Annals of Telecommunications (2019) 74:157–165
https://doi.org/10.1007/s12243-018-0684-x

Additively homomorphic encryption and fragmentation scheme for data aggregation inside unattended wireless sensor networks

Katarzyna Kapusta[1] · **Gerard Memmi**[1] · **Hassan Noura**[2]

Received: 15 May 2018 / Accepted: 20 November 2018 / Published online: 4 January 2019

Abstract
Data fragmentation and dispersion is recognized as a way of providing confidentiality and availability of data stored inside unattended wireless sensor networks. This paper introduces an additively homomorphic encryption and fragmentation scheme (AHEF). AHEF replaces additively homomorphic secret sharing used in state-of-the-art techniques with additively homomorphic fragmentation. This change has a significant impact on the volume of data stored inside the sensors which in turn supports lower transmission costs. Both, storage and transmission costs are reduced by a factor of at least 2. Moreover, by reducing the number of required computations, AHEF allows sensors to save energy.

Keywords Unattended wireless sensor networks · Internet-of-Things · IoT · UWSN · Data fragmentation · Data aggregation · Data protection · Additively homomorphic encryption · Information dispersal

1 Introduction

Wireless sensor networks are widely employed to capture all kinds of environmental information. They usually operate in a quasi real-time mode where sensors first acquire data and then push it towards a central network static node named *sink*. However, in some situations, the presence of the sink may not be ensured. For instance, this may happen when sensors are deployed over vast or hostile areas like national parks, battlefields, international border zones, etc. The term of *unattended wireless sensor networks* (UWSN) was introduced in [1] to define a class of sensor networks where data are stored inside the sensors waiting to be collected by a *mobile sink* (for instance a drone) that periodically visits the network.

UWSN raise security challenges as data kept inside the nodes may be exposed to various types of attackers wanting to read, destroy, or corrupt the stored information. Several strategies for data protection have been proposed. For instance, in [2], data are replicated and moved around the network in order to achieve data survivability. A different approach ensuring data confidentiality, consists in making the sensors encrypt, fragment, and disperse data over their neighbors. Data reconstruction is then impossible unless a given threshold of k out of n data fragments is gathered. Therefore, data are protected against an attacker as long as they are unable to compromise the required amount of sensors between consecutive visits of the sink.

Several fragmentation schemes were designed to be used in UWSN. They propose a processing based on secret sharing [3] or combination of data encryption and error-correction codes [4–6]. Some of the proposals use additively homomorphic schemes in order to reduce storage and cope with energy consumption and transmission costs [3, 7] which are an important issue in UWSN. In the additively homomorphic encryption and fragmentation (AHEF) scheme presented in this paper as well as in the

✉ Katarzyna Kapusta
katarzyna.kapusta@telecom-paristech.fr

Gerard Memmi
gerard.memmi@telecom-paristech.fr

Hassan Noura
hn49@aub.edu.lb

[1] LTCI, Telecom ParisTech, Paris, France

[2] Department of Electrical and Computer Engineering, American University of Beirut, Beirut, Lebanon

HEHSS scheme [3], fragmented and encrypted data are aggregated inside the neighbors' nodes. However, AHEF significantly revisits and improves the HEHSS scheme; instead of using additively homomorphic secret sharing for data fragmentation, AHEF fragments data using an additively homomorphic information dispersal algorithm. This change has a big impact on the resulting storage overhead which in turn supports scalability and lowers transmission costs. Indeed, fragments obtained by using secret sharing have the same size as the data itself. Information dispersal produces fragments of optimal size (see the explanation in [8]), avoiding an increase in the volume of data to be protected. In addition, AHEF supports the aggregation of fragments from a single sensor as well as fragments coming from a cluster of sensors.

Outline This paper is organized as follows: Section 2 describes related works. Section 3 presents a simplified view of the architecture of the considered network, describes the chosen threat model, and summarizes the notation used in the paper. Section 4 describes in detail the proposed scheme. Section 5 compares it with relevant works in terms of storage overhead, transmission costs and performance. The paper ends with into possible future works.

2 Related work

A general study on fragmentation, encryption, and dispersal to protect data can be found in [9]. This section focuses on fragmentation schemes used inside UWSN to ensure data confidentiality and availability. In the descriptions, the following terminology is used: a sensor $s_i, i = 0, \ldots, s-1$, captures data during events named *rounds*, denoted as r. R denotes the number of rounds between consecutive visits of the mobile sink. d_i^r denotes the data of size $|d_i^r|$ bits captured by a single sensor S_i during the round event r. When a fragmentation scheme is applied, data d is fragmented into n fragments, k of which are needed for data reconstruction.

2.1 Data moving ([1, 2])

Attackers inside UWSN come in different flavors. A *curious* attacker will try to learn as much as possible about stored data. A *polluter* will try to mislead the sink by introducing fraudulent data. A *search-and-erase* or *search-and-replace* will destroy or modify certain target data. Finally, an *eraser* will erase as much data as possible.

