has not been altered, either maliciously or accidentally, during the transmission process. Data integrity can be provided by a simple mechanism, such as a CRC (Cyclic Redundancy Code), but, usually, a MAC is used to fulfil simultaneously authentication and integrity.

Data freshness

In order to protect the network against replay attacks (see section §2.2.2 in the current chapter), messages with the same information content (i.e. measured values) should look different every time they are transmitted, so that the attacker cannot exploit the confidentiality, the authenticity or the integrity of a message at time t for successive transmissions at time $t+\delta$. Data freshness can be provided by simply embedding a counter or a pseudo-random value (shared by the sender and the receiver) in the packet payload before applying the cryptographic primitive used to provide confidentiality or authentication.

Availability

If this property holds, the network is able to ensure the availability of resources and measured data even in the presence of some compromised nodes. It is nearly infeasible to provide 100% availability of radio resources in a WSN, since sensor nodes' transmissions are low-power and can be easily jammed by a powerful attacker. Assuming that the attacker is not interested in jamming all communications in the network, in a secure data aggregation protocol it is possible to ensure a reasonable level of security with respect to availability, using mechanisms such as:

- *Self-healing* - diagnose and react to the attacker's activities and take corrective actions on defined policies to recover the network or a node.

- *Aggregator rotation* - rotate the aggregation duties between honest nodes in order to reduce the impact of an attacker compromising an aggregator node (the role of aggregator would be only temporary) and balance the energy consumption.

Non-repudiation

This property ensures that a transferred packet has been sent (received) by the party claiming to have sent (received) the packet. In secure aggregation schemes, once the aggregator sends the aggregation results, it should not be able to deny the sent data. This gives to the base station the opportunity to determine what caused the changes in the aggregation results.

2.2.2 Attack Types

Preface: An exhaustive classification of possible attacks in a WSN is beyond the purpose of this thesis and, therefore, the reader interested in a comprehensive overview should refer to some specific work, such as the one presented in [11]. In the following, the focus is mainly on attacks related to data aggregation.

Denial of Service (DoS) attack

Within this class, a wide range of possible attacks is included. DoS attacks could be performed by means of radio jamming, but also by simply deviating the behaviour of a sensor node from its specified policy, in such a way to prevent the system to work properly. In a data aggregation protocol, an example of DoS attack can be that of an aggregator which refuses to aggregate packets received from its children, thus preventing data from travelling towards the next hop.

Node compromise

Compromising a node allows an attacker to gain access to all of its contents (such as cryptographic keys) and to manage the communication with its neighbours. A compromised node can alter sensed data at its own will, without being detected as malicious by authenticity or decryption checks.

Sybil attack

By means of a Sybil attack, the adversary is able to impersonate more than one identity within the same network; in this way, he can alter aggregation protocols in several ways. For example, he could create multiple identities to generate additional votes in the aggregator's election phase (assuming that aggregators are not chosen in advance), allowing the election of a compromised node. Otherwise, the attacker could decide to contribute several times to the aggregation process, exploiting his different identities.

Selective forwarding attack

Using selective forwarding, a node drops some of the incoming messages , thus preventing the network to function properly (DoS attack) or simply by altering the aggregation result.

Replay attack

An attacker could record some traffic from the network, without necessarily understanding its content (i.e. without violating its confidentiality), and replay sniffed packets later on, exploiting their authenticity and integrity to mislead the aggregator in order to, for example, alter the aggregation result.

Stealthy attack

The adversary aims to inject false data into the network without revealing its existence. In a data aggregation scenario, the injected false data leads to a false aggregation result. A compromised node can report significantly biased or fake values and perform a Sybil attack to affect the aggregation result.

2.2.3 Adversarial Model

An adversary is characterized by several features, which enable him or not to perform certain types of attacks. In the following the main differences in the attacker's capabilities are described.

Adversary type

A first classification of an attacker could be done basing on which security requirements he aims to threaten:

- **Passive attacker** - the threatened security requirement is confidentiality. The adversary, in fact, simply takes advantage of the nature of the wireless medium and eavesdrop the traffic in order to extract sensitive information about sensed data, without really interacting with the network entities. Of course, a strong encryption algorithm can ensure security with respect to this type of attacker, with a cost in terms of energy consumption.
- **Active attacker** - the adversary aims to threaten the integrity and the authenticity properties in the network. In order to do it, he interacts actively with the network nodes: he may inject packets, modify overheard packets and then replay them, destroy nodes, stop or delay packets from being delivered to the destination node, tamper nodes and extract their sensitive information, etc.

Network access

The network scale over which the adversary can perform his attacks is another important feature. Two main categories can be distinguished:

- **Total network access adversary** - the attacker has access to the whole WSN. If he is passive, he can overhear all the communication between nodes; if he is active, he can interact with all the components of the network, thus including sensor nodes, aggregator and base station.
- **Partial network access adversary** - the attacker can access only a portion of the WSN, hence limiting the potential of his attacks. If the attacker is passive, he can eavesdrop only some of traffic in the network; if he is active, he can interact only with a subset of nodes in the WSN.

Adversary (coarse) classification

Summarising, the authors of [12] propose an approximate classification of attackers into three categories:

- **Strong adversary** - Active adversary with total network access.
- **Medium adversary** - Active adversary with partial network access.
- **Light adversary** - Passive adversary with partial network access.

2.3 Security Vs. Data Aggregation

In the previous sections, data aggregation and security in wireless sensor networks were introduced.
Data aggregation is necessary in order to improve the overall network lifetime, that is indeed a fundamental design parameter for a WSN. At the same time, though, a careful designer should consider security threats that could arise in such a network, which is easily accessible to the attacker and must face serious resource constraints.

Implementing data aggregation and security in a single protocol is a great challenge, since, somehow, they have contrasting goals. Data aggregation aims to minimise energy consumption, by means of reducing the amount of transmitted data; on the contrary, security adds computational and communication overhead, in order to ensure a set of properties such as confidentiality, authenticity, etc.

Two major difficulties in the creation of a secure data aggregation protocol can be distinguished:

- data aggregation requires to process information at the aggregator node; therefore, ensuring end-to-end verification of security requirements (which, in fact, would be desirable, as usually no sensor node can be trusted to be honest) is extremely challenging. Therefore, a relaxation of security requirements and particular assumptions on the attacker or on the network capabilities could be necessary in order to make the aggregation process feasible and efficient even in the presence of security mechanisms.

- sensor nodes are severely limited in terms of computational, storage and energy resources; each WSN protocol should be designed with this limitation in mind. When it comes to security, it is even more difficult to comply with these constraints, as traditional strong security mechanisms are computationally too intensive for most sensor nodes.

In the following chapter, some proposals for secure data aggregation protocols are presented.

Chapter 3

Secure Data Aggregation - Review of the State of the Art

The need for secure data aggregation protocols has been widely introduced in the previous chapter. In the following, a review of the current state of art in secure data aggregation protocols, to the best of the author's knowledge, is presented, according to the classification proposed in §3.1. After a detailed description of the major proposals, a structured comparison is performed in order to underline achievements and weaknesses of each protocol, with a focus on future developments.

3.1 Classification

Secure data aggregation protocols can be distinguished into two main categories, namely *hop-by-hop* and *end-to-end* protocols.

In the first class, security primitives are applied in a hop-by-hop fashion, i.e. security requirements (confidentiality, authenticity, integrity, etc.) are verified at each step. This kind of protocols allows a simpler implementation of aggregation functions, and does not impose any bound on their nature (sum, mean, max, min, etc.); moreover, it does not require specific cryptographic algorithms[1]. On the other hand, hop-by-hop protocols suffer from a considerable inefficiency, as they require aggregator nodes to deal with cryptographic primitives, especially when confidentiality is required. In general, however, hop-by-hop protocols are much more easily implementable in a resource-scarce WSN, as they require fewer computational (and, hence, energy) resources with respect to end-to-end protocols.

The second class, i.e., end-to-end protocols, is based on particular cryptographic algorithms, known as *Privacy Homorphisms*, which allow direct computations (usually addition and/or multiplication) on encrypted data. The main advantage of these protocols is that they ensure end-to-end confidentiality and they do not require cryptographic operations at the aggregator node. Conversely, because of the use of privacy homomorphisms, they need a significantly increased

[1] therefore, traditional cryptographic algorithms, both SKC and PKC, can be used.

amount of energy for the encryption phase with respect to traditional cryptographic algorithms and, with privacy homomorphisms available nowadays, only addition-based and multiplication-based aggregation operations are feasible. Further details are given in §3.3.

In the following, some of the major proposals for secure data aggregation protocols are presented; this review does not aim to be comprehensive of all protocols proposed by the scientific community, but rather to give an extensive overview on possible approaches to secure data aggregation by selecting some representative candidates.

3.2 Hop-by-hop protocols

3.2.1 SIA, Secure Information Aggregation [36]

SIA [36] is one of the first protocols for performing secure data aggregation proposed by the scientific community. SIA addresses the problem of "how to enable secure information aggregation, such that the user accepts the data with high probability if the aggregated result is within a desired bound, but that the user detects cheating with high probability and rejects the result if it is outside of the bound" [36]. The authors consider the case where compromised sensor nodes or aggregators can deviate their behaviour arbitrarily with respect to the predefined policy; the aim of their protocol is to avoid with the highest possible probability that the user accepts an aggregate that is deviating consistently from the correct value.
SIA assumes that each sensor has a unique ID and shares with the base station a pair of secret keys; these keys enable the sensor to authenticate messages and, if confidentiality is required, to encrypt them[2].
The paradigm of SIA is the so-called *aggregate-commit-prove* approach: the aggregators do not only have to perform aggregation tasks, but also have to prove that they perform these tasks correctly. Specifically, to prevent the aggregators from cheating, the protocol uses cryptography-based commitment techniques and constructs efficient *random sampling* mechanisms and *interactive proofs*; these mechanisms enable the user to verify that the answer given by the aggregators is a good approximation of the true value even when the aggregators and/or a fraction of the sensor nodes may be corrupted.
As the name suggests, the *aggregate-commit-prove* algorithm consists of three phases, namely:

1. AGGREGATE - the aggregator gathers data from the sensor nodes and locally computes the corresponding aggregated value.

[2]please note that the authors do not suggest any cryptographic algorithm to be used for the sake of confidentiality, as confidentiality is not the main target of the protocol, but it is rather a collateral property that may be implemented. On the other hand, they suggest the use of HMAC [38] for authentication.

2. COMMIT - the aggregator commits to the collected data; this commitment is a proof for the base station that the aggregator is using the sensor readings submitted by sensor nodes. The authors of SIA suggest the use of a *Merkle hash tree* [37] in order to implement efficiently this mechanism. In figure 3.1, it is shown how this mechanism works.

3. PROVE - the aggregator sends to the BS the result of the aggregation process and the commitment to the sensor readings, and then proves to the BS that the reported results are correct using *interactive proof* protocols, which usually consist of two steps:

   (a) the BS checks whether the committed data is a good representation of the true data values in the sensor network.

   (b) the BS checks whether the aggregator is cheating, that is if the aggregation result does not comply with the correct result aggregated from the committed data values.

The authors then propose some protocols, that they claim to be secure, for computing the median and the mean, for extracting the maximum and the minimum and finally for counting distinct elements in the reported sensor readings. It is shown that these protocols have sub-linear complexity.

SIA is completed by a mechanism which grants a property named *Forward Secure Authentication* (FSA). Specifically, FSA ensures that, if the BS queries some previously sensed data, the attacker is not able to alter the data collected in the past before the sensor was compromised. This mechanism assumes that each sensor is *loosely-time synchronized* with the BS and that time is divided into constant time intervals.
It should be noticed, by the way, that FSA is not easily implementable in a WSN, as sensor nodes are given a small amount of memory to store previously sensed data and have limited computational capabilities (the mechanism is based on the iterative use of hash functions).

3.2.2 SecureDAV, Secure Data Aggregation and Verification [39]

In many aspects, SecureDAV is a protocol similar to SIA: both of them aim to ensure protection against stealthy attacks (i.e. attacks in which the aggregator tries somehow to alter the aggregation result without being discovered by other nodes), use a verification mechanism to find out possible cheating on the aggregation result and have a *cluster-based* approach.
Nevertheless, SecureDAV introduces some important new features. First of all, SecureDAV uses Elliptic Curve Cryptography ([8]), which enables the use of PKC in a WSN, maintaining the computational cost at a reasonable level. In particular, ECC enables sensor nodes to efficiently generate a signature, while the verification of this signature (which is computationally much more complex) is devolved

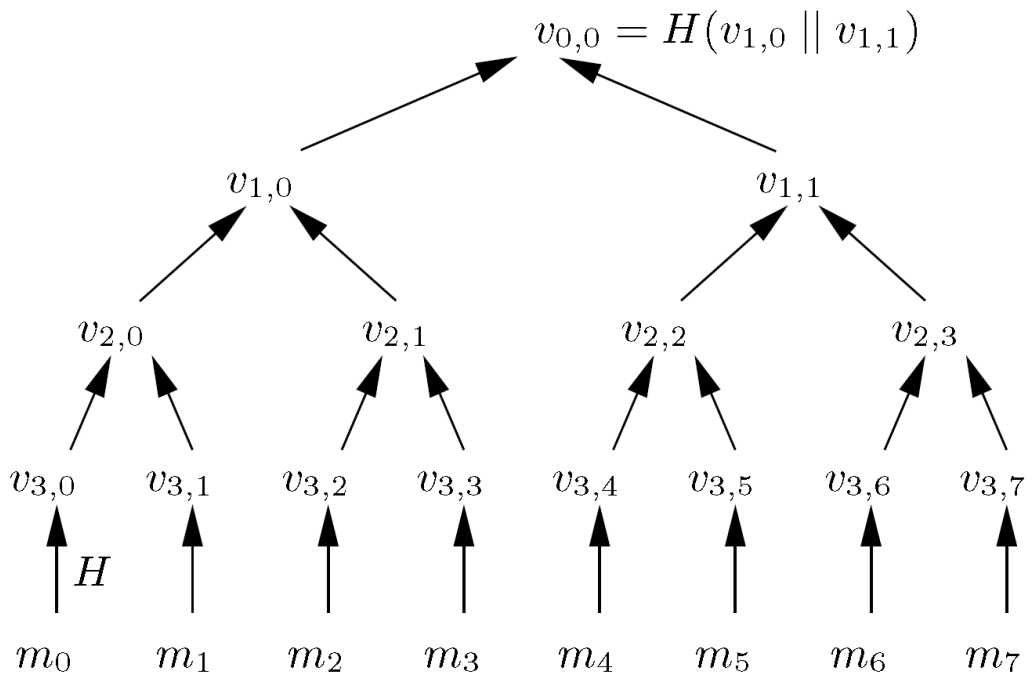


Figure 3.1: **Merkle hash tree used to commit to a set of values.**
The aggregator constructs the Merkle hash tree over the sensor measurements $m_0, ..., m_7$. To lower the size of verification information, the aggregator first hashes the measurements with a cryptographic hash function, e.g. $v_{3,0} = H(m_0)$, assuming that the size of the hash is smaller than the size of the data. To construct the Merkle hash tree, each internal value of the Merkle hash tree is derived from its two child nodes: $v_{i,j} = H(v_{i+1,2j}||v_{i+1,2j+1})$ (where $||$ denotes the concatenation operation). The Merkle hash tree is a commitment to all the leaf nodes and, given the authentic root node $v_{0,0}$, a verifier can authenticate any leaf value by verifying that the leaf value is used to derive the root node. For example, to authenticate the measurement m_5, the aggregator sends m_5 along with $v_{3,4}, v_{2,3}, v_{1,0}$ and m_5 is authentic if the following equality holds:

$$v_{0,0} = H(v_{1,0}||H(H(v_{3,4}||H(m_5))||v_{2,3}.$$

to the BS (which does not suffer from the strict limitations of the sensor nodes). The solutions proposed in [39] is composed mainly by two blocks: the *Cluster Key Establishment* (CKE) protocol and the *SecureDAV* protocol itself.

The CKE protocol generates a secret cluster key for each cluster[3]. Each node has only a share of the secret cluster key and the cluster key is hidden from each node. The cluster key is used to generate partial signatures using ECDSA (Elliptic Curve Digital Signature Algorithm) [42]. The cluster-head then gathers all the

[3]it is assumed that the corresponding public key is known by all the nodes in the cluster as well as by the BS

partial signatures of its cluster-nodes, combines them into a single signature and sends it, together with the aggregated data, to the BS. The BS, who knows the corresponding public key, can finally check the authenticity. Please note that, since the attacker does not know the complete cluster-key, he cannot generate the whole signature for the aggregated data. Furthermore, the authors of [39] prove that compromising less than t nodes in a cluster (where $t < n/2$, with n being the total number of nodes in the cluster) does not enable the adversary to form a full signature.
Once CKE has been successfully executed, SecureDAV uses its output in order to perform secure data aggregation. In [39] the implementation of SecureDAV is provided when the required aggregation function is the mean between the sensed readings. The process can be summarised into the following steps:

1. in each cluster, the cluster-head aggregates the data reported by sensor nodes and computes the mean; it then broadcasts the result to all cluster members.

2. every cluster node compares its reading with the value reported by the cluster-head. If the difference between the two values is below a fixed threshold (that is if the aggregation result is "reliable" according some predefined criteria), the node creates a partial signature of the aggregated data using its part of the secret key and sends it to the cluster-head.

3. the cluster-head combines all the partial signatures into the complete signature and sends it, together with the computed mean, to the BS, which finally verifies it.

SecureDAV, at last, has an *integrity check* mechanism based on a *Merkle hash tree*, definitely similar to the one used in [36].

3.2.3 SDAP, Secure Data Aggregation Protocol [43]

The work presented in [43] is quite complete and well-articulated. Besides inserting their work in a structured framework, the authors propose diverse solutions for collateral problems involved in secure aggregation, such as the tree construction (the network topology is assumed to be *tree-based*) and the key distribution. Moreover, after an accurate description of the proposed protocols and of their theoretical background, an extensive performance analysis[4] and simulations are conducted.

While designing SDAP, its authors observed, first of all, that compromising a node closer to the root of the tree has a more significant impact on the final aggregation result with respect to compromising a lower level node; the aggregation result of a higher level node depends, in fact, on a bigger number of lower-level sensor nodes. SDAP tries to limit this problem by using a *divide-and-conquer*

[4] in terms of computational, storage and communication overhead.

approach[5]. Specifically, SDAP uses a probabilistic grouping method to dynamically partition the network tree into multiple logical sub-trees of comparable size. Hop-by-hop aggregation is then performed separately in each group and results are then encrypted (with a different key for each sub-tree) and transmitted to the BS. Through this arrangement, in each group less nodes are positioned below high-level nodes, thus limiting the potential impact of a high-level node compromise. In figure 3.2 an example of tree-partitioning is shown.

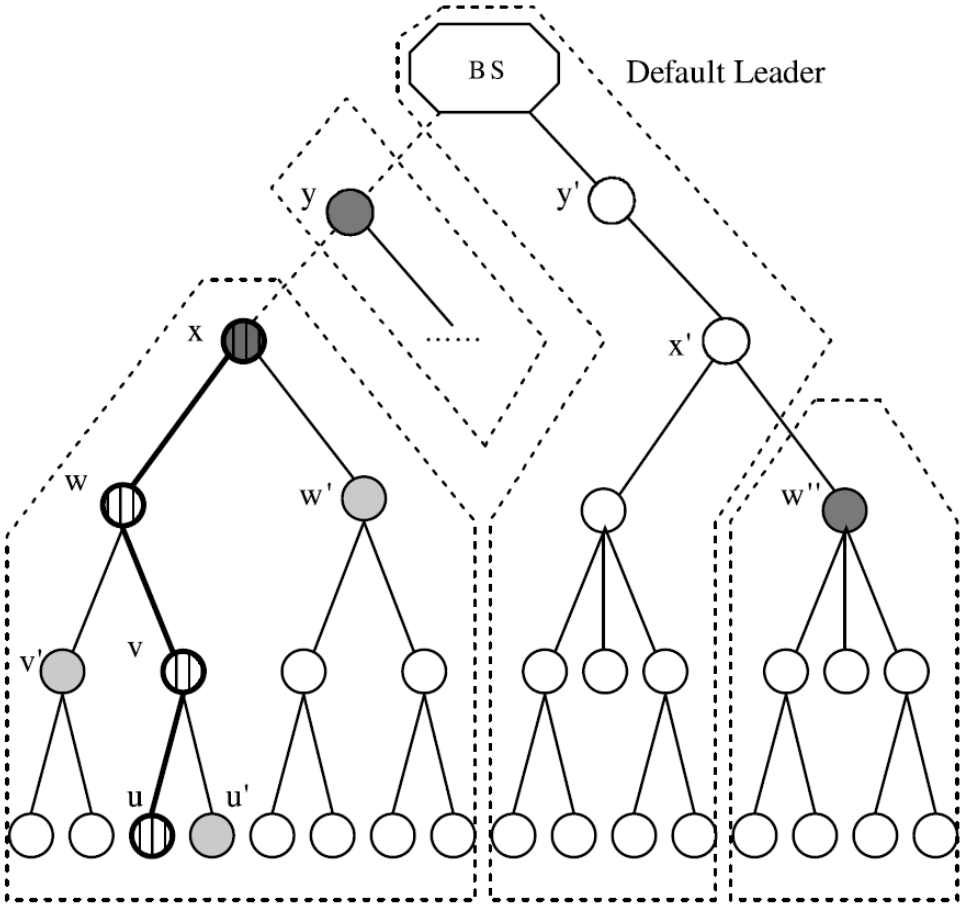


Figure 3.2: An example of grouped aggregation tree. The nodes x, y and w with the dark-gray color are leader nodes and the nodes included in the dashed line are corresponding group members. The BS is the default root.

Furthermore, the probabilistic approach, ensures another important advantage on the attacker; namely, the latter cannot compromise nodes *selectively* according to his optimal attack strategy (e.g. by positioning a compromised node in each sub-tree or gathering all compromised nodes in a single group). Each node, in fact, is not preassigned to a specific group and could be a group-leader

[5]the aggregation problem is decomposed into many sub-problems , which solutions are then reassembled at the BS.

(i.e. aggregator) or he could not: it depends on dynamic and probabilistic partitioning performed by the SDAP protocol. As mentioned earlier, SDAP performs an hop-by-hop aggregation in each logical subgroup and generates an aggregate for each of them; it is then up to the BS to verify the validity of the provided results and, at last, compute the final aggregation result. The BS mainly uses three mechanisms in order to detect potential attacks:

1. *Authenticity check* - the BS checks the received aggregate MAC, verifying that it was actually provided by a legitimate group leader.

2. *Bivariate multiple-outlier algorithm* - using an extended version of the Grubbs's test [44], the BS detects potential outliers in the data aggregated by each group; each suspected group will be required to prove the commitment that it has previously submitted to the BS.

3. *Commit-and-attest* - each group commits to its aggregate; the idea is similar to that presented in SIA and ensures that no group can repudiate aggregates that it has actually sent to the BS. Groups under suspicion (due to a failed authenticity check or detection of a potential outlier by the extended Grubb's test) are required to prove their commitment. In order to implement this mechanism, the authors of [43] use an approach similar to the *Merkle hash tree*, adapted, though, to SDAP specifications.

The BS discards aggregates provided by groups that are not able to prove their commitment; the final aggregation result is in fact computed on those aggregates that successfully passed the three tests described above.

Through theoretical analysis and several simulations, the authors of [43] claim that their protocol is effective against attacks that aim to alter the aggregation result and efficient with respect to the reasonable overhead that it entails: the energy used for the aggregation and attestation processes is nearly the same of that required for transmitting 6 bytes, while the storage overhead is limited to few kilobytes.

3.2.4 ESPDA, Energy-efficient Secure Pattern-based Data Aggregation [45]

A different approach is pursued in [45]. The authors propose a secure data aggregation protocol in a cluster-based WSN, but with a key difference with respect to the solutions presented so far: this protocol does not perform aggregation in strict meaning but rather selects only non-redundant data reported by sensor nodes to the cluster head (what this means exactly will be explained later), which then simply forwards it to the BS. Therefore, there is no aggregation function in ESPDA, but just an optimised redundancy reduction in each cluster head data. Specifically, ESPDA uses *pattern codes* in order to perform aggregation. As explained by its authors, "pattern codes are basically representative data items that

are extracted from the actual data in such a way that every pattern code has certain characteristics of the corresponding actual value". Pattern codes extraction may vary depending on the type of considered data[6]. Whether a single sensor node dispose of multiple sensing units (e.g. temperature, humidity, pressure, etc.), the pattern code corresponding to a given set of measurements is composed by the union of different pattern codes generated by each sensing unit (please refer to §2.2 in [45] for further details).
It is important to underline that, in the pattern codes extraction process, a certain approximation (whose entity depends on the precision of the pattern code generation) is assumed. This means, for example, that if two distinct sensor nodes measure two slightly different temperatures, let's say 15°C and 17°C respectively, the corresponding pattern code may be identical if the granularity is coarser than 2°C.
ESPDA could be summarised in the following steps:

1. each sensor node sends its pattern code to the associated cluster-head.

2. the cluster-head identifies distinct patterns and asks a single node for each different pattern to send its actual data to the BS, passing through the cluster-head itself; therefore the cluster-head acts more like a *data forwarder* than like a *data aggregator*.

3. each sensor node selected at the previous step encrypts its actual sensor readings with a session key (shared only with the BS) and sends it to the BS.

According to this procedure, the cluster-head does not have to decrypt the data received by its children, since the selection process is performed on pattern codes. ESPDA bypasses the main limit of hop-by-hop aggregation protocols, though it does not allow the implementation of aggregation functions in a traditional sense.

In conclusion, some other features of ESPDA are briefly listed:

- *Blowfish* - ESPDA uses this SKC algorithm in order to provide data confidentiality; the authors claim that this algorithm is optimal in terms of power consumption and data storage, as compared to other algorithms such as AES, DES, RC5;

- *Data freshness* - ESPDA ensures this fundamental property through two expedients: the use of different cryptographic keys (obtained by XORing a pre-loaded key and a session key broadcasted by the base station) and the use of a different pattern seed at each new session;

[6] the example presented in [45] takes as actual data some images of human beings sensed by surveillance sensors, for which the key parameters for *face* and *body recognition* are taken as patterns.

- *NOVSF-BH (Non-blocking Orthogonal Variable Spreading Factor - Block Hopping)* - ESPDA uses this technique (inherited and adapted from the OSFV codes used as channelisation codes by UMTS [53]) in order to further improve protocol security and spectral efficiency, without requiring additional energy;

- *Sleep-active mode protocol* - ESPDA introduces an algorithm for coordinating activation and deactivation of sensing units on sensor nodes with overlapped sensing ranges, in order to reduce redundancy in the sensed data and save energy as well.

3.2.5 SELDA, Secure and rELiable Data Aggregation protocol [46]

Another interesting and different approach to the secure data aggregation problem is proposed in [46]. The authors start from a simple statement of fact: since compromised nodes have access to cryptographic keys that are used to secure the aggregation process, cryptographic primitives alone are not enough to provide secure aggregation. Therefore, a *web-of-trust* approach [47] is proposed. The basic idea of SELDA is that sensor nodes observe actions of their neighbouring nodes to develop trust levels (trustworthiness) for both the environment and the neighbouring nodes. Monitoring mechanisms are used to detect node availability and misbehaviours of the neighbours; an example is shown in figure 3.3.

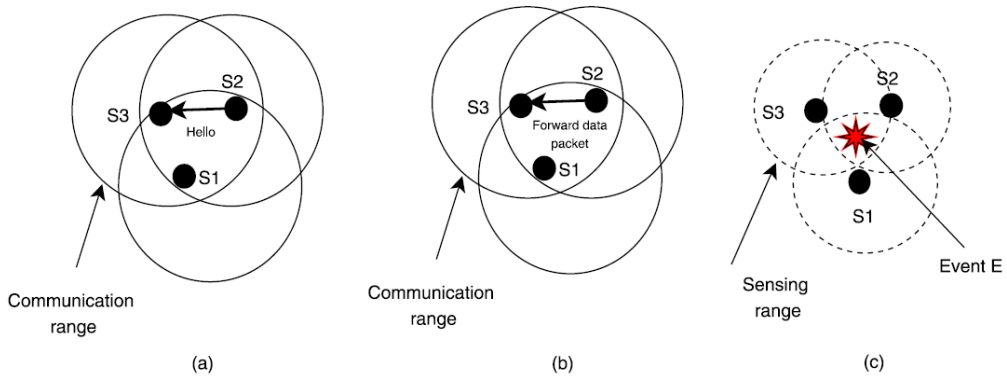


Figure 3.3: SELDA monitoring mechanisms

(a) S1 detects node availability misbehaviour of S3, if S3 does not respond S2's hello messages over a period of time .
(b) S1 detects routing misbehaviours of S3, if S3 does not want to forward S2's data packets properly.
(c) Event E is detected by S1, S2, S3 if event E is reported falsely by anyone of these nodes, as the sensing misbehaviour of an eventually compromised node is detected by the other two nodes.

This reputation scheme is based on *Beta distribution functions* ([48],[49]). Trust informations are exchanged by each node with its neighbours in order to create a web-of-trust that allows them to find secure and reliable paths towards data aggregators. Furthermore, in order to improve the reliability of the aggregated data, data aggregators weight the sensor readings they received according to the corresponding trust levels.

Finally, thanks to the monitoring mechanisms, SELDA is able to detect if a data aggregator is under DoS attack. The authors claim that SELDA involves a tolerable communication overhead, while increasing considerably the security in the aggregation process. An improvement of SELDA is achieved in [50] by introducing the concept of *functional reputations*, which ensure a more effective evaluation of sensor nodes trustworthiness.

