A novel secure aggregation scheme for wireless sensor networks using stateful public key cryptography

Omar Rafik Merad Boudia, Sidi Mohammed Senouci, Mohammed Feham

▶ **To cite this version:**

Omar Rafik Merad Boudia, Sidi Mohammed Senouci, Mohammed Feham. A novel secure aggregation scheme for wireless sensor networks using stateful public key cryptography. Ad Hoc Networks, 2015, 32, pp.98-113. 10.1016/j.adhoc.2015.01.002 . hal-02873889

HAL Id: hal-02873889

https://hal.science/hal-02873889v1

Submitted on 11 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ACCEPTED MANUSCRIPT

A Novel Secure Aggregation Scheme for Wireless Sensor Networks Using Stateful Public Key Cryptography

Merad Boudia Omar Rafik[1,*,†], Senouci Sidi Mohammed[2] and Feham Mohammed[1]

[1]*STIC Laboratory, University of Tlemcen, Algeria*
[2]*Drive Laboratory, University of Burgundy, Nevers, France*

Abstract

Wireless sensor networks (WSNs) are nowadays considered as an important part of the Internet of Things (IoT). In these networks, data aggregation plays an essential role in energy preservation. However, WSNs are usually deployed in hostile and unattended environments (e.g. military applications) in which the confidentiality and integrity security services are widely desired. Recently, homomorphic encryptions have been applied to conceal sensitive information during aggregation such that algebraic operations are done directly on ciphertexts without decryption. The main benefit is that they offer the end-to-end data confidentiality and they do not require expensive computation at aggregator nodes since no encryption and decryption are performed. However, existing solutions either incur a considerable overhead or have limited applicability to certain types of aggregate queries. This paper presents a novel secure data aggregation protocol for WSNs. The scheme employs Stateful Public Key Encryption (StPKE) and some previous techniques in order to provide an efficient end-to-end security. Moreover, our solution does not impose any bound on the aggregation function's nature (`Maximum`, `Minimum`, `Average`, etc.). We present and implement our scheme on TelosB as well as MicaZ sensor network platforms and measure the execution time of our various cryptographic functions. Simulations are also conducted to show how our scheme can achieve a high security level (by providing the above security services) with a low overhead (in terms of computation and communication) in large-scale scenario.

KEY WORD: wireless sensor networks; secure data aggregation; homomorphic encryption; simple power analysis.

1. INTRODUCTION

Wireless sensor networks have received much attention over the last few years, not only in academia, but also in industries for the study and development of a plethora of potential applications in various domains such as military, health, environmental, etc. Nowadays considered as one of the main elements in the Internet of Things (IoT) [1], WSNs are composed of many sensor nodes where the resource limitation represents the most important feature. In fact, the sensors are constrained in battery power, communication, and computation capability; therefore, every possible solution that aims to conserve these resources is extensively sought [2]. Data aggregation is one of the techniques that is actually considered as an essential paradigm for WSNs since it tends to save both computation and communication resources. With such technique, data are captured by sensor nodes and fused as they flow in the network by intermediate nodes and then transmitted to the sink over the wireless link. In IoT context, the sink node can be considered as an Internet powered device that gathers readings (aggregated data) and sends them to the cloud. In a nutshell, the sink node manages all the interaction between the WSN and the outside world (known as *Front-End solution*, in which the WSN is completely independent from the Internet [3]). Data aggregation allows in-network processing which leads to lesser packet transmissions and reduces redundancy, and therefore, helps in increasing the WSN's overall lifetime [4].

[*]Correspondence to: Merad Boudia Omar Rafik, STIC Laboratory, University of Tlemcen, Algeria.
[†]Telephone: (+213) 557 884 999.
[†]E-mail: om_meradboudia@mail.univ-tlemcen.dz

ACCEPTED MANUSCRIPT

The major problem of data aggregation is when we aim to provide security. On one hand, it should be noted that aggregation and security have opposite goals. While the first attempts to reduce the number of packet transmitted, the second adds a non-negligible cost in order to ensure some security properties. On the other hand, WSNs have some special features that are different from other networks, (i) they are limited in terms of resources, which makes the choice of the adequate security algorithm somewhat difficult. Algorithms that are simple and efficient in terms of resources utilization are the most suitable, or sometimes by sacrificing some security in order to render possible the implementation on a wireless sensor,(ii) they are often deployed in hostile and unattended environments, which makes them subject to several kinds of attacks. In an aggregation process for example, due to the amount of data to be fused, the nodes that perform the aggregation function are the most attractive to an adversary, and (iii) they use a wireless link, which may allow an attacker to monitor the transmitted data and even participate in the communication. To summarize, data aggregation protocols must be designed in conjunction with security protocols in order to reach a good compromise between the overall protocol complexity and the provided security level [5]. Guaranteeing security for data aggregation is therefore an intriguing challenge.

Traditional secure aggregation protocols use hop-by-hop encryption [6-8], in which sensor nodes encrypt the captured data and send the ciphertext to the aggregator node; the aggregator node decrypts, performs the aggregation function, and then sends the encryption of the result to the upper aggregator node. Therefore, while these data aggregation protocols improve the bandwidth and energy utilization of the network, and especially allow a simpler implementation of aggregation functions (`Maximum`, `Minimum`, `Average`, etc.), they incur not only a high computation overhead but also delay (due to encryption/decryption effort). Besides, the aggregator nodes can access to the plaintext data, so the end-to-end confidentiality is not provided which is mandatory, especially for military applications. Therefore, the development of efficient schemes with stronger security becomes primordial.

Recently, several solutions [9-15] that provide data confidentiality without inducing delay have been proposed. Known as *Concealed Data Aggregation (CDA)*, they are based on a particular cryptographic algorithm, namely *Privacy Homomorphism*, which enables direct calculations (addition and/or multiplication) on enciphered data. The main benefit of these schemes is that they offer the end-to-end data confidentiality and they do not require expensive computation at aggregator nodes since no encryption and decryption are performed. Conversely, actual privacy homomorphisms increase significantly the energy required for encryption and also have limited applicability to certain types of aggregate queries. In fact, only addition-based and multiplication-based aggregation operations are possible. Schemes based on symmetric encryption have been proposed, but most of them were cryptanalyzed [16]. In [11], the authors study and analyze a selected set of asymmetric algorithms for end-to-end privacy in WSNs. Their results show that the Elliptic Curve El Gamal (ECEG) is the most suitable algorithm for WSNs. In [17], we proposed an efficient implementation of ECEG on MicaZ motes. The encryption takes about 1.29s. However, for an application where the sink needs to continuously collect information about the target area e.g. every 20 seconds, such a scheme is impracticable and leads to energy depletion. Furthermore, ECEG is additive homomorphic and hence, supports only a limited number of aggregation functions related to addition operation [18]. Another security service namely, the end-to-end data integrity is an interesting issue that becomes a challenge to the cryptographic community. The hop-by-hop verification does not ensure that the aggregator performs correctly the aggregation function. A compromised aggregator can produce a fake aggregate and authenticate it with its legitimate key. Therefore, the end-to-end integrity is another service that is widely desired. Due to the special characteristics of WSNs, it is very challenging to provide these two security services at the same time [18].

Paper contributions. In this paper, we propose a Secure Aggregation scheme for WSNs using Stateful Public Key Cryptography (SASPKC). This novel protocol employs Stateful Public Key Encryption (StPKE) proposed by Bellare et al. [19] and some previous techniques to overcome the above problems. The contributions of the present paper are three-fold:

- First, SASPKC adopts StPKE for efficiency in terms of computation and communication costs. Simulation and experimental results show the huge decrease of energy utilization of the network in comparison with related work,
- Second, SASPKC aggregates not only ciphertexts but also signatures, the end-to-end data confidentiality and integrity security services are provided using symmetric homomorphic encryption and aggregate Message Authentication Code (MAC), respectively. In our proposal, the base station is

ACCEPTED MANUSCRIPT

able to extract individual data, verify the integrity of all messages, authenticate the senders and eventually identify the malicious node,

- Finally, our implementation on TelosB and MicaZ motes uses a fast algorithm for elliptic curve scalar multiplication to reduce the execution time of SASPKC, the algorithm is also secure against side channel attacks, in particular Simple Power Analysis.

Paper organization. The remainder of the paper is organized as follow: Section 2 presents the relevant literature review on secure data aggregation in WSNs. Section 3 presents the background knowledge. Section 4 details our protocol for secure data aggregation and Section 5 provides its security and performance analysis. Finally, Section 6 concludes our work and gives some future directions.

2. Related Work

There is an extensive research on secure data aggregation in WSNs [6-15]. Existing works are designed for different security requirements.

At the beginning, the researchers focused mostly on data integrity. Du et al. [6] propose a scheme in which some nodes called *witnesses* are involved in order to monitor the aggregate. Hu et al. [7] propose delayed aggregation and delayed authentication to protect against one compromised node. Przydatek et al. [8] use an *interactive proof* session in which the base station interacts with aggregator node in order to verify the correctness of the aggregation result. These schemes focus on aggregation of plain data and *the stealthy attack*, an attack where the adversary aims to force the base station to accept a deceiving aggregation result.

