Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Lehrstuhl für Rechnernetze und Telematik

WS 2009

**Seminar**

# Efficient Aggregation of encrypted data in Wireless Sensor Networks

**Claude Castellucia, Einar Mykletun, Gene Tsudnik**

Refik Hadzialic

January 30, 2010

Ad-Hoc Networks Seminar for Prof. Dr. Christian Schindelhauer

## Abstract

In a paper written by Casteluccia C., Mykletun E. and Tsednik G., an efficient approach is presented for utilizing the aggregation of data in a Wireless Sensor Network and the assuring of end-to-end encryption of data between the leaves and sink. One of the goals of the paper was to minimize the bit transmission between the sensor nodes and therefore to find an efficient encryption algorithm which is simple to implement and in turn would prolong the life of batteries.

## Contents

**1 Introduction 3**

**2 Wireless Sensor Networks 3**
2.1 WSN Requirements . . . 3
2.2 Problems and Issues . . . 3
2.3 Data aggregation as a solution . . . 3

**3 Encryption of data 4**
3.1 End-to-end encryption . . . 4
3.2 Suggested encryption algorithm . . . 4
3.2.1 Homomorphic property . . . 5
3.3 Concept idea of the algorithm . . . 5
3.3.1 Decryption . . . 6
3.4 Pseudocode . . . 6

**4 Calculations and bit-length of transmitted data 6**
4.1 Average and Variance . . . 6
4.2 Transmitted bit-length analysis . . . 7

**5 Results and analysis 7**
5.1 Theoretical analysis . . . 7
5.2 Results . . . 8
5.2.1 Bit-length transmission results . . . 8
5.2.2 Bandwidth gain results . . . 8

**6 Ideas how to improve this solution 9**

**7 Conclusion 9**

# 1 Introduction

Wireless Sensor Networks (WSN), by the definition of Holger K. and Willig A. [5], are devices that integrate simple processing power (CPU), memory (storage), and sensing and communication capabilities into a low cost device. WSNs work on the principle of Ad-Hoc networks; each node is a transceiver (it can receive and transmit data.) There are tremendous number of applications of WSN. Some are “Disaster relief applications” [5] where the employed WSN is equipped with thermal sensors measuring the average temperature e.g. in a forest, automatically alarming the fire department if the temperatures get too high. Another field where WSNs could be applied is in military applications, they could measure some important information for the army. In the second case it is obvious that a certain level of security is required; therefore there is a need for encryption algorithms. The need for encryption is always present, especially after the recent incident in which US Army surveillance airplanes, Predator MQ-1’s, were not using any encryption algorithms for their surveillance video data [2]. The videos and images of territories under surveillance were monitored by Iraqi militants as well, by just using a shareware windows application, satellite card receiver and a satellite parabola dish [2].

In the following sections first the WSN requirements will be explained, then the encryption method will be discussed, followed by explaining the suggested aggregation method, variance and average calculation, and then the results of the tested model of the WSN. The author of this seminar will introduce a few ideas on how battery consumption could be reduced even more and how the results of this paper could be outperformed.

# 2 Wireless Sensor Networks

## 2.1 WSN Requirements

WSNs have different requirements depending on their application. Usually WSNs are battery powered applications, so one should have in mind that lifetime of a WSN is important and thus the algorithms, which run on them, should be optimized and tested. These algorithms run on tiny Microcontrolling devices (MCU) which are limited in processing and storage space. Since the goal of a WSN is to make a collective conclusion [3], it is important that all sensor nodes work properly and that their lifetime is almost identical for most of them.

## 2.2 Problems and Issues

A main issue with all wireless devices is their battery power consumption; the more data are being transmitted, the larger the battery consumption is [3]. One way to attack this problem is to reduce the bit transmission. Bit transmission can be reduced by aggregating sensor data [3]. This approach to the problem cannot be applied in all WSNs, but it can be applied where the average, variance, max or min temperature, humidity or some other sensing property is of vital improtance for the WSN.