3.2.6 Secure hop-by-hop aggregation of end-to-end concealed data [52]

For completeness, the proposal in [52] is reviewed in this paragraph. The approach is definitely interesting, though there seem to be some security issue, at least unless some adjustment is applied. The idea is that of creating a protocol that has the typical advantages of a hop-by-hop protocol, while providing end-to-end data confidentiality. The authors aim to achieve the following results:

- *Resilience* - an adversary who compromises a few nodes of an aggregated WSN must not spy-out or gain any impact on the final aggregation outcome, at least not beyond the influence of readings and results of its (eventually) compromised nodes.

- *Efficient data integrity, commitment and attestation* - the aggregation results must be verified to be the authentic union of sensor readings and intermediate results. Such verification and attestation processes should not impose significant overhead over the WSN (i.e., over aggregation communication overhead).

- *Generality* - the protocol should apply to any aggregated WSN with arbitrary tree topology. Moreover, the protocol should support expandable WSNs without any extra reconfiguration.

- *Status monitoring* - BS must determine when a sensor node becomes unreachable, by knowing and maintaining a list of all nodes that contributed in each aggregation query.

The main ingredient of the proposed protocol is a process named *data diffusion*. Data diffusion is a transformation that preserves the mathematical relationships between different values which are all bounded by a defined interval; it is defined as a function of some input value (e.g. the sensor reading) and of a secret key, shared only by the two communication end-points, namely the sensing

node and the BS.
Being D the sensing range interval, K the space of possible keys, $m^j_{S_i}$ the j-th reading of sensor node S_i and $PS : D \times K \mapsto D$ a one-way-keyed function (initialised with a secret seed $m^0_{S_i}$, shared by the sensing node and the BS) that performs the mapping $D_j = PS(K_{S_i}, D_{j-1})$, the authors of [52] define the diffusion function $F : D \times D \mapsto D$ as follows:

$$F_{K_{S_i}}(m^j_{S_i}) = PS(K_{S_i}, D_{j-1}) \odot m^j_{S_i}.$$

As said in [52], "there is no strict definition of operand $\odot$; it refers to any reversible operation that takes two inputs and produces an output that belongs to D. Examples of $\odot$ could vary between trivial operators such as the simple addition, to more complex bijective functions".
Through the diffusion process, a weak notion of confidentiality is achieved and, hence, information is somehow concealed. In fact, in order to perform the inverse-diffusion (and thus extract the actual value), the designated receiver needs to know the seed $m^0_{S_i}$ of the one-way function PS and the secret shared key K_{S_i}. Of course the security level of this kind of this primitive is not even comparable to the one ensured by a proper encryption protocol, but, in a low-powered WSN, it appears to offer a good trade off between confidentiality and energy consumption.
Since mathematical relationships are preserved by the diffusion process, aggregation can be performed hop-by-hop directly on the concealed data. The actual result of the aggregation process can be successively obtained by the inverse-diffusion at the BS.
In [52] each sensor node S_i has pair of secret symmetric keys, K_{S_i} and K'_{S_i}, shared only with the BS. Once the reading from the sensing units is terminated, the sensor performs the diffusion process of the read data twice, once for each of the secret keys; this pair of diffused values, together with a MAC, are then forwarded to the data aggregator. The latter performs aggregation on the diffused data, computes the overall MAC and sends the result to the BS.
The BS can then apply the inverse-diffusion process on received aggregated data and perform the so-called *IPET*, i.e. Identical Pair Equality Test; the authors claim that this test gives, with a computational cost of $O(1)$, an immediate feedback about the integrity of the aggregation result. IPET is therefore used, together with MACs, to build a particularly efficient *commit-and-attest* algorithm that does not require the reconstruction of the whole MAC tree (as for the Merkle hash tree [37]).

In conclusion, the core idea of [52] is quite interesting and would allow to build a simple and functional protocol, which would have the advantages of both hop-by-hop and end-to-end protocols.
Unfortunately, however, the definition of the diffusion function is rather elusive. This definition allows, for example, to chose a simple modular addition in place of the operand $\odot$; that would enable the attacker to perform a co-ordinate attack on the diffused pair, e.g. by adding the same offset to both of the diffused values, thus invalidating the security of the whole protocol, whose expedients (double

diffusion and IPET) would become definitely useless (both for the sake of security and overhead).
A further investigation on diffusion functions could solve these issues, but, as long as these points are not clarified, the implementation of this protocol remains uncertain.

3.3 End-to-end protocols

A brief introduction to Privacy Homomorphisms

A privacy homomorphism (PH) is a cryptographic transformation which allows direct computation on encrypted data. Let Q and R be two rings, with addition denoted by "+" and multiplication denoted by "$\cdot$" in both cases, and let K be the space of the keys. Moreover, let $E : K \times Q \to R$ be the *encryption* operation and $D : K \times R \to Q$ the *decryption* operation.
A privacy homomorphism can be based on symmetric keys or on asymmetric keys and can be *additive* and/or *multiplicative*. Given $a, b \in Q$ and k, k_{priv}, k_{publ}, we define a PH:

- additive PH with symmetric keys if
  $a + b = D_k(E_k(a) + E_k(b))$
- additive PH with asymmetric keys if
  $a + b = D_{k_{priv}}(E_{k_{publ}}(a) + E_{k_{publ}}(b))$
- multiplicative PH with symmetric keys if
  $a \cdot b = D_k(E_k(a) \cdot E_k(b))$
- multiplicative PH with asymmetric keys if
  $a \cdot b = D_{k_{priv}}(E_{k_{pub}}(a) \cdot E_{k_{pub}}(b))$

In general, symmetric key PHs (such as [55]) are computationally more efficient, but they suffer a serious limitation: the compromise of a single sensor node (and the consequent extraction of its keys) involves confidentiality violation of the whole network, as a symmetric key PH, in order to perform data aggregation, requires all the ciphertexts to be encrypted with the same shared secret key. On the other hand, asymmetric key PHs (such as the ones proposed in [35],[56]) address this severe security problem, but they are found to be too computationally intensive for most of low-powered sensor nodes and, hence, they are hardly usable on traditional WSNs.
At the present moment, known privacy homomorphisms involve a significant computational overhead with respect to traditional symmetric key cryptography[7]; within some context, however, they offer a good trade off between ensured confidentiality and implementation complexity.

[7] an intense research activity is pursued by the scientific community.

3.3.1 CDA, Concealed Data Aggregation [54]

CDA is one of the first proposals for an *end-to-end* secure data aggregation protocol. Its authors assume that only aggregator nodes can be compromised, while sensor nodes cannot. Furthermore, in [54] no authenticity and/or integrity mechanism is included; the authors just suggest complementary algorithms.
The basing element of CDA is an additive and multiplicative privacy homomorphism proposed by Domingo-Ferrer [10]; Wagner proved in [57] that this PH is vulnerable to chosen *plaintext attacks*[8], but, nonetheless, the authors of CDA believe that it ensures sufficient security for a WSN; attackers are assumed to be interested in violating confidentiality in a reasonably small time, while currently known attacks to the considered PH need an amount of time in the order of one day to be accomplished.

CDA enjoys all the main advantages of end-to-end protocols (no cryptographic operation required to the aggregator node, end-to-end confidentiality under certain conditions), but it suffers, at the same time, of the two severe limitations of symmetric key PHs: on one side, the attacker can violate the confidentiality of the whole network by compromising a single sensor node[9]; on the other side, the power consumption of the adopted PH is considerably higher than the one of a traditional SKC algorithm such as RC5. The authors estimate the additional overhead entailed by CDA to be up to 22% and they claim that this additional complexity is compensated by the power saving granted by CDA in the aggregation phase.

3.3.2 CDAP [58]

The authors of CDAP observe that the symmetric key PH used in [54] is vulnerable to *plaintext attacks* and, therefore, they suggest to recur to an asymmetric key PH, e.g. the one proposed in [56]. Such an homomorphism, however, is computationally too intensive for an ordinary sensor node. Hence, the authors propose the use of a group of enhanced, more powerful nodes (such as the Intel Motes [59]), called *AGGNODEs*. Unlike traditional sensor nodes, these nodes are not subject to strict computational and energetic bounds and are committed to encryption and aggregation operations. The authors state that the proportion of required AGGNODEs over the number of traditional sensor nodes is in the order of 1 out of 50.

CDAP can be summarised in the following steps:

1. AGGNODEs receive the BS's public key and the network is deployed (a *static cluster-based* topology is assumed).

2. AGGNODEs establish pairwise secret keys with neighbouring sensor nodes.

[8] attacks where the adversary can chose a *plaintext* input to the encryption algorithm and obtain the corresponding *ciphertext*.

[9] the authors of [54] assume, as stated earlier, that only aggregator nodes are tamperable, but, unfortunately, such an hypothesis is not realistic in most cases.

3. each node encrypts its sensed data using a SKC algorithm (e.g. RC5) and sends them to the designated AGGNODE.

4. AGGNODEs decrypt received data, aggregate them and, at last, encrypt them using the PH described in [56].

5. encrypted data are forwarded by sensor nodes and further aggregated by AGGNODEs on the path to the BS.

Observing that in the first data gathering phase sensor nodes use a SKC algorithm, it is quite obvious that compromising an AGGNODE invalidates the confidentiality of all the sensor readings sent by the underlying nodes and allows, anyway, fake data injection. The authors of CDAP, by the way, claim that such an attack would have only a local effect and, thus, it is an acceptable drawback.

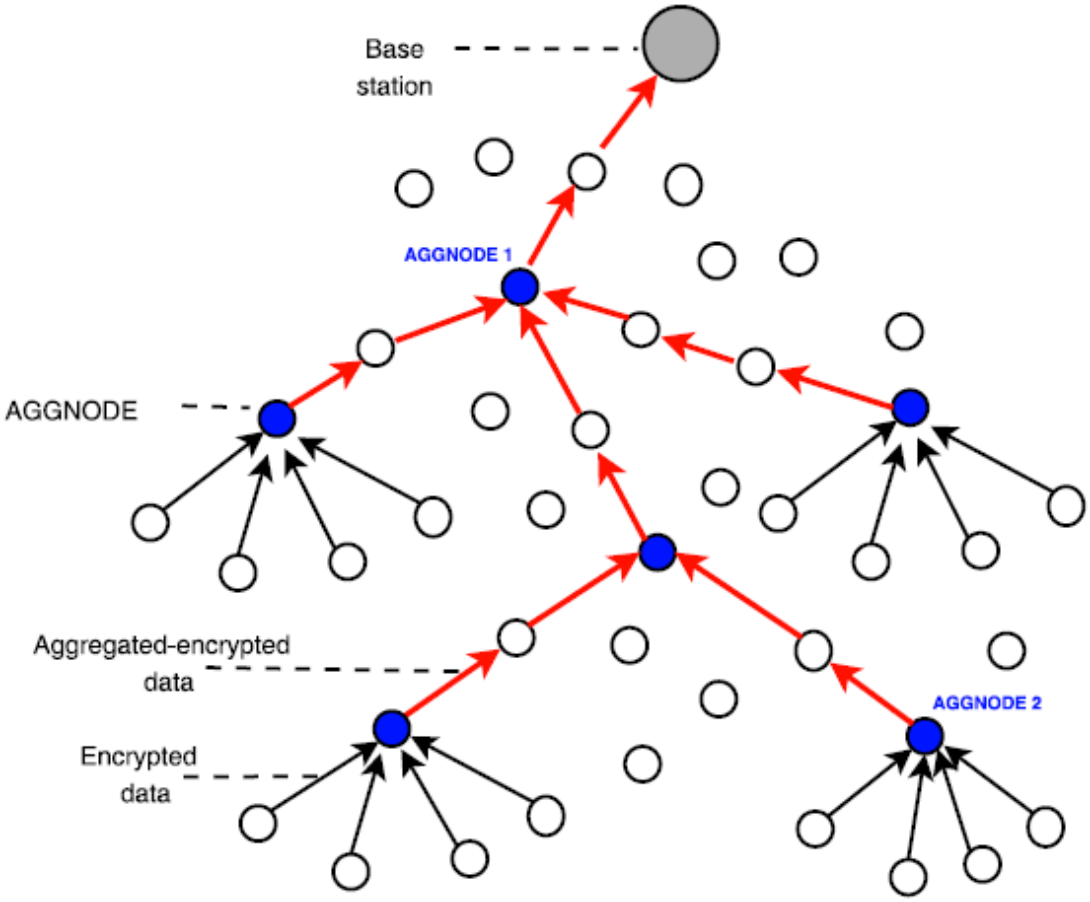


Figure 3.4: Example of the CDAP aggregation tree.

Referring to figure 3.4, it can be noticed that, if AGGNODE2 is compromised, it is empowered to alter only those data received by its children. On the other hand, if AGGNODE1 was compromised and no mechanism for protecting aggregated data was embedded in the protocol, the potential damage caused by the

attacker would be much wider. However CDAP, through the asymmetric key PH, ensures that AGGNODEs cannot manipulate or violate the confidentiality of the received aggregated data. Therefore, a compromised AGGNODE can cheat only on single sensor readings received by its children.
By means of simulations, it is shown that, using at least 6 AGGNODEs out of 100 traditional sensor nodes, CDAP is even more efficient than a traditional hop-by-hop secure aggregation protocol, despite the computational overhead involved by the used PH is considerably higher with respect to the one of a traditional SKC scheme.

3.3.3 EAED, Efficient Aggregation of Encrypted Data in WSNs [60]

The last surveyed proposal is the one of Castelluccia et al. [60]. The protocol is very similar to that presented in CDA, but it is based on a different PH, i.e. an additive homomorphic stream cipher, whose security proof is provided in the appendix of [60].

The basic idea is that of replacing the XOR operation, traditionally used by stream ciphers, with a simple modular addition. In the following, this new PH is defined. Let:

- M be a large integer;
- m be the message to encode, represented as an integer $m \in [0, M-1]$;
- k be a random keystream, with $k \in [0, M-1]$;
- c be the ciphertext, output of the encryption algorithm run on m;

The authors define the following operations:

1. *Encryption* - $c = E_k^M(m) = m + k \ (mod\ M)$;
2. *Decryption* - $D_k^M(c) = c - k \ (mod\ M)$;

Given two ciphertexts c_1 and c_2, where $c_1 = E_{k_1}^M(m_1) = m_1 + k_1 \ (mod\ M)$ and $c_2 = E_{k_2}^M(m_2) = m_2 + k_2 \ (mod\ M)$, It is straightforward to verify that

$$D_{k_1+k_2}^M(c_1 + c_2) = m_1 + m_2 \ (mod\ M).$$

Please note that if n different ciphertext c_i are added, then M must be larger than $\sum_{i=1}^{n} m_i$, otherwise correctness is not provided. In fact, if $\sum_{i=1}^{n} m_i$ is larger than M, decryption will result in a value m' that is smaller than M.
From the point of view of performance, the authors of [60] claim that their protocol is just slightly less efficient, with respect to used bandwidth, than a traditional symmetric key hop-by-hop protocol, though it provides a much higher level of security, being an end-to-end secure aggregation protocol. Moreover, encryption

and aggregation operations require just a small amount of additions and the encryption process expands the packet size of a few of bits.

Unfortunately, there is a non-negligible drawback in the proposed algorithm. In fact, each aggregate message is coupled with the list of nodes that failed to contribute to the aggregation. When the aggregation tree is large, the list of sensor nodes become larger and results in a significant communication overhead.

3.4 Comparison of the proposals

As a conclusion to the review performed in this chapter, tables 3.1 and 3.2 summarise the properties of the surveyed protocols, in terms of verified security requirements (§2.2.1) and attack vulnerabilities (according to the classification presented in §2.2.2).

Protocol	Confidentiality	Authentication	Integrity	Freshness	Availability
SIA	✓	✓	✓	✓	X
SecureDAV	✓	✓	✓	X	X
SDAP	✓	✓	✓	✓	X
ESPDA	✓	✓	✓	✓	X
[52]	✓	✓	✓	✓	X
SELDA	X	✓	✓	n.d.	✓
CDA	✓	X	X	X	X
CDAP	✓	X	X	X	X
EAED	✓	X	X	X	X

Table 3.1: Security requirements ensured by the presented protocols.

Protocol	DoS	Node compromise	Replay attack	Sybil attack	Selective forwarding	Stealthy attack
SIA	X	X			X	
SecDAV	X	X	X		X	X
SDAP	X	X			X	X
ESPDA	X	X			X	X
SELDA		X	n.d.			X
[52]	X	X			X	
CDA	X	X	X			
CDAP	X	X	X		X	
EAED	X	X				

Table 3.2: Attack vulnerabilities of the examined protocols.

Referring to tables 3.1 and 3.2, it is important to underline some points:

- the security level achieved by each protocol with respect to security requirements is not the same for all of the proposal; in particular, end-to-end

confidentiality can be assumed to be a stronger property when compared with hop-by-hop one. In general, the ensured security requirements should be always evaluated in the context of the considered protocol.

- end-to-end protocols ensure only end-to-end data confidentiality; the accomplishment of other security requirements is left to different protocols.
- none of the proposed protocols, except SELDA [46], addresses the availability requirement (and therefore protection against DoS attacks). Even if it could appear as a serious security flaw, one should observe that, due to transmitting power bounds of WSN nodes, an attacker can jam all the communications within the network without big efforts. Therefore, if an adversary aims at disrupting all network services (with an impact proportional to his power, see §2.2.3) through a DoS attack, we can assume that no security mechanism can protect the network against his threats. This is the reason why, somehow, it is not very important to look for availability (intended as protection against DoS attacks) in these protocols.
- all the proposed protocols are vulnerable to node compromise. This is a direct consequence of the fact that nodes are not tamper-proof; therefore, nothing can be done against this threat.

Chapter 4

Authentication Techniques applied to Data Aggregation

Authenticity is an essential requirement in communication systems pursuing security. In the field of wireless communications, the quest for authenticity becomes even more urgent, as the communication medium itself is intrinsically unsafe. Furthermore, if we consider that communication systems could be deployed in hostile environments, it is clear that authentication mechanisms cannot be given up.

Typically, in order to provide *data authenticity*, a MAC, i.e. *Message Authentication Code* is used. A MAC is nothing but an authentication tag, derived by applying a symmetric authentication scheme provided with the data to authenticate and a secret symmetric key, known only to authorized parties (namely to the MAC generator and to its intended recipients). Additionally to the authenticity property, MACs ensure also data integrity.

In this chapter, two of the most used and solid MAC algorithms (CBCMAC and HMAC) are presented after a brief, focused introduction to their theoretical foundations. Extensive references to the formal framework (definitions, algorithms, etc.) provided in [1] are made.
Afterwards, the relationship between data aggregation and authentication is investigated, and the difficulties arising when providing authenticity to a data aggregation protocol are shown and, finally, a protocol addressing end-to-end authenticity in data aggregation is proposed. The considered protocol, ESAWN [51], is thoroughly described, some enhancements are introduced and a set of metrics for its evaluation is proposed.

4.1 Message Authentication Codes (MACs)

As stated in [1], "MAC algorithms may be viewed as (*cryptographic*) hash functions which take two functionally different distinct inputs, a message and a secret key, and produce a fixed size (say n-bit) output, with the design intent that it is infeasible in practice to produce the same output without knowledge of the key".

In first place, it is therefore convenient to provide a formal definition of what a cryptographic hash function is, in order to successively define a MAC algorithm.

Definition 2 (Cryptographic hash function [1])**.**

A cryptographic hash function is a function h which has the following properties:

1. *compression* - h maps an input x of arbitrary finite bit-length, to an output $h(x)$ of fixed bit-length n.
2. *ease of computation* - given h and an input x, $h(x)$ is easy to compute.

Furthermore, one or more of the properties listed below may be required:

3. *preimage resistance* - for essentially all pre-specified outputs, it is computationally infeasible[1] to find any input which hashes to that output, i.e. to find any preimage x' such that $h(x') = y$, when given any y for which a corresponding input is not known.
4. *second preimage resistance* - it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a second preimage $x' \neq x$ such that $h(x') = h(x)$.
5. *collision resistance* - it is computationally infeasible to find any two distinct inputs x and x' which hash to the same output, i.e. such that $h(x) = h(x')$.

Definition 3 (MAC Algorithm [1])**.**

A *message authentication code* (MAC) algorithm is a family of functions h_k parametrised by a secret key k, with the following properties:

1. *ease of computation* - for a known function h_k, given a value k and an input x, $h_k(x)$ is easy to compute. The result is called MAC.
2. *compression* - h_k maps an input x of arbitrary finite bit-length to an output $h_k(x)$ of fixed bit-length n.

Furthermore, given a description of the function family h, for every fixed and allowable value of k (unknown to the adversary), the following property holds:

3. *computation resistance* - given zero or more text-MAC pairs $(x_i, h_k(x_i))$, it is computationally infeasible to compute any text-MAC pair $(x, h_k(x))$ for any new input $x \neq x_i$ (including possibly for $h_k(x) = h_k(x_i)$ for some i) for a party who does not know the secret key.

[1] what *computationally infeasible* means depends on the context of reference; in general, this definition entails the actual impossibility in performing a certain computation, even if, theoretically, it is feasible.

4.1.1 MAC Forgery

Whether a MAC algorithm is not compliant with property 3 (computation resistance), it is subject to *MAC forgery* and, being "forgeable", the security of the algorithm has been violated. As it can be easily understood, an adversary who wants to attack a MAC algorithm, aims to compute a new text-MAC pair $(x, h_k(x))$ for some $x \neq x_i$, given one or more legitimate pairs $(x_i, h_k(x_i)$ but, of course, without knowing the secret key.

Possible attacks on a MAC algorithm

We distinguish three possible types of attacks against MAC algorithms, for adversaries with increasing advantages:

1. *known-text attack* - one or more text-MAC pairs $(x_i, h_k(x_i))$ are available to the attacker.
2. *chosen-text attack* - one or more text-MAC pairs $(x_i, h_k(x_i))$ are available to the attacker, with the x_i chosen by the attacker.
3. *adaptive chosen-text attack* - the attacker can adaptively choose the inputs to the MAC algorithm, thus adjusting his strategy on-the-fly.

Types of MAC forgery

Two different types of MAC forgery can be distinguished, namely:

1. *selective forgery* - the adversary can choose an input to the MAC algorithm and obtain a correct text-MAC pair.
2. *existential forgery* - the adversary can produce a legitimate text-MAC pair, but is not allowed to choose the text.

Obviously, *selective forgery* is the most serious threaten to the security of the system, but also *existential forgery* can have a devastating impact. For example, let's think of a system for bank transactions: if an attacker is able to forge (*existentially*) consistent text-MAC pairs, he can replace the correct amount of money with a random one.

A limitation of MAC algorithms

As stated in [38], "the successful verification of a MAC does not completely guarantee that the accompanying message is authentic: there is chance that a source with no knowledge of the key can present a purported MAC on the plaintext message that will pass the verification procedure". Trivially, this type of attack can be achieved through a traditional brute-force approach and a forged t-bit MAC can be successfully verified with an expected probability equal to $(1/2)^t$, independently on the underlying MAC algorithm.

Of course, whether non-authentic messages can be repeatedly presented to the verifying authority, this limitation becomes even more consistent, since the success probability increases at each successive trial.

Also, if the MAC is truncated (i.e. the length of the used MAC is smaller than the output of the MAC function) the success probability improves. Hence, the authors of [38] suggest that if the MAC is truncated, then its length t should be chosen as large as possible and, anyways, it shall be at least $L/2$, where L is the bit-length of the MAC output block size.

The length of a truncated MAC, however, could be relaxed to 32 bits for applications in which either the number of repeated trials for message authentication is limited by design or the bandwidth of the communication channel is low enough to preclude too many trials. This could be the case of a WSN.

4.2 Block cipher-based authentication

4.2.1 Block ciphers

Note : block ciphers can actually be either symmetric-key or public key. For the purposes of this thesis, only symmetric-key block ciphers are considered.

According to the definition provided in [1], "a block cipher is a function which maps n-bit plaintext blocks to n-bit ciphertext blocks. The function is parametrised by a k-bit key K, taking values from a subset $\mathcal{K}$ (the *key-space*) of the set of all k-bit vectors $\mathcal{V}_k$. It is usually assumed that the key is chosen at random". In [1], a formal definition is provided as well:

Definition 4 (Block Cipher)**.**

A n-bit *block cipher* is a function $E : \mathcal{V}_n \times \mathcal{K} \mapsto \mathcal{V}_n$, such that for each key $K \in \mathcal{K}$ and plaintext $P \in \mathcal{V}_n$, $E(P, K)$ is an invertible mapping (the *encryption function* for K) from $\mathcal{V}_n$ to $\mathcal{V}_n$, written $E_K(P)$. The inverse mapping is the *decryption function*, denoted $D_K(C)$, where $C = E_K(P)$ denotes the ciphertext resulting from encrypting plaintext P under K.

The fixed length parameter n is called *block size* and, for many block ciphers, it is equal to 64 bits. Whether the plaintext block P or the key K have less than n bits, padding is required. Conversely, if the plaintext message exceeds one block in length, various modes of operation can be used in order to address this problem. Some available block cipher modes of operation are:

- Electronic CodeBook (ECB)
- Cipher-Block Chaining (CBC)
- Cipher FeedBack (CFB)
- Output FeedBack (OFB)

As stated earlier, these modes of operation allow block ciphers to provide confidentiality for arbitrary length messages. In figure 4.1 schemes for different modes of operation are shown, while a detailed overview is beyond the purpose of this thesis (the interested reader can find in [1] an exhaustive description).

CBC Mode

Referring to [1], we define the algorithm of CBC mode of operation:

Algorithm 1 (CBC mode)**.**

INPUT: k-bit key K; n-bit plaintext blocks $x_1, x_2, ..., x_t$, n-bit initialisation vector IV.

OUTPUT: ciphertext blocks $c_1, ..., c_t$

1. *Encryption* - $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $c_j \leftarrow E_K(c_{j-1} \oplus x_j)$.
2. *Decryption* - $c_0 \leftarrow IV$. For $1 \leq j \leq t$, $x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j)$.

Properties of the CBC mode of operation

- *Identical plaintexts* - if the *same* plaintext is encrypted twice using the same key and initialisation vector IV, the two block cipher outputs will be identical. Therefore, changing the IV, the key or the first plaintext block (e.g. through incrementing a counter or generating a random field) is necessary in order to achieve the *data freshness* property (see §2.2.1).
- *Chaining dependencies* - each ciphertext block c_j depends on x_j as well as on $x_{j-1}, x_{j-2}, ..., x_1$; therefore, for a successful decryption process, it is imperative that the ciphertext ordering is coherent with the one adopted during the encryption phase.
- *Error propagation* - if a single bit error occurs in ciphertext block c_j, the decryption of c_j and of c_{j-1} will be faulty (as x_j depends on c_j and c_{j-1}). Typically, the block x'_j recovered from c_j is totally random, while plaintext x'_{j+1} has bit errors exactly where c_j was corrupted; this means that the adversary can alter bits in x_{k+1} in a predictable way, simply by changing the corresponding bits in c_j.
- *Error recovery* - As said in [1], "CBC mode is *self-synchronizing* or *ciphertext autokey*, in the sense that if an error (including loss of one or more entire blocks) occurs in block c_j but not c_{j+1}, c_{j+2} is correctly decrypted to x_{j+2}".

4.2.2 The RC5 block cipher

A particularly efficient (and thus suitable for being used in WSNs) block cipher is RC5, proposed by Ronald L. Rivest[2] in 1997 [4]. In the following, a quite

[2]Rivest is one of the creators of the RSA algorithm.

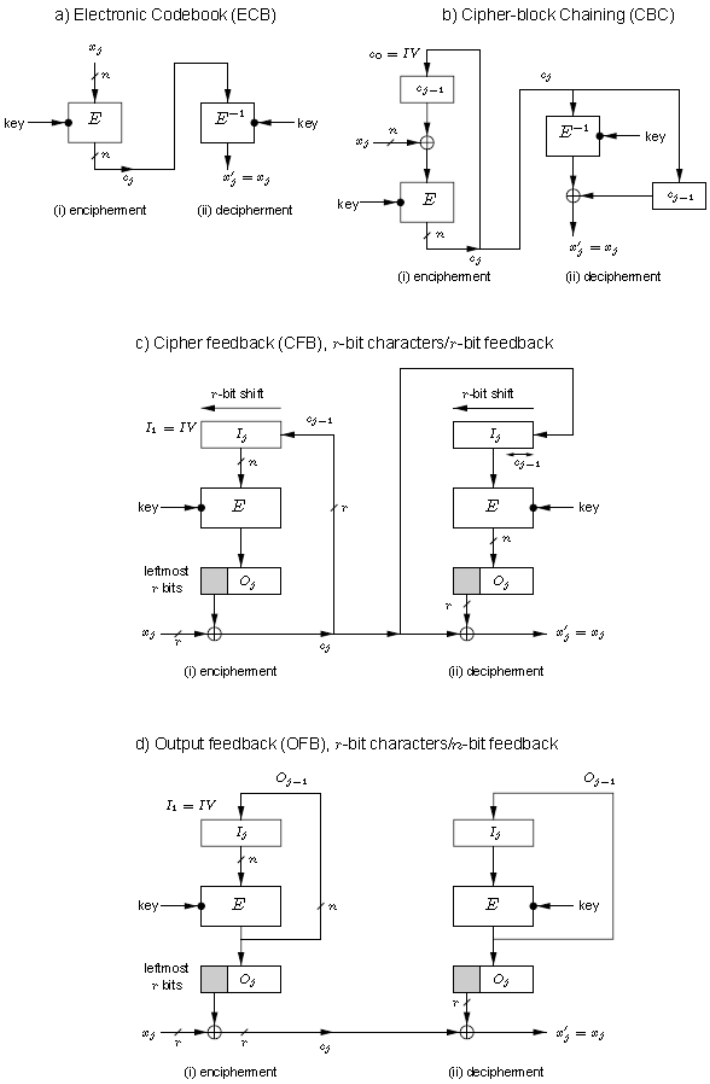


Figure 4.1: Common modes of operation for an n-bit block cipher. (figure taken from [1])

accurate description of the algorithm is presented.