Data confidentiality service becomes attractive when Westhoff et al. [9] introduce the concept of Concealed Data Aggregation. CDA uses homomorphic encryption that allows intermediate nodes to homomorphically aggregate their enciphered data, and thus providing an end-to-end data confidentiality. However, a common key is used by every node, which makes the system insecure in the case where a single node is compromised. This problem is fixed by Castelluccia et al. [10] by using a different key for every node. The authors address both confidentiality and efficiency; they propose a symmetric homomorphic encryption in which the Xor operation of stream cipher is replaced with a simple modular addition. The ciphertext expansion is efficient and therefore improves computational and communication efficiency. However, problems related to sensors identities and malleability are the main drawbacks of the system. This is the only symmetric homomorphic scheme that has not been cracked [16].The authors of [11]study the suitability of a selected set of asymmetric encryption algorithms that have the homomorphic property and are especially based on Elliptic Curve Cryptography (ECC). In fact, compared to RSA algorithm, ECC provides the same security with a smaller key size and smaller ciphertext. A 160-bit ECC provides the same security as a 1024-bit RSA. The authors show that the best candidate for WSNs is ECEG where the main advantages compared to other asymmetric algorithms are the highest level of security it can reach against passive adversary and its smaller packet size. However, as all other asymmetric algorithms, the major drawback remains the computation overhead, which makes it only suitable for a limited number of applications [17].

Most of recent works focus on both security services [12-15]. In [12], Zhu et al. propose a secure data aggregation scheme using a scrambling method, which allows intermediate nodes to perform data aggregation without any decryption. In their system, the verification is integrated with the aggregation results. However, several rounds are needed to verify data of one aggregation session. The protocol in [13] provides both, end-to-end confidentiality and end-to-end integrity, by using ECEG and aggregate signatures based on bilinear maps, respectively. The feature of an aggregate signatures scheme is that the final verifier has to know not only individual data but also the public key used for the signature in order to verify the final aggregate [20]. For this purpose, the authors employ an encoding function that allows the base station to extract individual data. The public keys are assumed to be known by the base station. Then, each packet includes an encryption and a signature on data. The scheme incurs an important computation and communication overhead due to the use of identity-based signature scheme. In [14], the authors propose a scheme that is similar to [13] because they make use of ECEG for data confidentiality, but a different scheme for data integrity. The authors propose a modified version of the Elliptic Curve Digital Signature Algorithm (ECDSA) [21] in order to allow aggregation over signatures. The hash of plaintext is replaced with the plaintext itself in the original ECDSA. This solution introduces not only a significant calculation time, but also the packet size is very important, since each packet contains ciphertext, signature and public key. The latter is required for verification at

ACCEPTED MANUSCRIPT

the base station. However, it is not practical to send such large packets due to the high bit error rates of wireless links such as in WSNs. In addition, this solution can only verify the final result of aggregation. In [15], IPHCDA provides integrity for concealed data aggregation. Based on a homomorphic encryption algorithm introduced in [22], IPHCDA has the particularity to enable decryption with different public keys. The authors apply this algorithm to a hierarchical WSN, where a different public key is assigned to each region in order to allow the base station to classify aggregated data based on the corresponding public key. Hop-by-hop data integrity is ensured using MAC. The major drawbacks are the computation overhead and the packet size, which increase with the number of regions in the network.

In the aforementioned schemes, there are only a few schemes that consider the two security services in an end-to-end manner [12-14]. This is widely desirable, as usually no sensor node can be trusted to be honest. These schemes either incur a considerable overhead in terms of computation and communication (due to the cryptographic algorithms employed) or have limited applicability to certain types of aggregate queries (due to the nature of the homomorphic property). Our contribution is motivated by the above facts, which also justify the importance of this work.

3. Background, System Model and Objectives

An exciting feature of SASPKC is the use of hybrid encryption, namely the combination of asymmetric and symmetric primitives. For that, SASPKC adopts StPKE [19] and makes use of the Castelluccia et al.'s scheme [10]. In this section, we first briefly introduce each of these concepts as well as other cryptographic tools, and then we present our system models and design goals.

3.1. Stateful Public Key Encryption (StPKE)

The computational cost of Public Key Encryption (PKE) is dominated by modular exponentiations (scalar point multiplications in ECC-based PKE), which affect dramatically the energy consumption. In [19], the authors propose a stateful encryption, which can significantly improve the computational cost of traditional PKE. In StPKE, the sender keeps a state that is re-used across different encryptions. In the following, we describe the scheme.

Public parameter: DSA group or elliptic curve group with prime order p, with generator "g"
Public key: $Y=g^x$, H// H: hash function
Private key: x
Encryption: $r \in Zq, C_1=g^r, K=H(C_1,Y,Y^r)$, $C_2=E(K,M)$
Ciphertext: (C_1, C_2)
Decryption: $K=H(C_1, Y, C_1^x)$, $M= D(K,C_2)$

Figure 1. Bellare et al.'s cryptosystem [19]

The state (C_1, r) is maintained by the sender and the same state is re-used for future encryptions. The symmetric key K is derived from g^{xr} (i.e. Y^r) by hash function, and used in the symmetric encryption of M under a secure symmetric scheme(E, D).The use of state reduces significantly the computation cost because C_1 is calculated just once, which consequently saves energy. The security is based on the Diffie-Hellman assumption (Given g^a, g^b, it is computationally infeasible to compute g^{ab}, for more detail see [19]). In [23], the authors realize the StPKE on WSNs and show its applicability on 8 bits platform. However, the authors only consider a simple one-hop communication. In this work, the StPKE is utilized to efficiently secure the convergecast traffic toward the base station, in a way that the state is used to share a secret with the base station and the network nodes use this secret to ensure the end-to-end security for aggregated data. Meanwhile, ECC is adopted to provide efficiency.

3.2. Castelluccia et al.'s scheme

Castelluccia et al. [10] propose one of the most studied algorithms, a symmetric homomorphic encryption that requires a small number of single precision additions (See Figure 2). The plaintext added to the current key (shared only with sink node) modulo the length of key space is performed for encryption, and for decryption, the sink node needs exactly the same keys used for encryption to obtain

ACCEPTED MANUSCRIPT

the plaintext. The ciphertext expansion is efficient and therefore improves computational and communication efficiency.

Public parameter: a large enough integer M
Encryption: $m \in [0,M-1], k \in [0,M-1]$, $C=k+m \pmod M$
Decryption: $m= C-k \pmod M$
Aggregation: let $C_1=Enc(m_1,k_1)$ and $C_2=Enc(m_2,k_2)$
Retrieve $m_1+m_2=Dec(C_1+C_2, k)$ for $k=k_1+k_2$

Figure 2. Castelluccia et al.'s scheme [10]

The security proof is provided in the appendix of their article. However, the authors in [24] state that to achieve the IND-CPA[1] (*INDistinguishability under a Chosen Plaintext Attack*), the output of the Pseudo Random Function (PRF) must be directly used, which means that the main advantage of the algorithm will be lost. In the same article, the authors propose (*Hashed CMT*), an algorithm that employs a second function namely, a length-matching hash function. The role of this function is to reduce the size of the PRF output similar to the size of the maximum possible of an aggregate value. In [25], the same authors report that the size of the second function output should still be chosen large enough to ensure reasonably low probability of success for a random guess. Besides the problems of sensors identities[2] and malleability, the scheme can only achieve additive or multiplicative aggregation.

3.3. Cryptographic tools

Homomorphic Encryption (HE): This property allows calculations on ciphertexts, which have the same effect as performing these calculations on the underlying plaintext data [16]. An encryption algorithm is accepted to be homomorphic if and only if the following equation holds:

$$D(E(x) \vartriangle E(y)) = D(E(x \vartriangle y))$$

The operation Δ can support either addition or multiplication or both, it depends on the features of the encryption scheme. The HE that supports any function on ciphertexts is known as Fully Homomorphic Encryption (FHE).Introduced by Gentry [26], it is the most significant advance in cryptography in the last few years. It is promising, but the time complexity of its algorithms is still too high for practical use. The other class of HE is Partially Homomorphic Encryption (PHE), which includes encryption schemes that have homomorphic property with respect to one operation. In [16], several PHE have been presented. However, the most efficient scheme in terms of computation and communication overhead is [10], previously described.

Aggregate MAC: It is clear that MAC cannot verify the additive property:

$$MAC(\alpha+\beta) \neq MAC(\alpha) + MAC(\beta)$$

Therefore, MAC cannot ensure the addition over authenticated data. In [27], the authors suggest an idea where they proved that the MACs can be aggregated using *Xor* operation and that the result allows integrity and authenticity verification with the condition to have all individual data to enable verification.

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

Hash-based Message Authentication Code (HMAC): The HMAC is a mechanism used to verify data integrity and source authentication, involving cryptographic hash functions. HMAC can be used with any iterative cryptographic hash function (e.g.MD5, SHA-1, etc.) in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function [28].We use HMAC *(K,m)* to denote message digest of *m* using a key *K*, assuming that the underlying hash function is SHA-1, which produces 20 bytes digest [28].

Pseudo Random Function (PRF): A PRF is an efficient deterministic function (i.e. computable in polynomial time) and takes two inputs *K* and *m*, where *K* is a hidden secret key and *m* is a variable. Its output is computationally indistinguishable from truly random output. HMAC is extensively considered as PRF [29]. In our work, we implement PRF as HMAC.

[1] IND-CPA is the highest security level against passive adversary; it is equivalent to semantic security [16].
[2] To be able to decrypt, the base station must know exactly which sensor did or did not contribute to the aggregate.

ACCEPTED MANUSCRIPT

Key Derivation Function (KDF): A KDF is a function that takes a key and other data as input and generates other keys required for cryptographic algorithms. In our work, we consider the secure KDF that uses PRF (HMAC) recommended by NIST, namely NIST SP800-108 HKDF (HMAC-based KDF) [30].