## 2.3 Data aggregation as a solution

Aggregating data is a way of compressing the transmitted packet, in a sense that the packet is comprised of only necessary information [3]. Aggregation of data was first introduced in Digital Signal Processing (DSP) applications [6]. There was a need to have an optimal calculation of the average/mean value of all the samples in real time because the DSP Processors did not have enough space in the fast static RAM (SRAM) to store everything. All the sample values of $X$ could not be stored in the SRAM, instead, by summing them (sample values $Xn$) all up and keeping in mind the number of taken samples ($n$), it was easy to calculate the average [6].

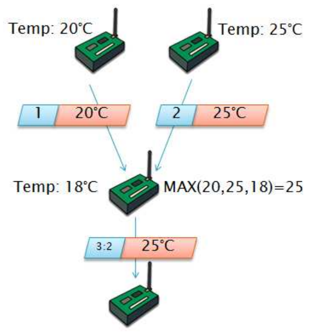

Figure 1: An example of aggregating data. If the node in the middle has access to data, it can choose by itself the maximal temperature value from the three given values and send only the maximal value with the ID of the node whose temperature it is.

Aggregation does not work for every application, i.e. where single reading samples are required e.g. perimeters control [3]. On figure 1, a simple example of data aggregation can be seen (determining the maximal temperature)[1].

## 3 Encryption of data

The second point in the research paper of Casteluccia C., Mykletun E. and Tsednik G. was encryption of data. Encrypting data is a way of encapsulating the information and protecting it from the outside world, in that sense that nobody should be able to know what information is inside the packet beside the device/person who should receive it. The authors goal was to achieve an end-to-end encryption between the nodes and the sink[2].

### 3.1 End-to-end encryption

In end-to-end encryption no node should be capable of knowing or being able to extract the information from the received packets beside the sink. Using this approach it is possible to guarantee that it will be more difficult for an eavesdropper[3] to gain access to the data. Another way of addressing the encryption problem would be to use a global encryption key or only keys between neighboring nodes, but in that case the end-to-end encryption is lost. For the first approach, having one global key, an eavesdropper could gain access to all information by just hacking one node and determining the global key.

### 3.2 Suggested encryption algorithm

As mentioned in the introduction, encryption plays a major role in all today's communication systems, especially in military or WSNs applications. In WSNs some sensors are expensive (e.g. sensors for measuring radioactivity) or the sensed data should be protected because of various rival companies who should not gain access to the data [3]. If every single node could have its own

[1]The node images are taken from Prof. Schindelhauers slides

[2]Sink is usually the last node in the WSN tree which collects all the data; and in most of the cases, it is more powerful than the rest of the nodes

[3]An eavesdropper is a passive attacker in the middle, who is only listening to the data being transferred between the nodes

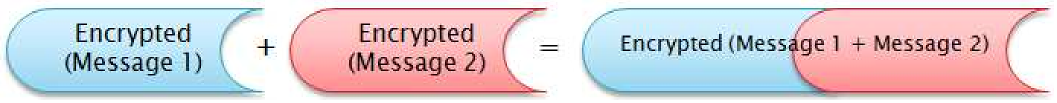

Figure 2: Homomorphic property, $\xi(X+Y) = \xi(X)+\xi(Y)$, where $\xi(X)$ denotes encryption of some value $X$

key that would mean only the sink could gain access to all of the data[4]. Even the eavesdropper, who might know one secret key of node, is not capable to read all the data from the rest of the nodes. In the suggested algorithm by Casteluccia C., Mykletun E. and Tsednik G. this property will be used (each node will have a unique secret key) [3].

### 3.2.1 Homomorphic property

The basic idea of homomorphic encryption algorithms is the power to operate on data while they are still encrypted, i.e. a rented super computer could calculate scientific data without knowing what they are from inside and still produce a perfectly valid result. For instance, "a sum of numbers can be computed without revealing the single numbers" [4]. This property will later be used for the aggregation of statistical data. The basic idea can be seen in figure 2.

