Data protection in Multipath WSNs

Quentin MONNET
Lab. LACL, Université Paris-Est
LACL (EA 4219), UPEC
F-94010 Créteil, France
quentin.monnet@lacl.fr

Lynda MOKDAD
Lab. LACL, Université Paris-Est
LACL (EA 4219), UPEC
F-94010 Créteil, France
lynda.mokdad@u-pec.fr

Jalel BEN OTHMAN
Lab. L2TI, Université Paris 13
L2TI (EA 3043), UP13
F-93430 Villetaneuse, France
jbo@univ-paris13.fr

***Abstract*—Used in areas such as pollution measurement or data gathering over battlefields, wireless sensor networks have attracted more and more attention over the last years. The deployment of such a network is accompanied by several security issues, including data confidentiality. Robust encryption algorithms addressed to network communication exist, but they do not always match the low resources restrictions — low processor, memory, limited energy — set upon the sensors. To overcome this, other, simpler solutions have been proposed, such as the *Securing Data based on Multi-Path routing* method, or an application of the *Shamir's Secret Sharing Scheme*, which both use distinct paths in the network to send pieces of data obtained by splitting the original message. This paper addresses the two methods named above, and proposes a solution based on traffic classification, using alternatively the *Securing Data based on Multi-Path routing* method, the *Shamir's Secret Sharing Scheme*, and strong encryption algorithms.**

***Index Terms*—Wireless communication, Sensor networks, Network-level security and protection.**

I. INTRODUCTION

Wireless sensor networks, often abbreviated *WSN*s, consist in sets of self-deployed small devices used to perform measurement on their immediate environment, and to forward collected data to a so-called *base station* through wireless communication. Sensors are supposed to have low computation capabilities, few available memory, and limited energy — to spend with care (in most cases batteries won't be reloaded). Over the last decades, the use of those networks have been introduced into a variety of application domains, including, but not limited to: pollution measurement, detection of forest fires, monitoring of thresholds of nuclear radioactivity, military communication over battlefields.

Amongst the deployed WSNs, some need to afford guaranties regarding data confidentiality. When the requirements are high, the use of strong cryptography to cipher the messages becomes unavoidable. But let's imagine an application which produces some messages of critical importance, along with trivial data. Is it worth encrypting the whole thing, knowing that encryption consumes both computational and memory resources, and costs processing time? In this article we expose in detail two existing methods based on multi-path routing and providing several degrees of confidentiality, the *Securing Data based on Multi-Path routing* (*SDMP*) and a *Threshold Sharing Scheme*: *Shamir's Secret Sharing Scheme* (*SSSS*). We also propose a solution based on traffic classification to assign one of the three methods — the two mentioned above, and encryption — to each packet in the network according to its level of required confidentiality (that is, to the criticalness of its content).

Each of those methods have already been presented for use in wireless networks, and in particular in WSNs. A *Threshold Sharing Scheme*, for example, consists roughly in splitting the message into n shares such that k or more shares out of the n leads to the message reconstruction, whereas any set of $k-1$ shares is unexploitable. Although the scheme was originally designed for sharing a secret between several participants, it has also been proposed for securing network transmissions. A recent suggestion is that MPLS networks security, for example, could be enhanced with this method [1]. By sending the n shares through n distinct paths, often available in networks based on this protocol, the transmitter ensures that an attacker observing the traffic on at most $k-1$ paths between the nodes will not be able to recover the message content. Similar proposals had been made for securing wireless ad-hoc networks, without MPLS, but still relying on distinct paths [2], [3].

The general use of cryptography to ensure multiple aspects of privacy in WSNs has of course drawn interest from researchers, and led to the creation of many protocols, for both confidentiality [4] and authentication [5]. Resistance to “denial of service” attacks, coming from the outside as well as from the inside of the network, is also a deeply investigated topic concerning WSNs security. For instance we propose in [6] a solution for detecting and reacting to compromised nodes trying to saturate bandwidth in clustered networks. To this aim, some nodes are assigned a monitoring role, which is periodically renewed so as to distribute the energy load evenly among the cluster. Also, the solution is modeled in terms of Markov chains and Petri networks. More generally, most of the identified attacks regarding WSNs and protocols designed to circumvent them are resumed in several state-of-the-art articles [7], some being dedicated to multi-path routing solutions [8]. In this paper we propose a solution based on traffic classification, using alternatively the *Securing Data based on Multi-Path routing* method, the *Shamir's Secret Sharing Scheme*, and strong encryption algorithms.

The organization of the rest of the article comes as follows: in section II we present our solution based on traffic classification, and explain in detail how the application of the

SDMP (subsection II-B) and SSSS (subsection II-C) methods may ensure confidentiality in WSNs. Some aspects of those methods, along with the use of "classical" encryption, are compared in section III. Finally, we conclude and give future research perspectives in section IV.