3.4. Network model

We consider a WSN with a large number of sensor nodes and one base station. Sensor nodes are resource-limited devices (e.g. TelosB motes [31]) firstly deployed in a geographical area to perform some special monitoring function. They are organized in several static clusters (See Figure 3). Some sensor nodes are dynamically elected as Cluster-Heads (CHs) to aggregate data from their members. The dynamic election can be performed using algorithms such as [32,33]. The base station is assumed to be a powerful device and widely trusted. After aggregation done, CHs forward the results to the next hop. All notations used in this paper with their definition are summarized in Table 1.

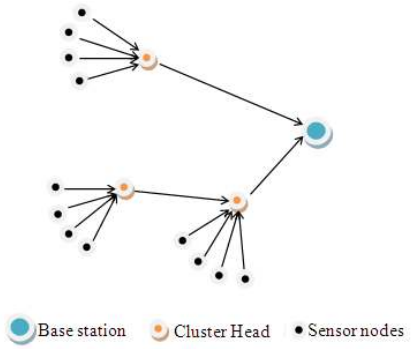


Figure 3. Network model

Notation	Definition
NR	Number of sensor nodes
R	Number of cluster heads in the network
CH_j	Cluster head CH_j, $j \in \{1,\ldots,R\}$
L	Maximum number of nodes per cluster
S_{ij}	Sensor node i belong CH_j, $i \in \{1,\ldots,L\}$
BS	The base station
Y	The base station's public key
$\|\|0^z$	Concatenation with z serial 0 bits
λ	Number of bits needed to represent the data captured
SK_{ij}^{BS}	The key shared between S_{ij} and BS
N_{ij}	A sequence number for data freshness
HKDF(K,N)	A KDF using HMAC as PRF
HMAC(K, m)	Message digest of m using SHA with key K

Table 1.Notations and their definitions.

3.5. Attack model

We categorize the attacker's abilities as follows:

Category.1. The attacker can eavesdrop and monitor the transmitted data in the WSN.
Category.2. The attacker can produce an illegal data, modify the transmitted packets and replay packets already transmitted.
Category.3. The attacker can stealthy compromise sensor nodes using power analysis.

The first two categories are described in the security analysis of CDA schemes in [18] and the last has been practically demonstrated in [34]. The first category concerns the passive attacker where its

ACCEPTED MANUSCRIPT

aim is to deduce the secret key. The most basic attack is *the Ciphertext Analysis (CA)*, in which the attacker tries to obtain information only by interpreting ciphertexts. The attacker can launch *the Known Plaintext Attack (KPA)*. In such an attack, the adversary tries to determine secret information with a known plaintext and corresponding ciphertext. In the same category, the attacker can also choose arbitrary plaintexts to be encrypted and study the resulting ciphertexts, known as *the Chosen Plaintext Attack (CPA)*. In the context of WSN, *CPA* is considered to be less practical than *CA* and *KPA* but is still a very dangerous attack. In the second category, the attacker can actively interfere the communication. Active attacks include *Malleability, packet forgery* and *replay*. Malleability allows the attacker to modify packets without necessarily knowing the content; particularly homomorphic encryptions are inherently malleable. The attacker can forge packets or even replay valid packets already "used" in the WSN, in order to deceive the base station. For more details about these attacks, refer to Peter et al.'s analysis [18]. In the third category, the attacker can perform a *stealthy compromise* or also known as side channel attack. This attack allows the adversary to use the information leaks during cryptographic operation to obtain all or part of the secret key [34]. In this category, we consider an attacker that takes power traces from the node without removing it from the network or disturbing it's functioning. In other words, the attacker is able to perform a *Simple Power Analysis (SPA).* More details are presented in Section 5.2.1.

3.6. Design goal

Under the aforementioned system models, our design goal is to develop an efficient and versatile CDA scheme for WSNs. More specifically, the following three objectives should be achieved.

Security: in the area of secure data aggregation, the two following security services are widely desired.

End-to-end confidentiality: prevents the intermediate nodes to access to the plaintext data.

End-to-end integrity: prevents the attacks that target the integrity of packets.

Efficiency: a secure data aggregation must be efficient in the two following aspects:

Computation overhead: the cryptographic operations must be efficient and do not require heavy computations, especially when the base station needs to continuously collect data from each sensor node about the target area.

Communication overhead: a secure aggregation scheme must highlight the advantage of using data aggregation in WSNs, and the security protocol does not affect this advantage.

Versatility: a secure data aggregation scheme must be versatile i.e. allows the base station to calculate any aggregation function on sensors data. This property is very important since it is needed to serve a wide range of sensor network applications.

4. PROPOSED SASPKC SCHEME

SASPKC is composed of two main phases namely, the *forwarding phase* and the *aggregation phase*. In the former, all sensors send their states, which will be used in the aggregation phase. In this latter phase, sensor nodes encrypt and authenticate their captured data using the state shared with BS. After that, the CH combines all ciphertexts and signatures into one ciphertext and one signature using the homomorphic operation and the *Xor* operation, respectively. Finally, the BS verifies the aggregated data by first, decrypting the aggregate and retrieving the plaintexts and then, by invoking the verification process. Details of the above two procedures, with the setup phase are showed in the following.

4.1. Setup phase

Bellare et al. [19] recommends ECC for StPKE for an efficient security. We suppose that before deployment, the BS generates its pair of keys (x, Y) where $Y=xG$, and keeps the private key x secret. Each sensor S_{ij} is loaded with a secret key SK_{ij}^{BS} shared only with BS and the elliptic curve domain parameters that are the set (Y, E, p, G, n), where Y is the public key and E the elliptic curve over prime

ACCEPTED MANUSCRIPT

field p with the base point G of order n. The sensors are also loaded with a large number M, a PRF-based KDF (NIST SP800-108 HKDF) and a secure MAC (HMAC).

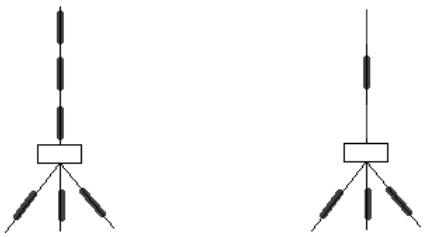
4.2. *Forwarding phase*

In this phase, sensor nodes send *the state* St_{ij}, which will be used to generate the sub-keys needed during aggregation phase. The authentication key is obtained using HKDF. In fact, the output K_{ij} of the PRF[3] (HMAC in our case) is computed with a nonce as the iteration variable, and then used as keying material for authentication. Each sensor S_{ij} executes Algorithm 1 and then sends the outputs .i.e. St_{ij} and MAC_{ij} to the next hop. In order to extract all keys by BS, all packets must be transmitted. Therefore, in this phase, the CH acts as a data forwarder and not as a data aggregator, as shown in Figure 4(a).

Algorithm 1: Forwarding phase (S_{ij})

Input: (Y, E, p, G, n), SK_{ij}^{BS}, Nonce
Output: St_{ij}, MAC_{ij}

1. Generate a random $r_{ij} \in [1, n-1]$
2. Compute $St_{ij}=r_{ij}G$
3. Compute $K_{ij}=HKDF(St_{ij} \| r_{ij}Y \| SK_{ij}^{BS}, N_{ij})$
4. Compute $MAC_{ij}=HMAC(St_{ij}, K_{ij})$



(a) (b)

Figure 4. Data transmission in SASPKC: (a) Forwarding phase and (b) Aggregation phase

Each sensor keeps (r_{ij}, St_{ij}) as state and uses this state for future encryptions. Once received, the CH forwards all data (including its own state) to the base station or the nearest CH. The BS then verifies the integrity and authenticates all the senders by using its private key x (correctness is guaranteed since $xSt_{ij}=xr_{ij}G=r_{ij}Y$). The verification is done by calculating all the keys corresponding to the received states and then by comparing the MAC_{ij}' computed with the one received (See Algorithm 2). If the verification holds, then the corresponding state St_{ij} will be stored in BS's database and used for future decryptions and verifications. Otherwise the state will be rejected. As a result of this phase, the BS shares a state St_{ij} with each node in the network.

Algorithm 2: Verification (BS)

Input: (Y, E, p, G, n), SK_{ij}^{BS},x , Nonce, All pairs (St_{ij} , MAC_{ij})
Output: MAC verification

1. For each $i \in \{1,\ldots,L\}$and $j \in \{1,\ldots,R\}$
   1.1.Compute $K_{ij}= HKDF(St_{ij}\|xSt_{ij}\|SK_{ij}^{BS},N_{ij})$
2. For each $i \in \{1,\ldots,L\}$ and $j \in \{1,\ldots,R\}$
   2.1. Compute $MAC_{ij}' = HMAC(St_{ij}, K_{ij})$
   2.2.if $MAC_{ij}' = MAC_{ij}$ Then accept
   Otherwise reject

4.3. *Aggregation phase*

[3]The PRF is iterated just once, since no encryption is considered in this phase

ACCEPTED MANUSCRIPT

This phase consists of three steps: *encrypt-aggregate-verify*. These steps work as follow:

Encrypt: In this step, the data value m_{ij} captured by S_{ij} is encoded before encryption. We slightly modify the Castelluccia et al.'s scheme in such a way that the encryption involves an encoded plaintext. The encoding function is similar to that used in [13], but the difference is that the resulting code e_{ij} is ciphered using symmetric encryption that is very faster than asymmetric one and also provides short ciphertexts. Unlike the forwarding phase, the HKDF outputs two keys in the aggregation phase namely, K_{ij1} and K_{ij2}, where $K_{ij1} < M$. The sensors encrypt the encoded plaintext and calculate the corresponding MAC using K_{ij1} and K_{ij2}, respectively. The encryption is performed using addition modulo the large number M, preloaded on sensors before deployment (See Algorithm 3). We note that M must be greater than $e_{agg} = \sum_{i=1\ldots L}^{j} e_{ij}$ otherwise correctness is not provided, if this property is verified then the decryption will result in a message e_{agg} that is smaller than M. The nonce N_{ij} used in HKDF ensures the dynamic keys needed for the security of encryptions. The MAC is then calculated on ciphertext (*encrypt-then-MAC*). Finally, the resulting ciphertext and MAC are sent to the corresponding CH_j.