The right defense strategy has to take into account the type of attacker. Three non-cryptographic strategies were introduced and analyzed in the context of search-and-erase and eraser attackers: DO-NOTHING (DN), MOVE-ONCE (MO), and KEEP-MOVING (KM). In DN, captured data are just waiting inside the sensor for the mobile sink. In MO, a sensor offloads newly captured data to some other randomly picked sensor right after the capture. In KM, data are moved continuously; each sensor moves each data item individually to another sensor at each round. Moreover, MO and KM may be combined with data replication in order to increase the probability of data survival. For an attacker of the search-and-replace type, the choice of strategy will depend on the frequency of sink visits (MO or KM). MO is more efficient than KM when $R < \frac{s}{k-1}$. When no replication is applied, DN is the best strategy against an eraser. However, data migration strategies become better than DN even with a single replica. More works on ensuring data survivability in UWSN using replication were done in [10, 11].

2.2 HybridS or RSSS ([5, 6])

HybridS is a scheme for secure and dependable storage inside UWSN, combining secret sharing with erasure coding. A sensor encrypts round data using a random key. In the next step, the Reed-Solomon scheme is used to encode the encrypted data into n fragments, k of which are needed for data reconstruction. The secret key is also fragmented into n fragments using a secret sharing scheme like Shamir's scheme presented in [12]. Finally, the sensor distributes the produced data and key fragments over n randomly selected neighbors. Such processing achieves lower storage and transmission costs than simple data replication. Indeed, the only overhead comes from the key fragments that are of size of the encryption key (typically 128 bits). When the round data is large, the HybridS achieves lower storage and transmission costs than fragmenting data using secret sharing. However, when the round data is very small, the overhead coming from key fragments may be significant. In [6], a similar scheme (RSSS) combining secret sharing with data encryption was presented. The main difference with HybridS is that it provides data integrity by including algebraic signatures within data fragments.

2.3 HKS ([7])

HKS (Homomorphic Key evolution Scheme) provides backward and forward secrecy of data stored inside the sensors by combining dynamic key generation with data aggregation. At each round, a sensor updates the encryption key K_i^r by hashing the key used during the previous round: $K_i^r = h(K_i^{r-1})$. Forward secrecy is provided as the previous key cannot be obtained from the current key. HKS encrypts data using the additively homomorphic encryption presented in [13], and thus significantly reduces the volume of data stored inside the sensors. Moreover, to decrypt the aggregated data, an attacker is obliged to have all the

dynamic keys that were used to encrypt the round data. Therefore, backward secrecy is also provided. The goal of HKS it to protect the network from curious intruders—those who want to read the stored data. It is inefficient against attackers who want to erase data as no data resilience is provided.

2.4 HEHSS ([3])

The homomorphic encryption and homomorphic secret sharing (HEHSS) scheme improves the HKS scheme by providing data reliability in addition to forward and backward secrecy. Round data are first fragmented into a set of n fragments using an additively homomorphic scheme (the Shamir's scheme [12]). Fragments are first encrypted using additively homomorphic encryption and then distributed over n sensor neighbors where they can be aggregated with fragments coming from previous rounds. Storage overhead and transmission costs in HEHSS are lower than in RSSS. This is mostly due to data aggregation. However, the use of secret sharing increases the size of the data fragments (Shamir's scheme produces fragments of size $|d_i^r|$, leading to an n-fold increase in data volume). Therefore, in HEHSS, the transmission costs during communication with neighbors are increased when compared to RSSS.

3 Problem formulation

This section presents assumptions about the considered sensor network architecture and the anticipated adversaries.

3.1 Network model

A network architecture scheme is presented in Fig. 1. In the considered scenario, the network is composed of s sensor nodes. Each sensor node has a fixed location and is denoted as S_i, where $i = 0, 2, \ldots, s-1$. Data captured by the sensors could be collected by one or more authorized mobile sinks visiting the network periodically. To simplify the description of the proposed approach, it is considered that only one mobile sink is present inside the network. We assume that a sensor captures data R times between consecutive visits of the sink (this is a simplified approach as sensors could have different intervals between rounds, as data capture could very well be event-triggered). The event of data capture by the sensor is denoted as a round r, where $r = 0,\ 1,\ \ldots,\ R-1$. Round data captured at a round r by a sensor S_i is denoted as d_i^r.

Sensors are limited in terms of computational power and memory. Apart from capturing and processing data, it is assumed that the sensors are able to communicate with at least n others sensors that are located in their neighborhood. It is a simplified approach as not all sensors may have n other nodes in their neighborhood. Two solutions may be adopted in a situation where there is a lack of neighbors. First, n can be variable and dependent on the sensor location (an example is shown in Fig. 1). The same solution could be applied in a situation of loss of neighbors due to interferences, obstacles, or issues of sensor mobility. Second, a multi-hop transmission may be used for sensors without enough neighbors. In the worst-case scenario, a sensor could function without neighbors for some period of time and then re-transmit collected data with the reappearance of the neighbors. In order to save energy, a classic sleep and live technique can be applied as sensors are alive only during and shortly after each round.