RC5 main features

The main features of RC5 are listed below:

- RC5 is a *symmetric block cipher*, suitable for hardware or software implementation.

- While designing RC5, the author wanted it to be *fast*: thus the choice to create a *word-oriented* architecture. Furthermore, in order to allow easy adaptation for processors with different word lengths, RC5 is parametrised

in the word size w. This feature makes RC5 particularly suitable for implementation on wireless sensor nodes.

- RC5 is iterative in structure, with a variable number of rounds. The user can explicitly manipulate the trade off between higher speed and higher security, adjusting the number of rounds r.

- RC5 accepts variable length cryptographic keys, depending on the desired level of security; the higher the key length b (in bytes), the higher the achievable security.

- RC5 is easily implementable.

- RC5 uses *data-dependent rotations*, in which "one word is cyclically rotated by an amount determined by the low-order bits of another intermediate result"[4].

Being a *word-oriented* algorithm, implies that all basic computational operations in RC5 have w-bit words as input and output. Moreover, input and output to the RC5 block cipher are blocks of length $2w$ (i.e. the plaintext and the ciphertext size is $2w$).

Another adjustable RC5 parameter is the number of rounds r. As the author says "choosing a larger number of rounds presumably provides an increased level of security".
RC5 uses an *expanded key table*, S, derived from the user's supplied secret key and whose size depends on r. Specifically, the size t of table S is defined as $t = 2(r+1)$ words; this means that choosing a higher r, though ensuring (at least theoretically) an increased level of security, entails a higher memory utilisation as well.

RC5 parameters

As mentioned in the previous paragraph, RC5 is adjustable according to several parameters. For the sake of clearness, tunable RC5 parameters are:

- w - the word size, expressed in bits; allowed values of w are $16, 32$ or 64 bits.

- r - the number of rounds, on which the size t of the expanded key table S depends, as $t = 2(r + 1)$; allowed values of r are $0, 1, 2, ...255$.

- b - the number of bytes in the secret key K; allowed values of b are $0, 1, 2, ...255$.

- K - the b-byte secret key, $K[0], K[1], ...K[b - 1]$.

RC5 primitive operators

RC5 is based only on three primitive operations (and their inverses), namely:

- $\boxplus$: addition modulo-2^w of two words;
- $\oplus$: bitwise exclusive OR (XOR) of words;
- $\hookleftarrow$, $\hookrightarrow$: respectively left and right rotation of words.

$x \hookleftarrow y$ denotes cyclically shifting a w-bit word left by y bits, where the rotation-count y may be reduced mod w. Rotations are performed according to *variable* amounts, depending on the plaintext; the author claims that, on modern microprocessors, a variable rotation takes a constant time, i.e., the computation time is independent on the span of the rotation.

Please note that the author assumes a standard *little-endian* convention for packing bytes into input/output blocks, i.e., the first byte occupies the low-order bit positions and so on.

The RC5 algorithm

The RC5 algorithm consists of three components, specifically a *key expansion* algorithm, an *encryption* algorithm and a *decryption* algorithm. All of these components are described below, according to the definitions provided in [4] and [1].

Algorithm 2 (RC5 Encryption)**.**

INPUT:

✓ $2w$-bit plaintext $M = (A|B)$, where A and B are two w-bit words;

✓ number of rounds r;

✓ key $K = K[0]...K[b-1]$.

OUTPUT:

✓ $2w$-bit ciphertext C.

ALGORITHM:

1. compute $2(r+1)$ subkeys $K_0, ..., K_{2r+1}$ through the *key expansion* algorithm;
2. $A \leftarrow A \boxplus K_0$, $B \leftarrow B \boxplus K_1$;
3. for $i = 1$ to r do
   $A \leftarrow ((A \oplus B) \hookleftarrow B) \boxplus K_{2i}$;
   $B \leftarrow ((B \oplus A) \hookleftarrow A) \boxplus K_{2i+1}$;
4. $C \leftarrow (A, B)$.

Algorithm 3 (RC5 Decryption)**.**

INPUT:

✓ $2w$-bit ciphertext $C = (A|B)$, where A and B are two w-bit words;

✓ number of rounds r;

✓ key $K = K[0]...K[b-1]$.

OUTPUT:

✓ $2w$-bit decrypted plaintext C.

ALGORITHM:

1. compute $2(r+1)$ subkeys $K_0, ..., K_{2r+1}$ through the *key expansion* algorithm;
2. for $i = r$ downto 1 do
   $B \leftarrow ((B \ominus K_{2i+1}) \hookleftarrow A) \boxplus A$;
   $A \leftarrow ((A \ominus K_{2i}) \hookleftarrow B) \boxplus B$;
3. $B \leftarrow B \ominus K_1$, $A \leftarrow A \ominus K_0$;
4. $M' \leftarrow (A, B)$.

Algorithm 4 (RC5 Key expansion)**.**

INPUT:

✓ word bit-size w;

✓ number of rounds r;

✓ key $K = K[0]...K[b-1]$.

OUTPUT:

✓ subkeys $K_0, ..., K_{2r+1}$, where K_i has w bits.

ALGORITHM:

1. Copy the secret key $K[0, ..., b-1]$ into an array $L[0, ..., c-1]$ of $c = \lceil b/u \rceil$ words, where $u = w/8$ is the number of bytes per word (if necessary, apply zero-padding to K in order to achieve a byte-count divisible by u);

   for $i = b-1$ downto 0 do
   $L[i/u] \leftarrow (L[i/u] \hookrightarrow 8) + K_i$;

2. Let $P_w = Odd((e-2)2^w$, $Q_w = Odd((\phi - 1)2^w)$ be two "magic constants"[3] . During the second phase of the key expansion, the subkeys are initialised to a particular, fixed, key independent pseudo-random bit pattern; this is done using an arithmetic progression modulo 2^w determined by P_w and Q_w;

   $K_0 = P_w$;
   for $i = r$ to $2r+1$ do
   $K_i \leftarrow K_{i-1} \boxplus Q_w$;

3. $i \leftarrow 0$,$j \leftarrow 0$;
   $A \leftarrow 0$, $B \leftarrow 0$;
   do $3 \times max\{2(r+1), c\}$ times
   $K_i \leftarrow (K_i \boxplus A \boxplus B) \hookleftarrow 3$, $A \leftarrow K_i$, $i \leftarrow i+1 \bmod 2(r+1)$;
   $L_j \leftarrow (K_j \boxplus A \boxplus B) \hookleftarrow (A \boxplus B)$, $B \leftarrow L_j$, $j \leftarrow j+1 \bmod c$;

4. Output $K_0, ..., K_{2r+1}$.

[3]with $e = 2.718281828459...$ (base of natural logarithms) and $\phi = 1.618033988749...$ (golden ratio).

RC5 (essential) cryptanalysis

In 1998 [5], 12-round RC5 (with 64-bit words) was found to be susceptible to a *differential attack* using 2^{44} chosen plaintexts. Adjusting the block cipher parameters, however, solves this issue ($18-20$ rounds offer a sufficient protection, at least according to the authors of [5]). Therefore, in the context of this thesis, it is assumed that RC5 offers a suitable security level for a WSN.

4.2.3 The CBCMAC algorithm

CBCMAC is a well-known authentication algorithm that is based on block ciphers in CBC mode of operation; it provides authenticity, founding on resilient and performing block ciphers. The CBCMAC algorithm is described below, according to the definition provided in [1].

Algorithm 5 (CBCMAC algorithm)**.**

INPUT:

✓ data x;

✓ specification of block cipher E, operating on n-bit blocks;

✓ secret MAC key k for E.

OUTPUT:

✓ n-bit MAC of x.

ALGORITHM:

1. *Padding and blocking*
   pad input x if necessary (in order to obtain a number of bits divisible by n). Split the padded input into a number t of n-bit blocks, denoted by $x_1, ..., x_t$;
2. *CBC processing*
   let E_k denote the encryption using E with key k. Compute the block H_t (i.e. the MAC) as follows:

   $H_1 \leftarrow E_k(x_1)$
   for $i =$ to t do
   $H_i \leftarrow E_k(H_{i-1} \oplus x_i)$;
3. *Optional process to increase strength of MAC*
   using a second key $k' \neq k$, compute $H'_t \leftarrow E^{-1}_{k'}(H_t), H_t \leftarrow E_k(H'_t)$
4. *Completion*
   output the n-bit MAC H_t

In figure 4.2, the CBCMAC algorithm is graphically shown. Please observe that the CBCMAC algorithm described above allows the designer to choose a generic block cipher for implementation. For example, the previously presented RC5 block cipher is suitable to be embedded in a CBCMAC-RC5 authentication algorithm (CBCMAC-RC5 is actually the scheme which has been implemented in this thesis for further analysis).

A quick insight into CBCMAC Security

Once the CBCMAC algorithm has been defined, a natural question emerges: is the CBCMAC secure and under which hypothesis? A detailed dissertation is beyond the purpose of this thesis, but some points have to be explored.

First of all, as proven in [62], the security of a CBCMAC depends on the one of the underlying block cipher. This means that CBCMAC cannot be considered alone while evaluating its security, but has to be considered together with its embedded block cipher.

Secondly, if its input messages are not of fixed size, CBCMAC could be vulnerable to *existential forgery* through *adaptive chosen-plaintext* attacks. Specifically, let x_i be an n-bit block and let $y^{(n)}$ be the n-bit representation of y. The attacker can request the MAC t_1 for a block x_1, obtaining the text-MAC pair (x_1, t_1); then, inputting the block $x_2 = t_1$, he obtains the MAC t_2 which is also the MAC for the two-block message $x_1||0^n$. Therefore, given two text-MAC pairs (x_1, t_1) and (x_2, t_2), with $x_2 = t_1$, the adversary is able to forge a new pair (x_3, t_3), where $x_3 = x_1||0^{(n)}$.

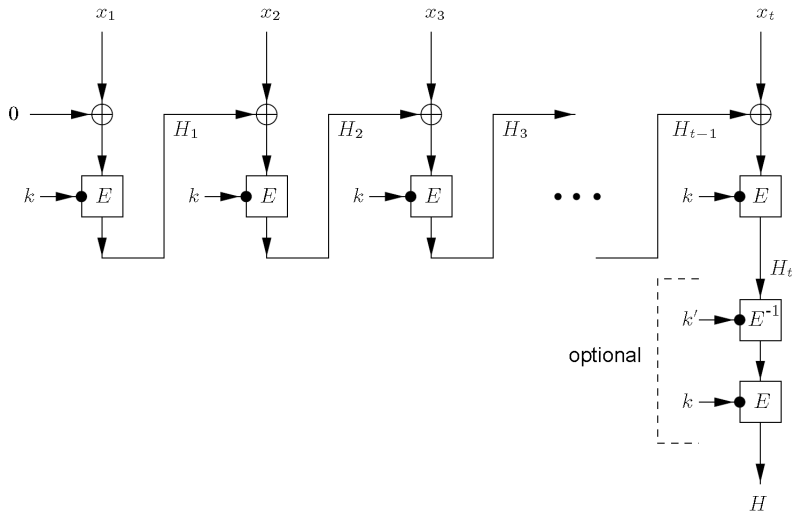


Figure 4.2: The CBCMAC algorithm. (figure taken from [1])

4.3 Hash function-based authentication

In the previous section, a possible way of ensuring message authentication using block ciphers was shown. Block ciphers provide good performance and a well-studied platform to lean on, but they are not the only, nor the best (in absolute terms), means to ensure message authenticity.
In particular, there is a family of cryptographic primitives that suit well to the MAC generation problem: *un-keyed cryptographic hash functions*. This group of

hash functions, which includes primitives such as *MD4, MD5, SHA1, RIPEMD*, can be even faster than block ciphers in software implementation and is readily and freely available to the public.

The scientific community defined a standard [38] for creating keyed-hash message authentication codes. This standard is described in §4.3.3, but, firstly, an overview on un-keyed cryptographic hash functions is provided. A definition of what a cryptographic hash function is, was already given in §4.1.
In the next paragraph two of the most used representatives of this class of functions are described, namely *MD5* and *SHA1*.

4.3.1 The MD5 message digest algorithm

MD5 is a 128-bit cryptographic hash function proposed in 1991 by Ronald Rivest [63].
MD5 was designed for 32-bit machines as an extension of the *MD4* message-digest algorithm and, with respect to *MD4*, is a bit more conservative, in the sense that it sacrifices some speed for improved security. *MD5* is indeed a very fast hashing algorithm, though it was found to be vulnerable to several attacks (a detailed cryptanalysis is not provided in this thesis; the interested read could refer to [64, 65, 66]), that make its use questionable for the sake of robust security. It is interesting, however, to test its performances in a WSN, particularly in order to compare it against different or newer solutions.

Algorithm 6 (MD5 algorithm [1] [63])**.**

INPUT:

✓ b-bit message $x = x[0]....x[b-1]$, where b is an arbitrary non-negative integer;

OUTPUT:

✓ 128-bit message-digest of x.

ALGORITHM:

1. Preprocessing

   - *Padding*
     pad input x so that its length in bits is congruent to 448 modulo 512; padding is always performed, even if the length of the message is already congruent to 448 modulo 512.
     Padding is performed according to the following procedure: firstly, a single bit "1″ is appended to the message and, then, $r-1$ "0″ bits are appended so that the bit-length of the padded message is equal to 448 modulo 512. Therefore, at least 1 bit and at most 512 bits are appended.
   - *Append length*
     A 64-bit representation of b (the length of the message before padding) modulo 2^{64} is appended to the padded bit-string.

   The result is a bit-string with a length that is an exact multiple of 512 bits, i.e. 16 32-bit words. Summarising, let m be the number of 512-bit blocks in the resulting string, being $b + r + 64 = 512m = 32 \cdot 16m$. The formatted input consists of $16m$ 32-bit words: $x_0, x_1, ..., x_{16m-1}$.

- *Initialise MD buffer*
  A four-word buffer (A, B, C, D) is used to compute the message digest. A, B, C, D are 32-bit registers, initialised with the following hexadecimal values:
  (a) $A =$ 0x01234567;
  (b) $B =$ 0x89ABCDEF;
  (c) $C =$ 0xFEDCAB98;
  (d) $D =$ 0x76543210.

2. Processing

- *Preliminary definitions*
  Firstly, define four auxiliary functions, each taking three 32-bit words as input and producing a 32-bit word as output:
  – $f(x, y, z) = xy \vee x\overline{z}$
  – $g(x, y, z) = xz \vee y\overline{z}$
  – $h(x, y, x) = x \oplus y \oplus z$
  – $i(x, y, z) = y \oplus (x \vee \overline{z})$

  where uv denotes the bitwise AND between u and v, $\vee$ denotes the bitwise inclusive OR, $\oplus$ the bitwise exclusive OR and $\overline{u}$ denotes the bitwise complement of u.

  Secondly, define 64 different 32-bit constants $y[i]$ = first 32 bis of binary value $abs(sin(i + 1))$, $0 \leq i \leq 63$, where j is in radians and abs denotes the absolute value.

  Thirdly, define the order for accessing source words:
  (a) $z[0, ..., 15] = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$,
  (b) $z[16, ..., 31] = [1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12]$,
  (c) $z[32, ..., 47] = [5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2]$,
  (d) $z[48, ..., 63] = [0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9]$.

  Finally, define the number of bit positions for left shifts:
  (a) $s[0, ..., 15] = [7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22]$,
  (b) $s[16, ..., 31] = [5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20]$,
  (c) $s[32, ..., 47] = [4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23]$,
  (d) $s[48, ..., 63] = [6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21]$.

- *Processing*

for $i = 0$ to $m - 1$ {
  for $j = 0$ to 15 {
    $X[j] \leftarrow x_{16i+j}$;
  }
  $(AA, BB, CC, DD) \leftarrow (A, B, C, D)$;
  Round 1
  for $j = 0$ to 15
    $t \leftarrow (A + f(B, C, D) + X[z[j]] + y[j])$;
    $(A, B, C, D) \leftarrow (D, B + (t \hookleftarrow s[j], B, C)$;
  Round 2
  for $j = 16$ to 31 {
    $t \leftarrow (A + g(B, C, D) + X[z[j]] + y[j])$;
    $(A, B, C, D) \leftarrow (D, B + (t \hookleftarrow s[j], B, C)$;
  }
  Round 3

for $j = 32$ to 47 {
$t \leftarrow (A + h(B, C, D) + X[z[j]] + y[j])$;
$(A, B, C, D) \leftarrow (D, B + (t \hookleftarrow s[j], B, C)$;
}
Round 4
for $j = 48$ to 63 {
$t \leftarrow (A + i(B, C, D) + X[z[j]] + y[j])$;
$(A, B, C, D) \leftarrow (D, B + (t \hookleftarrow s[j], B, C)$;
}
Update chaining values
$(A, B, C, D) \leftarrow (A + AA, B + BB, C + CC, D + DD)$. }

3. Completion
The final hash value is the concatenation: $A||B||C||D$.

4.3.2 SHA1 - Secure Hash Algorithm 1

SHA hash functions are cryptographic hash functions designed by the U.S. National Security Agency (NSA) and published by the National Institute of Standards and Technology(NIST). *SHA1* (Secure Hash Algorithm 1) was introduced in 2001 [67] and is the most widely used member of the SHA family.

SHA1 has a structure very similar to that of the *MD* family and is actually modelled on the *MD4* message-digest algorithm. Concerning security issues, *SHA1* was found to be subject to some *collision attacks* (i.e., attacks violating the collision resistance property of the cryptographic hash function), but these attacks require a huge amount of computations and are thus considered nearly infeasible nowadays[4].
Therefore in a WSN, with the currently available attacks on the algorithm, *SHA1* can provide sufficient security, since we can reasonably assume that the attacker is interested in violating security systems within a small amount of time[5] and, most likely, he cannot dispose of outstanding computational power, as he would like to compromise the system from the inside (e.g., by compromising a node and using its resources).
Anyways, in the near future, it would be wise to introduce a new hashing primitive to overcome the weaknesses of the SHA family.

Algorithm 7 (SHA1 algorithm [1, 67])**.**

INPUT:

✓ b-bit message $x = x[0]....x[b-1]$, where b is an arbitrary non-negative integer;

OUTPUT:

✓ 160-bit message-digest of x.

[4]for certain applications, a stronger hash function would be desirable; hence, other functions such as *SHA2*[68] were designed.

[5]data transmitted by sensor nodes typically consists of locally and instantly sensed values (e.g., temperatures) that are meaningful only for a limited period of time.

ALGORITHM:

1. PREPROCESSING

   - *Padding*
     (identical to *MD5* padding) pad input x so that its length in bits is congruent to 448 modulo 512; padding is always performed, even if the length of the message is already congruent to 448 modulo 512.
     Padding is performed according to the following procedure: firstly, a single bit "1″ is appended to the message and, then, $r-1$ "0″ bits are appended so that the bit-length of the padded message is equal to 448 modulo 512. Therefore, at least 1 bit and at most 512 bits are appended.
   - *Append length*
     Append the two 32-bit words specifying the bit-length b, with the most significant word preceding the least significant one.

   As in *MD5*, the result is a bit-string with a length that is an exact multiple of 512 bits, i.e. 16 32-bit words. Summarising, let m be the number of 512-bit blocks in the resulting string, being $b+r+64 = 512m = 32 \cdot 16m$. The formatted input consists of $16m$ 32-bit words: $x_0, x_1, ..., x_{16m-1}$.

   - *Initialise SHA1 buffer*
     Similarly to *MD5*, *SHA1* uses a five-word buffer $(h_0, h_1, h_2, h_3, h_4)$ to compute the message digest. $(h_0, h_1, h_2, h_3, h_4)$ are 32-bit registers, initialised with the following hexadecimal values:
     (a) h_0 = 0x67452301;
     (b) h_1 = 0xEFCDAB89;
     (c) h_2 = 0x98BADCFE;
     (d) h_3 = 0x10325476;
     (e) h_4 = 0xC3D2E1F0;

2. PROCESSING

   - *Preliminary definitions*
     SHA1 uses three auxiliary functions, each taking three 32-bit words as input and producing a 32-bit word as output:
     – $f(x,y,z) = xy \vee x\overline{z}$
     – $g(x,y,z) = xz \vee y\overline{z}$
     – $h(x,y,x) = x \oplus y \oplus z$

     where uv denotes the bitwise AND between u and v, $\vee$ denotes the bitwise inclusive OR, $\oplus$ the bitwise exclusive OR and $\overline{u}$ denotes the bitwise complement of u.

     Furthermore, in the *SHA1* algorithm 4 per-round additive constants are defined, namely:
     (a) y_1 = 0x5A827999;
     (b) y_2 = 0x6ED9EBA1;
     (c) y_3 = 0x8F1BBCDC;
     (d) y_4 = 0xCA62C1D6;

     Conversely to *MD5*, no order for accessing bit positions, nor definition of bit positions for left shifts are defined.

   - *Processing*

for $i = 0$ to $m - 1$ {
  for $j = 0$ to 15 {
    $X[j] \leftarrow x_{16i+j}$;
  }
  for $j = 16$ to 79 {
    $X_j \leftarrow ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{j-16}) \hookleftarrow 1)$
  }
  $(A, B, C, D, E) \leftarrow (h_0, h_1, h_2, h_3, h_4)$;
  Round 1
  for $j = 0$ to 19 {
    $t \leftarrow ((A \hookleftarrow 5) + f(B, C, D) + E + X_j + y_1)$;
    $(A, B, C, D, E) \leftarrow (t, A, B \hookleftarrow 30, C, D)$;
  }
  Round 2
  for $j = 20$ to 39 {
    $t \leftarrow ((A \hookleftarrow 5) + h(B, C, D) + E + X_j + y_2)$;
    $(A, B, C, D, E) \leftarrow (t, A, B \hookleftarrow 30, C, D)$;
  }
  Round 3
  for $j = 40$ to 59 {
    $t \leftarrow ((A \hookleftarrow 5) + g(B, C, D) + E + X_j + y_2)$;
    $(A, B, C, D, E) \leftarrow (t, A, B \hookleftarrow 30, C, D)$;
  }
  Round 4
  for $j = 60$ to 79 {
    $t \leftarrow ((A \hookleftarrow 5) + h(B, C, D) + E + X_j + y_4)$;
    $(A, B, C, D, E) \leftarrow (t, A, B \hookleftarrow 30, C, D)$;
  }
  Update chaining values
  $(h_0, h_1, h_2, h_3, h_4) \leftarrow (h_0 + A, h_1 + B, h_2 + C, h_3 + D, h_4 + E)$.
}

3. Completion
   The final hash value is the concatenation: $h_0||h_1||h_2||h_3||h_4$.

4.3.3 The keyed-hash message authentication code (HMAC)

As widely anticipated in §4.3, the standard for implementing hash-based authentication is the HMAC. This standard is described in the FIPS (Federal Information Processing Standards) publication 198 [69].

HMAC shall be used in combination with an approved cryptographic hash function and needs a secret key for the calculation and the verification of the MACs. While designing it, the authors planned to achieve the following goals:

- *Black-box approach*: use available hash functions without modifications; enable easy replacement of the underlying hash function.
- *Preserved performance*: HMAC should essentially have the same performance of the underlying hash function; additional complexity should be nearly negligible.
- *Simple key management*: keys should be used and handled in a simple way.

- *Provable security*: it should be easy to prove the algorithm security, assuming the security of the underlying hash function.

The HMAC algorithm

Algorithm 8 (HMAC algorithm)**.**

PRELIMINARY DEFINITIONS:

- B: block size (in bytes) of the input to the approved hash function.
- H: an approved hash function.
- L: block size (in bytes) of the output of the approved hash function.
- *ipad*: inner pad = 0x36 repeated B times.
- *opad*: outer pad = 0x5C repeated B times.
- t: number of bytes of the MAC.
- K: secret key shared between the originator of the message and the intended receiver(s).

INPUT:

✓ *text*: n-byte message, where $0 \leq n < 2^B - 8B$;

✓ K: secret key shared between the originator of the message and the intended receiver(s).

OUTPUT:

✓ t-bytes message-digest of x.

ALGORITHM:

1. *Key derivation*
   (a) if $length(K) = B$: $K_0 \leftarrow K$;
   (b) if $length(K) > B$: hash K to obtain a L-byte string, then append $B - L$ zeros in order to create a B-byte string: $K_0 \leftarrow H(K)||00...0$;
   (c) if $K < B$: append zeros at the end of K in order to create a B-byte string;
2. *Inner processing*
   (a) XOR K_0 with *ipad* to produce a B-byte string: $K_{in} \leftarrow K_0 \oplus ipad$;
   (b) Append *text* to K_{in}: $data_{in} \leftarrow K_{in}||text$;
   (c) Apply H to $data_{in}$: $hash_{in} \leftarrow H(data_{in}) = H(K_{0,ipad}||text)$;
3. *Outer processing*
   (a) XOR K_0 with *opad*: $K_{out} \leftarrow K_0 \oplus opad$;
   (b) Concatenate K_{out} with $hash_{in}$: $data_{out} \leftarrow K_{out}||hash_{in} = K_0 \oplus opad||H(K_{0,ipad}||text)$;
   (c) Apply H to $data_out$: $hash_{out} \leftarrow H(data_{out}) = H(K_0 \oplus opad||H(K_{0,ipad}||text))$;
4. *Completion*
   Select the leftmost t bytes of $hash_{out}$ as the MAC.

Summarising, the final result of the HMAC algorithm can be expressed as:

$$MAC(text)_t = HMAC(K, text)_t = H((K_0 \oplus opad)||H((K_0 \oplus ipad)||text))_t.$$

On HMAC cryptographic keys. The size of the secret key K should be equal to or greater than $L/2$. In [69], the authors advise that:

- keys greater than L bytes do not increase significantly HMAC security.
- keys longer than B-bytes should first hash the key using H and then use the resulting L-bytes string as the HMAC key.
- keys should be chosen at random using an approved key generation method and shall be changed periodically.

HMAC Security. The security of the described authentication scheme is lower-bounded by the one of the underlying hash function (a theoretical proof is given in [38]), but the actual security is higher than the one of the hash function, especially thanks to the use of the random cryptographic key.

As stated in [70], "the strongest attack known against HMAC is based on the frequency of collisions for the hash function H (i.e. a *birthday attack* [1]) and is totally impractical for minimally reasonable hash functions".

Suffice it to say that HMAC-MD5 and HMAC-SHA1 (at least at presently(do not suffer from those attacks the respective cryptographic hash functions are vulnerable. This is due to the way these hash functions are used in the construction of HMAC. This means that, as long as a specific attack against HMAC-MD5 or HMAC-SHA1 is not discovered, both of them can ensure authenticity; the use of HMAC-SHA1 would be, however, recommendable, as the $MD5$ security flaws are by far more exploitable.

4.4 The quest for the aggregated MAC

By definition, the message authentication codes described above require the original data on which the MAC was computed in order to be verified. If we want to ensure end-to-end data authenticity in a data aggregation protocol, this becomes a major issue.

First of all, why would we want to achieve end-to-end and not just hop-by-hop authenticity in a data aggregation protocol? The answer is straightforward: because *no* assumptions can be made on the trustworthiness of sensor nodes and, in particular, of aggregator nodes. Under these conditions, hop-by-hop authentication does not ensure that aggregators perform aggregation only using authentic submitted data; for example, a compromised aggregator node[6] could receive authentic data, forge a fake aggregate, authenticate it with its legitimate key (shared with the upper level aggregator) and forward the result to its next hop, with the forgery being unnoticed (as the message that it has generated is

[6] a compromised node, as explained in §2.2.2, has access to all keying material of the corrupted sensor.

indeed authentic from a hop-by-hop point of view). Therefore, in order to provide effective authentication of aggregated data, we need to keep track, somehow, of all the authenticity information of the data involved in the aggregation process. Thus, to some extent, we need to ensure end-to-end authenticity, as, ideally, the BS needs to verify the authenticity of all the sensor readings which took part in the aggregation process.

Clearly, this requirement is in sharp contrast with the base principle of data aggregation. In fact, while in-network aggregation aims at reducing the amount of data transmitted to the next hop of the aggregation tree, end-to-end authenticity verification requires the whole non-aggregated data to be available at the designated verifying end-point (e.g., the BS).

Providing stringent end-to-end authenticity while performing in-network aggregation is therefore a challenging task and, to the best of the author's knowledge, no effective solution has been found yet. In addition [73] shows that even a straightforward and intuitive refinement of the MAC security model (in the data aggregation setting) is not achievable.

Given the present impossibility of ensuring end-to-end authentication within a data aggregation protocol, an effective relaxation of the authenticity notion should be found. Several approaches can be pursued while looking for the best trade off between efficiency and security guarantees.
In the next section, a possible solution, the ESAWN protocol [51], is thoroughly described and analysed.
Before describing this proposal, however, two more considerations on the problem of aggregated MAC are provided.

On signing schemes

A partial solution to the authentication problem could be the use of a signing scheme like ECDSA [42] (used for example in [39]). This approach, though, is no longer based on traditional MACs, but on public key cryptography.

The basing principle of a signing scheme is that all the sensor nodes participating in the aggregation process, should sign with their own keys the aggregate produced by the aggregator node. The partial signatures produced by sensor nodes are then merged into a single signature, that is sent to the BS. The BS, which knows the corresponding public key, can finally check the authenticity. Note that, since the attacker does not know all the keys used by sensor nodes to sign the aggregate, he cannot generate the whole signature for the aggregated data by himself. However, there are at least two reasons why a signing scheme is not optimal for the purposes of this thesis.
Firstly, it suits to networks where only one aggregation cycle is performed before sending the result to the BS. If consecutive aggregations must take place on the path to the BS, the signing scheme does not provide end-to-end authenticity anymore.
Secondly, it is based on public key cryptography and, hence, energy consumption is not comparable to that required by traditional MACs.