Algorithm 3: Encrypt & sign (S_{ij})

Input: m_{ij} , (r_{ij} , St_{ij}) , Y , M , Nonce
Output: C_{ij} , MAC_{ij}

1. Encode m_{ij} into $e_{ij}=m_{ij}||0^z$, where $z= \lambda *(i-1)$.
2. Compute the current K_{ij}= HKDF ($St_{ij}||r_{ij}Y||Y$, N_{ij}) where $K_{ij} = K_{ij1}|| K_{ij2}$
3. Compute $C_{ij}= K_{ij1}+ e_{ij} \bmod M$.
4. Compute MAC_{ij}= HMAC (C_{ij}, K_{ij2})

Aggregate: In this step, the CH acts as a data aggregator, unlike forwarding phase (See Figure 4(b)). The CH combines all *L-1* ciphertexts including its own ciphertext into one ciphertext C_{agg}, the *L-1* MACs and its own MAC into one MAC_{agg} (See Algorithm 4). The ciphertexts are homomorphically aggregated using addition operation modulo M, and the MACs are Xored. After that, the outputs of Algorithm 4 i.e. C_{agg} and MAC_{agg} are sent to the BS or the nearest CH. For a CH that receives a packet from another CH, it just forwards the packet to the BS. In addition, the CH can notify the BS, the sensors that fail to send their packets. Each CH_j performs the homomorphic aggregation as follow:

Algorithm 4: Homomorphic aggregation (CH_j)

Input: All pairs (C_{ij} , MAC_{ij})where $i\in\{1,\ldots,L\}$
Output: C_{agg} , MAC_{agg}

1. For L ciphertexts ($C_{1j}\ldots C_{Lj}$)
   1.1.Compute $C_{agg}= \sum_{i=1\ldots L}^{j} C_{ij} \bmod M$
2. For L MACs ($MAC_{1j}\ldots MAC_{Lj}$)
   2.1.Compute $MAC_{agg}= \oplus MAC_{ij}$

Verify: In this step, after receiving all data packets i.e. corresponding to the aggregate of each cluster, the BS invokes the decryption and verification processes. The BS first computes the current keys corresponding to all network nodes using the states (shared in forwarding phase) stored in the BS's database. After that, the BS decrypts the aggregated ciphertext and retrieves the individual plaintexts (See Algorithm 5). Finally, after the calculation of each pair (C_{ij}, MAC_{ij}), the BS checks the end-to-end integrity. If the verification holds, the aggregated data e_{agg} will be accepted, otherwise reject. The BS can identify the malicious node by notifying the CH_j (corresponding to the infected aggregate) to send all pairs (C_{ij}, MAC_{ij}) and then by verifying each pair. Another advantage of our scheme is that there is no need to send the list of responding nodes to the BS since all sensors participate in the aggregate. In our scheme, each sensor produces an encryption and a signature MAC even if there is no sensed data. In fact, the non responding node simply uses a zero value of m in Algorithm 3, which does not influence the final result and, thus, avoiding the problem of [10] (See Section 3.2). Therefore, as the BS is provided with all sensors data[4], it is then able to perform any aggregation function on them, which is the main advantage of hop-by-hop solutions. Our scheme is then versatile and does not impose any bound on this function's nature. To summarize, i) the confidentiality service is provided through our modified additively homomorphic scheme, ii) the integrity service is ensured through MAC, and iii) the versatility aspect is guaranteed since the original individual messages are available for verification.

[4]The individual data can be obtained from e_{agg} using the decoding function (See Algorithm 5).

ACCEPTED MANUSCRIPT

Algorithm 5: End-to-end verification (BS)

Input: All pairs $(C_{agg}, MAC_{agg})_j$, Where $j \in \{1,\ldots, R\}$
Output: MAC verification

1. Compute all currents K_{ij}= HKDF(St_{ij}||xSt_{ij}|| Y , N_{ij})
2. For each pair $(C_{agg}, MAC_{agg})_j$
   2.1.Compute $e_{agg}= C_{agg}- \sum_{i=1\ldots L}^{j} K_{ij1} \mod M$
   2.2.Decode (e_{agg},L, λ): m_i=e[(i-1)* λ , λ *i-1], where i=1,…,L
   2.3.For each m_i, where i=1,…,L
      2.3.1.Compute MAC_i
   2.4. Compute $MAC'_{agg} = \oplus MAC_i$
   2.5. if $MAC'_{agg} = MAC_{agg}$ Then e_{agg} is accepted
   Otherwise e_{agg} is rejected

The aggregation phase is performed several times until that the states St_{ij} expire. Application's developers design the expiration date. The key expiration is a very important security measure where it allows a *key refresh*, which involves a new forwarding phase and then improves security. In our scheme, the node lifetime can be viewed as a sequence of epochs where each epoch consists of two phases namely forwarding and aggregation phase (see Figure 5). The expiration date is defined depending on the target application. In fact, the application's developers can judge that the security level used is not secure enough (for long term) for their sensitive information (e.g. against Brute Force Attack). So, a *key refresh* is important to improve the security. In this case, the expiration date can be preloaded on sensor before deployment, as used in our implementation in Section 5.2. Also, the synchronization with some nodes can be lost due to some malfunctioning, then, a novel forwarding phase can refresh synchronization. In this case, a novel forwarding phase can be requested from BS by using a specific active message.

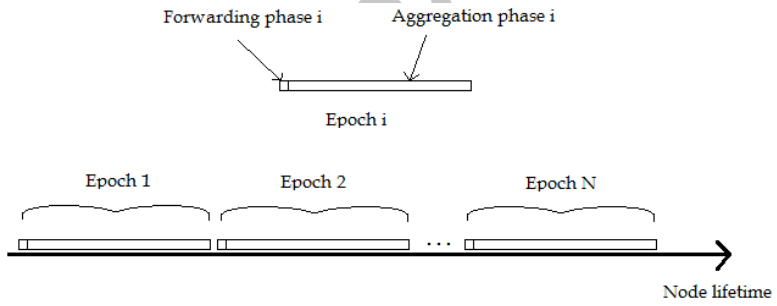


Figure 5. Node lifetime.

4.4. An illustrative example

In the following, we give an example to show how SASPKC works. As shown in Figure 6, we suppose a network consisting of a base station and 3 sensor nodes S_1, S_2 and S_3 where S_3 is selected as CH. Assume that the forwarding phase is achieved so, each node has a state shared only with BS. We suppose that the data sensed by these nodes are respectively 7, 8 and 10, thus the number λ=4 (needed to represent the data captured). Each sensor first encodes the plaintext and then performs an encryption by adding the encoded plaintext to the current key (obtained using HKDF) modulo a large number, and finally sends the resulting ciphertext as well as the corresponding MAC to the CH. We note that in this example, small numbers are utilized in order to simplify the comprehension. However, very large numbers are used in practice. After that, the aggregator performs the homomorphic operation on ciphertexts and transmits the result along with aggregated MAC to the BS. Finally, the BS extracts the keys, decrypts and decodes the aggregated result to obtain the individual data (7, 8 and 10), and finally invokes the verification process.

ACCEPTED MANUSCRIPT

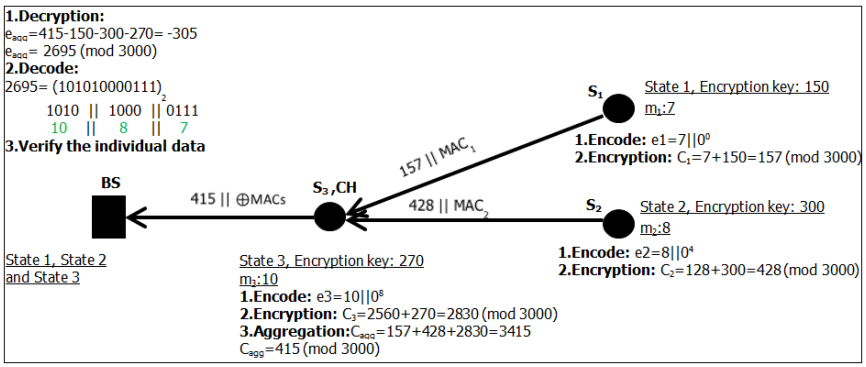


Figure 6. An illustrative example

5. Performance Evaluation and Analysis

In this section, we provide the security and performances analysis of SASPKC protocol. For security, we analyze the confidentiality and integrity and for the performances evaluation, we analyze the computation and communication overhead, energy consumption, scaling and portability.