The mobile sink is assumed to be a trusted party that cannot be compromised. It is also assumed that the collected data are processed in an environment not limited in terms of computation power, memory constraints, or energy consumption. For instance, once collected, data could be sent to a cloud for further processing.

Static configuration is a simplified approach. A more sophisticated scheme would include a dynamic selection of neighbors. Such a dynamic network topology would be motivated by two factors. First, the sensors' nature may be mobile, leading their neighbors to continuously

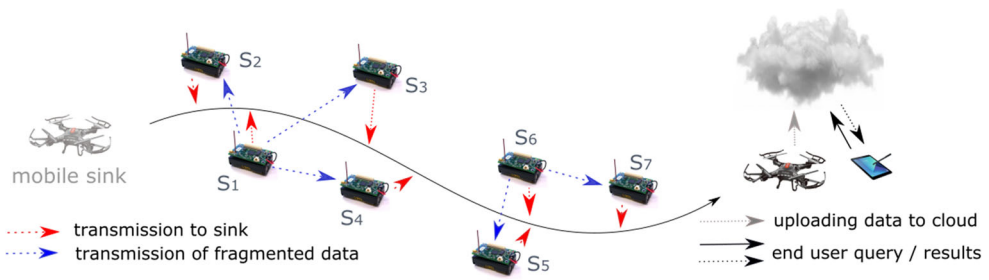


Fig. 1 Simplified scheme of the network architecture. Each sensor communicates with its neighbors (during each round, it is shown just for two sensors: S_1 and S_6) and with the mobile sink (during data collection). Depending on the network density, some sensors may have less neighbors than others; S_1 has three neighbors, but S_6 only two

change. Second, in a static configuration, attackers who know the topology of the network will first most likely try to compromise a set of sensors presumed to be neighbors. A mobile network makes the fragment distribution complicated, but at the same time increases the level of data protection. Indeed, moving sensors naturally add randomness to the fragment distribution, confusing the attacker about the location of the fragments belonging to same round data. Some works on data fragmentation using secret sharing inside mobile UWSN have already been done in [14] showing how parameters of the secret sharing should be chosen in function of the mobility degree of the network.

During the system's initialization, each sensor receives information about the location of its n neighbor's node (as at each round, the sensor will produce n data fragments and disperse them over its neighbors). Information about the network topology should be transmitted securely as it can provide hints to an attacker about the groups of sensors that will store fragments of the same round data.

3.2 Threat model

The threat model presumes that a roaming adversary is present inside the network and has the ability to compromise most $k-1$ sensors during consecutive visits of a mobile sink. The setting of k and R must be carefully chosen during the network's dimensioning while also considering an estimation of the system's vulnerability. Once the attacker compromises a sensor, they are able to fully control it and consequently obtain the data collected during their occupation of the sensor. However, unless they manage to compromise a set of k sensors storing fragments of the same round data, they are not able to obtain the data collected before the compromise. There is no way for a sensor to distinguish if its neighbor is currently under attack or not.

An attacker is assumed to not be able to compromise k sensors between consecutive visits of the sink. However, an additional countermeasure may be established. It would consist of encrypting the first round data fragments stored at the sensor with a temporary key distributed during the system initialization and known to the end user. This temporary key would be deleted from the sensor's memory after the first round. This way, an attacker would only dispose of a short interval of time at the beginning of the data collection to compromise k sensors and obtain their temporary keys.

The attacker is assumed to be curious: they want to read the stored data and are less interested in erasing or modifying them. Therefore, the AHEF scheme focuses mainly on providing data confidentiality and availability (data resilience has to be ensured as data could be lost due to some accident) and neither treats the problem of fragments authentication, integrity verification, or of an attacker strongly focused on data erasure. However, AHEF could be complemented with a different scheme, for instance the one presented in [15] for data authentication or integrity.

4 The proposed scheme

The goal of the scheme is to protect the data stored inside the sensors until the arrival of the mobile sink while minimizing storage overhead, the complexity of processing, as well as the transmission costs.

4.1 System initialization

During system initialization, a secure hash function (for instance SHA-512) denoted as $h(.)$ is chosen and preloaded to the sensors along with a symmetric cipher algorithm (for instance AES). In accordance with Kerckhoffs's principle, the hash and the cipher function are publicly known. Moreover, each sensor S_i receives its own initial key denoted as K_i^0 (key distribution protocol may be based on one of the several solutions proposed in [16]). The initial key K_i^0 is refreshed after each visit of the mobile sink by being exclusive-ored with a nonce produced using a secure cryptographic deterministic pseudorandom generator (DPRG) [17]. The seed of the employed DPRG can be constructed by hashing the secret key.