In conclusion, in the context of this thesis, signing schemes are not considered to be a suitable solution for the considered problem.

On aggregating MACs

A widely used expedient to approach authentication in a data aggregation protocol is that of aggregating MACs. A possible technique is suggested in [72], where the authors prove that MACs can be aggregated by means of bitwise XORing operations and that the result still enables authenticity verification. Specifically, given n MACs, $MAC_1, ..., MAC_n$, the aggregated MAC can be computed as

$$MAC_{agg} = MAC_1 \oplus MAC_2 \oplus ... \oplus MAC_n.$$

It is proven that an adversary, in order to forge MAC_{agg}, should be able to forge at least one MAC_i (which is assumed to be unfeasible).
Unfortunately, while achieving the compression of multiple MAC tags into a single one, the aggregated MAC, MAC_{aggr}, still needs all the original data on which all MACs where computed in order to be verified.

4.5 ESAWN - Relaxed authenticity for data aggregation in WSNs

The ESAWN protocol [51] addresses the problem described in the previous section, proposing a probabilistic relaxation of the authenticity requirement in the presence of a fraction of compromised nodes; its objective is to provide an adjustable trade off between probabilistic authenticity and data aggregation.
Specifically, the authors of [51] observe that, in the presence of compromised nodes, an honest node cannot verify the authenticity of a received aggregate without additional data from nodes contributing to this aggregate. Yet, sending additional data to the verifying node is in contrast with the idea of data aggregation.

The basic idea of ESAWN is to use a configurable number of redundant aggregation paths along the aggregation tree towards the BS. The aggregated values travel along the redundant paths and are compared by different nodes. The comparison can be used to *detect* and (under certain conditions) *correct* false aggregation values.
An efficient parametrised protocol is proposed; ESAWN performances can be tuned according to the following parameters:

- β, the percentage of compromised nodes;
- $\mathcal{P}$, the percentage of authentic aggregates finally received at the BS.

According to the framework proposed in [76], ESAWN, with slight modifications, can provide one or more of the following properties:

- ✓ *Completeness*: honest nodes detect forged aggregates.
- ✓ *Soundness*: by means of a majority vote, forged aggregates are discarded and correct aggregates are accepted.
- ✓ *Non-repudiation*: previously sent messages cannot be repudiated, thus allowing malicious aggregators identification.

In this thesis, however, only the first two properties are considered, as ensuring the third one is believed to introduce an excessive communication overhead. Moreover, the original ESAWN protocol is based on authenticated encryption [77]; here, only traditional authentication primitives, such as CBCMAC and HMAC, are considered, as confidentiality is not the main interest of the present work. The choice of using plain authentication leads to a protocol optimisation, which will be described in §4.5.2.

ESAWN supports arbitrary aggregation functions, scales well with increasing network complexity and is based on efficient symmetric key cryptographic primitives.
In order to provide an in-depth analysis of the protocol, preliminary definitions and model assumptions are provided in the following paragraphs.

4.5.1 Definitions and model assumptions

Aggregation tree and network topology

The sensor network is modelled as an aggregation tree, i.e. a connected graph $G(V, E)$, where V is the set of vertices and E the set of directed edges; $|V| = n$ and it is assumed that there are no cycles in the tree. Edges represent aggregation relationships, which are denoted by "$\rightarrow$" and satisfy the following properties:

1. there exists a unique node $r \in V$ that has only incoming edges; this node is called *root node*.
2. every node $v \in V \backslash \{r\}$ has exactly one outgoing edge, $v \rightarrow v'$. The node v' is called the *parent* of v and v is called *source* of v'.

Furthermore, the following definitions are given:

- a *path* in T is a sequence of nodes $P = \{v_1, v_2, ..., v_l\}$ such that $\forall i \in \{1, ..., l-1\} : v_i \rightarrow v_{i+1}$.
- for a path $P = \{v_1, v_2, ..., v_l\}$, node v_{i+1} is called the *1-ancestor* of v_i; the *k-ancestor* of a node v_i is defined recursively as the *(k-1)-ancestor* of the *1-ancestor* of v_i.
- the *i-descendant* of v_i is the node for which v_i is the *i-ancestor*.
- F_V is the aggregation function.

- $\mathcal{A}_V$ is the set of aggregation nodes.

- δ_v denotes the number of source nodes for every aggregation node $v \in \mathcal{A}_V$.

- $\delta = mean_{v \in \mathcal{A}_V}(\delta_v)$.

According to the definitions provided above, the total number of nodes is approximately

$$n = \sum_{i=0}^{h} \delta^i = \frac{\delta^{h+1} - 1}{\delta - 1} \tag{4.1}$$

and therefore the tree depth can be expressed as

$$h = log_\delta(1 + (\delta - 1)n) - 1. \tag{4.2}$$

Finally, the following assumptions are made:

- all communications in the network are performed by means of *multi-hop* transmissions.

- in-network aggregation is accomplished according to the *periodic per-hop aggregation* paradigm described in 2.1.1; therefore aggregation is performed periodically as soon as the aggregator hears from all of its sources.

- *pair-wise keys* are deployed a-priori, among those nodes that need to communicate with each other, through the use of a suitable secure key distribution algorithm [74, 75].

- nodes that take part in the same verification process has to be synchronised; *synchronisation* can be implemented providing a pseudo-random number generator (PRNG) with a shared seed or an internal state. This issue will be analysed in detail later on.

- any node v can take measurements, perform aggregation, or both at the same time.

Attacker model and security parameters

The authors of [51] assume a *uniformly distributed, static attacker*. Thereby, the attacker is assumed to be capable of compromising a certain fraction β of nodes in advance, but he cannot modify his choice dynamically (i.e. *adaptive* attacks are not allowed[7]). Furthermore, compromised nodes are chosen uniformly at random among the set of all sensor nodes.

The choice of β determines the *security parameter* (t, k), which should be interpreted as follows: ESAWN is able to detect aggregation misbehaviour when at

[7] the authors of ESAWN claim that it is not possible, under the specified conditions (arbitrary aggregation functions, end-to-end authenticity requirement, etc.), to provide full protection against an adaptive attacker.

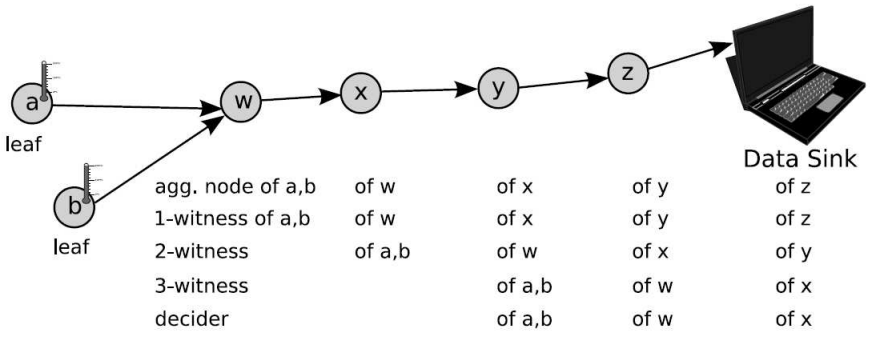


Figure 4.3: ESAWN nomenclature for $t = 3$.

most k nodes out of t consecutive nodes on any path in the aggregation tree are compromised. The actual meaning of this security parameter will be clarified as soon as the protocol is thoroughly described; intuitively, however, t is the redundancy introduced by ESAWN in the aggregation process (e.g. $t = 2$ means that aggregation is performed twice at two different aggregators), while k indicates the resiliency of the scheme with respect to node compromise (e.g. $k = 1$ means that at most 1 node out of t can be compromised, in order to allow ESAWN to detect forgeries).

A further assumption of ESAWN is that the adversary does not perform any kind of Denial of Service attack, such as selective forwarding. This is a reasonable assumption, as the adversary presumably aims to remain unnoticed while trying to inject false data in order to alter the final aggregation result. A deeper analysis of this topic, however, has not been accomplished yet.

The BS, of course, is assumed to be honest, as authentic aggregation would be impossible otherwise.

4.5.2 ESAWN Protocol

Overview

As widely anticipated, the basic idea of ESAWN is that of propagating aggregates (and the original data on which they were computed) on redundant paths, thus allowing other nodes to check if the aggregation result is authentic in a strict sense (i.e., if it was computed using all the submitted authentic data).

The protocol is founded on two fundamental concepts: witness and decider nodes. A *witness node* of v is one of v's j-descendants, where $0 \leq j \leq t$. On the other hand, a *decider node* of v is the t-ancestor of v. It should be noticed that the t-witness of v is the decider node of v as well. A clarifying representation can be found in figure 4.3.

Intuitively, v's witnesses verify that v performed the aggregation in a legitimate way. Then, the decider node of v uses the information provided by its witnesses to determine the correct aggregate (performing a majority vote) or to

detect a forgery attempt. Whether the decider is able just to detect forgery or to correct the altered aggregate depends on the security parameter (t, k). In the first case only the *completeness* property is ensured, whereas in the second on also *soundness* is provided. Of course, if correction is required, a higher redundancy (and hence a higher energy consumption) is needed. How these considerations translate in terms of the security parameter (t, k) will be clarified in the following sections.

ESAWN step-by-step description

The ESAWN protocol, as presented in [51, 76], follows the rules listed below:

1. a measuring node v regularly measures its environment and sends the measurements to its t witnesses and to its decider.

2. an aggregating node v waits for sensor readings from all of its sources, applies the aggregation function F_v to the received values and sends the resulting aggregate to v's decider.

3. a witness of a node v waits for values from all the sources of v, computes the aggregation function F_v on those values and sends the resulting aggregate to the decider of v.

4. the decider v' of a node v waits for messages from all the witnesses of v. If only *completeness* is required, the protocol raises an alarm whether two different aggregates are reported, otherwise no action is taken (the aggregate is assumed to be authentic, w.r.t. the relaxed authenticity notion). If *soundness* is required, a majority vote is performed and the aggregate with most instances is chosen as the correct one.

5. witnesses calculations and the majority vote at the decider node are performed synchronously with probability p (the synchronisation mechanisms is described later on).

In the following, ESAWN pseudocode is reported:

Algorithm 9 (ESAWN Protocol)**.**

Code for aggregating node v
// v as decider of v'
for all t-descendants v' of v {
  wait for data from v' and all j-witnesses of v', $0 < j < t$;
  if (SOUNDNESS is required)
    perform majority vote;
  if (mismatch is detected)
    if (only COMPLETENESS is required)
      stop protocol;
}
// v as j-witness of v', $0 < j < t$

for all j from $t-1$ downto 1 do {
    for all j-descendants v' of v {
        wait for data from all sources of v';
        compute the result r of the aggregation function F_v with received data;
        send r to decider of v';
    }
} // v as aggregator
wait for data from all sources of v;
compute the aggregation function F_v with the received data;
send result to decider of v

Code for measuring node v
// v as leaf node
wait for next timing period to begin;
measure value;
send value to all j-witnesses of v, $0 < j < t$, and to the decider of v.

On the use of plain authentication in place of authenticated encryption

The ESAWN protocol proposed in [51] was slightly modified for the purposes of this thesis. Specifically, while the original ESAWN instance is based on authenticated encryption [77], the one considered here is based on plain authentication (i.e. on traditional MACs such as CBCMAC and HMAC). Confidentiality, in fact, is not regarded as a primary goal.

This leads to an important protocol optimisation, which enables a potentially considerable performance improvement[8]. In fact, since the payload sent by each node is not encrypted, each node can send it only once (instead of t times, as required by the original ESAWN instance) in a single message together with t MACs. Furthermore, sending one message per node instead of t, decreases the overhead due to message headers, which for a WSN are typically of 10 bytes [78] per message. Thus, the overall network energy consumption can be notably reduced.

On the choice of the security parameter

The security parameter (t, k) should be chosen carefully by the user, according to the level of security that is required to the authentication scheme. There are mainly two elements that jointly influence the choice of (t, k), namely:

- the fraction of compromised nodes β, which binds the choice of k;
- the required security properties, i.e. *completeness* and/or *soundness*, which bind the choice of t.

If the user wants to provide protection against a fraction β of compromised nodes, he should set $k = \beta \cdot t$. With such a choice, the protocol can ensure authenticity in the presence of a fraction β of compromised nodes on any path of

[8] the (eventual) performance improvement depends on the performances of the authentication scheme chosen in place of the authenticated encryption primitive.

t consecutive nodes in the aggregation tree. If the adversary compromises more than k nodes, instead, authenticity cannot be guaranteed any more.

On the other hand, the value of t must be chosen as the minimum value needed to assert the security requirements. If only *completeness* has to be provided, $t \geq k+1$ has to be chosen, so that at least one honest node out of t consecutive nodes on any path to the root is trustworthy and can detect eventual forgeries. Instead, in order to achieve *soundness* together with *completeness*, it takes $t \geq 2k+1$, so that the majority of nodes in any path of t consecutive nodes is honest and, thus, the voting scheme can extract the correct value.

Summarising, choosing the lower bound in the aforementioned inequalities and combining them, the following reference values are obtained:

- if *completeness* is required

$$\Rightarrow t = k+1,\ k = \lceil \frac{\beta}{1-\beta} \rceil. \tag{4.3}$$

- if *completeness* and *soundness* are required

$$\Rightarrow t = 2k+1,\ k = \lceil \frac{\beta}{1-2\beta} \rceil. \tag{4.4}$$

Therefore, requiring *soundness* in addition to *completeness* involves a much higher redundancy to be introduced in the network. At the same time, though, *soundness* allows the protocol to work properly even in the presence of compromised nodes, while, if only *completeness* is ensured, ESAWN just detects forgeries and is successively forced to stop the current aggregation process.

Probabilistic relaxation

Since the redundancy introduced by ESAWN can significantly increase the network communications required with respect to a non-authentic aggregation protocol (which is indeed a lower bound for an authentic one, at least in terms of communication overhead), the authors of [51] propose a probabilistic relaxation of the authenticity notion. Specifically, verifications are triggered only with a certain probability p, $0 < p \leq 1$. Therefore p is a further parameter by which the ESAWN protocol can be adjusted.

Synchronisation among sensor nodes. Nodes taking part into the verification process need to be time-synchronised, so that they can activate the verification protocol simultaneously. In ESAWN, *synchronisation* is achieved by means of a pseudo-random number generator (PRNG), fed with a seed or internal state shared by all the sensor nodes that take part in the verification.

The random verification, referred to an aggregation node v, is performed according to the following steps:

1. all sources of v, the k-ancestors of v and v itself pick the same pseudo-random number r, $0 < r \leq 1$, using a PRNG initialised with a shared seed.

2. if $r \leq p$, the ESAWN verification described in algorithm 9 (§4.5.2) is executed, otherwise v just forwards its aggregate to its 1-ancestor in G.

3. all children, the k-ancestors and v itself update their PRNG's internal state.

As at most k nodes in a row are compromised, the attacker possibly knows the PRNG seed and can then forecast when the verification process is performed. The authors of ESAWN claim that this is not a security problem, since "[...] although the attacker will know when successful forgery is possible , he cannot control p or whether an actual aggregation from v is verified. The attacker can still successfully forge aggregates with probability $(1 - p)$". This argument can be argued, as the possibility for a node v to predict verifications within a range of t nodes could be a serious security flaw. Therefore, we believe that other synchronisation techniques should be investigated[9].

Defining $\mathcal{P}$. Given p, i.e. the probability of triggering the verification process, it is desirable to estimate the overall probability of having authentic aggregates at the root node; this probability is denoted by $\mathcal{P}$.
First of all, the probability that an aggregate is authentic can be expressed as the sum of the probability that the aggregating node is honest and of the probability that the aggregating node is compromised, but verification is triggered. Namely:

$$p_{auth} = (1 - \beta) + (\beta \cdot p).$$

Assuming that each individual aggregation is statistically independently authentic with probability p, the overall probability $\mathcal{P}$ is given by:

$$\mathcal{P} = (1 - \beta + \beta \cdot p)^{|\mathcal{A}_v|},$$

where $|\mathcal{A}_v|$ is the cardinality of the set of aggregators $\mathcal{A}_v$, which, recalling eq.4.2, can be computed as:

$$|\mathcal{A}_V| = \sum_{i=1}^{h-1} \delta^i = \sum_{i=1}^{log_\delta(1+(\delta-1)n)-2} \delta^i = \frac{n-1}{\delta} - 1.$$

therefore the following equalities hold:

$$\mathcal{P} = (1 - \beta + \beta \cdot p)^{\frac{n-1}{\delta}-1}, \tag{4.5}$$

$$p = 1 - \frac{1 - \sqrt[n-\delta-1]{\mathcal{P}^\delta}}{\beta}. \tag{4.6}$$

[9]for example, assuming that the any t nodes taking part in the verification process of the aggregate submitted by v are in transmission range of one another, the decider node could broadcast a verification request; this, however, would violate the *multihop* nature of communication and, furthermore, is not always a reasonable assumption.

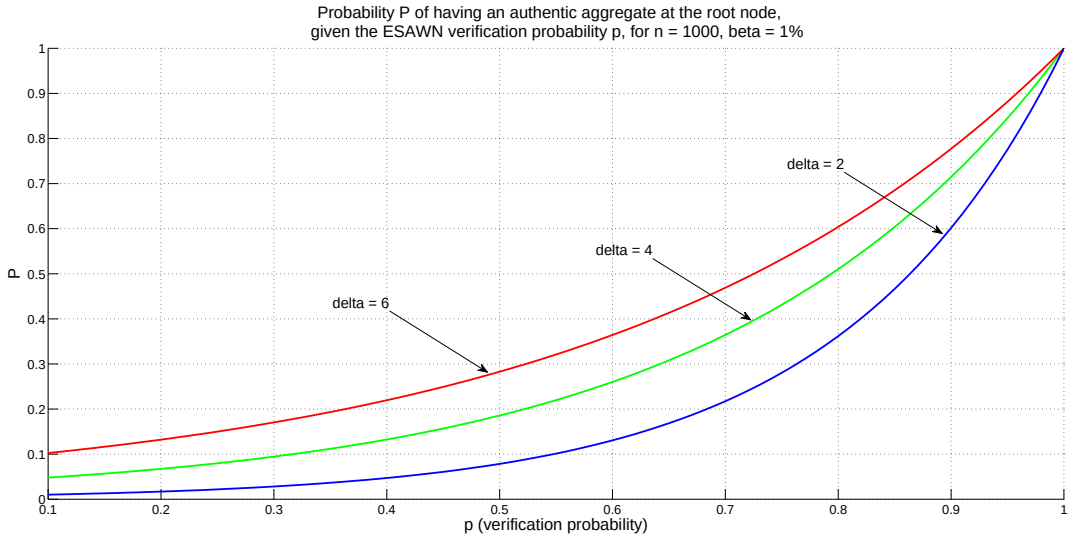


Figure 4.4: Overall probability $\mathcal{P}$ of having authentic aggregates at the BS as a function of the ESAWN verification probability p, given $\beta = 1\%$.

Figure 4.4 shows how $\mathcal{P}$ scales with p, for different values of δ and for $\beta = 0.01$.

4.5.3 Protocol evaluation

In order to evaluate its performances, ESAWN is compared with two simple protocols, which somehow provide an upper and a lower bound. The two considered protocols are:

- *Non-authentic aggregation (NON-AUTH)* - the protocol performs standard aggregation and does not include any security or authenticating mechanism; thus, *NON-AUTH* provides a lower bound to ESAWN complexity.
- *Authentic without aggregation (NON-AGGR)* - the protocol does not perform any data aggregation, but provides full authentication. *NON-AGGR* provides an upper bound to ESAWN complexity.

Under a suitable parameter choice, ESAWN is expected to have better performances than NON-AGGR, whereas it cannot outperform the non-authentic aggregation (NON-AUTH).

Evaluation metrics

A *set of metrics* has been chosen in order to evaluate ESAWN performances. Firstly, some preliminary definitions are given; in particular, let

- n_{nodes} be the total number of nodes in the tree, including the root node.

- $node(i, j)$ represent the j-th node at level i, $1 \leq i \leq h$.
- $n_{src}(i, j)$ be the number of sources (i.e., children) of the j-th node at level i, $1 \leq i \leq h$.
- $t'_i = min(t, i-1)$ (for a given tree level i, $1 \leq i \leq h$) be the minimum between t and the distance (in number of hops) from the root (i.e. $i-1$).
- $nodes(i)$ be the number of nodes at level i.
- hdr be the size (in bytes) of the packet header, pld the size (in bytes) of the packet payload and MAC the size (in bytes) of the authentication code.

Futhermore, please note that all communications are assumed to be multi-hop.
Hence, the following metrics are defined:

1. Number of sent bytes:

   (a) NON-AGGR: Since there is no aggregation in the network, each node (except the root) generates one message and each message, whose size is equal to $(hdr + pld + MAC)$, has to be transmitted over $(i-1)$ hops in order to reach the root node. Hence, the transmitted bytes generated by $node(i, j)$ is:

   $$n_{bytes}^{(1),node(i,j)} = (hdr + pld + MAC) \cdot (i-1). \tag{4.7}$$

   Starting from the previous equation, the overall number of bytes transmitted over the network can be computed as

   $$n_{bytes}^{(1),TOT} = \sum_{i=2}^{h} \sum_{j=1}^{nodes(i)} n_{bytes}^{(1),node(i,j)}. \tag{4.8}$$

   Finally, the average number of sent bytes per node is

   $$n_{bytes}^{(1),AVG} = n_{bytes}^{(1),TOT} / n_{nodes}. \tag{4.9}$$

   Given (4.2) and considering that the dominating term of (4.8) is $(h-1) \cdot nodes(h)$, the overall number of sent bytes using the NON-AGGR protocol scales with $O(n \cdot \log(n))$ and, therefore, the number of sent bytes per node scales with $O(\log(n))$.

   (b) NON-AUTH: Thanks to aggregation, each node transmits only one non-authenticated message, whose size is equal to $(hdr + pld)$. Therefore, the number of bytes transmitted due to $node(i, j)$ is

   $$n_{bytes}^{(2),node(i,j)} = (hdr + pld), \tag{4.10}$$

the overall number of transmitted bytes over the networks, which scales with $O(n)$, is

$$n_{bytes}^{(2),TOT} = \sum_{i=2}^{h} \sum_{j=1}^{nodes(i)} n_{bytes}^{(2),node(i,j)}, \tag{4.11}$$

and the average number of sent bytes per node, which scales with $O(1)$, is

$$n_{bytes}^{(2),AVG} = n_{bytes}^{(2),TOT} / n_{nodes}. \tag{4.12}$$

(c) ESAWN: Each node $node(i,j)$ causes the transmission of just one message of size $hdr+pld+MAC$ with probability $(1-p)$ (i.e., if upper level witnesses do not prompt for an ESAWN verification, see §4.5.2). If a verification is requested (probability p), $node(i,j)$ transmits its payload plus t'_i MACs to its parent node, who has to transmit $node(i,j)$'s payload plus $(t'_i - 1)$ MACS to its parent and so on; therefore, the overall number of bytes transmitted due to $node(i,j)$ is

$$\begin{aligned} n_{bytes}^{(3),node(i,j)} &= hdr + (pld + MAC)(1-p) + (\sum_{l=1}^{t'_i}(pld + l \cdot MAC))p = \\ &= hdr + (pld + MAC)(1-p) + (t'_i \cdot pld + \frac{t'_i(t'_i+1)}{2} \cdot MAC) \cdot p \end{aligned} \tag{4.13}$$

Hence, the overall number of transmitted bytes over the network, which scales with $O(n)$, is

$$n_{bytes}^{(3),TOT} = \sum_{i=2}^{h} \sum_{j=1}^{nodes(i)} n_{bytes}^{(3),node(i,j)}, \tag{4.14}$$

and the average number of sent bytes per node, which scales with $O(1)$, is

$$n_{bytes}^{(3),AVG} = n_{bytes}^{(3),TOT} / n_{nodes}. \tag{4.15}$$

2. Number of aggregation operations.

(a) ***NON-AGGR***:

Since there is no aggregation operation in NON-AGGR, all the relative metrics are identically equal to zero, that is:

$$n_{agg.ops.}^{(1),node(i,j)} = 0, \tag{4.16}$$

$$n_{agg.ops.}^{(1),TOT} = \sum_{i=2}^{h-1} \sum_{j=1}^{nodes(i)} n_{agg.ops.}^{(1),node(i,j)} = 0, \tag{4.17}$$

$$n_{agg.ops.}^{(1),AVG} = n_{agg.ops.}^{(1),TOT}/n_{nodes} = 0, \tag{4.18}$$

where $n_{agg.ops.}^{(1),node(i,j)}$ is the number of aggregation operations caused by $node(i,j)$, $n_{agg.ops.}^{(1),TOT}$ is the overall number of aggregation operations performed in the whole network and $n_{agg.ops.}^{(1),AVG}$ is the average number of aggregation operations per node.

(b) ***NON-AUTH***:

In NON-AUTH protocol, each node, except the leaves and the BS (which is not considered in this analysis, as it is not a classical sensor node, but rather a powerful device, possibly with unlimited energy resources), performs exactly one aggregation operation. Therefore, the overall number of aggregation operations, which scales with $O(n)$, is

$$n_{agg.ops.}^{(2),TOT} = \sum_{i=2}^{h-1} \sum_{j=1}^{nodes(i)} n_{agg.ops.}^{(2),node(i,j)}, \tag{4.19}$$

and the average number of aggregation operations per node, which scales with $O(1)$, is

$$n_{agg.ops}^{(2),AVG} = n_{agg.ops.}^{(2),TOT}\ n_{nodes}. \tag{4.20}$$

(c) ***ESAWN***:

In ESAWN, each node $node(i,j)$, except the leaves and the BS (for the same reasons as for NON-AUTH), performs only one aggregation operation with probability $(1-p)$ and $t'_i = min(t, i-1)$ operations with probability p. Summarising, the number of aggregation operations performed by $node(i,j)$ is

$$n_{agg.ops.}^{(3),node(i,j)} = 1 \cdot (1-p) + t'_i \cdot p = (1-p) + min(t, i-1) \cdot p, \tag{4.21}$$

and the overall number of aggregation operations performed in the whole network is

$$n_{agg.ops.}^{(3),TOT} = \sum_{i=2}^{h-1} \sum_{j=1}^{node(i)} n_{agg.ops.}^{(3),node(i,j)}. \tag{4.22}$$

Again, while the overall number of aggregation operations scales with $O(n)$, the number of aggregation operations per node scales with $O(1)$.

3. Number of computed MACs.

(a) ***NON-AGGR***:

In the NON-AGGR protocol, each node, except the BS, computes a single MAC, that is

$$n_{mac.ops.}^{(1),node(i,j)} = 1. \tag{4.23}$$

Therefore, the overall number of computed MACs is given by

$$n_{mac.ops.}^{(1),TOT} = \sum_{i=2}^{h} \sum_{j=1}^{nodes(i)} n_{mac.ops.}^{(1),node(i,j)} = n_{nodes} - 1. \tag{4.24}$$

(b) ***NON-AUTH***:

Of course, in NON-AUTH no MAC is computed, therefore

$$n_{mac.ops.}^{(2),node(i,j)} = 0. \tag{4.25}$$

$$n_{mac.ops.}^{(2),TOT} = \sum_{i=2}^{h} \sum_{j=1}^{nodes(i)} n_{mac.ops.}^{(2),node(i,j)} = 0. \tag{4.26}$$

(c) ***ESAWN***:

In ESAWN, each node $node(i,j)$ has to compute only two MACs with probability $(1-p)$, while, with probability p, it causes $(2 \cdot t'_i)$ MAC computations, of which t'_i are performed by $node(i,j)$ itself and t'_i are performed by $node(i,j)$'s t'_i witnesses. Therefore, the average number of MAC computations caused by node $node(i,j)$ is

$$n_{mac.ops.}^{(3),node(i,j)} = 2 \cdot (1-p) + 2 \cdot t'_i \cdot p. \tag{4.27}$$

The overall number of MAC computations in the network is

$$n_{mac.ops.}^{(3),TOT} = \sum_{i=2}^{h} \sum_{j=1}^{nodes(i)} n_{mac.ops.}^{(3),node(i,j)}. \tag{4.28}$$

Using the metrics defined above, the overall protocol energy consumption for NON-AGGR, NON-AUTH and ESAWN can be approximated with the sum of the energy spent for radio transmissions (which is directly proportional to the number of sent bytes) and of the energy spent for local computations (which is mainly influenced by MAC computations and, thus, is directly proportional to the number of computed MACs). In the following chapter experimental results, based on different authentication schemes and on random aggregation trees, are provided.

On ESAWN memory consumption

ESAWN's memory consumption scales with $O(1)$ with respect to the number of network nodes. Each $node(i, j)$, in fact, has to manage at most $t \cdot n_{src}(i, j)^t$ incoming messages and at most t authentication keys.

4.5.4 Summary

In the previous sections the ESAWN protocol was thoroughly described and some contributions were given. Firstly, the use of traditional authentication schemes in place of authenticated encryption, which, together with some modifications in the way of assembling data for transmission, leads to a considerable performance improvement, as described in §4.5.2. Secondly, a suitable framework for performance evaluation was defined. In particular, the adjustable protocol parameters were considered together with their potential impact on the protocol security and on its performance.

ESAWN, while trying to address the problem of end-to-end authenticity in a data aggregation protocol, offers a good trade off between energy consumption and probabilistic authenticity. Unfortunately, there are some issues that should be further investigated; in particular, the synchronisation mechanism has to be improved, in order not to be predictable by the attacker.