5.1. Security analysis

SASPKC involves two phases; we begin with the security of forwarding phase:

Theorem 1. *The states St_{ij} transmitted toward the BS in forwarding phase are secure.*

Proof. The security is provided assuming that the Gap Diffie-Hellman GDH problem is hard[5] and the MAC is unforgeable. The MAC is calculated using the key K_{ij}, which is derived via HKDF from $r_{ij}Y$ (unknown to attacker assuming that the GDH problem is hard) and SK_{ij}^{BS}, the preloaded secret key shared with BS. The fact that $xSt_{ij} = r_{ij}Y$, the BS can verify the integrity of all packets and authenticate the senders using x and SK_{ij}^{BS}. Meanwhile, considering that SASPKC uses an unforgeable MAC protocol such as HMAC [28], attackers cannot successfully forge packets in transmission phase.□

We note that, unlike St_{ij}, which expires after an expiration date, SK_{ij}^{BS} is a key that never expires. After the expiration date, a sensor node will use the same key along with a nonce to send the new state.

SASPKC aims to provide the captured data with end-to-end confidentiality and end-to-end integrity:

Theorem 2. *SASPKC provides the end-to-end data confidentiality in the presence of an attacker of category.1.*

Proof. Our encryption is a slight modification of Castelluccia et al.'s scheme, which has indistinguishable encryptions. The difference is that the encryption involves an encoded plaintext. It is clear that this modification does not negate the security since our modification is in the plaintext domain. So, SASPKC is as secure as the Castelluccia et al.'s scheme [10]. Our encryption is information-theoretically secure i.e. it cannot be broken even against a computationally unbounded adversary. The authors of [10] claimed that to provide security, the keys used for encryption must be dynamic i.e. change from one message to another. In order to provide the chosen plaintext security for our symmetric encryption, we use a nonce in our PRF, which ensures the pseudo randomness of keys. In fact, the encrypting device uses in addition to the state, a nonce for encryption, and similarly, the BS uses the same state and nonce for decryption. The nonce is a public value, which does not need to be hidden from the attacker; the only requirement is that the pair (state, nonce) is only used to encrypt one plaintext. In other words, the pair (state, nonce) must change from one message to another. In SASPKC, the state is calculated just once and a new nonce[6] is generated for each message. The key K_{ij} is used to

[5]The GDH problem refers to a computational problem where an adversary, given (G,aG,bG) for unknown a,b∈ F_p, tries to compute a key abG with the help of DH-oracle, that is able to decide whether c=ab or not, given the tuple (G,aG,bG,cG) [35].
[6]The nonce used is an implicit sequence number, which is initialed and incremented from one packet to another.

ACCEPTED MANUSCRIPT

encrypt only one plaintext, since it changes in every round, and all keys are unique with a very high probability. Furthermore, the output of the PRF is directly used thanks to the encoding function employed. Roughly speaking, the choice of M still depends on e_{agg} in our case, even if the output is directly used. Thus, the IND-CPA level is achieved in SASPKC without the requirement of a length-matching hash function [24]. Therefore, the security against attacker of category.1 is provided.□

Recall from Section 3 that the attacker of category.2 targets the integrity of data, and try to deceive the base station by accepting a malicious result by replaying, modifying or forging packets.

Theorem 3. *SASPKC provides the end-to-end data integrity in the presence of an attacker of category.2.*

Proof. An attack of category.2 is directly tackled by the aggregate MAC scheme; if a malicious action against integrity is occurred, then the end-to-end verification will fail. The security proof of the scheme is provided in [27]. In the following, we prove the security of SASPKC through an analysis of the security against attacks that can be launched by an adversary of category.2:

Replay attack: SASPKC uses different keys for encryption and authentication from one packet to another. After an amount of time t, the attacker who replays the packet of the i-th round of aggregation phase, it is obvious that the check process will fail because the base station uses the key k of the $(i+ t)$-th round (no longer the key of the i-th round) to verify the data.

Malleability: As previously mentioned, the malleability is a serious threat for HE. The Castelluccia et al.'s scheme is vulnerable to this attack. In fact, a ciphertext $(m + k) \bmod M$ can easily be altered by: $(m+10)+k \bmod M = (m+k)+10 \bmod M$. SASPKC uses aggregate MACs which allows the base station to verify the data end-to-end and authenticate the senders. Consequently, if the encrypted data is modified, the check process will fail and the base station will reject the fake aggregate.

Packet forgery: There are two cases: i) forge packets without node compromise, and ii) forge packets with node compromise. In the first case, the attacker cannot produce a valid ciphertext and MAC without knowing the current corresponding keys, because SASPKC utilizes different keys for every message. In the second case, after node capture, the attacker can use the node to (i) act as a legitimate node (ii) forge a plaintext that is close to the legitimate plaintext, and (iii) forge a plaintext that is quite different from the plaintext. For the two first, there is no secure aggregation solution actually that detects such actions, but for the last, the base station in SASPKC is able to detect the malicious node through the encoding function employed because the forged message exceed the number of bits allowed to represent the data. Also, if the compromised node is a CH, the latter cannot access to the plaintext data of his member nor deceive the base station by accepting a malicious plaintext, because the corresponding keys are unknown to the attacker. In other words, the compromised aggregator can only modify its own plaintext to try to mislead the base station.□

5.2. Performance evaluation

Our scheme aims to reduce computation and communication overhead. In the following, we analyze our proposal's performances:

5.2.1. Computation overhead analysis

The implementation is done on TelosB motes [31], the 16-bit Texas Instruments MSP430F1611 microcontroller with 48 KB of flash ROM, 10 KB of static RAM, and running at a clock frequency of 8 MHz. To analyze the computation complexity, we denote symbol *SM* is the cost of one scalar multiplication, *MA* is the cost of one modular addition and *SG* is the cost of one signature. Note that *SG* is also considered for one HKDF because HMAC is used for both key derivation and signature generation. In forwarding phase, each sensor has to compute its state and forwards it to the base station. The computation involves *2SM+2SG* operations. In aggregation phase, each sensor takes *MA+2SG* operations to compute an encryption and a signature. The CH nodes spend *2*(L-1)*MA* operations (considering that the *Xor* operation is also a modular addition (mod 2)) to perform the homomorphic aggregation. We note that BS operations are not analyzed because it is considered as a powerful device. Therefore, the total computation cost of SASPKC for one epoch is *2SM+2SG+ NS*(NR*(MA+2SG) + 2*R*MA*(L-1))* where *NS* is the number of session in one aggregation phase.

ACCEPTED MANUSCRIPT

To give a sense to this, we implement the above operations on TelosB. In our implementation, we use TinyOS and TinyECC, TinyOS is an open source designed for low-power wireless devices [36] and TinyECC is a freely library that provides operations of elliptic curve over prime field F_p [37], the domain parameters secp160r1 is used. We develop an application, which provides state calculation, symmetric encryption and MAC calculation. For our HKDF and signatures, we use the HMAC provided in the library that considers SHA-1 as hash function. The HKDF produces a 20 bytes key in forwarding phase and two 20 bytes keys (for encryption and authentication) in aggregation phase. Note that M is chosen to be 2^{160}, in order to avoid any overflow. Before calculating the computation cost of our cryptographic functions, we focus on scalar point multiplication, its efficiency and its security against side channel attacks.

The *SM* is the major operation of ECC, its represents 80% of the key calculation time [38]. TinyECC provides some optimization switches that can be turn on or off based on application needs, including the switch of Sliding Windows Method (SWM). This method uses some precomputation to improve the performance of *SM* especially when the involved point is fixed and known a priori. In SASPKC, the *SM* is performed with the two points G and Y namely, the base point and the BS's public key, which are both, fixed and known a priori. The execution of *SM* involves the execution of both, *point addition* (A) and *point doubling* (D) if the corresponding bit of the expansion of the scalar is 0, and only a *point doubling* in the other case. Among the contributions of the present work is to propose a protected implementation of SASPKC against SPA (Attacker of category.3). The purpose of SPA attack is to deduce the sequence of A and D from a single power trace obtained during the execution of *SM*. These two operations have different costs where A takes longer than D [39]. In order to prevent SPA, some countermeasures must be incorporated in the implementation in order to make the processing time of the *SM* algorithm independent from the operands. With unprotected implementation, the state in SASPKC is compromised and consequently all past and future encryptions will be compromised. In our implementation, we use the fast and SPA-secure scalar multiplication proposed in [40], used to secure the implementation of ECEG in [17]. This method not only improves performance but also avoids SPA. In fact, a new representation of the scalar is utilized, which is represented by a sequence of non-zero bit-strings with 1 and -1. Also, the algorithm stores only 2^{w-1} points instead of 2^w-1 for the original comb method [41], where w is the window used, and this without increasing the execution time. More details about comb method and its SPA-secure version can be found in [17]. Note that the two methods are added to the library.

Theorem 4. *SASPKC provides the security against attacker of category.3.*

Proof. The implementation considers the SPA-secure algorithm [40] in which the operations performed consist of a sequence of alternative point doubling and point addition, namely, DA|DA|…DA|DA. Therefore, our implementation does not leak any information about the secret r_{ij} to SPA attacker.□

Table 2 shows the execution time of our cryptographic functions implemented on TelosB mote. The precomputations for both G and Y are performed in the *Stpke.init* () function. The execution time is taken by the average of several executions. Since the curve points Y and G are fixed, the overhead of *Stpke.init* () can be neglected by performing the precomputations offline and distribute the points to every sensor node prior to deployment. We note that SPA attack is only considered for state calculation namely, *Stpke.state* (). In fact, the state in SASPKC is used for long term (aggregation phase), and with unprotected implementation, SPA attacker can recover the state, which consequently compromises all communications in aggregation phase. However, the keys used for encryption and authentication change from one message to another. Therefore, the operations performed in aggregation phase namely, *Stpke.encrypt* () and *Hom.Add* () are not vulnerable to SPA.