4.1.1 Fragments distribution and aggregation

Two configurations of fragment distribution and aggregation are possible. In the first configuration, only data coming from the same sensor is aggregated. At each round, a sensor captures round data, fragments it into a set of n fragments, and disperses the fragments over its neighbors where they are aggregated with fragments distributed during previous rounds. In the second configuration, sensors are organized into clusters of n nodes. At each round, each sensor from the cluster sends their data to the neighbors (that are also belonging to the cluster). Then, round data coming from different nodes of the cluster are aggregated.

4.1.2 Key evolution

At the beginning of each round, the round index r and the sensor's round key K_i^r are updated. The current round key is obtained by hashing the previous round key: $K_i^r = h\left(K_i^{r-1}\right)$, where $r = 0, \ldots, R-1$ and K_i^0 is the initial key. Thanks to the one-way property of the hash function, attackers who compromise a sensor and obtain its current round key will not be able to derive the previously used

round keys. Thus, *forward security* is provided. The mobile sink stores the initial key K_i of each sensor, allowing it to derive the round keys as needed.

4.2 Processing round data

At each round, data captured by a sensor are processed using the AHEF scheme composed of two steps: data fragmentation and data encryption.

4.2.1 Step 1—additively homomorphic fragmentation of the round data

In the first step, round data is fragmented into n fragments, k of which are needed for data reconstruction. Instead of the additively homomorphic secret sharing that was applied in HEHSS, an additively homomorphic dispersal algorithm—similar to the one presented in [18]—is used. More precisely, the first step does the following operations:

1. Represent collected round data d_i^r as an integer $d_i^r \in [0, \ldots, D-1]$ of size $|d_i^r|$ bits.
2. Split round data into k data parts $d_i^r(j)$, where $j = 0, \ldots, k-1$ and $d_i^r(j) \in [0, \ldots, D_{part}-1]$. The size of each data part is of $\frac{|d_i^r|}{k}$ bits. This can be done in a straightforward way by cutting data into chunks of $\frac{|d_i^r|}{k}$ bits.
3. Choose an integer M in function of the predicted number of rounds R, aggregation configuration (aggregate only data from one sensor or data from a cluster of sensors), and the value of D_{part}: $M = 2^{\lceil \log_2(pR) \rceil}$ when the aggregation is performed only on data coming from the same sensor or $M = 2^{\lceil \log_2(pRn) \rceil}$ when the data will be aggregated inside a cluster of n sensors. p is the maximum data value: $p = \max(d_i^r(j))$.

   The proposed scheme uses the additively homomorphic encryption scheme (AHE) presented in [13]. In AHE, encryption/decryption operations consist in addition/subtraction modulo a big integer M. The value of M has to be not only larger than the size of the data to be encrypted but also has to take into account the number of rounds R and the number of fragments that will be aggregated together at each round in order to prevent the overflow coming from the addition of multiple fragments. Indeed, with each round, the value of the sum of the round data increases. Therefore, data fragments have to be large enough to contain the sum of all fragmented round data.
4. Construct an encoding polynomial $y_i^r(x) = \sum_{j=0}^{k-1} d_i^r(j) x^j \pmod{M}$. The k data parts are used as the coefficients of the encoding polynomial.
5. Evaluate $y_i^r(x)$ at n different evaluation points x_m ($x_m > 0$ and $m = 0, \ldots, n-1$) to obtain n different data points $p_{i,r}^m = \left(x_m, y_i^r(x_m)\right) = (x_m, y_m)$ that are the result of the fragmentation of round data d_i^r. Data parts (and in consequence the round data) can be obtained by interpolating the polynomial using any k of the n points.

4.2.2 Step 2—additively homomorphic encryption

The second step encrypts each of the data points obtained after the first step using additively homomorphic encryption:

1. Generate n pseudorandom keystreams $t_{i,r}^m$ ($m = 0, \ldots, n-1$ and $t_{i,r}^m \in [0, M-1]$) using a stream cipher and the current round key K_i^r.
2. Transform a point $p_{i,r}^m$ into the final fragment $f_{i,r}^m$ by encrypting the y-axis values of the point. The encryption process entails the modular addition of the y-axis value of the point and of its corresponding keystream:
   $c_m = Enc(y_m, t_{i,r}^m, M) = y_m + t_{i,r}^m \pmod{M}$
   $f_{i,r}^m = (x_m, c_m)$

4.3 Fragments aggregation

Let $F_a = \text{AHEF}(d_i^r) = f_{i,r}^a = (x_a, c_a) | a = \{0, \ldots, n-1\}$ and $F_b = \text{AHEF}(d_{i'}^{r'}) = f_{i',r'}^b = (x_b, c_b) | b = \{0, \ldots, n-1\}$, be two sets of n fragments obtained during the fragmentation of two data sets obtained during different round (r and r') and/or inside different sensors (s_i and s_i'). Then, the sum of both sets when $x_a = x_b$ is equal to a result of fragmentation of the sum of $d_i^r + d_{i'}^{r'}$:
$F_{a+b} = (F_a + F_b) = (x_a + x_b, (c_a + c_b) \pmod{M}) = \text{AHEF}(d_i^r + d_{i'}^{r'})$.
This is possible because of the additively homomorphic properties of both, fragmentation and encryption. Additively homomorphic fragmentation allows the addition of multiple points of different polynomials evaluated within the same set of evaluation points. Furthermore, additively homomorphic encryption allows the addition of multiple encrypted points even if they were encrypted using different keys.