In the next chapter, performance of ESAWN is experimentally evaluated by means of simulations. Its achievements and its weaknesses are analysed in detail as applied to a randomly generated wireless sensor network; the results are combined with those obtained by the implementation of some MAC algorithms on real sensor nodes, in order to yield a more accurate estimate of ESAWN's performance.

Chapter 5

Experimental results

As a conclusion to the study of joint secure data aggregation and authentication techniques in wireless sensor networks, some experimental results are presented in this chapter.

Firstly, three authentication algorithms, CBCMAC-RC5, HMAC-MD5 and HMAC-SHA1, has been implemented for TelosB motes [81] and their performances are evaluated directly on this platform.

Secondly, the ESAWN protocol has been analysed. Specifically, its metrics has been evaluated by means of *Monte-carlo simulations* and energy consumption has been estimated by exploiting the experimental results achieved for the three selected MAC schemes.

In the remainder of this chapter, the overall outcomes of the aforementioned analysis are presented.

5.1 Evaluation of MAC protocols on TelosB motes

5.1.1 Experimental settings

TelosB motes

For the implementation of authentication algorithms, the chosen platform are *TelosB* motes [81]. TelosB mote is an open source platform designed for low power wireless sensor networks. The one used in this thesis is TPR2420, which is provided with a CC2420 radio chip [82] together with a MSP430 (MSP430F1611) microcontroller. TPR2420 main features are:

- IEEE 802.15.4/ZigBee compliant RF transceiver.
- 2.4 to 2.4835 GHz transmission band (globally compatible ISM).
- 250 Kbps data rate.
- integrated onboard antenna.
- 8 MHz TI MSP430 16-bit microcontroller with 10 KB of RAM.

- low current power consumption.
- 1 MB external flash for data logging.
- programming and data collection via USB.
- sensor suite including integrated light, temperature and humidity sensor.
- runs TinyOS 1.1.10 or higher.

TPR2420 is powered by two AA batteries. If the TPR2420 is plugged into the USB port for programming, power is provided by the host computer. A detailed description is beyond the scope of this thesis; further information can be found in [81].

TinyOS and nesC

TelosB motes operate on TinyOS, which is the *de-facto* standard for wireless sensor nodes operating systems. TinyOS, developed by UC Berkeley, is a lightweight operating system specifically designed for low-power wireless sensors and differs from most other operating systems in that its design focuses on ultra low-power operation.

TinyOS applications and systems, as well as the OS itself, are written in the *nesC* language. nesC is a C dialect optimised to reduce RAM and code size, enable significant performance improvements and help in preventing low-level bugs like race-conditions. A nesC application consists of one or more *components* wired to form an application executable. Each component defines a set of used and provided *interfaces* and provides their implementation.

There are two types of components in nesC: *modules* and *configurations*. Modules provide implementations for one or more interfaces, while configurations wire other components together, connecting interfaces used by components to interfaces provided by others. On the other hand, interfaces may specify a set of *commands*, i.e., functions to be implemented by the interface provider, and/or a set of *events*, i.e., functions to be implemented by the interface user.

An in-depth presentation of TinyOS programming is not in the purposes of this thesis; thereby, the reader interested in a comprehensive description should refer to, e.g, [83].

5.1.2 MAC schemes experimental results

Three MAC schemes were implemented on TelosB motes, namely CBCMAC-RC5, HMAC-MD5 and HMAC-SHA1 (theoretical descriptions can be found in chapter 4). The algorithms were written in nesC, basing on the ones provided in [80, 79]; further improvements to these algorithms performances would be desirable, though they already ensure good performances. In the following paragraphs, a performance evaluation and a comparison among the different schemes is provided. Three main metrics are chosen to compare the aforementioned MAC

schemes: ROM and RAM size, time complexity and throughput, complexity at the aggregator node. The algorithms were run on TelosB motes and metrics has been computed and averaged over 250 iterations.

ROM and RAM size

Memory consumption, both in ROM and in RAM, is a crucial aspect to consider while evaluating an algorithm. The memory consumptions of the considered MAC schemes are listed in table 5.1.

Algorithm	ROM Size (bytes)	RAM Size (bytes)
CBCMAC-RC5	2924	132
HMAC-MD5	11578	88
HMAC-SHA1	4438	28

Table 5.1: ROM and RAM occupation of the considered MAC schemes.

Furthermore, considering that TinyOS default memory occupation is equal to 1398 bytes in ROM and 4 bytes in RAM, the introduced overhead can be computed; results are shown in table 5.2.

Algorithm	% ROM Overhead	% RAM Overhead
CBCMAC-RC5	109	3200
HMAC-MD5	728	2100
HMAC-SHA1	217	600

Table 5.2: ROM and RAM occupation overhead of the considered MAC schemes.

We observe that HMAC-MD5 consumes much more ROM than the other two schemes. This is due to the MD5 hash function (see §4.3.1), which requires a considerable amount of ROM for storing its predefined parameters, such as the order for accessing source words and the number of bit positions for left shift. On the contrary, CBCMAC-RC5 consumes less bytes in ROM, but needs a considerable amount of RAM to be executed, probably in order to perform the key expansion algorithm (algorithm 4) and to store the temporary encrypted blocks of the CBCMAC scheme. HMAC-SHA1 appears to be the best trade off between ROM and RAM usage, though, as described later on, this comes at the price of a considerably increased time complexity.

MAC Time complexity and throughput

In figure 5.1 the time complexity of the considered MAC schemes is plotted against the number of bytes over which the MAC is computed. The time values were computed as the mean of the times measured over 250 iterations performed by a TelosB mote. Several observations arise while looking at this figure.

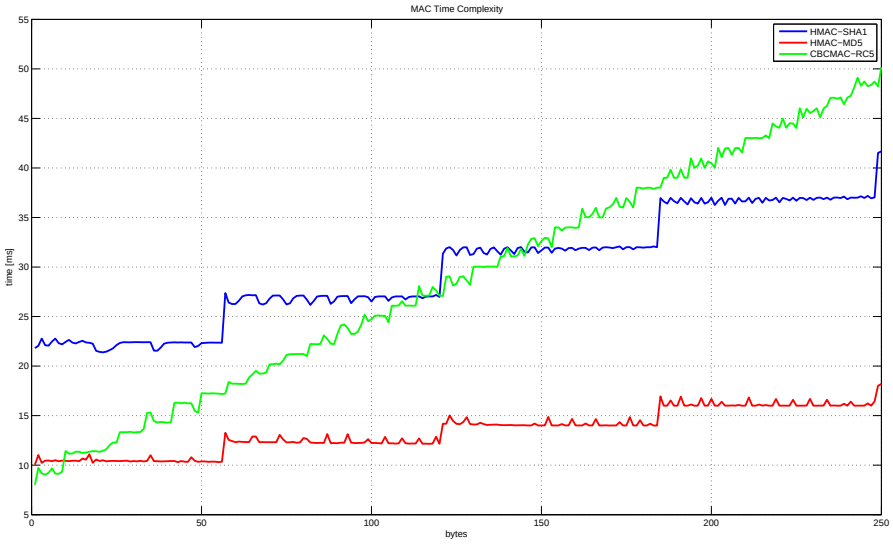


Figure 5.1: Average MAC time complexity as a function of the input size (average over 1000 algorithm iterations).

First of all, it is evident that HMAC-MD5 definitely outperforms the other two schemes. The notable speed achieved by HMAC-MD5 is mainly due to the one of the underlying hash function, i.e., MD5, which was indeed designed to be a very fast algorithm. This comes at the price of a higher ROM usage and of weakened security level (as explained in §4.3.1), that, anyways, at the present moment, is suitable for WSN applications.

The second important feature that emerges from figure 5.1 is the different trend of HMAC-MD5 and HMAC-SHA1 with respect to the one of CBCMAC-RC5. In fact, while the latter scales almost linearly with the size of the authenticated data, the one of HMAC-MD5 and HMAC-SHA1 is locally constant with a step in correspondence of each multiple of 56 bytes. This is due to the definition of MD5 and SHA1, which operate on blocks whose length has to be a multiple of 448 bits (i.e. 56 bytes); therefore, if their input length does not comply with this condition, the message is padded so that its length becomes a multiple of 448 bits.

Finally, it should be noticed that, for inputs with size greater than approximately 150 bytes, CBCMAC-RC5 yields the worst performance among all considered schemes. Therefore, if authentication of long messages is required, HMAC-MD5 and HMAC-SHA1 are preferable.

Similar considerations can be made by observing figure 5.2, which shows the average throughput of the considered MAC schemes. Again, while the trend of CBCMAC-RC5 is regular, the one of HMAC-MD5 and HMAC-SHA1 presents discontinuities in correspondence of multiples of 56 bytes, for the same reason explained above. The higher throughput is the one achieved by HMAC-MD5,

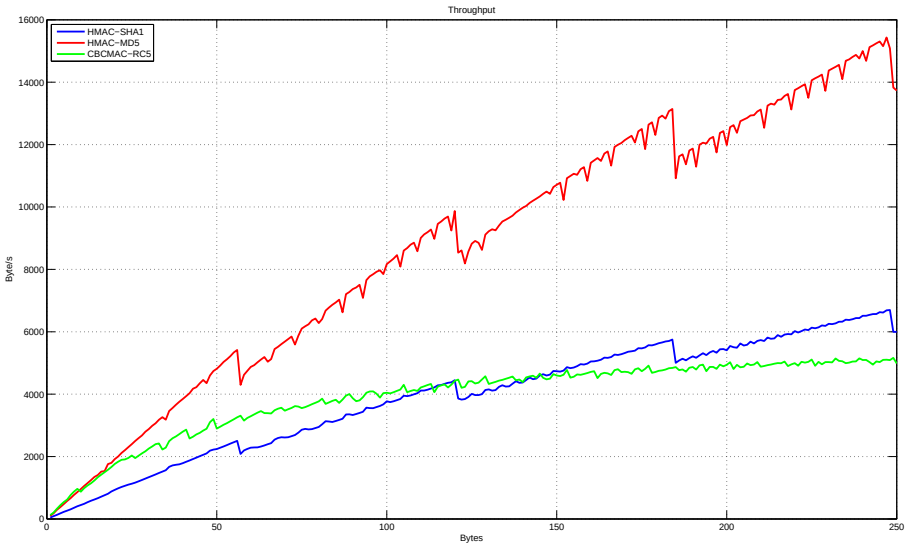


Figure 5.2: MAC throughput as a function of the payload size.

while CBCMAC-RC5 ensures better performances of HMAC-SHA1 only up to 150 bytes; after this threshold CBCMAC-RC5 throughput degrades, though the disadvantage on HMAC-SHA1 is not remarkable.

MAC complexity at the aggregator node

Finally, the MAC time complexity at the aggregator node is considered. For the purposes of the present thesis, this time complexity is evaluated when authenticity verification is performed hop-by-hop, as the ESAWN protocol requires.
Figure 5.3 shows the overall time required by MAC computations at the aggregator node, depending on the payload size and on the number of sources of the considered aggregator node. These times include the time spent by all the aggregator sources for computing their own MAC and the time spent by the aggregator node itself in order to compute the MAC of the aggregate.
Of course, the time complexity grows linearly with the number of sources, but a couple of issues should be observed. Firstly, the time spent by an aggregator for MAC computations could range from a few milliseconds to some seconds, depending on the payload size and on the number of sources of the aggregator node. Spending several seconds of CPU time for MAC computations, involves an excessive overhead for a resource-scarce aggregator; thus, it would be desirable to lower the payload size (e.g. exploiting some efficient data representation) and to limit the number of children, so that the authentication overhead does not grow excessively. On the other hand, the higher the number of sources, the higher is the advantage ensured by data aggregation. Therefore, a suitable trade off between optimal data aggregation and acceptable authentication overhead should be found.

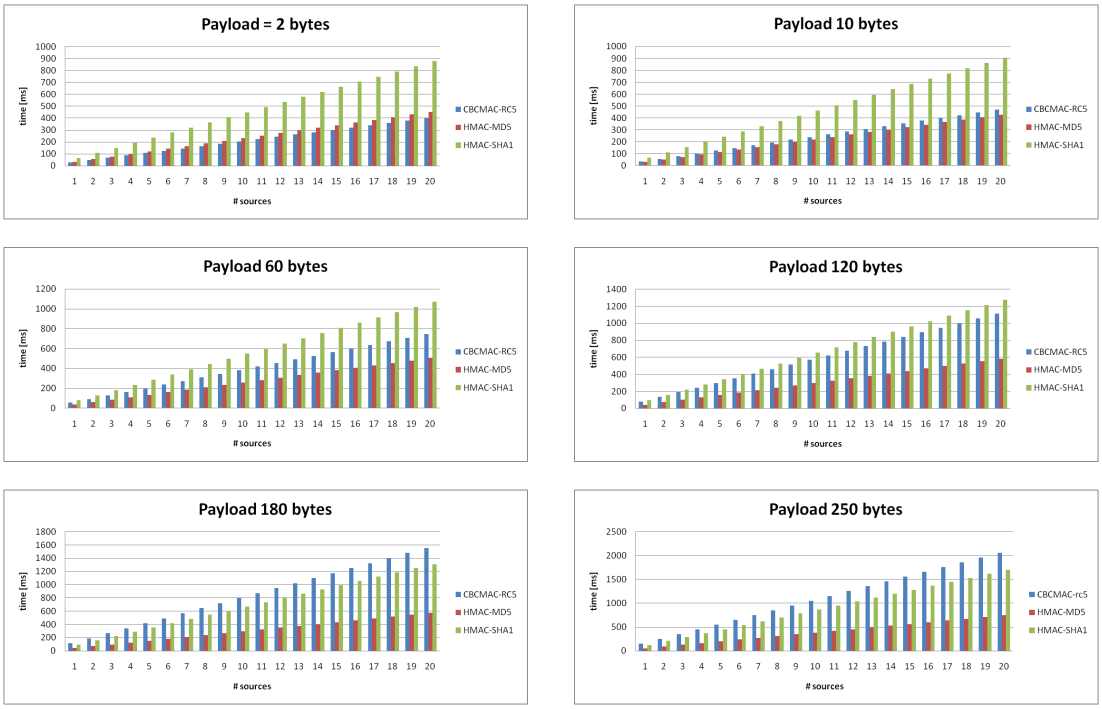


Figure 5.3: Overall MAC time complexity at the aggregator node, assuming hop-by-hop authentication, depending on the payload size and on the number of sources.

Summary and conclusions

From the previous metrics evaluation, the following facts emerges:

1. HMAC-MD5 is globally the fastest among the proposed authentication schemes. This comes at a price of a much higher ROM occupation, which could possibly be a problem in certain applications. Furthermore, HMAC-MD5 is presumably the weaker among the considered authentication schemes, as the MD5 hash function was found to be vulnerable to several attacks; it seems, anyways, that HMAC-MD5 does not suffer from the same vulnerabilities as the MD5 algorithm alone and, therefore, its use can still be considered a suitable choice.

2. HMAC-SHA1 can be regarded as the strongest authentication algorithm among the three presented here. Its ROM and RAM occupation is rather low and balanced, while its time complexity is typically higher than in the other schemes, which could be a major problem in certain applications.

3. CBCMAC-RC5 performance is between those of HMAC-MD5 and HMAC-SHA1 for inputs of size up to 150 bytes, but they degrade afterwards. Furthermore, even though CBCMAC-RC5 ROM occupation is the lowest among the three algorithms, its RAM occupation is considerably higher than for the other solutions.

Summarising:

- HMAC-MD5 is the best choice when ROM occupation is not a concern and as long as MD5 vulnerabilities are not found to threaten HMAC-MD5 security.

- CBCMAC-RC5 is a suitable choice for applications where RAM occupation is not a major concern and where the message to authenticate is smaller than approximately 150 bytes. Furthermore, as explained in §4.2.3, a CBCMAC can be used only when inputs are of fixed size; therefore, for applications with varying input sizes, CBCMAC-RC5 should be discarded a-priori.

- HMAC-SHA1 offers a good trade off between the two previous solutions, providing stronger security, a balanced ROM and RAM occupation and acceptable performance.

5.2 Experimental evaluation of ESAWN

5.2.1 Experimental settings

In order to evaluate the performance of ESAWN, an algorithm for creating a random tree was used. This algorithm takes as input the number of tree levels h and the range $[n_1, n_2]$ of children (sources) each node can have. At each iteration, a random value in $[n_1, n_2]$ is picked for each node, starting from the root node and descending to the leaves. For each set of values (h, n_1, n_2) the random tree generation algorithm is run several times. Furthermore, in order to compute the metrics described in §4.5.3, the specification of the path redundancy t and of the verification probability p are required.

Metrics are calculated for a single complete cycle of data gathering and aggregation, starting from data sensing and ending with the final aggregates reported to the BS. Metrics are evaluated for randomly generated trees over a number *iter* (in the order of a few hundreds or thousands) of iterations and finally the average of all gathered values is computed.

To summarise, metrics are estimated depending on the following parameters:

- h, the number of tree levels, with $h > 0$;

- $[n_1, n_2]$, the range of possible sources of each node;

- t, the path redundancy of ESAWN;

- p, the ESAWN verification probability;

- *iter*, the number of algorithm iterations

The performance of ESAWN is compared against those of NON-AGGR (the authenticated, non-aggregated protocol) and with the ones of NON-AUTH (the non-authenticated, aggregated protocol), which, ideally, should bound ESAWN performance, at least in terms of energy consumption. In the remainder of this section, the analysis of simulation results is presented.

5.2.2 ESAWN simulation results

Number of sent bytes

According to the metric definitions given in §4.5.3, the number of bytes transmitted by the ESAWN protocol depends on the following parameters:

1. the number of levels h in the aggregation tree,
2. the number of sources for each aggregation node (picked up uniformly at random in $[n_1, n_2]$,
3. the header size hdr, the payload size pld and the MAC size MAC,
4. the path redundancy t,
5. the verification probability p,

while for the other two protocols the number of bytes transmitted depends only on the first 3 parameters.

For the present evaluations, the following sizes for the packet header, payload and MAC are assumed:

- $hdr = 15$ bytes, as specified by TinyOS jointly with the 802.15.4 standard;
- $pld = 2$ bytes, as this value is assumed to be sufficient to store a single sensor reading;
- $MAC = 4$ bytes (truncated MAC), as it is considered to provide enough security for a WSN [38].

Please note that increasing the size of these fields improves the advantage of ESAWN over the NON-AGGR protocol, as ESAWN has to send less packets than NON-AGGR, thanks to its nature (each sensor measurement has to be forwarded for at most t levels) and to the optimisation introduced in §4.5.2.

In figure 5.4, the total number of transmitted bytes for each of the three protocols is reported, with h varying between 4 and 8, $[n_1, n_2] = [2, 8]$, $t = 2$ and $p = 1$. Results are averaged over 100 iterations.

It can be noticed that, using ESAWN and the NON-AUTH protocol, the overall number of bytes transmitted in the network does not increase its growth rate with the number of tree levels h, i.e., their slope remains constant for different tree depths. Furthermore, the straight line of ESAWN has a slope that is slightly

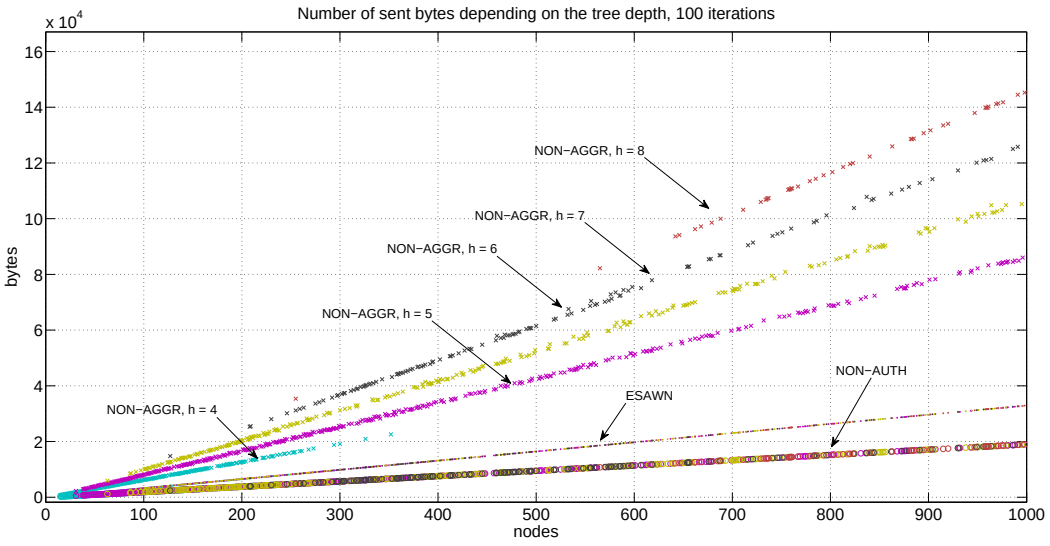


Figure 5.4: Total number of transmitted bytes for NON-AGGR, NON-AUTH and ESAWN, with h varying between 4 and 8, $[n_1, n_2] = [2, 8]$, $t = 2$ and $p = 1$

higher than NON-AUTH's, because of the MACs that have to be transmitted by ESAWN in addition to the payload.

On the contrary, the slope of the lines corresponding to NON-AGGR suffers a considerable increase for each newly added tree level; this is due to the fact that each forwarded packet has to be transmitted to the root node by means of multihop communication. For example, in a complete tree with δ children per node, raising the tree depth from h to $h + 1$ involves the transmission of $h \cdot \delta^h \cdot (hdr + payload + MAC)$ additional bytes.

In terms of sent bytes, the advantage of ESAWN with respect to NON-AGGR is shown in figure 5.5, where the percentage reduction of transmitted bytes achieved by ESAWN w.r.t. NON-AGGR is plotted for different values of h. The simulation parameters are $p = 1$, $t = 2$, $[n_1, n_2] = [2, 8]$, $iter = 100$. As it can be noticed, this advantage grows for increasing tree levels and ranges from approximately 25% for $h = 3$ to nearly 80% for $h = 8$. Furthermore, for a fixed value of t, the gap between percentage reductions relative to consecutive values of h gets smaller as the tree depth increases. This happens because the path redundancy t is spread across the whole tree; therefore, for higher values of h, the impact of a fixed t-redundancy over the total number of bytes sent in the network is reduced.

In ESAWN, the number of sent bytes depends also on t, i.e., the redundancy in the aggregation paths, and on p, i.e., the probability that the ESAWN verification is performed. Obviously, the higher t and p are, the higher the number of transmitted bytes is and, as a consequence, the advantage over NON-AGGR is lower. Figures 5.6 and 5.7 show the percentage reductions in sent bytes for varying t and p, respectively, while the other parameters are fixed. Specifically,

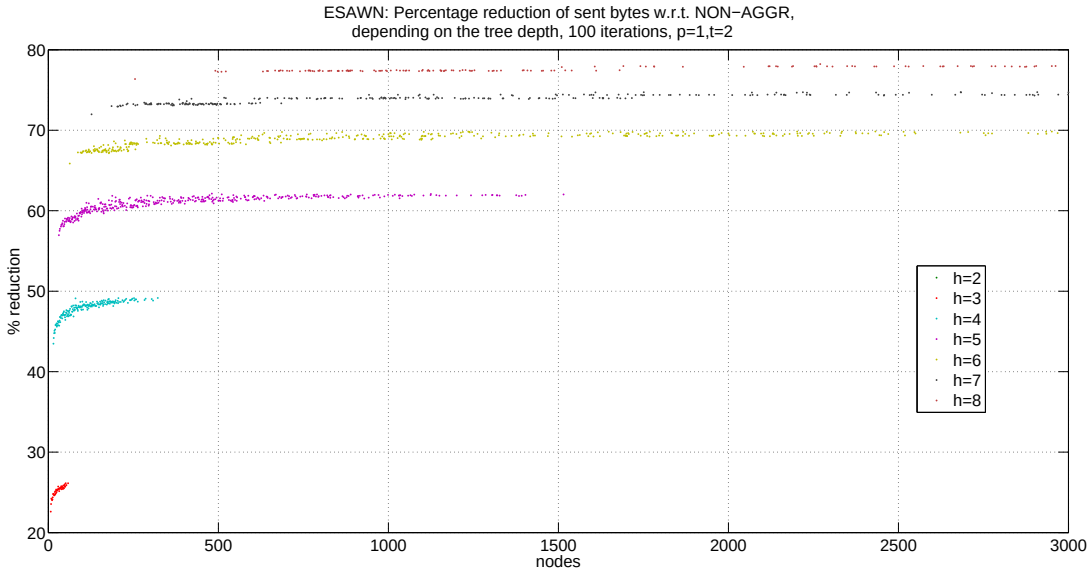


Figure 5.5: Percentage reduction of transmitted bytes achieved by ESAWN w.r.t. NON-AGGR, with h varying between 2 and 8, $[n_1, n_2] = [2, 8]$, $t = 2$ and $p = 1$

figure 5.6 assumes $h = 6$, $[n_1, n_2] = [2, 8]$, $t \in [2, 4]$, $p = 0.8$ and $iter = 100$; figure 5.7 assumes $p \in [0.2, 0.5, 0.8, 1]$, $h = 6$, $[n_1, n_2] = [2, 8]$, $t = 3$. There is no particular observation that has to be made with regard to these figures, except that their trend asymptotically tends to a constant value, which is reached as soon as the number of nodes allows a proper spreading of the redundancy across the whole tree.

Number of aggregation operations

The second evaluated metric is the number of aggregation operations; the results achieved by ESAWN are compared with those of NON-AGGR and NON-AUTH. According to the metric definition provided in §4.5.3, the number of in-network aggregation operations depends on:

- the tree depth h;
- the number of sources for each aggregator (picked up at random in $[n_1, n_2]$);
- the ESAWN path redundancy t;
- the ESAWN verification probability p

Figures 5.8 and 5.9 show how the overall number of aggregation operations and the average number of aggregation operations per node scale with the number of nodes, for different values of h and given $[n_1, n_2] = [2, 8]$, $t = 2$ and $p = 1$. As expected, for the same value of h, the number of aggregation operations performed by ESAWN is approximately t times the one of NON-AUTH, while NON-AGGR does not perform any aggregation operation.

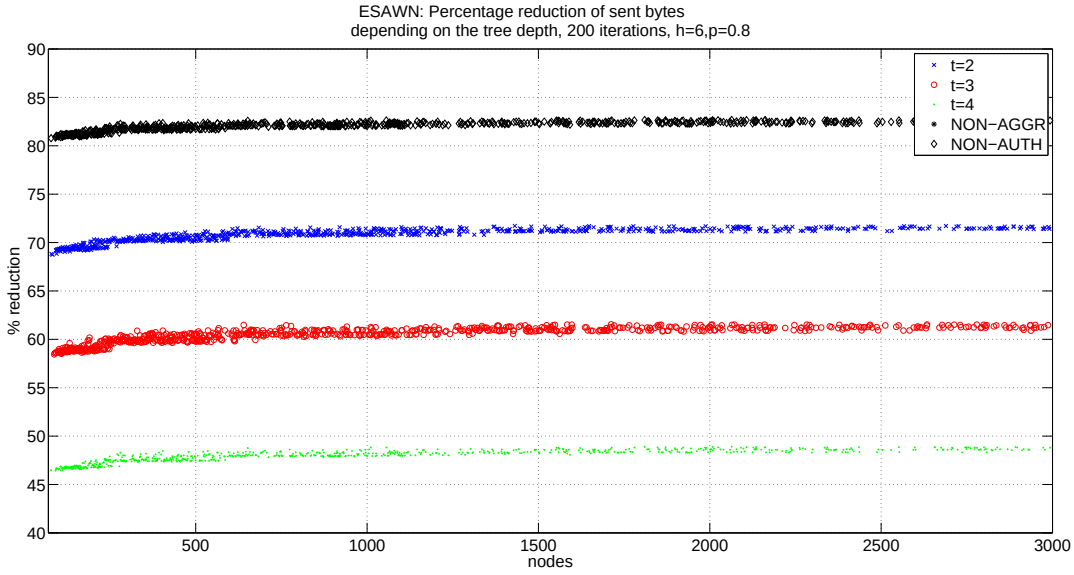


Figure 5.6: Percentage reduction of transmitted bytes achieved by ESAWN w.r.t. NON-AGGR, with t varying between 2 and 4, $[n_1, n_2] = [2, 8]$, $h = 6$ and $p = 0.8$

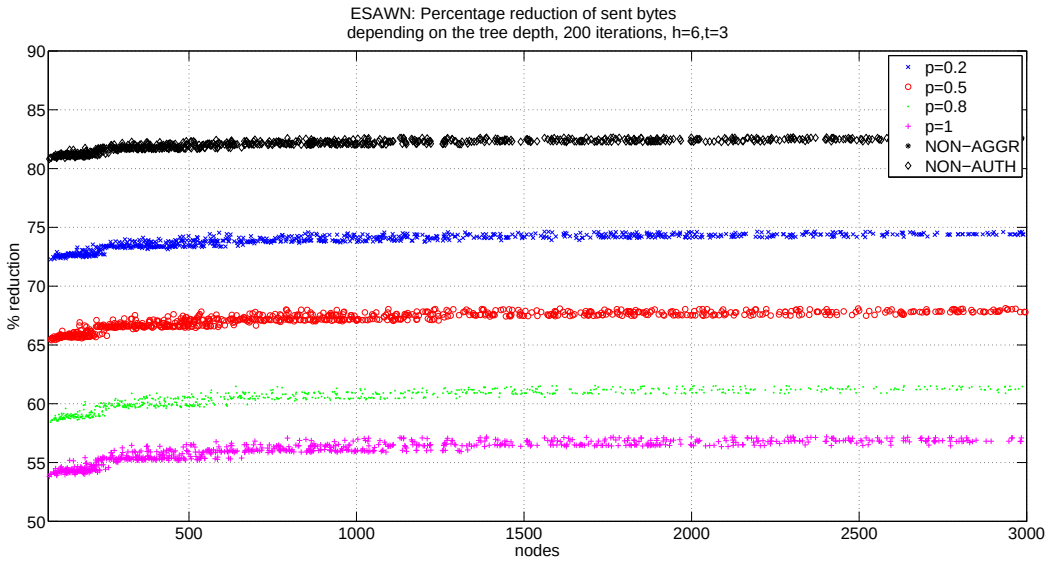


Figure 5.7: Percentage reduction of transmitted bytes achieved by ESAWN w.r.t. NON-AGGR, with $p \in [0.2, 0.5, 0.8, 1]$, $h = 6$, $[n_1, n_2] = [2, 8]$, $t = 3$.