Cryptographic function		Execution time	SPA
Stpke.state()	SWM (w=4)	5.82s	Yes
	Comb (w=4)	2.71s	Yes
	Sec.Comb (w=4)	2.85s	No
	Sec.Comb (w=5)	2.29s	No
Stpke.encrypt()		0.081s	No
Hom.add()		0.002s	No

Table 2. Execution time of our cryptographic functions

ACCEPTED MANUSCRIPT

Our results presented in Table 2 show that the Comb method can significantly improve the execution time of *SM* needed in forwarding phase. Also, it is showed that the SPA-secure version proposed in [40] is not only secure against SPA but also faster, namely, 61% and 15% than SWM and Comb, respectively, which are both vulnerable to SPA. Meanwhile, in SASPKC, the point compression technique is adopted to transmit the state, which allows representing a point using the minimum possible number of bits. The decompression effort requires the computation of one square root in prime fields [34], but since this operation is only performed at BS which is assumed to be a powerful device in terms of computation capabilities, energy resources, memory, etc. this operation can be done efficiently using algorithm such as [42].

In aggregation phase and for encryption, the sensor takes about 0.081s to encrypt and produce a MAC i.e. *MA+2SG*. If we compare with encryption effort of related works, we can find that our proposal is the most efficient. The works [13,14] both use ECEG for encryption, which takes approximately the same execution time as our forwarding phase. In our scheme, this time is consumed only in forwarding phase. The major advantage of stateful encryption is that two *SM* are saved for each encryption. In aggregation level, the aggregator performs the homomorphic operation on encrypted data and needs only a few amount of computation to perform the aggregation. In [13-15], this operation needs to perform calculation over elliptic curve, which causes not only an important computation overhead that leads to high energy consumption but also increases the end-to-end delay where the BS must to wait an important amount of time before receiving the aggregated data. Our scheme takes advantages of both kind of encryption to reduce the computation overhead, it uses El Gamal scheme to generate a state (shared only with BS), enabling encryption in all future transmissions. Our system also uses the best features of symmetric encryption that are the fast computation and small ciphertext.

5.2.2. Communication overhead analysis

Fisrt of all, the complexity of communication is *O(1)* for non CH nodes and *O(L)* for CH nodes in forwarding phase because all packets are transmitted toward the BS. Therefore, the total communication cost (i.e. the number of transmitted packets) in this phase is *R(2L-1)*. In aggregation phase, the communication is a convergecast traffic toward the BS where every sensor sends one packet .i.e. *O(1)* for both Non-CH and CH nodes. Therefore, the total communication cost is *N*. We note that we consider the case where CH nodes are directly connected to the BS. Otherwise, these costs could increase, the increase will depend on the depth, The number *L* and the actual phase.

In what follow, we present new simulation results that complement, in terms of communication overhead and energy consumption, the experimental and analytical study presented in the previous sections. Results fully confirm the suitability of our protocol to ensure a high level of security with minimal overhead. We select TOSSIM [43], a widely used simulator for sensor networks. We use the TOSSIM-CC2420 simulator provided in the TinyOS distribution. It models the CC2420 radio. We simulate the three schemes, SHA [14], EVCDA [13], and ours in order to properly make the comparison and to obtain accurate results. The reason to choose these two works for comparison is that they provide a comparable level of security (end-to-end confidentiality and integrity) as ours. In three simulation scenarios, 50, 100 and 150 sensor nodes are randomly deployed within a square area with a single base station centered.

In our scheme, the states are points that belong to the elliptic curve and they are transmitted in their compressed form which requires only 1+ $[\log_2 p]$ bits, which means 21 bytes in our implementation and the HMAC used outputs 160 bits (20 bytes) that can be truncated to 10 bytes following [28]. In aggregation phase, we use an encoded plaintext of 20 bytes, which results in a 20 bytes ciphertext. Hence, our proposal uses only short packet in order to provide both end-to-end confidentiality and integrity. The end-to-end security proposed in [13-15] incurs much communication overhead compared to our scheme. The overhead can be computed following the way used to send the corresponding packets, which corresponds to two different ways: i) send longer messages (since 802.15.4 supports up to 127 bytes) which increases the bit error rate and decreases the reliability of the network as used in [14], and (ii) split the packet into blocks and send each block separately which incurs not only delay but also additional headers overhead as used in [15]. Nevertheless, whatever the way used in these schemes, we argue that our solution provides greater reduction in energy consumption due to the number and size of messages needed to validate the sensors data. To give a sense to this, we run our three simulations, each simulation run lasts for 500s, and each result is averaged over several simulation runs. We use a simple TDMA-based clustering algorithm to transmit data to the base station. 4, 8 and 12 Cluster Heads (CHs) are selected for the three topologies 50, 100 and 150, respectively, thus, the number of nodes per cluster $L \in [8, 13]$. In our application, the non-CH node sends (every 20 seconds) a packet to the corresponding CH, the CH aggregates and sends the

ACCEPTED MANUSCRIPT

result to the BS. For our scheme, non-CH nodes first send their states and then securely report information (using the state) about the region where they are deployed to the BS. We note that for the same simulation time, an expiration date is considered 5 times in 5E (5 Epochs) and 10 times in 10E (10 Epochs), while 1E (1 Epoch) represents only one forwarding phase and one aggregation phase. We measure the communication overhead as the total data transmitted in the network. The results are presented in Figure 7 for different scenarios.

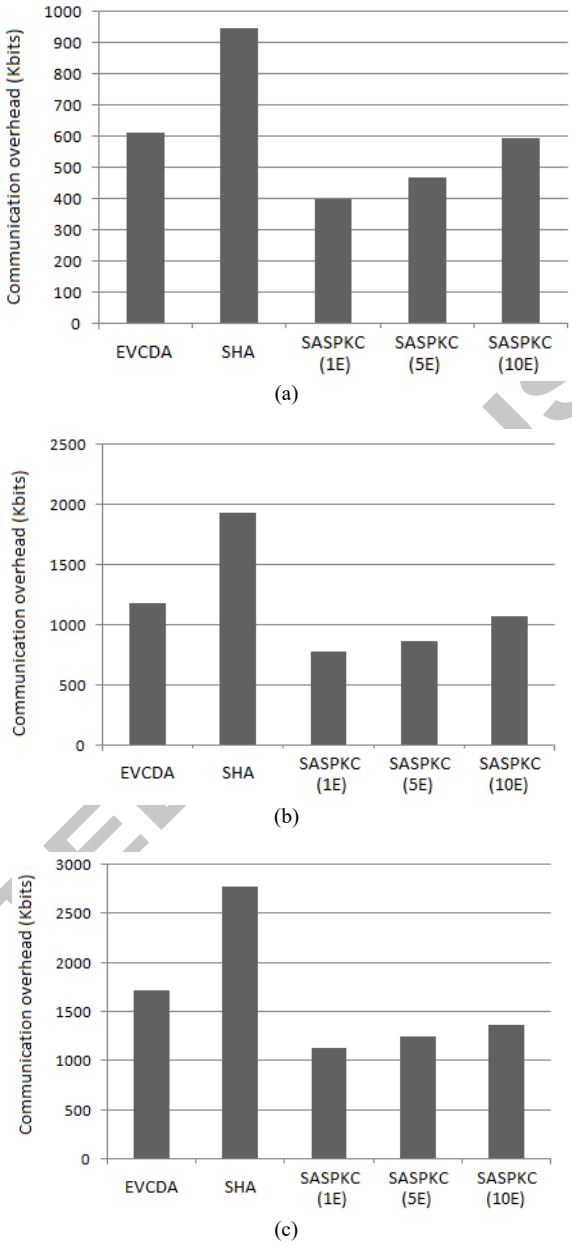


Figure 7. The communication overhead for different topologies: (a) 50 nodes, (b) 100 nodes and (c) 150 nodes

Simulations results show that our scheme incurs less communication overhead compared to related works. This is due to the use of stateful public key encryption in which data is encrypted using symmetric encryption that yields to short ciphertexts and consequently short packets. The results show also that even if the security is improved in our scheme (by using several epochs for the same time duration), the overhead remains acceptable. Furthermore, the advantage of using StPKE can be observed in the case where the density of the network increases, see Figure 7(b) and (c).

ACCEPTED MANUSCRIPT

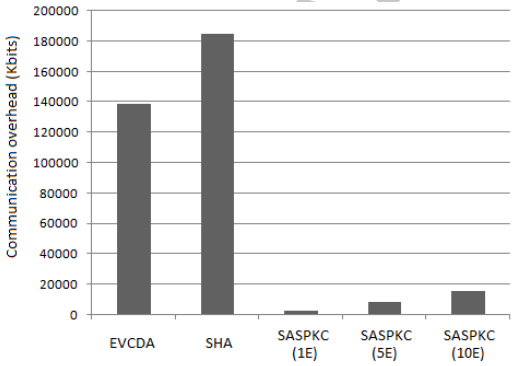
5.2.3. Energy consumption analysis

Energy consumption is the core issue in WSNs. Computation and communication are two aspects that have a direct impact on energy consumption and consequently the node's lifetime. TOSSIM-CC2420 incorporates PowerTOSSIM [44], a power modeling extension to TOSSIM. PowerTOSSIM includes a model of the power consumption of the TelosB motes. However, TOSSIM does not model CPU execution time; it cannot provide accurate information for calculating CPU energy consumption. For this aim, we add for every transmitted/received packets the corresponding computation overhead involved, we take the model used in [37], the energy consumption E can be calculated by using the formula E = U×I×t, whereby U denotes the voltage, I denotes the current, while the execution time is represented by t. For 2 AA batteries, the voltage is about 3.0V. As stated in [37], the amount of current draw for TelosB is 1.8mA for MCU On/Radio Off. Hence, the energy consumption of our cryptographic functions can be calculated and are presented in Table 3.