4.4 Data reconstruction

Data reconstruction is performed after the collection of the data by the mobile sink. It does the following operations:

1. Gather any k of n fragments $f_{i,r}^l, l = 0, \ldots, k-1$
2. Generate k keystreams $t_{i,r}^l$, using the same K_i^r as during encryption, corresponding to the k possessed fragments
3. Decrypt fragments to obtain l points. The decryption consists of decrypting the y-axis values of the fragments

Table 1 Quantitative analysis of relevant schemes

Scheme	Storage overhead	Transmission costs	Resilience	Agg.
Data M.	$\sum_{r=1}^{r_{\max}} \lvert D_i^r \rvert$	DN: $\sum_{r=1}^{r_{\max}} \lvert D_i^r \rvert$	No	No
Di Pietro et al. [1]		MO: $2\sum_{r=1}^{r_{\max}} \lvert D_i^r \rvert$		
Di Pietro et al. [2]		KM: $\sum_{j=1}^{r_{\max}} \sum_{r=1}^{j} \lvert D_i^r \rvert$		
RSSS	$\frac{n}{k} \sum_{r=1}^{r_{\max}} \lvert D_i^r \rvert$	$\frac{(n+k)}{k} \sum_{r=1}^{r_{\max}} \lvert D_i^r \rvert$	Yes	No
Ren et al. [5]				
Wang et al. [6]				
RSSS*	$\frac{n}{k} \sum_{r=1}^{r_{\max}} (\lvert D_i^r \rvert + k\lvert \text{key} \rvert)$	$\frac{(n+k)}{k} \sum_{r=1}^{r_{\max}} (\lvert D_i^r \rvert + k\lvert K \rvert)$	Yes	No
HKS	$\lvert D_i^r \rvert + \log_2(r_{\max})$	$\lvert D_i^r \rvert + \log_2(r_{\max})$	No	Yes
Ren et al. [7]				
HEHSS[1]	$n(\lvert D_i^r \rvert + log_2(r_{\max}))$	$(n+k)(\lvert D_i^r \rvert + \log_2(R))$	Yes	Yes
Ren et al. [3]				
HEHSS[2]	$(\lvert D_i^r \rvert + \log_2(r_{\max} n))$	$(n+k)(\lvert D_i^r \rvert + \log_2(r_{\max} n))$	Yes	Yes
AHEF[1]	$n(\frac{\lvert D_i^r \rvert}{k} + \log_2(r_{\max}))$	$n(\frac{\lvert D_i^r \rvert}{k} + \log_2(r_{\max}))$	Yes	Yes
AHEF[2]	$\frac{1}{k}(\lvert D_i^r \rvert + \log_2(r_{\max} n))$	$(n+k)(\frac{\lvert D_i^r \rvert}{k} + \log_2(r_{max} n))$	Yes	Yes

$RSSS^*$ A version of RSSS where the key is fragmented and transmitted among the data; $HEHSS^1$, $AHEF^1$ aggregation is performed only on fragments coming from the same sensor; $HEHSS^2$, $AHEF^2$ sensors are organized into clusters of n nodes and corresponding fragments from the n sensors are being aggregated

$$y_l = c_l - t_{i,r}^l \pmod M$$
$$p_{i,r}^l = \text{Dec}(f_{i,r}^l) = (x_l, y_l)$$

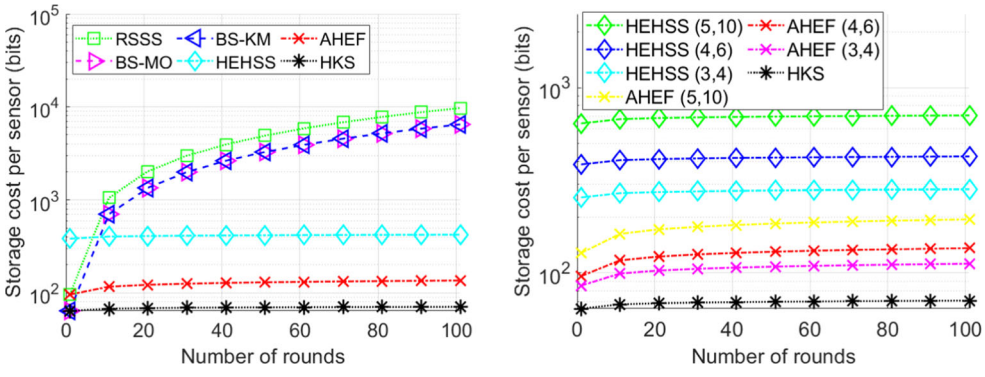
4. Interpolate the polynomial y_i^r from the reconstructed points to obtain the k coefficients of the polynomial that correspond to the round data

4.5 Decryption of aggregated fragments

The decryption of a sum of fragments consists in multiple subtractions of all the key streams used during the encryption of individual fragments. Decrypted points may then be used for the recovery of the sum of the coefficients of polynomials used to obtain the points. This is only possible when fragments with the same x-axis values are added.