II. Proposed solution

A. Traffic classification

The proposed solution relies on classification of the packets according to their respective importance. The emitting node must use a traffic shaper so as to mark the packets as of *low*, *middle* or *high* importance. Distinction between the degrees of importance is set by the user; it may depend on the packet type (protocol in use, destination port, *et cætera*) as well as on the nature of carried data. Each node of the network should embed a shaper to determine the category to which its enqueued packets belong.

Once the degree of importance of an outgoing packet has been determined, it is sent with the associated method:

- for low importance packets: weak (but fast) SDMP method (see subsection II-B);
- for middle importance packets: *Shamir's Secret Sharing Scheme* (see subsection II-C);
- for high importance packets: ciphering (see subsection II-D).

B. Low importance traffic: SDMP method

1) Sending: The SDMP (*Securing Data based on Multi-Path routing*) method [9] basically consists in:

1) dividing the packet into n pieces;
2) "hiding" nearly all pieces with a logical exclusive OR (that is, a XOR) operation;
3) sending the pieces to the target node *via* n distinct paths.

Of course this method requires n distinct paths to be available between the emitting node and the recipient of the packet.

The actual packet splitting for SDMP is very simple: a message of length l_m is split into n pieces of length $l_p = \lceil \frac{l_m}{n} \rceil$. We want all the pieces to have the same length. So if $\frac{l_m}{n}$ is not an entire value, padding must be added to the message before splitting. We obtain n pieces p_1, p_2, ..., p_n. Each piece p_i is then "hidden" by applying to its content a XOR operation with the content of the p_{i+1} piece (and p_n is XORed with p_1). There is one exception: a single piece p_k, with k being a random number such as $1 \leq k \leq n$, remains clear and will act as a "key" to recover the original message. We now have n pieces $p'_{1,2} = p_1 \oplus p_2$, $p'_{2,3} = p_2 \oplus p_3$, ..., $p'_{k-1,k} = p_{k-1} \oplus p_k$, p_k (unchanged), $p'_{k+1,k+2} = p_{k+1} \oplus p_{k+2}$, ..., $p'_{n-1,n} = p_{n-1} \oplus p_n$, $p'_{n,1} = p_n \oplus p_1$. Each piece p_i is sent over one of the n distinct paths in the network.

2) Receiving: The target node will not be able to recover the whole message unless it receives all the pieces. Starting with p_k — which was sent in clear text — and p'_{k-1}, the target node will retrieve p_{k-1}, then p_{k-2}, *et cætera*, until all pieces p_i are recovered. The last step is the reconstruction of the original message by concatenating the different pieces.

An attacker trying to intercept the exchanged message will not be able to recover the full content of the message unless it catches every single piece, which means that all the paths that were used are compromised.

3) New header: Actually, if one wants to implement SDMP, a few supplementary information must be sent along with the pieces. For each piece, the number i of the piece must be sent, so that the target node may be able to know in which order the recovered pieces must be concatenated. If padding was added to one or more pieces, it should also be indicated. A two (for instance) byte-long header containing those numbers should be added at the beginning of each piece to send.

The Figure 1 shows a concrete example of the effectuation of the SDMP method.

4) Signaling: Furthermore, the recipient node also needs to know:

- how many pieces constitute the original message;
- what the number of the "key" piece is.

The number of pieces and the designation of the "key" piece form what we call the *signaling* for this method. It could be fixed. That is to say, the user could initially choose one value n in which to split all the messages, and one value k that would represent the "key" piece for each set of pieces resulting from a splitting. But all nodes in the network do not have the same connectivity index, and it is very likely that some couple of nodes will have more distinct paths available to communicate between them than some other couples. So n should depend on the number of available distinct paths, and must be sent for each message. Regarding the k value, it indicates which piece of the original message is sent in clear text; so it should be chosen at random for each message, to improve security.

k and n are both necessary for the reconstruction of the original message. Hence those values should be sent with a minimum of caution, as they would help an attacker to reconstruct the message, were they to get captured. There are two ways to transmit signaling from the emitting node to the recipient:

- *out-of-band signaling:* the first way consists in sending the n and k values in a specific packet, different from the pieces containing the data. In no case it should be sent through the same path than the "key" piece of data. At least it should be sent through a distinct path; if possible, it is ciphered (encrypting the "key" piece only would require less resources than encrypting the whole message).
- *in-band signaling:* the second way to transmit signaling values is by including it to the pieces containing the data. Two associated fields must be added at the beginning of those pieces. n and k should not be inserted into each piece of the original message; instead, it should be "divided" so that the recipient node needs to combine several pieces to get the signaling values. So the n and k values are split into several sets of bits, each being included in a different piece of data. The recipient node retrieves n and k by XORing all n and k fields from the