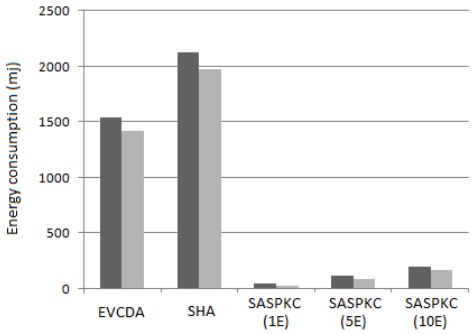
Cryptographic function	Energy consumption
Stpke.state()	12.366 mj
Stpke.encrypt()	0.437 mj
Hom.add()	0.009 mj

Table 3. Energy consumption of our cryptographic functions

The above measures are added to our energy model. We perform our simulations for the three schemes (we note that for SHA, we use the measures provided in [45] and for EVCDA, we consider an encryption cost of *3SM*, one *D* and one hash while an aggregation cost of *2*(L-1)*D* as stated in [46]), each simulation run lasts for 500s, the estimated energy consumptions in the whole network, and at CH and Non CH nodes are presented in Figure 8(a), and Figure 8(b), respectively.



(a)



(b)

Figure 8.The estimated total energy (a) in the whole network (b) at CH and Non-CH nodes in a network of 100 nodes.

It is apparent from Figure 8(a) that our scheme provides a great reduction of energy consumption compared with related works. This gain can be explained by the fact that much lesser computation overhead is incurred in our scheme due to the use of symmetric primitives and the efficient implementation of asymmetric operations (states calculation). Consequently, for the same level of security provided, the network lifetime is hugely improved. In Figure 8(b), we show that the average

ACCEPTED MANUSCRIPT

energy consumed by CH and Non-CH nodes is reduced considerably in our scheme and this is due to, on one hand, the nodes continuously (for each aggregation round) perform intensive calculations in EVCDA and SHA while in our scheme the costly operations are performed just in forwarding phases, on the other hand, in addition to participating in the aggregation process, the CH nodes perform aggregation function (the homomorphic operation). In SHA and EVCDA, this operation requires addition over elliptic curve while it requires only a few amount of computation in our proposal.

Recent works [47,48] showed that the energy cost of communications could be neglected with asymmetric computations. The authors, however, stated that if the energy required for cryptographic computations is reduced, this cost would become important. StPKE improves the computational cost of traditional PKE by saving two exponentiations (two ECC's scalar multiplications). Therefore, it gives us the opportunity and the possibility to consider the communication cost and consequently, to highlight the advantage of using data aggregation in WSNs. In Table 4, the estimated total energy costs of SHA, EVCDA and SASPKC at CH and Non-CH node are presented. The results show that as opposed to SASPKC, computations compose almost the overall cost of SHA and EVCDA on TelosB mote.

Scheme		**CH**	**Non-CH**
SASPKC(1E)	Comm	33.446 (60%)	15.029 (48%)
	Comp	21.999 (40%)	16.649 (52%)
	Total	55.465	31.678
SASPKC(5E)	Comm	35.711 (32%)	15.191 (20%)
	Comp	77.461 (68%)	63.038(80%)
	Total	113.172	78.229
SPSPKC(10E)	Comm	38.103 (21%)	15.284 (11%)
	Comp	141.145 (79%)	128.369 (89%)
	Total	179.248	143.653
EVCDA [13]	Comm	39.007 (3%)	16.748 (1%)
	Comp	1497.143 (97%)	1408.692 (99%)
	Total	1536.15	1425.44
SHA [14]	Comm	90.744 (4%)	27.488 (1%)
	Comp	2038.263 (96%)	1950.672 (99%)
	Total	2129.007	1978.16

Table 4. The estimated total energy costs (in mj) of SHA, EVCDA and SASPKC at CH and Non-CH node in a network of 100 nodes.

5.2.4. Scaling Analysis

The proposed scheme is scalable, we can add as clusters as we want. The only condition is that, the number of nodes per cluster L does not exceed the maximum number of nodes that can support the encoding function. In Table 5, we show how this number can change with the security level and λ (the number of bits needed to represent the data). We note that the reference technique such as used in [49] can significantly reduce the number λ. For multi-hop networks, the CH that receives an aggregated ciphertext from another CH has just to transmit the corresponding packet to the BS or the nearest CH.

Security level	**λ**	**L**
80 bits	1	80
	2	40
	4	20
	8	10
160 bits	1	160
	2	80
	4	40
	8	20

Table 5. Maximum number of nodes per cluster

ACCEPTED MANUSCRIPT

5.2.5. Portability Analysis

SASPKC has been also implemented on another popular platform namely MicaZ mote, equipped with the 8-bit ATmega128L processor clocked at 7.3728 MHz, 4 kBRAM, and 128 kB Flash memory [31], and in which encouraging results were obtained. In fact, the state calculation needs only 1.48s using the algorithm proposed in [40], the *encrypt&sign* function takes about 0.057s and the homomorphic operation requires only a few amount of computation namely 0.0012s. Following the model used in [37], the corresponding energy consumption is calculated and presented in Table 6. Since TelosB is more power-efficient than MicaZ, its energy consumption is lower. In fact, the results show that even if the execution time on MicaZ mote is 34% faster than TelosB, the latter requires only about 27% energy consumed by MicaZ mote for the same time duration.

Cryptographic function		**MicaZ**	**TelosB**
Stpke.state()	Time	1.48s	2.29s
	Energy	44.24mj	12.366mj
Stpke.encrypt()	Time	0.057s	0.081s
	Energy	1.7mj	0.437mj
Hom.add()	Time	0.0012	0.002s
	Energy	0.036mj	0.009mj

Table 6. Estimated energy costs of SASPKC computations on MicaZ and TelsoB motes.

5.2.6. SASPKC vs. Related Work

In Table 7, a comparison in terms of security (confidentiality and integrity), efficiency (computation and communication overheads) and versatility with related works is presented. The major advantages of our protocol are that all sensing data can be verified only at reader device (BS). Also, it employs symmetric key encryption and MAC, which lead to an efficiency in terms of computation and communication overhead. Finally, the versatility is provided. In fact, it is more interesting to provide the WSN's user with precise data instead of an aggregate representing the individual messages.

Scheme	**Security**		**Efficiency**		**Ver**
	C	**I**	**Comp**	**Comm**	
Hu et al. [6]		•	+	+	•
Castelluccia et al. [10]	•		+	+	
Zhu et al. [12]	•	•	+	+++	
Albath et al. [13]	•	•	+++	+++	
Sun et al. [14]	•	•	+++	+++	•
Ozdemir et al. [15]	•	•	+++	+++	
SASPKC	•	•	+	+	•

C : Confidentiality – **I** : Integrity
Comp : Computation cost-**Comm** : Communication cost
Ver : Versatility
+++: High
++ : Medium
+: Low
• : Provided

Table 7. Comparison of security, efficiency and versatility.

6. CONCLUSION AND FUTURE WORKS

We proposed SASKPC, a novel secure data aggregation scheme for WSNs based on stateful public key encryption and homomorphic encryption. In WSNs, performing data aggregation while ensuring data confidentiality and integrity is a challenge. The proposed scheme uses an additive homomorphic

ACCEPTED MANUSCRIPT

encryption and aggregate MAC to provide the end-to-end confidentiality and the end-to-end integrity, respectively. Experimental and simulation results confirmed the efficiency of our proposal in terms of energy used and show that, compared to other model proposed in current literature, our scheme achieves a comparable level of security with considerably better performance. To the best of our knowledge, this work is the first to address the security issues of data aggregation in wireless sensor networks by using stateful public key encryption. The proposed scheme can be very useful in a military application where the base station needs to continuously collect relevant data from each sensor node about the target area, and this, while providing a strong security against adversary that aims to disturb the network. In future work, we aim to extend our work to support nodes mobility, while considering new attacks such as selective forwarding.

REFERENCES

[1] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* 2013; *29*(7), 1645-1660. DOI:10.1016/j.future.2013.01.010

[2] Yick, J., Mukherjee, B., & Ghosal, D. Wireless sensor network survey. *Computer networks* 2008; *52*(12), 2292-2330. DOI: 10.1016/j.comnet.2008.04.002

[3] Alcaraz, C., Najera, P., Lopez, J., & Roman, R. Wireless sensor networks and the internet of things: Do we need a complete integration?. In *1st International workshop on the security of The internet of Things (SecIoT)*.November 2010

[4] Akkaya, K., Demirbas, M., & Aygun, R. S. The impact of data aggregation on the performance of wireless sensor networks. *Wireless Communications and Mobile Computing* 2008; 8(2), 171-193. DOI: 10.1002/wcm.454

[5] Chen, X., Makki, K., Yen, K., & Pissinou, N. Sensor network security: a survey. *IEEECommunications Surveys & Tutorials*2009; 11(2), 52-73. DOI:10.1109/SURV.2009.090205

[6] Du, W., Deng, J., Han, Y. S., & Varshney, P. K. A witness-based approach for data fusion assurance in wireless sensor networks. Proceeding of the *IEEE Global Telecommunications Conference, 2003. GLOBECOM'03.* December 2003. IEEE, Sans Francisco, 2003; 1435-1439.