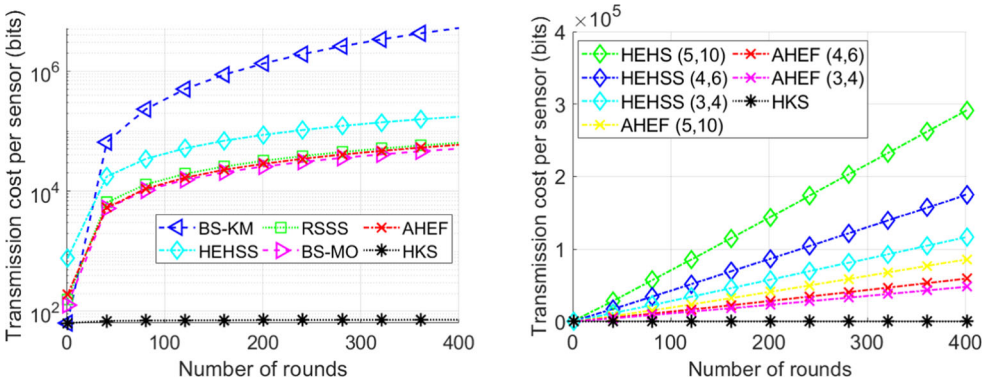
4.6 Fragments aggregation between rounds and sensors

The proposed scheme allows the aggregation of data fragments not only between different rounds but also between different sensors. The only rule is that two fragments may be aggregated only if they were obtained using the same evaluation points (x-axis values). In the proposed simplified architecture, it is presumed that, during the static configuration of the network, each sensor receives information about its



(a) *SO - fragmentation techniques.* (b) *SO - homomorphic techniques.*

Fig. 2 Storage cost per sensor. Data aggregation allows a significant reduction in storage cost (**a**). HKS does not provide data resilience and comes with limited data protection. AHEF achieves a lower storage cost per sensor (factor $\approx \frac{1}{k}$) than HEHSS, since it replaces secret sharing with information dispersal (**b**)



(a) *TC - fragmentation techniques.* (b) *TC - homomorphic techniques.*

Fig. 3 Transmission costs (TC) per sensor. Fragmenting data instead of replicating and applying error-correction codes helps limit the transmission costs while providing data resilience (**a**). AHEF significantly reduces TC in comparison to HEHSS as its data fragments are k-times smaller than in HEHSS. HKS does distribute data to its neighbors (and thus does not provide data resilience), so the only TC is the transmission to the sink (**b**)

n neighbors. It will then always transmit fragments with same x-axis values to the same neighbor (so fragments from different round can be aggregated). This is the same principle for the rest of the sensors. In order to enable the aggregation of fragments coming from different sensors, information about the evaluation points to be used during fragmentation and a map associating sensors with these points should be given during the initialization phase.

5 Comparison with relevant works

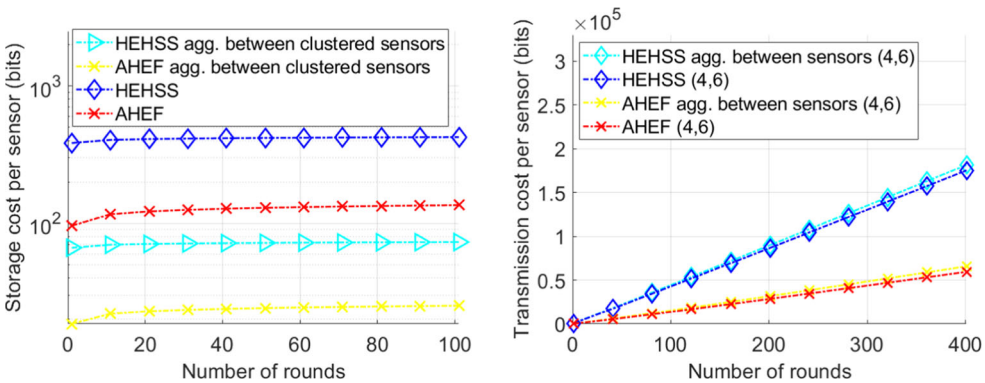
AHEF was analyzed in terms of storage overhead and transmission costs as well as in terms of performance of data processing. The results were compared with relevant works.

5.1 Storage overhead

A comparison of storage overhead (SO) and transmission costs (TC) of relevant schemes was done. Table 1 presents a summary of the quantitative analysis. To better illustrate the difference between algorithms, simulations of SO and TC are presented in Figs. 2 and 3. During simulations, a data sample of 1000 bits was created at each round. It was supposed that the end user is interested by the average value of the captured data as well as by their variance.