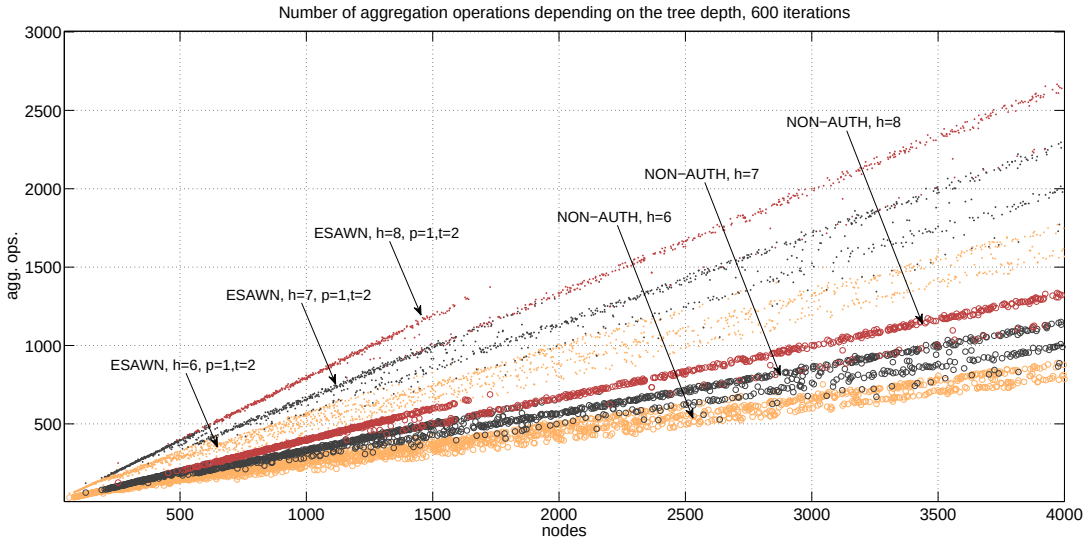


Figure 5.8: Overall number of aggregation operations, with $h \in \{6, 7, 8\}$, $[n_1, n_2] = [2, 8]$, $t = 2$ and $p = 1$

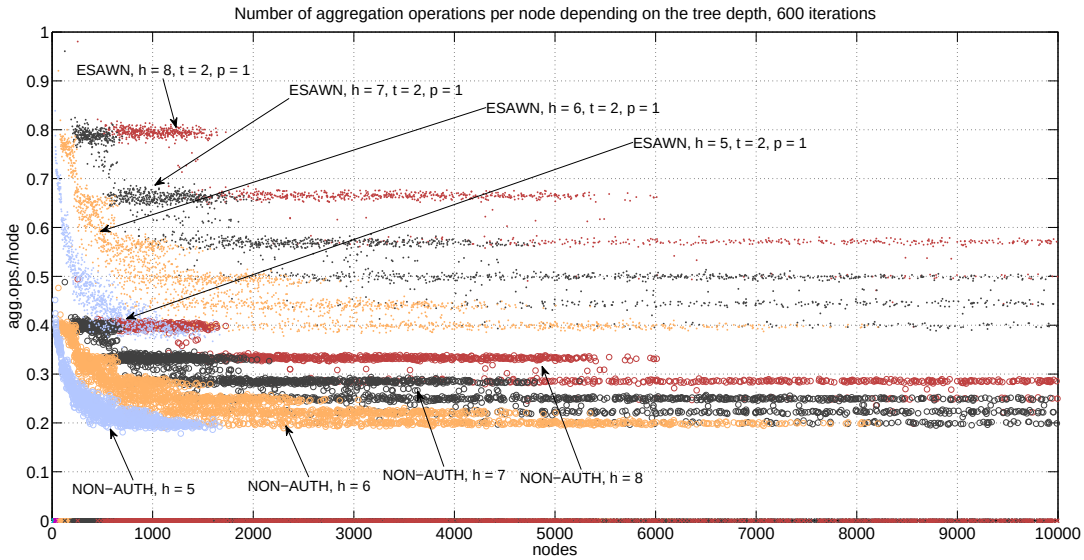


Figure 5.9: Average number of aggregation operations per node, with $h \in \{5, 6, 7, 8\}$, $[n_1, n_2] = [2, 8]$, $t = 2$ and $p = 1$

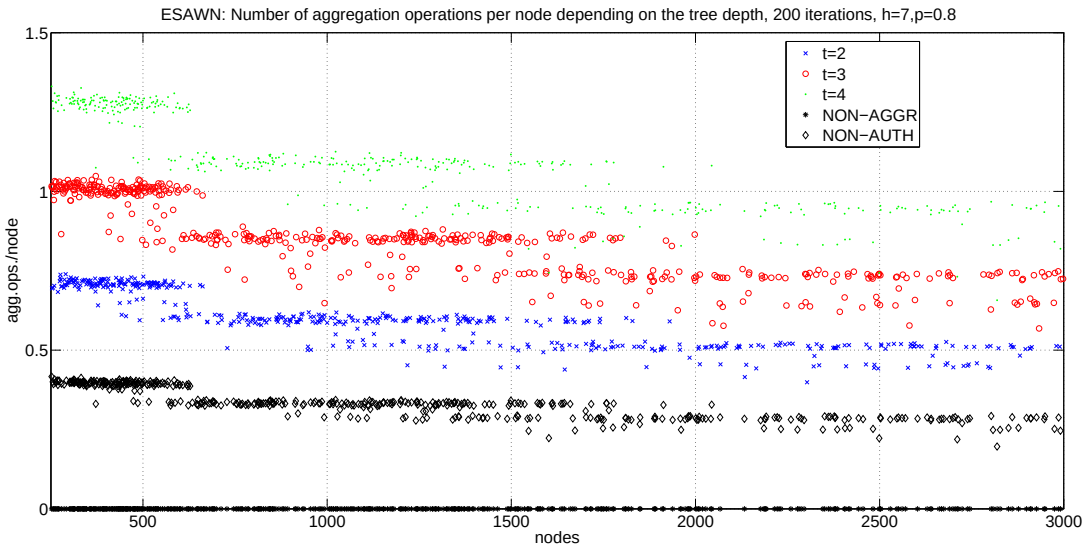


Figure 5.10: Average number of aggregation operations per node for $t \in [2, 4]$, with $h = 7$, $[n_1, n_2] = [2, 8]$, $p = 0.8$ and $iter = 200$

Furthermore a second important observation has to be made; the discontinuities that, for a fixed h, can be observed both in figure 5.8 and in figure 5.9, are due to the fact that the number of sources for each aggregator is picked up uniformly at random in $[n_1, n_2]$. This implies that, once h is fixed, a certain network size is achievable only for higher values of the average number of sources per aggregator node; at the same time, if the number of sources per aggregator node increases, the overall number of aggregation operations required in the whole network proportionally decreases, and so do the corresponding line slope and the average number of aggregation operations per node.

As stated earlier, the number of aggregation operations in ESAWN depends also on the value of t and on the value of p; again, the higher t and p, the higher the number of performed aggregation operations will be.
Figure 5.10 and figure 5.11 show how the number of aggregation operations scales for different values of t and p. Looking at figure 5.10, it can be noticed that, with respect to NON-AUTH, the average number of aggregation operations for ESAWN is approximately increased by $(1-p)+p \cdot t$ times. As a consequence, the discontinuity steps increase their amount for higher values of t, as the discontinuity of NON-AUTH is amplified proportionally to t. On the other hand, observing figure 5.11, we can see that the overhead introduced by ESAWN in terms of average number of aggregation operations per node is, analogously, approximately proportional to $(1 - p) + p \cdot t$.

Number of computed MACs

Finally, the number of computed MACs is considered. According to the metric definition provided in §4.5.3, the number of computed MACs depends on:

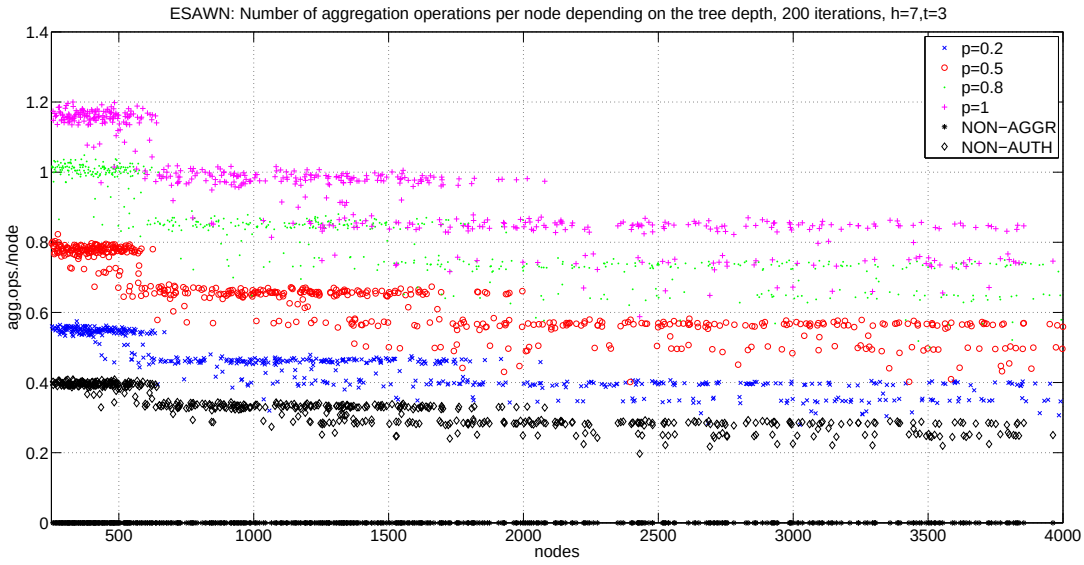


Figure 5.11: Average number of aggregation operations per node for $p \in \{0.2, 0.5, 0.8, 1\}$, with $h = 7$, $[n_1, n_2] = [2, 8]$, $t = 3$ and $iter = 200$

- the tree depth h,
- the number of sources for each aggregator (picked up at random in $[n_1, n_2]$);
- the ESAWN path redundancy t;
- the ESAWN verification probability p.

Figure 5.12 shows how the total number of computed MACs scales with the number of nodes in the network, for $h \in [4, 8]$, $t = 2$ and $p = 1$. Observing this figure, two facts should be noticed.
Firstly, both for NON-AGGR and ESAWN the line slope does not change for different values of h (for NON-AUTH the line is identically equal to 0).
Secondly, the ESAWN line slope is considerably higher than the NON-AGGR one; specifically it is approximately equal to $2\cdot(1-p)+2t\cdot p$ times the NON-AGGR one. This is a consequence of the ESAWN approach, which trades off communication complexity for computational complexity (since local computations are assumed to consume less energy than radio transmissions).

Looking at figure 5.13, the same phenomenon can be observed from another point of view, namely in terms of average MAC computations per node for different values of t. The considered figure assumes $h = 7$ and $p = 0.8$. Here, the proportion between ESAWN and NON-AGGR MAC complexity becomes even more apparent: while NON-AGGR performs only on one MAC computation per node, the average number of computed MACs in ESAWN asymptotically tends to $2 \cdot (1 - p) + 2t \cdot p$.
The asymptotic value, however, is reached only if the tree depth allows it. As the careful reader surely has noticed, in fact, the nodes at a distance smaller than t

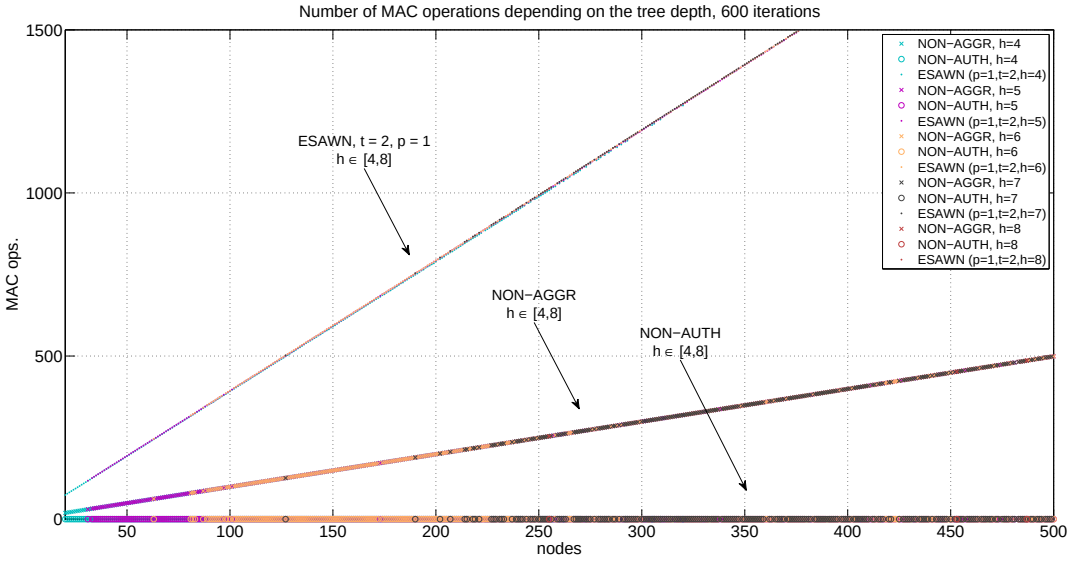


Figure 5.12: Overall number of MAC operations for $h \in [4, 8]$,with $[n_1, n_2] = [2, 8]$, $t = 2$, $p = 1$ and $iter = 200$

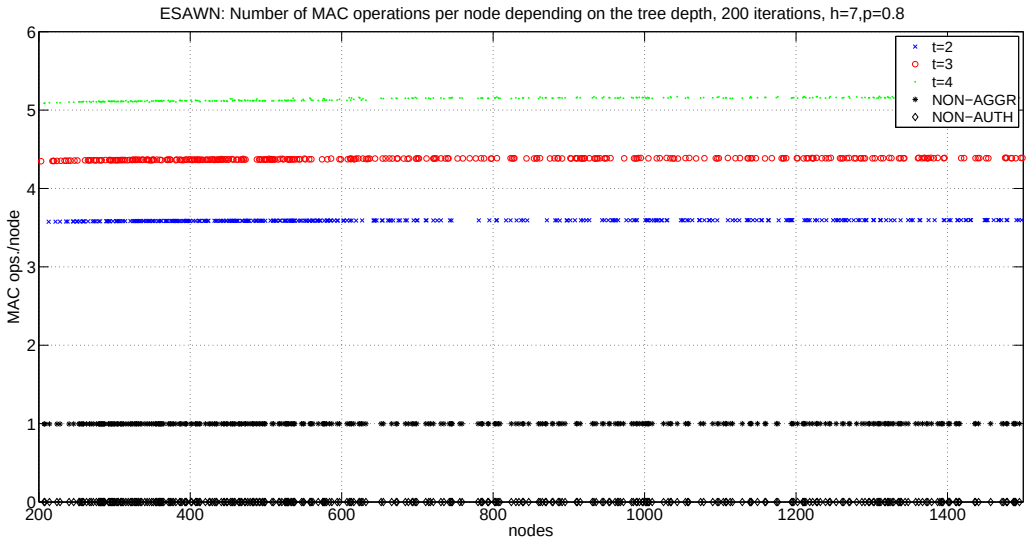


Figure 5.13: Average number of MAC operations per node for $t \in [2, 4]$, with $h = 7$, $[n_1, n_2] = [2, 8]$, $t = 3$ and $iter = 200$

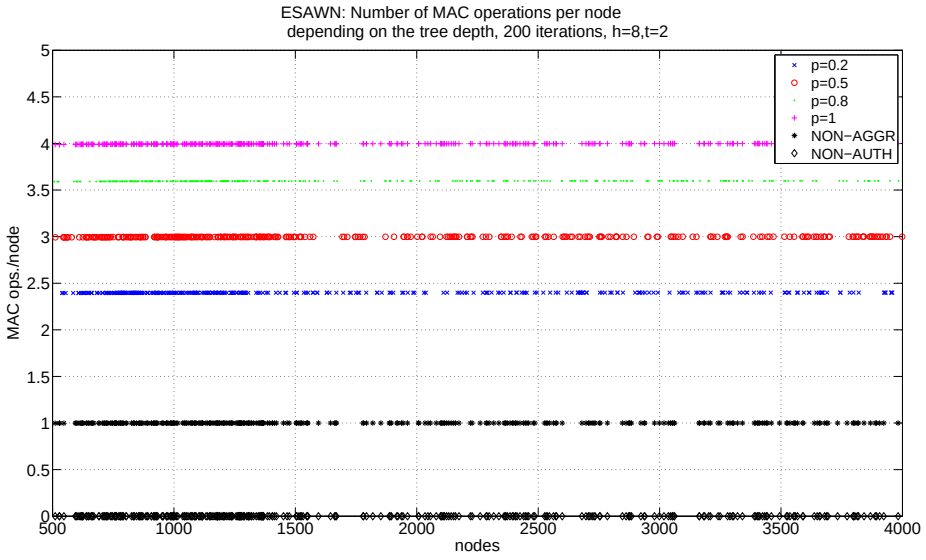


Figure 5.14: Average number of MAC operations per node for $p \in \{0.2, 0.5, 0.8, 1\}$, with $h = 8$, $[n_1, n_2] = [2, 8]$, $t = 2$ and $iter = 200$

from the root do not send t different MACs, but just $t'_i = min(t, i-1)$, where $(i-1)$ is the distance from the root. Therefore if h, i.e., the number of tree levels, is not high enough, the average number of computed MACs does not top $2 \cdot (1-p) + 2t \cdot p$, but it stabilises to a smaller value.

The number of performed MAC operations depends also on p and, in figure 5.14, this dependency is shown, when $h = 8$ and $t = 2$. Also in this case, the average number of MAC operations per node in ESAWN is, as expected, $2 \cdot (1-p) + 2t \cdot p$. This, however, is just an asymptotic value and, as explained for figure 5.13, is reached only if the tree depth allows it.

Summary

In the previous paragraphs ESAWN performance was evaluated according to different metrics and compared with the ones of the NON-AGGR and NON-AUTH protocols. The influence of the choice of t and p was shown, as well as the impact of the network topology (number of nodes, number of levels, children per node, etc.).

The results achieved in this section enable, together with the ones from §5.1, the evaluation of the ESAWN energy consumption, even if with some minor simplifications. This estimate is presented in the next section.

5.3 Energy consumption

Combining the results presented in §5.1 and in §5.2, it is finally possible to estimate the overall ESAWN energy consumption for the TelosB platform, with

respect to the NON-AGGR and to the NON-AUTH protocol and while authentication is provided either with CBCMAC-RC5, HMAC-MD5 or HMAC-SHA1.

The estimated energy consumption is expressed in terms of current absorption (as in [51]), i.e., in $mA{\cdot}s$, according to the reference values that can be found in CC2420 and MSP430 datasheets.

In order to perform this evaluation, some simplifying assumptions are made:

1. the estimated current absorption for radio transmissions considers only the current needed to transmit the plain messages at the predefined bitrate; specifically, the radio start up time, the time for transmitting the preamble, eventual retransmissions (managed by the link layer), etc. are assumed to be negligible with respect to transmission and thus not included in the computation. Disregarding this aspects actually reduces ESAWN advantage over NON-AGGR, as ESAWN sends less messages and, thus, the additional overhead involved by start-up time, preamble transmission, etc. would be smaller than the one entailed by NON-AGGR.

2. the time for performing aggregation is assumed to be negligible with respect to the one needed by MAC operations; this is a reasonable assumption if the aggregation operation is simple, e.g., a mean, and the number of sources per aggregator is not excessive; since in this evaluation it is assumed that the aggregation operation is the mean and that the number of sources per aggregator is at most 8, the aggregation complexity is in the order of 1 ms and, therefore, neglecting it does not have a considerable impact on the estimate.

3. only the radio and the CPU current absorption are considered; other circuitry is not taken into account.

4. the radio is assumed to transmit at 0 dBm.

According to these assumptions, in the following sections ESAWN current absorption is evaluated for a network based on TelosB motes. The current absorbed for transmissions and the one used for local computations are evaluated separately in first place and finally combined into the overall protocol current absorption. As for the ESAWN metrics evaluation, also the current absorption estimation is performed over a single cycle of data gathering and aggregation, i.e., starting from data sensing and ending with the final aggregates delivered to the BS.

5.3.1 Radio current absorption

According to [82], the CC2420 radio chip consumes 17.4 mA while transmitting at 0 dBm and 19.7 mA while receiving and it provides a bitrate of 250 Kbps. Assuming the simplifying hypothesis that CC2420 energy consumption is approximated by the energy needed for the plain transmission/reception of the message bytes

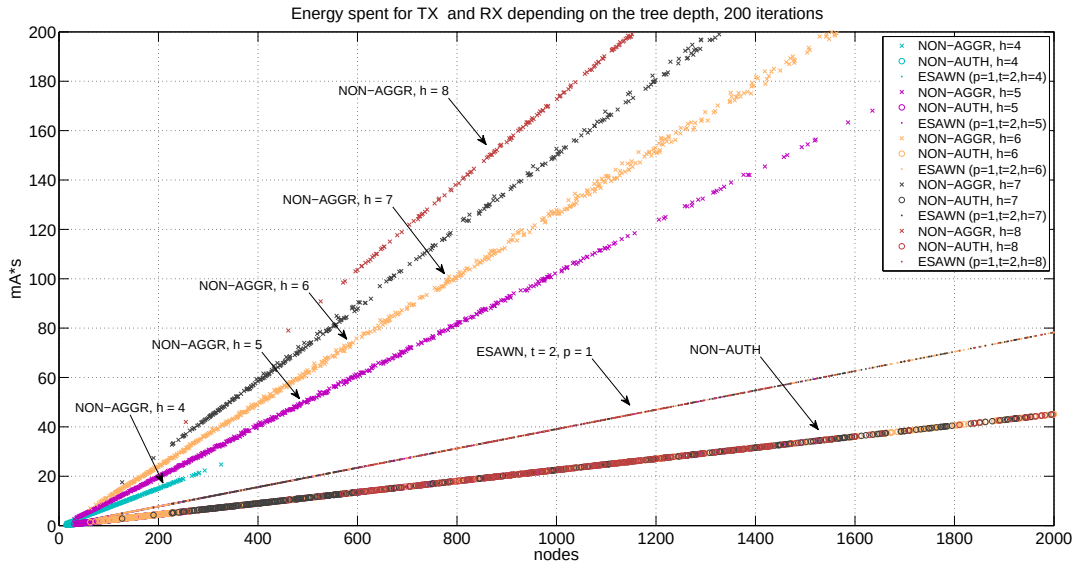


Figure 5.15: Overall TX and RX current absorption for $h \in [4, 8]$, $[n_1, n_2] = [2, 8]$, $t = 2$, $p = 1$ and $iter = 200$

at the predefined bitrate, the current absorption caused by the transmission of 1 byte is:

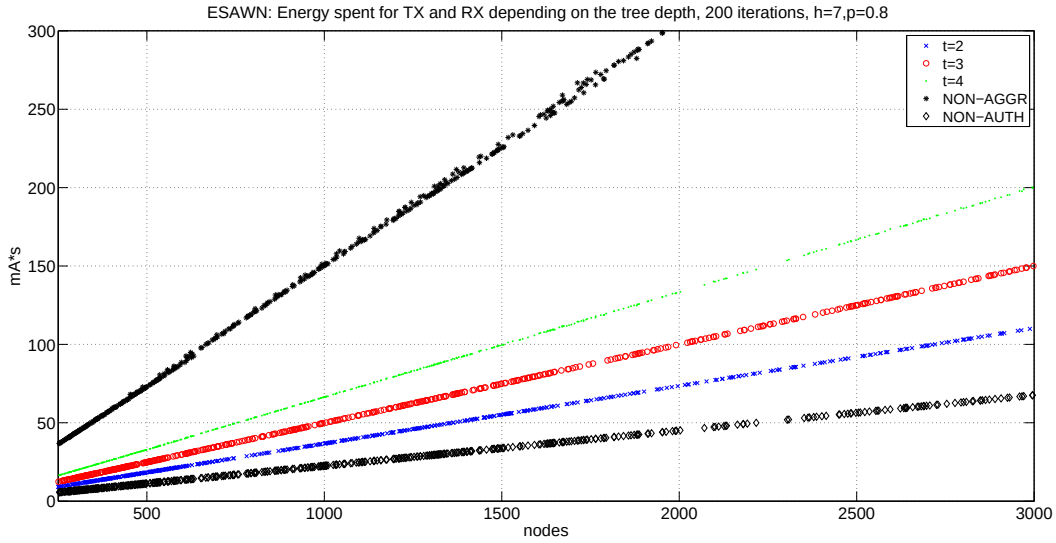
$$I_{radio}(1) = (17.4\ mA + 19.7\ mA) \cdot \frac{8\ bit}{250\ kbps} = 2.968 \cdot 10^{-4}\ mA/byte \qquad (5.1)$$

In order to estimate the current absorption associated to the three different protocols, $I_{radio}(1)$ should be multiplied for the overall number of transmitted bytes. Again, it is assumed that the sensor measurements are composed by 2 bytes and that the MAC size is 4 bytes. Figure 5.15 shows how the overall current absorbed in the network for sending and receiving data scales with the number of sensor nodes, for $h \in [4, 8]$, $[n_1, n_2] = [2, 8]$, $t = 2$ and $p = 1$. The behaviour of the three protocols is expectedly similar to the one observed for the number of sent bytes, as the radio energy consumption is directly proportional to that value. Again, the slope of lines associated to NON-AGGR increases while adding new tree levels; conversely, the one of ESAWN and NON-AUTH remains constant. Furthermore, the slope difference between the line associated to ESAWN and the one associated to NON-AUTH is due to the authentication overhead introduced by ESAWN.

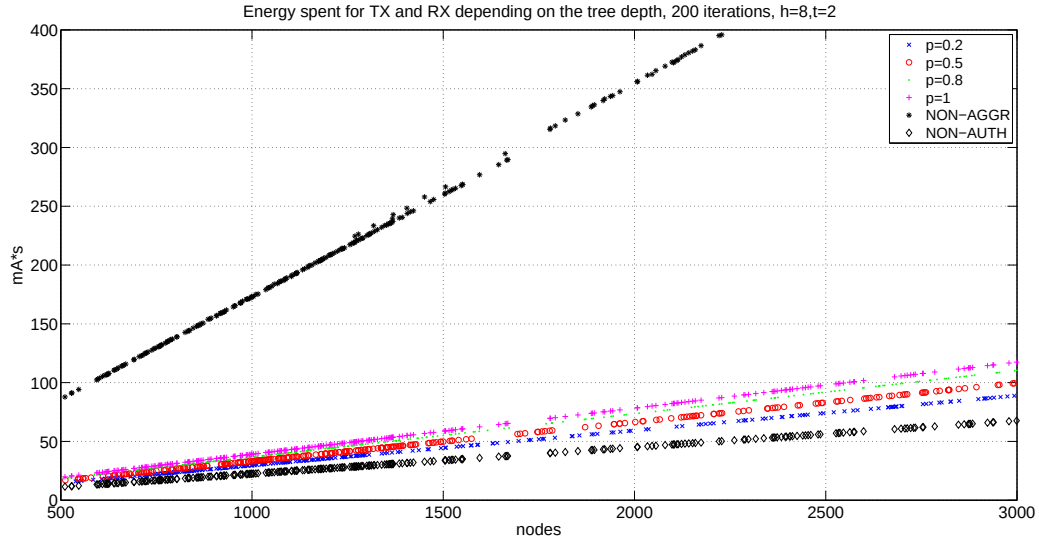
Figures 5.16(a) and 5.16(b) show how the current absorption scales for different values of t and p respectively. Considerations on these figures are analogous to that made for the overall number of sent bytes (see §5.2).

5.3.2 CPU current absorption

The MSP430 is the microcontroller used by TelosB motes in order to perform local computations. According to its datasheet, the MSP430 typically consumes



(a) For different values of t, $h = 7$, $p = 0.8$



(b) For different values of p, $h = 8$, $t = 2$

Figure 5.16: Overall TX and RX current absorption for $h \in [4, 8]$, $[n_1, n_2] = [2, 8]$, $t = 2$, $p = 1$ and $iter = 200$

500 μA while used in active mode. Therefore, in order to evaluate the CPU current absorption, one should multiply this reference value for the amount of time the CPU was used for. In this evaluation, only the CPU time for computing MACs is considered and, furthermore, it is assumed that the message over which the MAC is computed has a payload of 2 bytes.

In figures 5.17(a), 5.17(b), 5.18(a), 5.18(b), 5.19(a), 5.19(b), it is shown how the CPU current absorption scales for different MAC algorithms and for different values of t and p.
Assuming the hypothesis described in §5.3, the CPU current absorption is directly proportional to the number of computed MACs. Therefore the same considerations made for the MACs are valid. Nevertheless, here we can observe the influence that the chosen MAC algorithm has on CPU current absorption.