[7] Hu, L., & Evans, D. Secure aggregation for wireless networks. Proceeding of the *IEEE Symposium on Applications and the Internet Workshops*, January 2003. IEEE, Orlando FL, 2003 ;384-391.

[8] Przydatek, B., Song, D., & Perrig, A. SIA: Secure information aggregation in sensor networks. Proceedings of the *1st international conference on Embedded networked sensor systems*. November 2003. ACM, L.A, California, 2003; 255-265.

[9] Westhoff, D., Girao, J., & Acharya, M. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Transactions on Mobile Computing* 2006; *5*(10), 1417-1431. DOI: 10.1109/TMC.2006.144.

[10] Castelluccia, C., Mykletun, E., & Tsudik, G. Efficient aggregation of encrypted data in wireless sensor networks. Proceeding of *the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services(MobiQuitous),* July 2005. IEEE, San Diego, 2005; 109-117.

[11] Mykletun, E., Girao, J., & Westhoff, D. Public key based cryptoschemes for data concealment in wireless sensor networks. Proceeding of the*IEEE International Conference on Communications. ICC'06.*June 2006. IEEE, Istanbul, Turkey, 2006; 2288-2295.

[12] Zhu, W. T., Gao, F., & Xiang, Y. A secure and efficient data aggregation scheme for wireless sensor networks. *Concurrency and Computation: Practice and Experience*2011; *23*(12), 1414-1430. DOI: 10.1002/cpe.1615

[13] Sun, H. M., Lin, Y. H., Hsiao, Y. C., & Chen, C. M. An efficient and verifiable concealed data aggregation scheme in wireless sensor networks. Proceeding of *the International Conference on Embedded Software and Systems, ICESS.* July 2008. IEEE, Sichuan, 2008; 19-26.

[14] Albath, J., & Madria, S. K. Secure hierarchical data aggregation in wireless sensor networks. Proceeding of *Wireless Communications and Networking Conference, WCNC*. April 2009. IEEE, Budapest, 2009; 1-6.

[15] Suat Ozdemir, Yang Xiao. Integrity protecting hierarchical concealed data aggregation for wireless sensor networks. *Computer Networks*2011; 55(8), 1735-1746. DOI: 10.1016/j.comnet.2011.01.006

[16] Fontaine and Galand. A Survey of Homomorphic Encryption for Nonspecialists. *EURASIP Journal on Information Security*, volume 2007, 1-15, DOI: 10.1155/2007/13801

[17] Merad Boudia , OR., &Feham , M. Fast and secure implementation of ECC-based concealed data aggregation in WSN. Proceeding of *Global Information Infrastructure Symposium,GIIS*. October 2013. IEEE, Trento, Italy, 2013; 1-7.

[18] Peter, S., Westhoff, D., & Castelluccia, C. A survey on the encryption of convergecast traffic with in-network processing. *IEEE Transactions on Dependable and Secure Computing*2010; *7*(1), 20-34. DOI: 10.1109/TDSC.2008.23

[19] Bellare, M., Kohno, T., & Shoup, V. Stateful public-key cryptosystems: how to encrypt with one 160-bit exponentiation. Proceeding of *the 13th ACM conference on Computer and communications security*. October 2006. ACM, Alexandria, VA, 2006; 380-389.

[20] Boneh, D., Gentry, C., Lynn, B., & Shacham, H. Aggregate and verifiably encrypted signatures from bilinear maps. Proceeding of *Advances in cryptology—EUROCRYPT* 2003. Springer Berlin. 2003; 416-432.

[21] Don Johnson, Alfred Menezes, Scott A. Vanstone: The Elliptic Curve Digital Signature Algorithm (ECDSA). Int. J. Inf. Sec. 1(1): 36-63 (2001).

ACCEPTED MANUSCRIPT

[22] D. Boneh, E. Goh, K. Nissim, "Evaluating 2-DNF Formulas on Ciphertexts", TCC '05,LNCS 3378, pp. 325-341, 2005.

[23] J.Baek, H.Chiang Tan, J.Zhou and J.W.Wong: Realizing Stateful Public Key Encryption in Wireless Sensor Network, 23rdInternational Information Security Conference; (Boston: Springer), pp. 95–107. 2008.

[24] CHAN, Aldar C.-F. et CASTELLUCCIA, Claude., "On the privacy of concealed data aggregation", In : Computer Security–ESORICS 2007. Springer Berlin Heidelberg, 2007. p.390-405.

[25] Chan, A. C., & Castelluccia, C., "A security framework for privacy-preserving dataaggregation in wireless sensor networks", ACM Transactions On Sensor Networks (TOSN),7(4), 29. 2011.

[26] C. Gentry, Fully homomorphic encryption using ideal lattices, Symposium on the Theory of Computing (STOC), 2009, pp. 169-178.

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

[28] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashingfor Message Authentication", February 1997.

[29] J. Kim, A. Biryukov, B. Preneel, and S. Hong, "On the Security ofHMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 andSHA-1," Proc. Fifth Int'l Conf. Security and Cryptography forNetworks (SCN), 2006.

[30] CHEN, Lily. Recommendation for key derivation using pseudorandom functions. NIST Special Publication, 2008, vol. 800, p. 108.

[31] Crossbow technology. http://www.xbow.com.

[32] W. Heinzelman, A Chandrakasan, and H. Balakrishnan,"Energy-efficient communication protocol for wireless micro sensor networks," In Proceedings of the Hawaii, International Conference on System Sciences, January2000.

[33] O. Younis and S. Fahmy, "HEED: A hybrid, energyefficient,distributed clustering approach for ad-hoc sensornetworks," IEEE Transactions on Mobile Computing,vol. 3, no. 4, pp.366-379, October 2004.

[34] Giacomo de Meulenaer, François-Xavier Standaert: Stealthy Compromise of Wireless Sensor Nodes with Power Analysis Attacks. *MOBILIGHT* 2010: 229-242.

[35] T. Okamoto and D. Pointcheval. The Gap Problems: A New Class of Problems for the Security of Cryptographic Schemes. PKC '01, Lecture Notes in Computer Science Vol. 1992, K. Kim ed., Springer-Verlag, 2001.

[36] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay,D., Hill, J., Welsh, M., Brewer, E., Culler, D.: TinyOS: An operating system forWireless Sensor Networks. In: Weber, W., Rabaey, J., Aarts, E. (eds.) AmbientIntelligence, Springer, New York (2004).

[37] An Liu, Peng Ning, "TinyECC: Elliptic Curve Cryptography for SensorNetworks(Version2.0). http://discovery.csc.ncsu.edu/software/TinyECC/, August 2010.

[38] N. Gura, A. Patel, A. S. Wander, H. Eberle, and S. Chang Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science, pp. 119–132. Springer Verlag, 2004.

[39] Merad Boudia Omar Rafik and Mohammed Feham. The impact of ECC's scalar multiplication on wireless sensor networks. In : 11th International Symposium on Programming and Systems (ISPS), 2013. IEEE, 2013. p. 17-23.

[40] Hedabou, M., Beneteaus, L., & Pinel, P. Some ways to secure Elliptic Curve Cryptosystem. Advances in Applied Clifford Algorithm, 18, pp. 677–688. 2008.

[41] C. Lim and P. Lee, "More Flexible Exponentiation with Precomputation," *Advances in Cryptology* – CRYPTO'94, LNCS 839, pp. 95–107, Springer-Verlag, 1994.

[42] Billy Bob Brumley, Kimmo U. Järvinen: Fast Point Decompression for Standard Elliptic Curves. *EuroPKI* 2008: 134-149.

[43] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire tinyos applications. In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003). *ACM Press*, November 2003.

[44] Victor Shnayder, Mark Hempstead, Bor-rong Chen, and Matt Welsh, Harvard University. PowerTOSSIM: Efficient Power Simulation for TinyOS Applications. *SenSys2004*, November 2004

[45] Kumar, V., & Madria, S. K. Secure hierarchical data aggregation in wireless sensor networks: Performance evaluation and analysis. Proceeding of *the IEEE 13th International Conference on Mobile Data Management (MDM),* July 2012. IEEE, Bengaluru, Karnataka, 2012; 196-201.

[46] Chen, C. M., Lin, Y. H., Lin, Y. C., & Sun, H. M. RCDA: recoverable concealed data aggregation for data integrity in wireless sensor networks. *IEEE Transactions onParallel and Distributed Systems*2012*, 23*(4), 727-734. DOI: 10.1109/TPDS.2011.219

[47] De Meulenaer, G., Gosset, F., Standaert, O. X., & Pereira, O. On the energy cost of communication and cryptography in wireless sensor networks. Proceeding of *IEEE International Conference on Wireless and Mobile Computing Networking and Communications, WIMOB*. October 2008. IEEE, Avignon, 2008; 580-585.

[48] Piotrowski, K., Langendoerfer, P., & Peter, S. How public key cryptography influences wireless sensor node lifetime. Proceedings of *the fourth ACM workshop on Security of ad hoc and sensor networks*. October 2006. ACM, Alexandria, VA, USA, 2006; 169-176.

[49] Sanli, H. O., Ozdemir, S., & Cam, H. SRDA: secure reference-based data aggregation protocol for wireless sensor networks. Proceeding of *IEEE 60th Vehicular Technology Conference, VTC.* September 2004. IEEE, Los Angeles, CA, USA, 2004; 4650-4654.