Intuitively, for all schemes, the storage overhead depends on the size of round data $|d_i^r|$ and the number of rounds R. As presented in Fig. 2, data aggregation allows a significant saving of SO. Indeed, for basic scheme and RSSS at each round, the SO increases by the size of round data $|d_i^r|$. HKS, HEHSS, and AHEF allow data aggregation. For each of these schemes, at each round, the storage overhead increases only by the value coming from the bits added to avoid overflow during aggregation of data fragments. HEHSS applies secret sharing in addition to encryption which, in contrast to encryption increases the size of the data (the size of each data fragment is equal to the size of the round data, so it leads to a n-fold increase of SO during the first round). AHEF deals with this problem by replacing Shamir's scheme with Krawczyk's information dispersal.

As shown in Fig. 4a, for additively homomorphic fragmentation schemes (HEHSS and AHEF) the storage overhead can be reduced by aggregating fragments from multiple sensors.



(a) *Storage overhead.* (b) *Transmission costs.*

Fig. 4 Comparison of two aggregation variants: (1) aggregation of fragments only coming from a single sensor and (2) aggregation of data coming from a single sensor and its neighbors. This demonstrates that aggregating fragments from multiple sensors leads to an increase in transmission costs but a decrease of storage costs

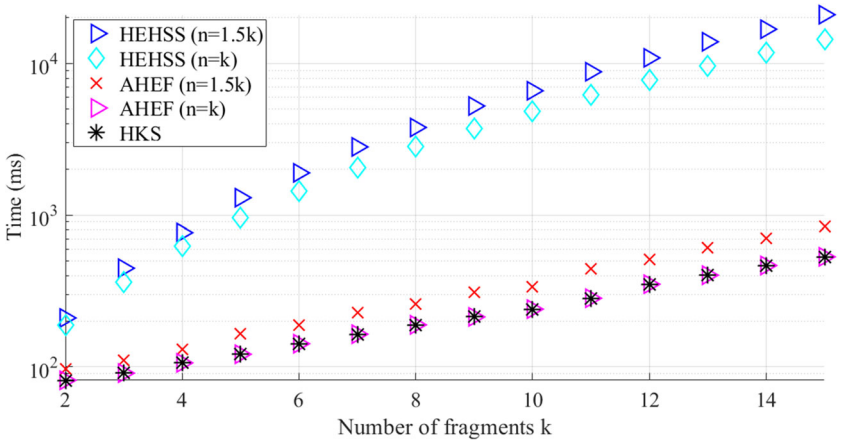


Fig. 5 Performance benchmarks. AHEF achieves a better performance than HEHSS and RSSS. It comes with a slightly higher performance overhead than HKS as it adds fragmentation to the encryption. HKS does not provide data resilience which is why it was not included in the comparison for $n = 1.5k$

5.2 Transmission costs

Transmission costs are expressed as the sum of bits transmitted from the sensor to its neighbors (during fragmentation) and of bits of stored data transmitted to the sink during data collection. As presented in Fig. 3, TC for the basic scheme in the KM mode are much higher than for the rest of the schemes (for which the increase is linear in function of the number of rounds). When $k = n$, RSSS and basic scheme in the mode MO have equal TC. Schemes allowing data aggregation comes with a significantly lower TC. HKS has the lowest TC; it does not spread data over the sensor's node since its only TC is the cost of uploading data to the sink. For AHEF, the cost is twofold: in addition to the sink's transmission, at each round, data are diffused over neighbors. Fragment size in HEHSS is k times larger than in AHEF (due to the use of Shamir's scheme). Therefore, at each round, its TC are k times larger than these of AHEF.

In contrast to the storage overhead, transmission costs slightly increase when fragments from multiple sensors are aggregated (see Fig. 4b).

5.3 Performance evaluation

Implementation details Relevant algorithms were implemented in JAVA using the following resources: JAVA 1.8 and Matlab 2011b on DELL Latitude E6540, X64-based PC running on Intel® Core™ i7-4800MQ CPU @ 2.70 GHz with 8 GB RAM, under Windows 7. AES-CTR with 128-bits key was used to generate the pseudorandom key streams and SHA256 as the hash function.

Benchmark results Execution time for data processing for $R = 10000$ rounds was measured for each algorithm in two configurations: when all the fragments are required for data recovery ($n = k$) and when redundant fragments are generated ($n = 1.5k$). The time between consecutive rounds was not taken into account, as it is the time when a sensor rests in sleep mode. Results are shown in Fig. 5. The proposed scheme (AHEF) achieves a better performance than HEHSS and RSSS. The performance's gain increases with the number of fragments. The clear lack of scalability of HEHSS and RSSS is mainly caused by the use of Shamir's secret sharing. In contrast to HKS, AHEF provides resilient data and generates interdependent fragments. Thus, it comes with a slightly larger performance overhead.

6 Conclusion

A novel scheme for data protection in unattended wireless sensor networks was introduced and analyzed. In order to provide a backward and forward secrecy, it combines a dynamic key evolution with the fragmentation of encrypted data. A combination of additively homomorphic fragmentation and encryption allows a secure aggregation of data fragments. In contrast to similar schemes, the proposed approach achieves lower storage overhead and transmission costs as well as better performance. Moreover, it is scalable and therefore can be integrated within networks composed of a multitude of nodes.