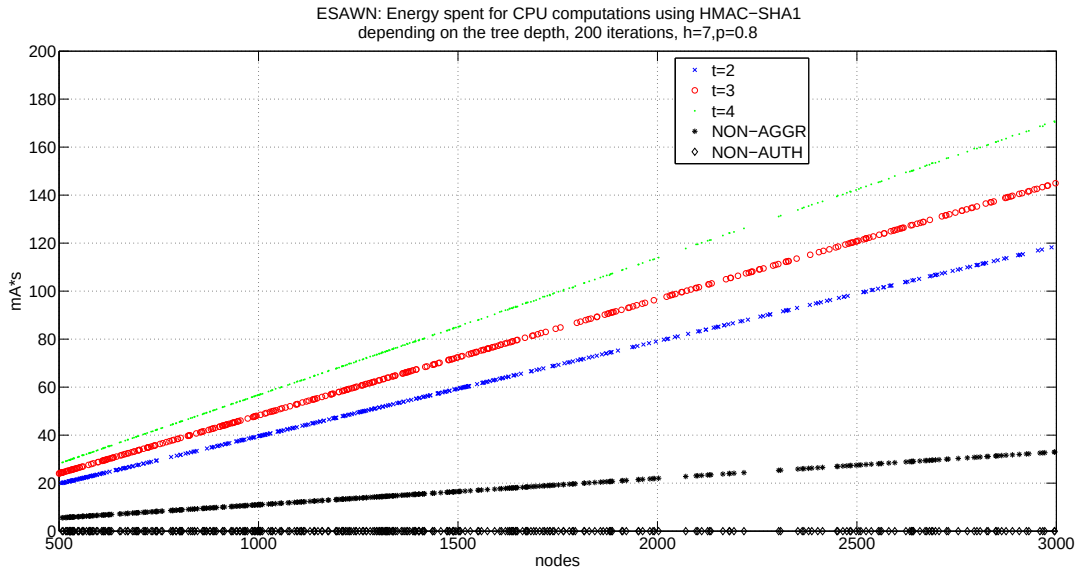
HMAC-SHA1 is, as expected, the most expensive authentication algorithm, while HMAC-MD5 and CBCMAC-RC5 have approximately the same cost, which, for the chosen parameters, is nearly the half of that of HMAC-SHA1.

5.3.3 Overall current absorption

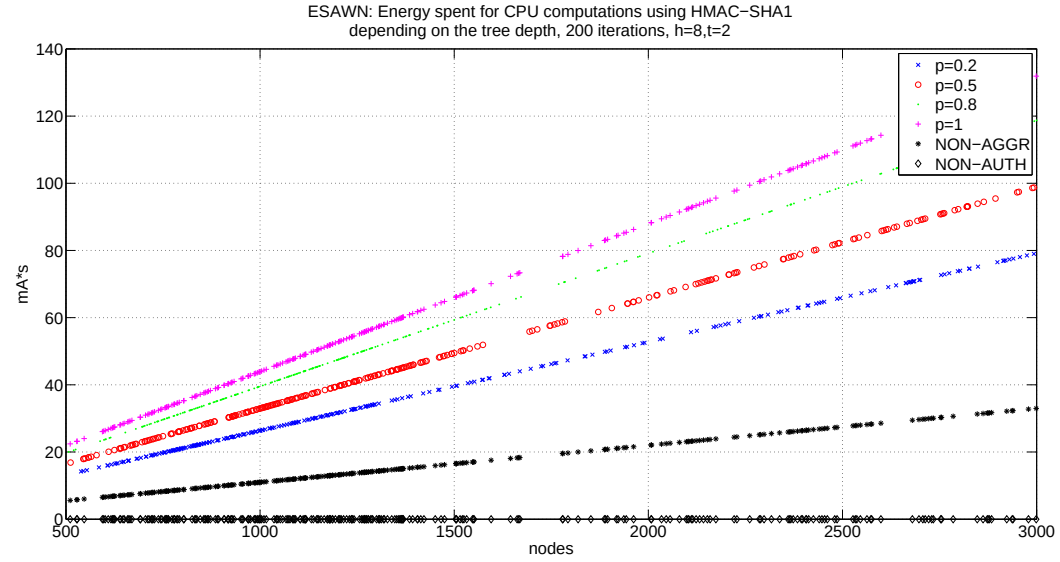
Exploiting the results achieved in the previous sections, it is now possible to estimate how the ESAWN overall current absorption scales with respect to the one of NON-AGGR and NON-AUTH, for different values of h, t and p. Please note that the radio and the CPU current absorption can be summed up, as the CC2420 and the MSP430 operate at the same voltage (i.e. 3 V). Furthermore, it is important to notice that the CPU current absorption overhead introduced by ESAWN w.r.t. NON-AGGR is proportionally smaller than the radio consumption overhead of NON-AGGR w.r.t. ESAWN. This fact confirms that trading off communication complexity for local computation complexity is advantageous in the context of WSNs.

Firstly, it is interesting to look at the percentage reduction of the overall current absorption achieved by ESAWN with respect to NON-AGGR. Figures 5.20(a), 5.20(b) and 5.20(c) show this percentage reduction for $h \in [3, 8]$, $t = 2$ and $p = 1$. As it can be noticed, for almost all the values of h, using HMAC-MD5 or CBCMAC-RC5 ensures an advantage of approximately the 10% over HMAC-SHA1. In fact ESAWN, together with HMAC-MD5 or CBCMAC-RC5, ensures the best performance, achieving up to a 80% reduction, while HMAC-SHA1 reaches at most a 70%. Furthermore, it should be observed that, for $h = 3$, $t = 2$ and $p = 1$, ESAWN causes the absorption of even more current than NON-AGGR (i.e., the percentage reduction is negative, as the current absorption is actually increased), due to the excessive redundancy introduced for the considered tree depth.

Figures 5.21 and 5.22 show how the overall current absorption scales with respect to different values of t and p respectively. It is evident that, even for high values of t and p, ESAWN consumes much less energy than the NON-AGGR protocol. In particular, the average current absorption per node is shown in figures 5.23 and 5.24 for different values of t and p.

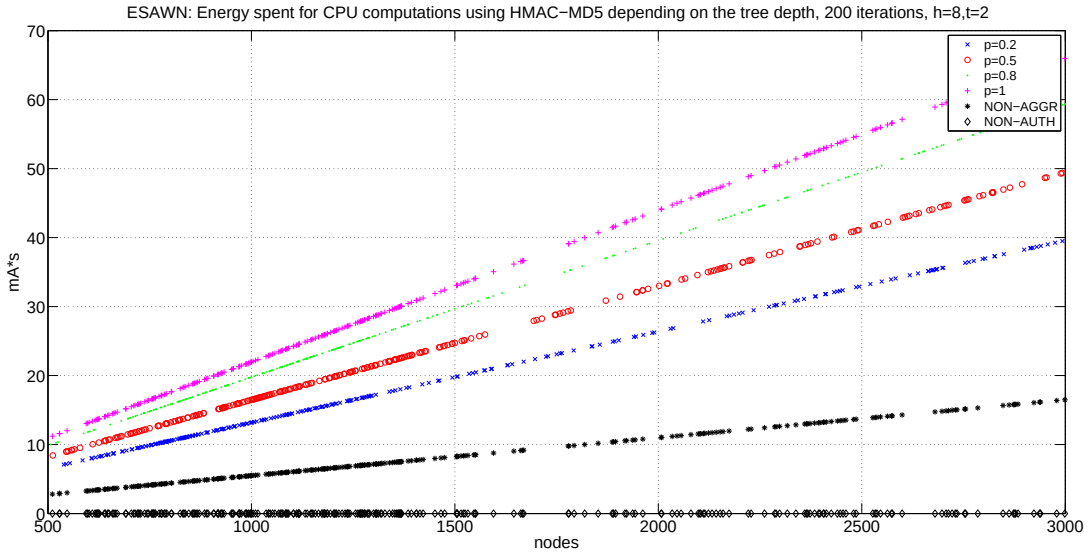


(a) For different values of t, $h = 7$, $p = 0.8$

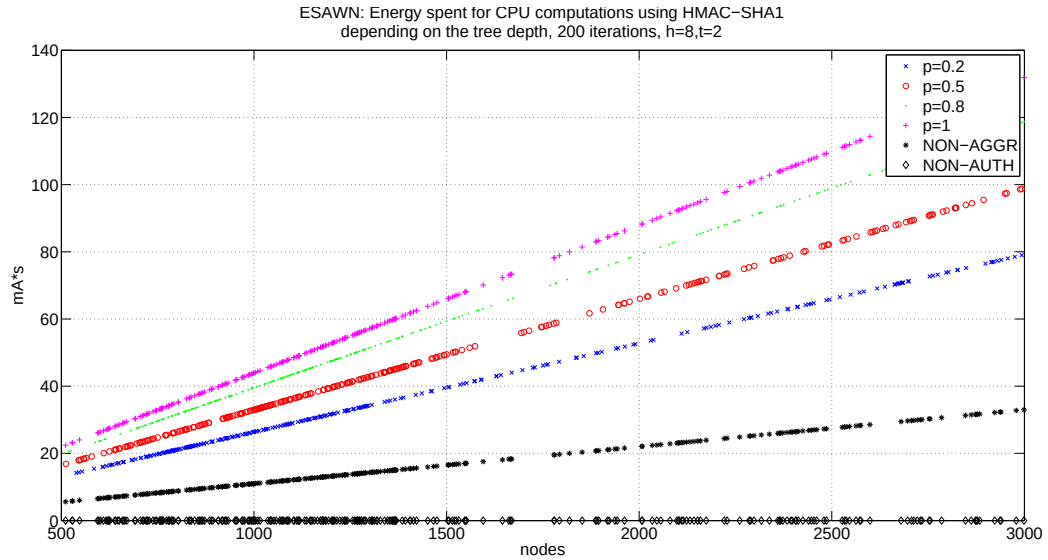


(b) For different values of p, $h = 8$, $t = 2$

Figure 5.17: Current absorbed for CPU computations using HMAC-SHA1, $[n_1, n_2] = [2, 8]$ and $iter = 200$

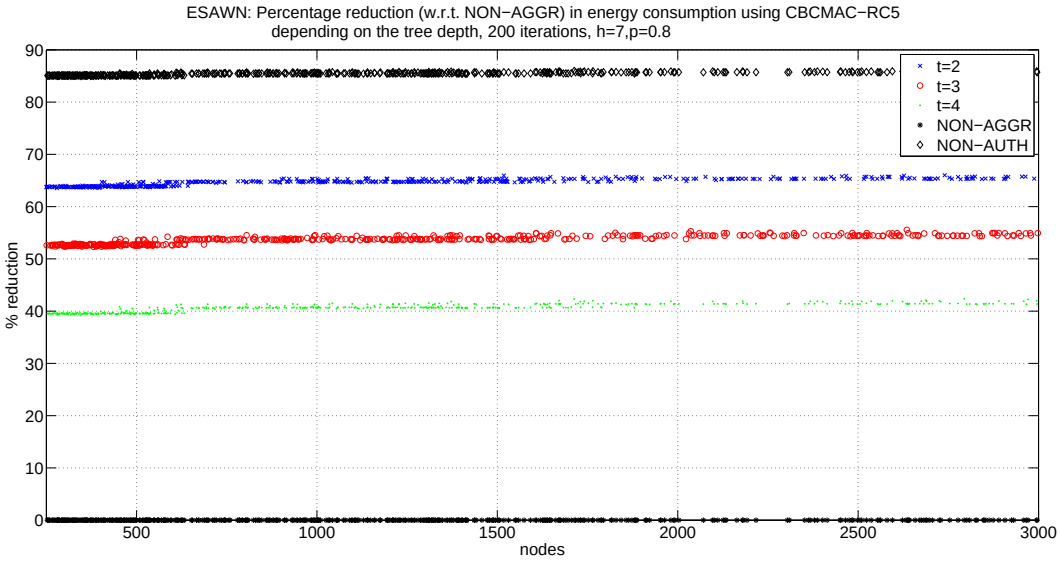


(a) For different values of t, $h = 7$, $p = 0.8$

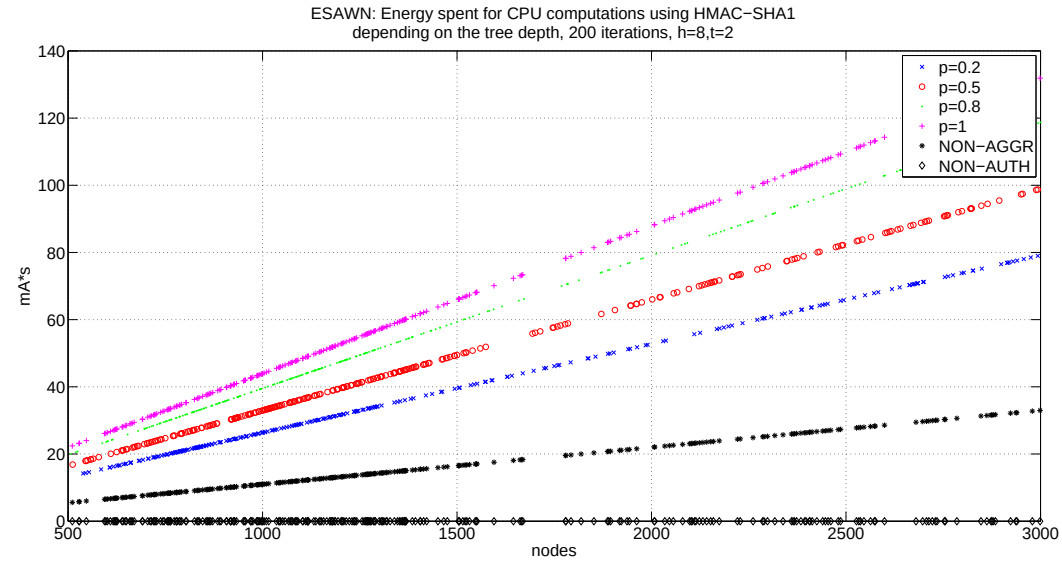


(b) For different values of p, $h = 8$, $t = 2$

Figure 5.18: Current absorbed for CPU computations using HMAC-MD5, $[n_1, n_2] = [2, 8]$ and $iter = 200$

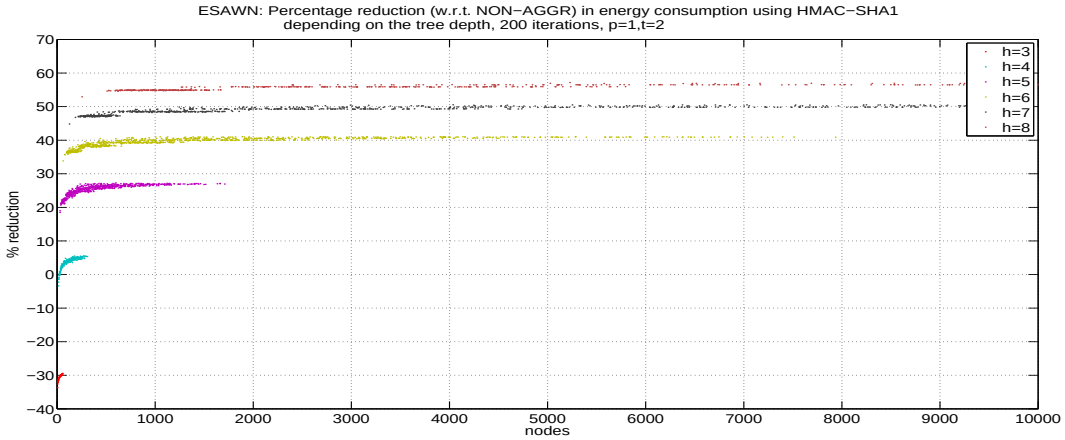


(a) For different values of t, $h = 7$, $p = 0.8$

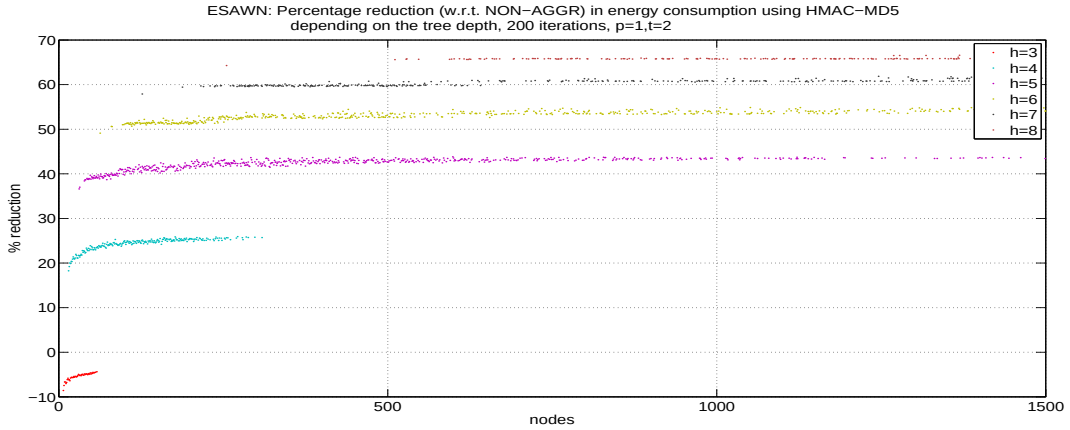


(b) For different values of p, $h = 8$, $t = 2$

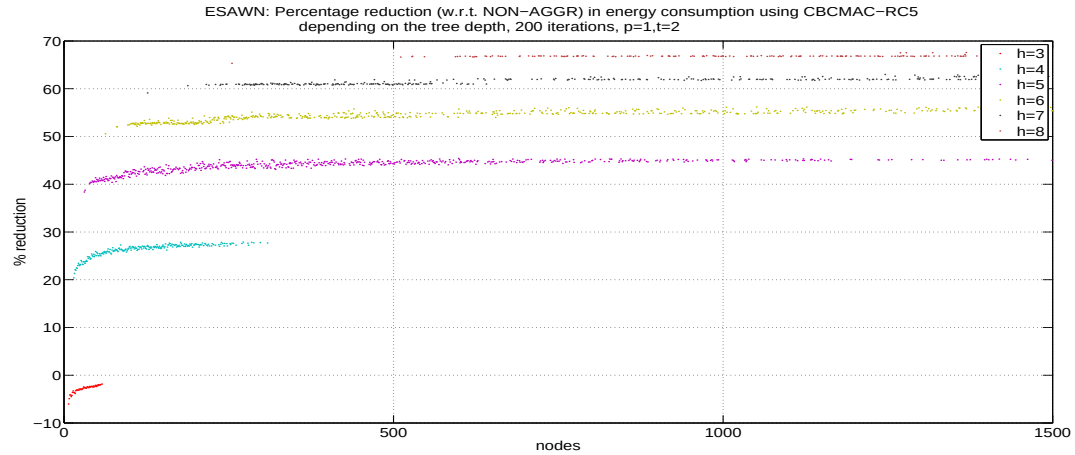
Figure 5.19: Current absorbed for CPU computations using CBCMAC-RC5, $[n_1, n_2] = [2, 8]$ and $iter = 200$



(a) HMAC-SHA1

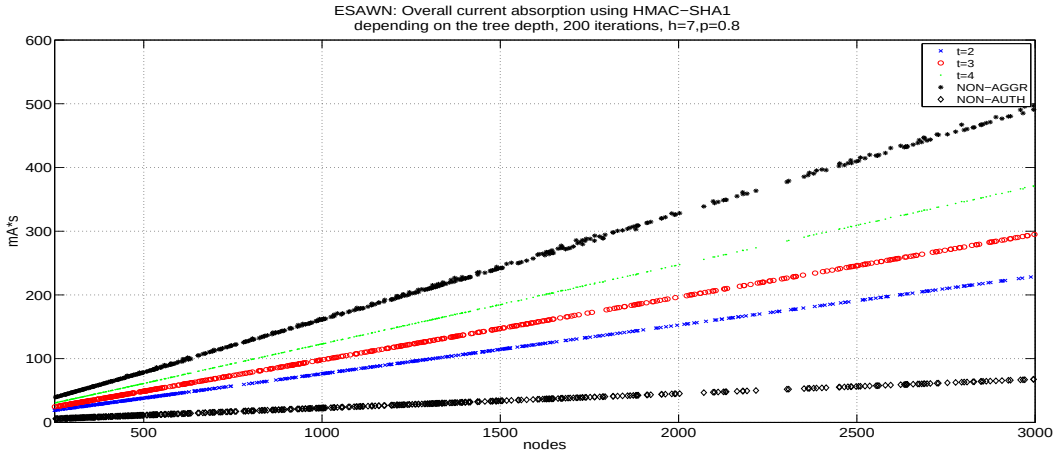


(b) HMAC-MD5

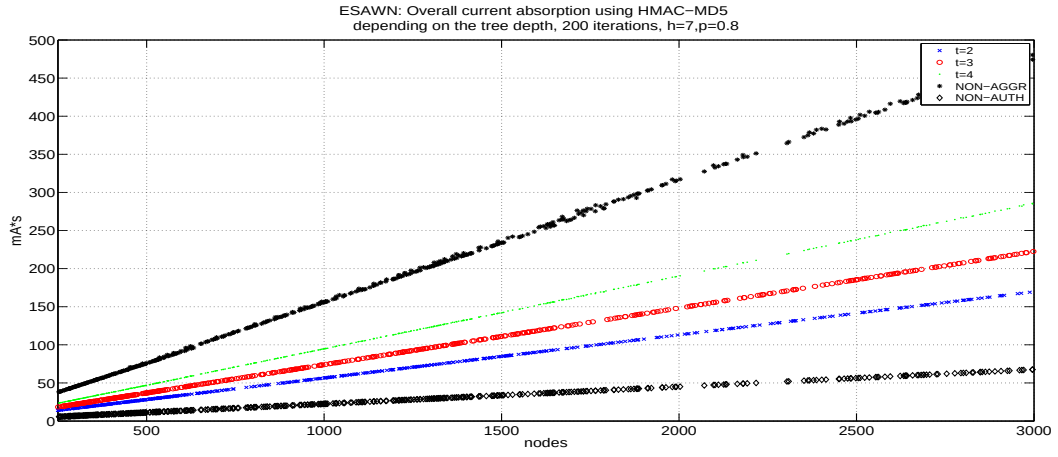


(c) CBCMAC-RC5

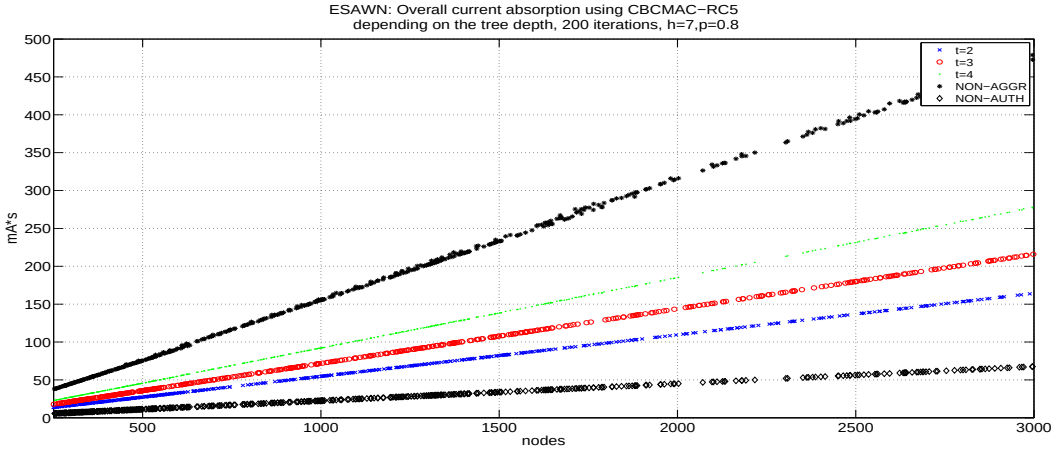
Figure 5.20: Percentage reduction (w.r.t. NON-AGGR) in current absorption with $h \in [3, 8]$, $[n_1, n_2] = [2, 8]$, $t = 2$, $p = 1$ and $iter = 200$



(a) HMAC-SHA1

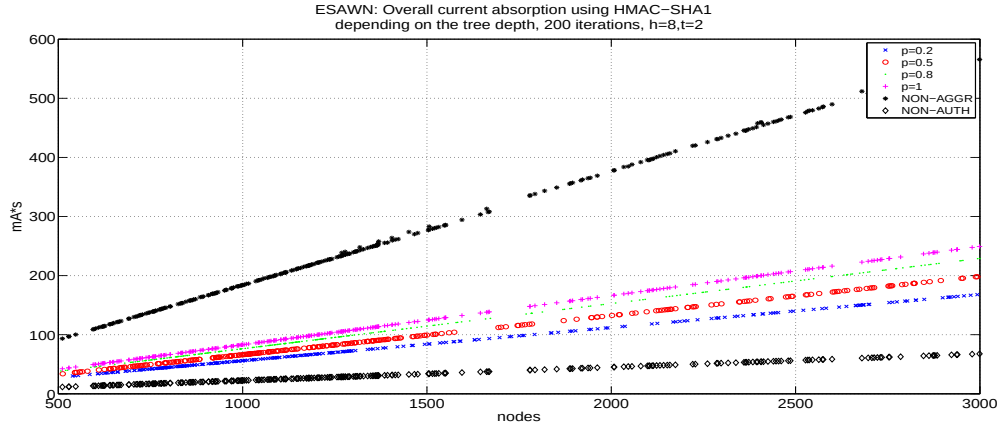


(b) HMAC-MD5

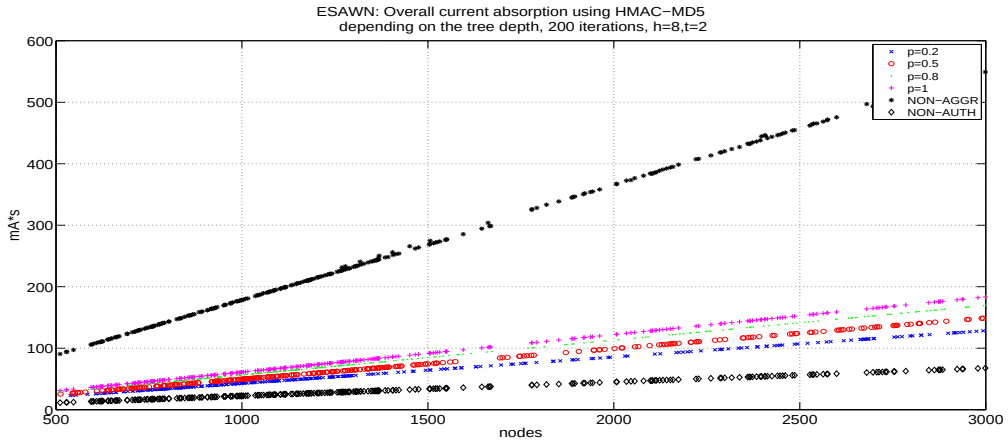


(c) CBCMAC-RC5

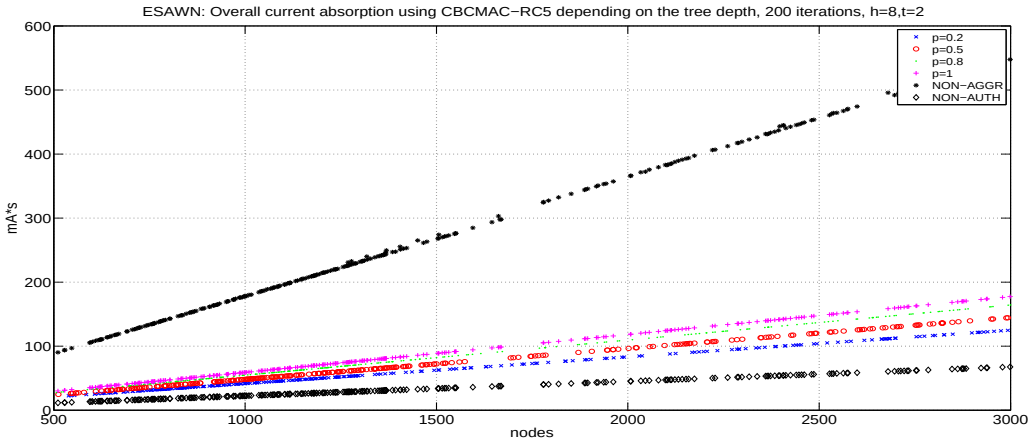
Figure 5.21: Overall current absorption for different values of t, $h = 7$, $[n_1, n_2] = [2, 8]$, $p = 0.8$ and $iter = 200$