Assume $Enc()$ is the function for encrypting some message $M$, and it produces a ciphertext $C$ (encoded message $M$) using a secret key $K$. Let $c_1$ denote the first encrypted message $m_1$ with the secret $key_1$, i.e. $c_1 = Enc_{k1}(m_1)$ and let $c_2$ denote the second encrypted message $m_2$ with the $key_2$, i.e. $c_2 = Enc_{k2}(m_2)$ [3]. Then, there exists such an secret key $k$ where $c_1 \cdot c_2 = Enc_k(m_1 + m_2)$ holds [3], i.e. if $Enc()$ would be a 2-tuple function, the result would be the same as $Enc(m_1 + m_2, k_1 + k_2)$. This property is based on the discrete logarithm scheme [4]. The same property is used in the RSA encryption algorithm, just in RSA the algorithm is not additively homomorphic, but rather multiplicatively. We will use this property to sum up encrypted data and then the sink will decrypt the message and get summed up values which later on will be used to calculate the average, variance, etc.

## 3.3 Concept idea of the algorithm

The suggested additively homomorphic algorithm consists of four parts:

- Choosing a large enough number
- Secret key generation
- Encryption
- Decryption

The explanation will not proceed in a descending order of the mentioned steps. First the concept ideas will be presented and then the pseudocode will be given.

The encryption function is a 3-tuple function and works as follows $c = Enc(m, k, M) = m + k(modM)$, where $m$ is the message we want to encrypt, $k$ is the secret key for the node and $M$ is the large number.

The question that should be raised at this point is, "how large does $M$ have to be". A large number will waste too many spare bits and reduce the battery life of the device, and a small number will make the mechanism fail. By looking at how the encryption function works, one can easy conclude why the mechanism will fail. If the modulo number $M$ would be smaller than the sum of all sample values and encryption keys, the sink could not reproduce the real sum but would produce a smaller number than $M$, in other words a number which was cut off by the

[4]Let us assume that only the sink knows all private keys from all the nodes.

modulo number $M$ (overflow.) Therefore, $M$ should be carefully selected. The paper's Authors suggest selecting $M$ by using the following formula, $M = 2^{\lceil log_2(p*n) \rceil}$, where $n$ is the number of nodes and $p = max(m_i)$, where $m_i$ is the size of the message, e.g. if 7 bit unsigned values are used for the messages, then $p = 128$. By choosing $M$ using this approach, the assumption by the authors $0 \leq m_i < M$ is satisfied[5]. The reason why $m_i$ should be greater or equal to zero is because otherwise the additively homomorphic property would not work properly. Instead of using exclusive-OR for the stream cipher, the authors of the paper decided to replace it with modular addition [3].

### 3.3.1 Decryption

Note that decryption is the inverse process of encryption. Let $Dec()$ be a 3-tuple decryption function. Let the input to the decrypt $Dec()$ function be $c$, the ciphertext; $k$, the secret key and $M$, the large number we already have defined. The $Dec()$ function works as follows: $Dec(c, k, M) = c - k(modM)$. The resulting value of the function is the originally encrypted value. Now assume two different values $m_1$ and $m_2$ are encrypted with two different secret keys, $k_1$ and $k_2$, let $c_1$ and $c_2$ be the resulting ciphertexts; $c_1 = Enc(m_1, k_1, M)$ and $c_2 = Enc(m_2, k_2, M)$. Now if the ciphertexts were summed up, $C = c_1 + c_2$, and of course the sink knows the keys and sums them up as well, $K = k_1 + k_2$, then by applying $Dec(C, K, M) = m_1 + m_2$, we get summed up the original messages/values. After the concept is understood, the pseudocode will be given.

## 3.4 Pseudocode

Encryption

1. Choose M large enough, i.e. $M = n \cdot t$, where $n$ is the number of nodes and $t$ is the maximal value which may appear in the measurements, $t = max(m_i)$
2. Let the messages $m_i$ be represented as an integer number, $m_i \epsilon [0, M - 1]$
3. Choose $k_i$ to be a random key-stream, where $k_i \epsilon [0, M - 1]$
4. Apply the encryption function to the values, $c_i = Enc(m_i, k_i, M) = m_i + k_i(modM)$

Decryption

1. Use the decryption function on the sink, $Dec(C, K, M) = C - K(modM)$, where $C = \sum_{i=0}^{n-1} c_i$ and $K = \sum_{i=0}^{n-1} k_i$[6]

# 4 Calculations and bit-length of transmitted data

Calculating the average and variance is straightforward. The analysis of the bit-length will follow after understanding the calculation of average and variance.