Our future work will focus on both the design and the evaluation of more complex IoT architectures that support a dynamic configuration including an energy-aware selection of the sensor's neighbors. Neighbors may be dynamically changed after each round or each sink's visit contributing in misleading an adversary. A multi-hop transmission of data will be considered to take into account the scarcity of neighbors and to broaden the choice of nodes able to receive fragments from a given sensor (where, for instance, some of the nodes would play the role of fragments aggregators). Such a sophisticated architecture will have to carefully set the right trade-off between the desired level of data protection and the energy savings of the sensor's batteries. Our team already started working in this direction [19].

Lastly another research track could be to integrate the AHEF inside the group-based security scheme proposed in [20].

Acknowledgments The authors would like to thank the reviewers of this work as well as professor Roberto Di Pietro for his remarks and encouragements.

References

1. Di Pietro R, Mancini LV, Soriene C, Spognardi A, Tsudik G (2008) In: Sixth annual IEEE international conference on pervasive computing and communications (PerCom 2008), 17-21 March 2008, Hong Kong, pp 185–194. https://doi.org/10.1109/PERCOM.2008.31
2. Di Pietro R, Mancini LV, Soriente C, Spognardi A, Tsudik G (2009) IEEE Trans Comput 58(11):1500. https://doi.org/10.1109/TC.2009.109
3. Ren Y, Oleshchuk V, Li FY (2009) In: 2009 2nd IFIP Wireless Days (WD), pp 1–6. https://doi.org/10.1109/WD.2009.5449662
4. Kapusta K, Memmi G, Noura H (2017) In: 1st cyber security in networking conference, CSNet 2017, Rio de Janeiro, Brazil, October 18-20, 2017, pp 1–8. https://doi.org/10.1109/CSNET.2017.8241998
5. Ren W, Ren Y, Zhang H (2008) In: 2008 IEEE/IFIP international conference on embedded and ubiquitous computing, vol 2, pp 318–323. https://doi.org/10.1109/EUC.2008.76
6. Wang Q, Ren K, Lou W, Zhang Y (2009). In: IEEE INFOCOM 2009, pp 954–962. https://doi.org/10.1109/INFCOM.2009.5062006
7. Ren Y, Oleshchuk V, Li FY (2009) In: 2009 3rd international conference on new technologies, mobility and security, pp 1–5. https://doi.org/10.1109/NTMS.2009.5384753
8. Rabin MO (1989) J. ACM 36(2):335. https://doi.org/10.1145/62044.62050
9. Memmi G, Kapusta K, Qiu H (2015) In: 2015 IEEE international conference on cyber security of smart cities, industrial control system and communications (SSIC) 2015, Shanghai, China, pp 1–9. https://doi.org/10.1109/SSIC.2015.7245680
10. Aliberti G, Di Pietro R, Guarino S, Netw J (2017) Comput Appl 99(C):146. https://doi.org/10.1016/j.jnca.2017.09.008
11. Di Pietro R, Verde NV (2011) In: Proceedings of the Fourth ACM conference on wireless network security, ACM, New York, WiSec '11, pp 11–22. https://doi.org/10.1145/1998412.1998417
12. Shamir A (1979) Commun ACM 22(11):612. https://doi.org/10.1145/359168.359176
13. Castelluccia C, Mykletun E, Tsudik G (2005) In: The Second annual international conference on mobile and ubiquitous systems: networking and services, pp 109–117. https://doi.org/10.1109/MOBIQUITOUS.2005.25
14. Di Pietro R, Guarino S (2013) In: 2013 proceedings IEEE INFOCOM, pp 205–209. https://doi.org/10.1109/INFCOM.2013.6566764
15. Bohli J, Papadimitratos P, Verardi D, Westhoff D (2011) In: 2011 IEEE 36th conference on local computer networks, pp 994–1002. https://doi.org/10.1109/LCN.2011.6114411
16. Gonçalves DdO, Costa DG (2015) J Imaging 1(1):4. https://doi.org/10.3390/jimaging1010004. http://www.mdpi.com/2313-433X/1/1/4
17. Barker EB, Kelsey JM (2011) Recommendation for random number generation using deterministic random bit generators (revised), US Department of Commerce, Technology Administration, National Institute of Standards and Technology, Computer Security Division, Information Technology Laboratory
18. Krawczyk H (1994) In: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology, Springer, London, CRYPTO '93, pp 136–146. http://dl.acm.org/citation.cfm?id=646758.705700
19. Choi HB, Ko YB, Lim KW (2018) In: Proceedings of the Third international conference on security of smart cities, industrial control system and communications, SSIC '18
20. Hamid MA, Sarkar AMJ (2012) Annales des Télécommunications 67(9-10):455. https://doi.org/10.1007/s12243-011-0278-3