(a) HMAC-SHA1

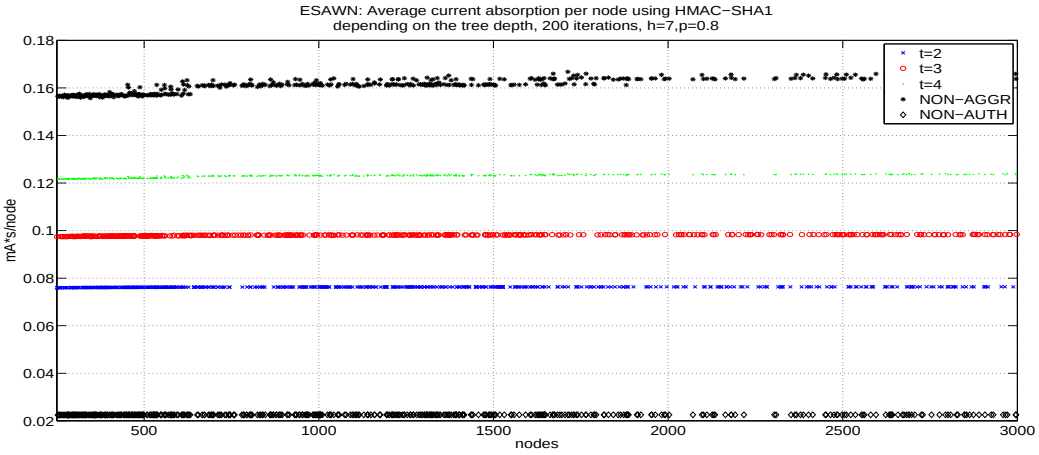


(b) HMAC-MD5

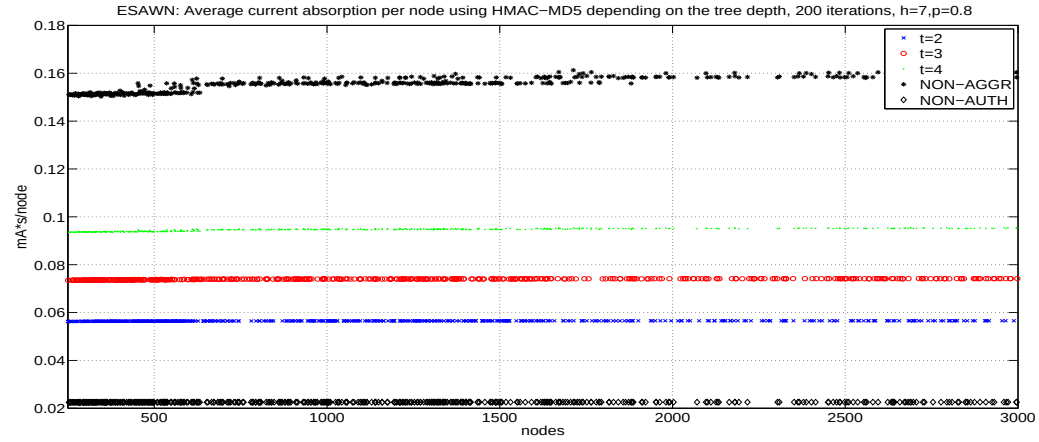


(c) CBCMAC-RC5

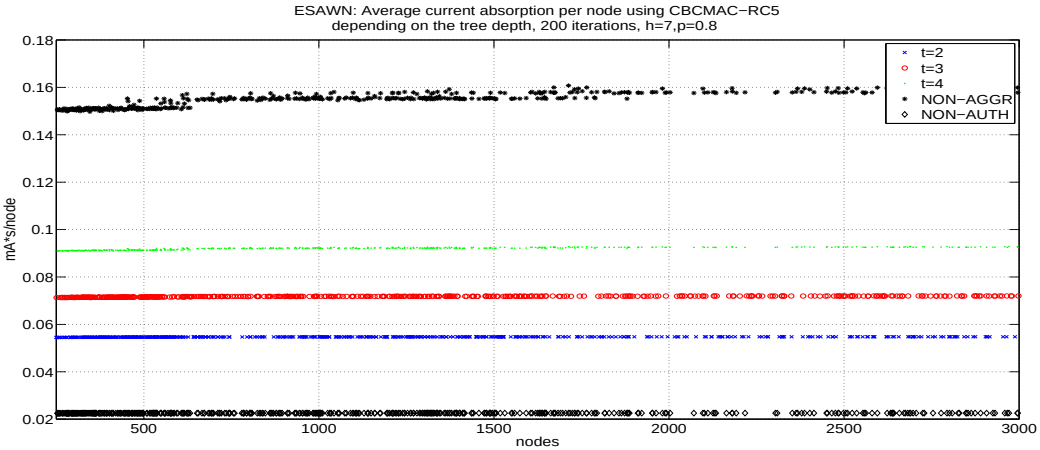
Figure 5.22: Overall current absorption for different values of p, $h = 8$, $[n_1, n_2] = [2, 8]$, $t = 2$ and $iter = 200$



(a) HMAC-SHA1

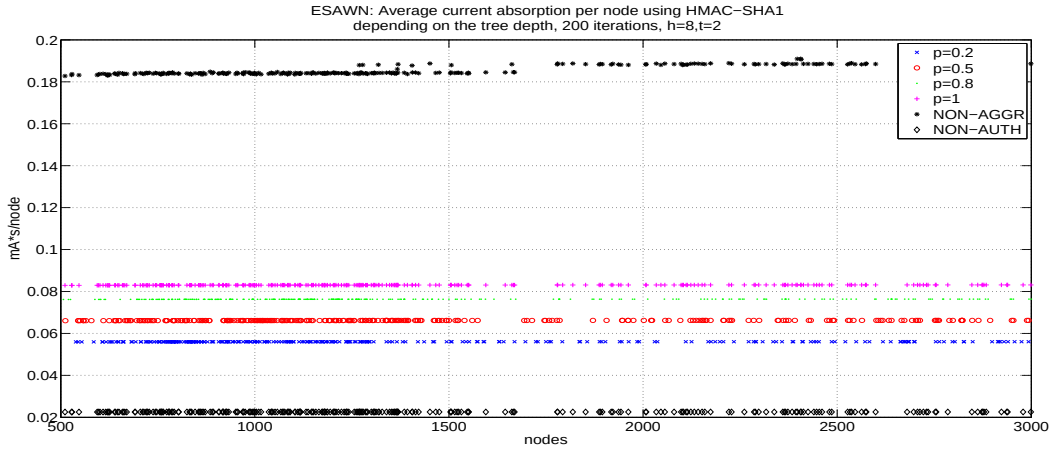


(b) HMAC-MD5

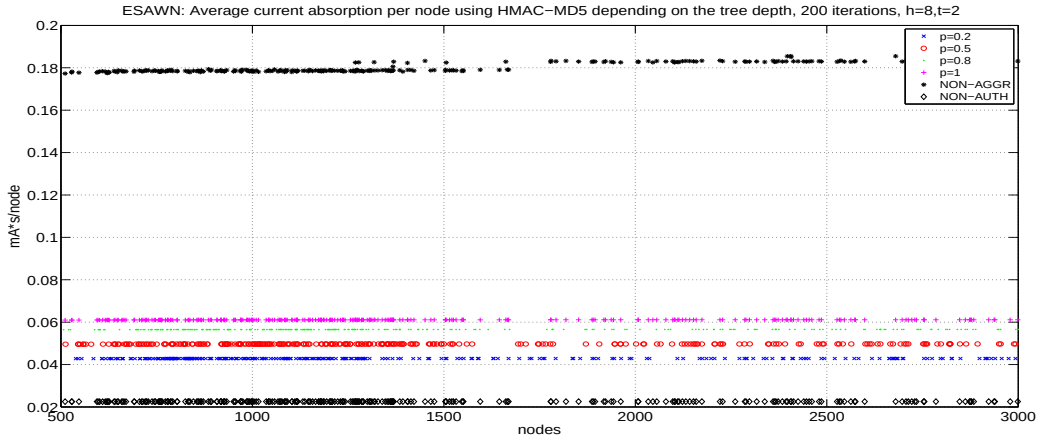


(c) CBCMAC-RC5

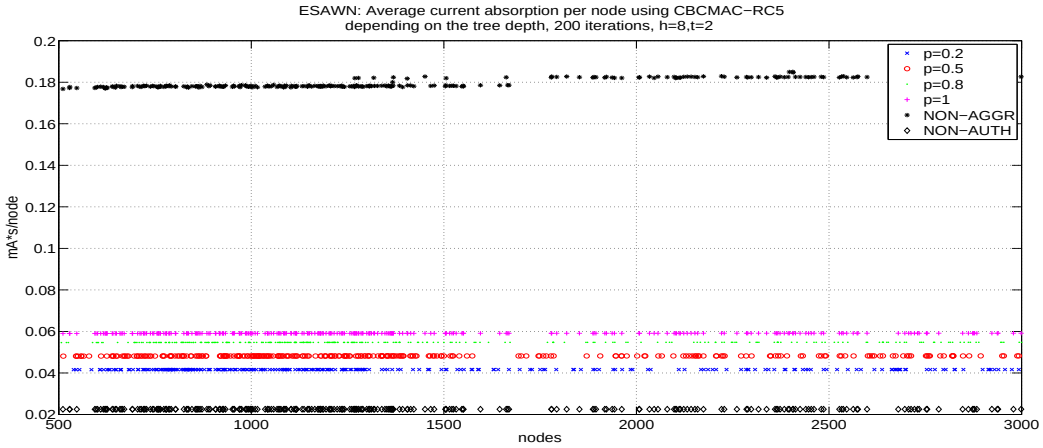
Figure 5.23: Average current absorption per node for different values of p, $h = 7$, $[n_1, n_2] = [2, 8]$, $p = 0.8$ and $iter = 200$



(a) HMAC-SHA1



(b) HMAC-MD5



(c) CBCMAC-RC5

Figure 5.24: Average current absorption per node for different values of p, $h = 8$, $[n_1, n_2] = [2, 8]$, $t = 2$ and $iter = 200$

5.4 Summary

In the previous sections it was shown that ESAWN achieves considerable reductions in terms of energy consumption. The advantage over the NON-AGGR protocol varies according to the considered tree depth h, to the introduced redundancy t, to the verification probability p and to the use MAC algorithm. Regarding the choice of this parameters, several observations should be made.

Firstly, the choice of the ESAWN parameters should be made in accordance with the required level of security (see §4.5.2), but without disregarding the tree topology. For example, choosing $t = 4$ in a tree with 4 levels would make no sense and, on the contrary, it would cause ESAWN to achieve worse performance than NON-AGGR. On the other hand, t should take into account the fraction β of compromised nodes (see §4.5.2). Therefore, in order to achieve optimal performances, the value of t and the value of h have to be considered jointly, recalling also that the security parameter (t, k) depends, firstly, on the value of β. In particular for values of h that are considerably higher (at least twice) than t, the introduced redundancy is compensated by a much more consistent reduction in terms of communication overhead with respect to NON-AGGR protocol (i.e., transmitting t packets per node and performing aggregation is more convenient than sending one packet per node without performing aggregation). As a consequence, topologies with higher values of h are better suited to the ESAWN protocol.

Secondly, the choice of p should comply with the required overall probability that the aggregates reaching the BS are authentic, as seen in section §4.5.2. Therefore, $p \geq 0.8$ would be desirable. Moreover, comparing the current absorption per node for different values of t and p, it is interesting to observe that t is the parameter that most influences the current absorption, i.e., the gaps between the lines associated with consecutive values of t are by far more consistent than the ones obtained, for a fixed t, for different values of p.

The final remark is focused on the choice of the MAC scheme to be embedded in ESAWN. It was shown, through the direct implementation of each scheme on a TelosB platform, that HMAC-MD5 and CBCMAC-RC5 achieve better performance than HMAC-SHA1. The following facts, however, should be considered:

- HMAC-SHA1 is expected, accordingly to the present studies in cryptography, to ensure a higher level of security.

- CBCMAC-RC5 is not actually suitable for implementation when its inputs are of variable size; embedding it in ESAWN (at least when the optimisation described in §4.5.2 is adopted) would thus compromise the security of the system, as, depending on whether the ESAWN verification takes place or not, messages can have variable sizes.

- as reported in §5.1, time complexity is not the only parameter that characterises a MAC algorithm, though it is for sure the most remarkable; also ROM and RAM consumption should be taken into account, especially when

the ESAWN protocol has to be integrated with other applications on a single sensor node.

Summarising, assuming a proper choice of its parameters and the use of one of the proposed MAC schemes, the ESAWN protocol allows a significant reduction in the overall network energy consumption, without introducing an excessive overhead with respect to the NON-AUTH protocol. Therefore, ESAWN can be considered as a suitable solution to the problem of ensuring end-to-end authenticity in a data aggregation protocol. Further improvements in its performance, both in terms of energy consumption and of ensured authenticity, should, however, be investigated, as described in the next chapter.

Chapter 6

Conclusion and future works

Data aggregation is a fundamental component in a wireless sensor network protocol. It enables a considerable reduction in communication overhead, at a price of a just slightly increased computational complexity. Since a wireless sensor network, however, could be subject to malicious attacks, at the same time it is important to provide a certain level of security.

In the previous chapters of this thesis, issues related to secure data aggregation protocols were investigated, starting from the definition of what data aggregation and security in a WSN are (chapter 2), and proceeding with the analysis of some of the most valuable proposals for secure data aggregation protocols (chapter 3). Starting from chapter 4, the focus was moved to the specific problem of providing authenticity in a data aggregation protocol. Three different MAC schemes were studied and implemented on a TelosB mote; in chapter 5, their performances has been evaluated, analysed and compared. Furthermore, ESAWN, a protocol proposed in [51], was described and analysed as a solution to the end-to-end authentication problem in a data aggregation protocol. Successively, ESAWN performances has been evaluated and commented.

The main contributions of the present work are the following:

- focused analysis of the end-to-end authentication problem, in the context of data aggregation protocols;
- evaluation of actual performance of some of the most widely deployed MAC schemes, namely CBCMAC and HMAC, as applied to the TelosB platform;
- optimisation of the ESAWN protocol for providing plain data authentication in place of authenticated encryption; by means of some adaptations (see §4.5.2), a considerable reduction of the number of transmitted bytes and, therefore, of energy consumption, is achieved.
- theoretical analysis (according to some specifically defined metrics) and experimental evaluation of the optimised ESAWN protocol.

Several issues, however, still have to be considered and provide a wide range of possible future works. Some of them are listed below:

- as stated in §4.5.2, the ESAWN verification synchronisation mechanism seems to be intrinsically insecure, as an attacker can predict verifications. Therefore, other techniques should be investigated.
- a filtering mechanism, able to detect eventual outliers in the submitted sensor readings, would be desirable to be implemented in ESAWN, as long as it is not too demanding in terms of system resources.
- additional cooperation techniques between sensor nodes, aimed to detect malicious nodes misbehaviours, could help in further reducing the ESAWN protocol overhead; thus, it would be interesting to investigate this direction.
- faster and less computationally expensive MAC schemes, optimised for implementation on a sensor node would be desirable, so that additional energy saving could be achieved.
- a more accurate evaluation of ESAWN power consumption would be convenient, namely:
  - the analysis should take into account eventual link layer retransmissions due to packet loss;
  - radio start up time, preamble transmission, etc. should be considered;
  - estimation of aggregation cost for different aggregation functions could be an interesting issue;
  - performance should be evaluated according to different payload and MAC sizes.

As a conclusion, the ESAWN optimised protocol appears to be an efficient solution for ensuring end-to-end authenticity (even if in a relaxed way) in a data aggregation protocol. Its performance evaluation, in fact, shows that it achieves a considerable advantage in terms of energy consumption with respect to an authenticated, non-aggregated protocol. Several improvements, though, can be investigated and could lead to a higher level of probabilistically ensured end-to-end authenticity, together with a further reduction in power consumption.

Acknowledgements (Ringraziamenti)

Nel congedare il presente lavoro, desidero rivolgere, innanzitutto, un grazie riconoscente al prof. Zorzi, per avermi offerto la possibilità di sviluppare questa ricerca. Un grazie anche all'ing. Angelo P. Castellani, per l'attenzione e il tempo che mi ha dedicato, e al dott. Paolo Casari, per la disponibilità, la cordialità e il prezioso sostegno, ma anche per la fiducia e la stima che ha saputo transmettermi.

Un grazie di cuore ai miei genitori, Roberto e Rita, per avermi accompagnato fino a qui, incoraggiandomi e sostenendomi sempre. Grazie a Chiara e Pietro, per essere sempre stati al mio fianco.

Infine, un grazie sincero ai miei compagni di corso, per avermi regalato anni meravigliosi in loro compagnia, pieni di stimoli e soddisfazioni condivise. In particolare, grazie a Bruno, Carlo, Caterina, Francesco, Giulia, Marco, Mattia e Nicola.

Bibliography

[1] A.J. Menezes, Paul C. van Oorscht, Scott A. Vanstone, "Handbook of applied cryptography", CRC Press, October 1996, ISBN 0849385237.

[2] J. D. Tygar, et al, "SPINS: Security Protocols for Sensor Networks", Wireless Networks, vol. VIII, no. 5, Sept. 2002, pp. 521-534.

[3] Joan Daemen and Vincent Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." Springer-Verlag, 2002. ISBN 3540425802.

[4] R.L. Rivest, "The RC5 Encryption Algorithm", Proceedings of the Second International Workshop on Fast Software Encryption (FSE) 1994. pp. 86-96, http://theory.lcs.mit.edu/ rivest/Rivest-rc5rev.pdf

[5] A. Biryukov and E. Kushilevitz , Improved Cryptanalysis of RC5, EUROCRYPT 1998.

[6] National Bureau of Standards, Data Encryption Standard, FIPS-Pub.46. National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[7] Original RSA Patent as filed with the U.S. Patent Office by Rivest, Ronald L. (Belmont, MA), Shamir Adi (Cambridge, MA), Adleman Leonard M. (Arlington, MA), December 14, 1977.

[8] V. Miller, Use of elliptic curves in cryptography, CRYPTO 85, 1985.

[9] Taher ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transactions on Information Theory, v. IT-31, n. 4, 1985, pp.469-472 or CRYPTO 84, pp.10-18, Springer-Verlag.

[10] W. Diffie and M. E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, vol. IT-22, Nov. 1976, pp: 644-654.

[11] T. Roosta, S. Shieh, S. Sastry , Taxonomy of security attacks in sensor networks, in 'The First IEEE International Conference on System Integration and Reliability Improvements', IEEE International, 2006, Washington, DC, USA.

[12] H. Alzaid, E. Foo, and J.G. Nieto, Secure data aggregation in wireless sensor network: A Survey, In Proceedings of the Sixth Australasian Conference on information Security - Volume 81 (Wollongong, NSW, Australia, January 01 - 01, 2008). L. Brankovic and M. Miller, Eds. ACM International Conference Proceeding Series, vol. 328. Australian Computer Society, Darlinghurst, Australia, 93-105.

[13] E. Fasolo, M. Rossi, J. Widmer, M. Zorzi, In-network aggregation techniques for wireless sensor networks: a survey, IEEE Wireless Commun. 14 (2) (2007) 70-87.

[14] M. Ding, X. Cheng, and G. Xue, "Aggregation Tree Construction in Sensor Networks", IEEE VTC '03, Orlando, FL, Oct. 2003

[15] K. Dasgupta, K. Kalpakis, and P. Namjoshi,"An Efficient Clustering-based Heuristic for Data Gathering and Aggregation in Sensor Networks", IEEE WCNC '03, New Orleans, LA, Mar. 2003.

[16] A. Harris III, R. Kravets, and I. Gupta, "Building Trees Based On Aggregation Efficiency in Sensor Networks", Med-Hoc-Net 2006, Lipari, Italy, June 2006.

[17] S. Madden et al.,"TAG: a Tiny AGgregation Service for Ad Hoc Sensor Networks", OSDI 2002, Boston, MA, Dec. 2002.

[18] C. Intanagonwiwat et al., "Directed Diffusion for Wireless Sensor Networking", IEEE/ACM Trans. Net., vol. 11, no. 1, Feb. 2002, pp. 2-16.

[19] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data Gathering Algorithms in Sensor Networks using Energy Metrics", IEEE Trans. Parallel Distrib. Sys., vol. 13, no. 9, Sept. 2002, pp. 924-35.

[20] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-Specific Protocol Architecture for Wireless Microsensor Networks", IEEE Trans. Wireless Commun., vol. 1, no. 4, Oct. 2002, pp. 660-70.

[21] Y. Yao and J. Gehrke, "Query Processing for Sensor Networks", ACM CIDR 2003, Asilomar, CA, Jan. 2003.

[22] B. Zhou et al., "A Hierarchical Scheme for Data Aggregation in Sensor Network", IEEE ICON '04, Singapore, Nov. 2004.

[23] A. Mahimkar and T. S. Rappaport, "SecureDAV: A Secure Data Aggregation and Verification Protocol for Sensor Networks", IEEE GLOBECOM 2004, Dallas, TX, Nov. 2004

[24] S. Nath et al., "Synopsis Diffusion for Robust Aggregation in Sensor Networks", ACM SenSys 2004, Baltimore, MD, Nov. 2004.

[25] A. Manjhi, S. Nath, and P. B. Gibbons, "Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Stream", ACM SIGMOD 2005, Baltimore, MD, June 2005.

[26] S. Chen and Z. Zhang, "Localized Algorithm for Aggregate Fairness in Wireless Sensor Networks", ACM/SIGMOBILE MobiCom 2006, Los Angeles, CA, Sept. 2006.

[27] A. Manjhi, S. Nath, and P. B. Gibbons, "Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Stream", ACM SIGMOD 2005, Baltimore, MD, June 2005.

[28] M. Sartipi and F. Fekri, "Source and Channel Coding in Wireless Sensor networks using LDPC Codes", IEEE SECON '04, Santa Clara, CA, Oct. 2004.

[29] K. Akkaya, M. Demirbas, R.S. Aygun, The impact of data aggregation on the performance of wireless sensor networks, Wiley Wireless Communications and Mobile Computing (WCMC), J.8, 2008, 171-193.

[30] Y. Wang, G. Attebury, B. Ramamurthy, A survey of security issues in wireless sensor networks, IEEE Communications Surveys and Tutorials, 2006.

[31] Y. Zhou, Y. Fang, Y. Zhang, Securing Wireless Sensor Networks:A Survey, IEEE Communications Surveys, Vol.10, No.3, 3rd Quarter 2008.

[32] J. D. Tygar, et al, "SPINS: Security Protocols for Sensor Networks", Wireless Networks, vol. VIII, no. 5, Sept. 2002, pp. 521-534.

[33] A. Perrig, R. Canetti, J.D. Tygar, D. Song, The TESLA broadcast authentication protocol, RSA CryptoBytes, 2002.

[34] Anthony D. Wood, John A. Stankovic, "Denial of Service in Sensor Networks", Computer, vol. 35, no. 10, pp. 54-62, Oct. 2002

[35] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, 1978.

[36] B. Przydatek, D. Song, A. Perrig, SIA : secure information aggregation in sensor networks, in: Proceedings of SenSys '03, 2003, pp. 255-265.

[37] Ralph C. Merkle. A certified digital signature. In Proc. Crypto '89, pages 218-238, 1989.

[38] Mihir Bellare, Ran Canetti, and Hugo Krawczyk, Keying hash functions for message authentication, In: Advances in Cryptology - CRYPTO '96, pages 1-15, 1996.

[39] A. Mahimkar, T.S. Rappaport, SecureDAV: a secure data aggregation and verification protocol for wireless sensor networks, in: Proceedings of the 47th IEEE Global Telecommunications Conference (Globecom), November 29 - December 3, Dallas, TX,2004.

[40] N. Koblitz, Elliptic curve cryptosystems, in Mathematics of Computation 48, 1987, pp. 203-209

[41] V. Miller, Use of elliptic curves in cryptography, CRYPTO 85, 1985.

[42] D. Johnson, A. Menezes and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", Springer-Verlag, 2001.

[43] Yang, Y., Wang, X., Zhu, S., and Cao, G. 2008. SDAP: A secure hop-by-hop data aggregation Protocol for sensor networks. ACM Trans. Inf. Syst. Secur. 11, 4, Article 18 (July 2008).

[44] Frank, G. 1969. Procedures for detecting outlying observations in samples. Technometrics 11, 1 (February), 1-21.

[45] H. Çam, S. Ozdemir, P. Nair, D. Muthuavinashiappan, H.O.Sanli, Energy-efficient and secure pattern based data aggregation for wireless sensor networks, Comput. Commun., Elsevier 29 (4) (2006) 446-455

[46] S. Ozdemir, Secure and reliable data aggregation for wireless sensor networks, in: H. Ichakawa et al. (Eds.), LNCS 4836, 2007, pag. 102-109.

[47] Niels Ferguson, Bruce Schneier, "Practical Cryptography", John Wiley & Sons (2003). ISBN 0471223573.

[48] A. Josang, E. Ismail, "The beta reputation system", in: Proceedings of the 15th Bled Conference Electronic Commerce, 2002.

[49] S. Ganeriwal, M.B. Srivastava, "Reputation-based framework for high integrity sensor networks", in: Proceeding of the Second ACM Workshop on Security of Ad Hoc and Sensor Networks.

[50] Suat Ozdemir, "Functional Reputation Based Reliable Data Aggregation and Transmission for Wireless Sensor Networks", Computer Communications, Elsevier, vol. 31, no. 17, pp. 3941-3953, Nov. 2008.

[51] BlaßErik-Oliver, Wilke Joachim and Zitterbart Martina, "Relaxed authenticity for data aggregation in wireless sensor networks", SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication networks, pag. 1-10, Istanbul (Turkey), 2008

[52] E. Mlaih, S.A. Aly, "'Secure Hop-by-Hop Aggregation of End-to-End Concealed Data in Wireless Sensor Networks, in: IEEE Infocom 2008 Proceedings.

[53] F. Adachi, M. Sawahashi, K. Okawa, "Tree-structured generation of orthogonal spreading codes with different length for forward link of DS-CDMA mobile radio", Electronic Letters 33 (1) (1997) 27-28.

[54] D. Westhoff, J. Girao, M. Acharya, "Concealed data aggregation for reverse multicast traffic in sensor networks: encryption key distribution and routing adaptation", IEEE Trans. Mobile Comput. 5 (10) (2006) 1417-1431.

[55] J. Domingo-Ferrer, A provably secure additive and multiplicative privacy homomorphism, in: Proceedings of the Information Security Conference, 2002, pp. 471-483.

[56] T. Okamoto, S. Uchiyama, "A New Public-key Cryptosystem as Secure as Factoring", in: Proc. Conf. Advances in Cryptology (EUROCRYPT '98), pp. 208-318, May 1998.

[57] D. Wagner, "Cryptanalysis of an Algebraic Privacy Homomorphism", in: Proceeding of the 6th Information Security Conference (ISC03), Bristol, UK, October 2003.

[58] S. Ozdemir, "Secure data aggregation in wireless sensor networks via homomorphic encryption", Journal of The Faculty of Engineering and Architecture of Gazi University 23 (2) (2008) 365-373. ISSN:1304-4915.

[59] R.M. Kling, Intel Mote: an Enhanced Sensor Network Node, http://www.intel.com/research/exploratory/motes.htm

[60] C. Castelluccia, E. Mykletun, G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks, in: Proceedings of the Conference on Mobile and Ubiquitous Systems: Networking and Services, 2005, pp. 109-117.

[61] M.J.B. Robshaw, Stream Ciphers, RSA Laboratories Technical Report TR-701, Version 2.0, July 25, 1995 RSA Laboratories

[62] Mihir Bellare, Joe Kilian, Phillip Rogaway, "The Security of the Cipher Block Chaining Message Authentication Code", Journal of Computer and System Sciences, Volume 61, Issue 3, December 2000, Pages 362-399, ISSN 0022-0000, DOI: 10.1006/jcss.1999.1694.

[63] RFC1321 - The MD5 Message-Digest Algorithm

[64] Xiaoyun Wang and Hongbo Yu, "How to Break MD5 and Other Hash Functions", Retrieved December 21, 2009

[65] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", Cryptology ePrint Archive Report 2004/199, 16 Aug 2004, revised 17 Aug 2004.

[66] J. Black, M. Cochran, T. Highland, "A Study of the MD5 Attacks: Insights and Improvements", March 3, 2006.

[67] RFC3174 - US Secure Hash Algorithm (SHA1)

[68] Secure Hash Signature Standard (SHS), FIPS PUB 180-2, 2002.

[69] The keyed-hash message authentication code (HMAC), FIPS PUB 198, issued March 6,2002 by the NIST Information Technology Laboratory.

[70] RFC2104 - H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", February 1997

[71] Wang, X., Yu, H., Wang, W., Zhang, H., and Zhan, T. , "Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC", In Proceedings of the 28th Annual international Conference on Advances in Cryptology: the theory and Applications of Cryptographic Techniques (Cologne, Germany, April 26 - 30, 2009). A. Joux, Ed. Lecture Notes In Computer Science, vol. 5479. Springer-Verlag, Berlin, Heidelberg, 121-133.

[72] J. Katz and Y. Lindell, "Aggregate Message Authentication Codes", In CT-RSA, Springer-Verlag (LNCS 4964), pages 155-169, 2008.

[73] Castelluccia, C., Chan, A. C., Mykletun, E., and Tsudik, G., "Efficient and provably secure aggregation of encrypted data in wireless sensor networks", ACM Trans. Sen. Netw. 5, 3 (May. 2009), 1-36.

[74] L. Eschenauer and V. Gligor, "A key management scheme for distributed sensor networks", In: Proceedings of ACM Computer and Communications Security, Washington D.C. USA, Nov 2002, pp. 41-47.

[75] M. Zitterbart and E.-O. Blaß, "An Efficient Key Establishment Scheme for Secure Aggregating Sensor Networks", In: ACM Symposium on Information, Computer and Communications Security, Taipei, Taiwan, Mar. 2006, pp. 303-310, ISBN 1-59593-272-0.

[76] Jochim Wilke, Erik-Oliver Blass, Felix Freiling, Martina Zitterbart, "A framework for probabilistic, authentic aggregation in wireless sensor networks", PIK Vol. 32(2), pages 116-126, April 2009. (ISSN 0930-5157)

[77] H.C. van Tilborg, Encyclopedia of Cryptography and Security, Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[78] http://www.tinyos.net/tinyos-2.1.0/doc/html/tep125.html

[79] http://www.ist-ubisecsens.org/download.php

[80] Karlof, C., Sastry, N., and Wagner, D., "TinySec: a link layer security architecture for wireless sensor networks", In Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems (Baltimore, MD, USA, November 03 - 05, 2004). SenSys '04.

[81] Moteiv Corporation, "Telos (Rev B) Datasheet", http://www.moteiv.com, Dec. 2004.

[82] CC2420 datasheet, January 2008, available at:
http://inst.eecs.berkeley.edu/ cs150/Documents/CC2420.pdf

[83] Levis, P. and Gay, D., "TinyOS Programming, 2009", 1st. Cambridge University Press.