## 4.1 Average and Variance

Only summed up samples have to be sent, as well as their square summation as well[7].

$S = \sum_{i=0}^{n-1} m_i$ and $V = \sum_{i=0}^{n-1} m_i^2$

Average is calculated by dividing $S$ with $n$, where $n$ is the number of nodes in the WSN without the sink. Variance is similarly calculated, $Var = E(m^2) - E(m)^2$ , where $E(m^2)$ and $E(m)^2$ one can easily obtain using the following two formulas: $E(m^2) = (\sum_{i=0}^{n-1} m_i^2)/n$ and $E(m) = (\sum_{i=0}^{n-1} m_i)/n$. The second formula is the average actually. By looking carefully at the

[5]By using unsigned numbers, $m_i$ cannot be negative and $M$ is always larger than the sum of all messages.

[6]This part of the decryption applies if data aggregation was used

[7]This is required for the variance.

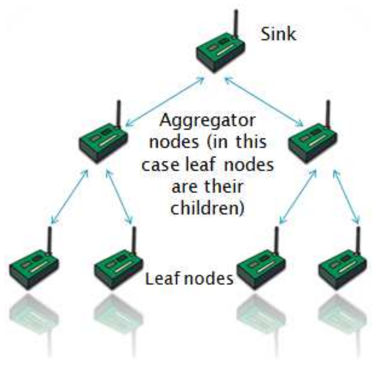
Sink
Aggregator nodes (in this case leaf nodes are their children)
Leaf nodes

Figure 3: A 2-ary or binary WSN tree, similar to theone used for the performed tests

two formulas, it is easy to see that all that has to be sent to the next node is the sum of $m_i$ and the sum of its squares, $m_i^2$ [3].

## 4.2 Transmitted bit-length analysis

The transmitted bit-length of the ciphertext used to calculate the average, $c_i$, will be $log(M)$ bits long. Let $M_A$ be the large modulo number for the average calculation and let $M_V$ be the large modulo number for variance calculation. Knowing that $M_A = n \cdot t$, we can easily get $log(M_A) = log(n) + log(t)$ to be the bit-length of the transmitted ciphertext. Similar calculation has to be done to get the bit-length for the ciphertext of the variance calculation. As this time, the squared sample values have to be summed up, so we must define $M_V$ in a different way. Let $M_V = n \cdot t^2$, where $n$ is the number of nodes and $t$ the maximal sample/measurement value. It is squared because we sum up squared values of $m_i$. By doing the same procedure as above, we obtain the following bit-length for the variance ciphertext: $log(M_V) = log(n)+log(t)^2 = log(n)+2\cdot log(t)$. If we sum up $M_A$ and $M_V$ we get, $M_A+M_V = log(n)+log(t)+log(n)+2\cdot log(t) = 2\cdot log(n)+3\cdot log(t)$ to be the bit-length of the data part of the sent packet[8].

# 5 Results and analysis

The efficiency of this method can be proven by comparing it with two other methods which the authors tried themselves. The tests were performed on a *k-ary tree*, where $k = 3$. A *k-ary tree* is a tree where each node has $k$ children, beside the leaf nodes. The depth of the tree was selected to be 7 [3].

## 5.1 Theoretical analysis

The no aggregation method just forwards the packets received towards the sink [3]. In this case, all packets are encrypted (it could be any encryption algorithm scheme as well as a homomorphic, but the property of aggregating data is not used) and beside the sink none of the nodes knows what is inside the packet. This method offers the best end-to-end privacy option, however the waste of bandwidth is obvious. In every next level of the k-ary tree, the transmission bit-length for the node in that level grows exponentially. In other words, in a 3-ary WSN tree of height 3, nodes in the third level transmit their encrypted samples, nodes in the second level transmit their own read sample and the other three sample values which they received from their predecessors,

[8]Without the header, footer and delimiter.

children. This geometric growth rate reduces exponentially the life of the nodes near the sink, the nearer the nodes to the sink are, the sooner their batteries die [3].

Another way to solve the bandwidth problem, instead of using no aggregation of data at all, is to try the hop-by-hop (HBH) approach. In this method, neighboring nodes create pair-wise keys with their neighboring nodes (children and parents) [3]. When they receive a data packet, they decrypt it first with the same key as their child had, sum it up (aggregate) with their information and then encrypt it, but with the key from their parent node [3]. This approach is more bandwidth efficient than the previous method without aggregation at all. No packet is sent twice [3]. Limitations of this method are: more battery power gets consumed because it encrypts and decrypts the data on the Microcontroller, but still less than the previous method[9], and there is a lack of end-to-end encryption of data [3]. The disadvantage of this method is its vulnerability to hackers. Hacking the node near the sink would result in knowing all the aggregated data, therefore this method is very vulnerable to external attackers[10].

## 5.2 Results

The Author's tests revealed similar results to the theoretical analysis in the preceding text. One test was conducted, to analyze how many bits are being transmitted at each level in the WSN tree, and out of which the bandwidth gain was calculated. The range of measured values was selected to be $t = 128$, i.e. temperatures $0 \sim +127°C$ or $-50 \sim +77°C$, in other words 7 bits are needed to represent a sample. In the WSN k-ary tree, $k$ was selected to be three, $k = 3$.

### 5.2.1 Bit-length transmission results

In the result table 1, the suffixes of A's denote only the transmission of averages whereas AV's, the averages and variances. A's and AV's alone with the percentage symbol denote the aggregation method suggested in the paper where the percentages denote how many nodes did not respond [3]. When a node does not respond, its parent node appends the ID of that node so the sink knows which nodes have failed to transmit their samples. The HBH method shows the best results[11], although the aggregation method, when each node responds, shows an equal distribution of load on every single node which can be considered as an eye-catching property [3]. The no aggregation method is the worst out of all three as it grows exponentially and the nodes with highest traffic load[12] die first. The first 56 bits in each packet are the header bits of the packet, used by the TinyOS, the rest of bits can be easily calculated using the equation $log(M)$, e.g. 75 bits are needed to transmit the average values with the aggregation method, $56 + log(t) + log(n) = 56 + log(128) + log(2187) = 56 + 7 + 12 = 75$ [3].

### 5.2.2 Bandwidth gain results

In the table numbered 2, one can see the bandwidth gains of the two methods, aggregation and HBH, compared to the no aggregation method. It is obvious that advantages exist over the no aggregation method. When the height of the tree is 8, the HBH-A method transmits up to 7.5 times less bits than the no aggregation method. The suggested solution by the authors saves around 6 times less bits which would be wasted in a tree of height 8 using the no aggregation method.

[9]The no aggregation method
[10]Each node stores the secret key to decode the data from its children.
[11]The least number of bits is transmitted
[12]Nodes near the sink

| Levels | # Nodes | A(0%) | A(10%) | A(30%) | AV(0%) | AV(10%) | AV(30%) | HBH-A | HBH-AV | No-Aggreg |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 75 | 950 | 2700 | 100 | 975 | 2725 | 73 | 97 | 68859 |
| 2 | 9 | 75 | 366 | 950 | 100 | 392 | 975 | 72 | 94 | 22932 |
| 3 | 27 | 75 | 172 | 366 | 100 | 197 | 392 | 70 | 91 | 7623 |
| 4 | 81 | 75 | 107 | 172 | 100 | 132 | 197 | 68 | 87 | 2520 |
| 5 | 243 | 75 | 85 | 108 | 100 | 111 | 132 | 67 | 84 | 819 |
| 6 | 729 | 75 | 78 | 85 | 100 | 103 | 110 | 65 | 81 | 252 |
| 7 | 2187 | 75 | 75 | 75 | 100 | 100 | 100 | 63 | 63 | 63 |

Table 1, bit-size lengths sent per node of the three methods, 3-ary tree, $t = 128$

| Levels | # Nodes | A(0%) | A(10%) | A(30%) | HBH-A | AV(0%) | AV(10%) | AV(30%) | HBH-AV |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 40 | 2.42 | 2.39 | 2.34 | 2.58 | 1.89 | 1.87 | 1.84 | 2.24 |
| 4 | 121 | 3.20 | 3.13 | 3.01 | 3.50 | 2.46 | 2.40 | 2.37 | 3.02 |
| 5 | 364 | 3.96 | 3.82 | 3.60 | 4.46 | 3.03 | 2.98 | 2.84 | 3.84 |
| 7 | 3280 | 5.46 | 5.13 | 4.58 | 6.41 | 4.10 | 3.90 | 3.60 | 5.52 |
| 8 | 9841 | 6.22 | 5.72 | 4.95 | 7.39 | 4.59 | 4.30 | 3.85 | 6.37 |

Table 2, bandwidth gain of the two methods compared to no aggregation method, 3-ary tree, $t = 128$

## 6 Ideas how to improve this solution

The author of this seminar had an idea how to improve the methods even more and to reduce power consumption by transmitting even less data bits. The author of the seminar looked online for his own idea and a paper was found which utilizes this approach and idea [7]. The basic idea is to compress the data by known natural facts, e.g. mornings and nights the temperatures are usually lower than during the day, or during summer the temperatures rarely go below zero. If we know which time of the season it is, we may expect some temperatures based on the previous statistical data measured. We may know the fact that the average highest temperature in June in Sarajevo is $+27°C$ and lowest $+14°C$ [1]. Instead of using 7 bits to represent this information, we may use the difference between 27 and 14, 13 to represent these temperatures, and we only need 4 bits to represent this information. "The basic idea of Huffman coding is that symbols that occur frequently have a smaller representation than those that occur rarely" [6]. Using this approach we could reduce the bit length for at least a double of the original size and prolong the life of our WSN. In the case where single readings are important, one could use the delta encoding algorithm [6]. Instead of sending readings like e.g. 54, 55, 67, 56, we could send the first value and then all the differences depending on the first value, in this case it would be 54,1,12,-11 ($54 + 1 => 55 + 12 => 67 - 11 => 56$.) Using this approach one could save more than the double[13] of bit-size length.

## 7 Conclusion

To summarize everything up, the goals of this paper were accomplished. The authors found a good way to use well the homomorphic property of their suggested encryption algorithm, which is simple to implement and at the same time they achieved an end-to-end encryption between the sensor nodes. They achieved bit-length reduction in the transmission process and automatically prolonged the lifetime of the WSN and saved spare bandwidth consumption. The author of this seminar work would like to emphasize their achievement of even communication load inside the WSN, as seen in table 1, packet sizes were equal, and this useful property will be definitely of further use. The author of this seminar suggested two other ideas how the bit transmission could be reduced even more by using the Huffman and Delta encoding methods. Finally, the author of this seminar will keep researching on his own in this field.

[13]The more readings there are, the more one can save on bit-size length.

## References

[1] Federal Hydrometeorological Institute. http://www.fhmzbih.gov.ba/engleski/index.php, 2008-03-1, 2008.

[2] Predator drones hacked in Iraq operations. http://www.cnet.com, 2009-12-24, 2009.

[3] Gene Tsudnik Claude Castellucia, Einar Mykletun. Efficient aggregation of encrypted data in wireless sensor networks. *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*, 2005.

[4] Hans Delfs and Helmut Knebl. *Introduction to Cryptography: Principles and Applications.* Springer, 2002.

[5] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons, 2005.

[6] Steven Smith. *Digital Signal Processing: A Practical Guide for Engineers and Scientists.* Newnes, 2002.

[7] C. Tharini and P. Vanaja Ranjan. Design of modified adaptive huffman data compression algorithm for wireless sensor network. *Journal of Computer Science 5*, 2009.