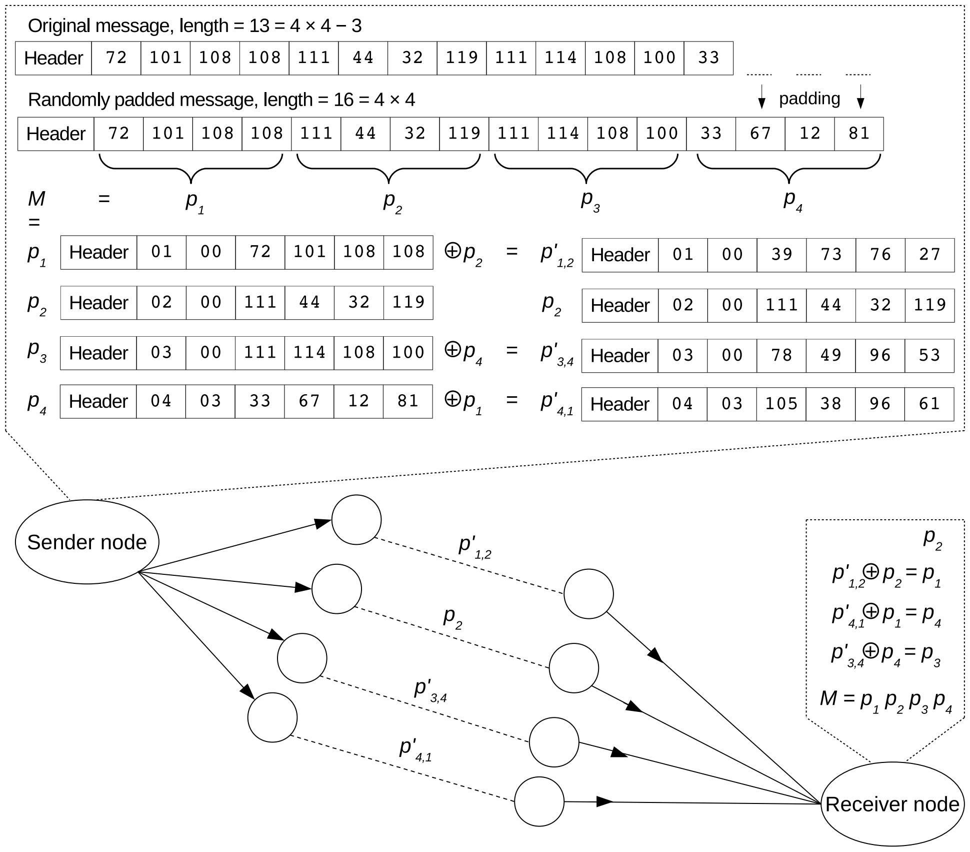
Original message, length = 13 = 4 × 4 − 3
Header 72 101 108 108 111 44 32 119 111 114 108 100 33
Randomly padded message, length = 16 = 4 × 4
padding
Header 72 101 108 108 111 44 32 119 111 114 108 100 33 67 12 81
M = p1 p2 p3 p4
p1 Header 01 00 72 101 108 108 ⊕p2 = p'1,2 Header 01 00 39 73 76 27
p2 Header 02 00 111 44 32 119 p2 Header 02 00 111 44 32 119
p3 Header 03 00 111 114 108 100 ⊕p4 = p'3,4 Header 03 00 78 49 96 53
p4 Header 04 03 33 67 12 81 ⊕p1 = p'4,1 Header 04 03 105 38 96 61
Sender node
p'1,2
p2
p'3,4
p'4,1
Receiver node
p2
p'1,2⊕p2 = p1
p'4,1⊕p1 = p4
p'3,4⊕p4 = p3
M = p1 p2 p3 p4

Figure 1. SDMP method: concrete example, with $n = 4$, key $= p_2$

different pieces. But the node can not be sure that all the sets of bit reached it, since it ignores how many pieces should be received. So a third field must be added to the pieces of data, containing a mask for the bits which are actually sent in this piece.

Note that for more clarity, signaling was not represented on Figure 1.

5) Possible ameliorations: Two improvements with regards to the original SDMP method are proposed to enhance the security:

- sending $p''_{k,k+1,k+2}$ instead of p_k as a "key piece". That is, the "key piece" should be created by XORing p_k and $p'_{k+1,k+2}$ (for example) so as not to send p_k in clear text. When receiving both $p'_{k+1,k+2}$ and $p''_{k,k+1,k+2}$, the recipient node will be able to recover p_k;
- changing the XORing (and recovering) order for the pieces. For instance, if n is equal to 5, instead of sending $p'_{1,2}$, $p'_{2,3}$, $p'_{3,4}$ and $p'_{4,5}$, one could create and send $p'_{1,4}$, $p'_{4,2}$, $p'_{2,3}$ and $p'_{3,5}$. It would make it more difficult for an attacker ignoring the order of reconstruction to recover the content of the pieces.

Of course, each one of these improvements implies more signaling data to send.

C. Middle importance traffic: Shamir's Secret Sharing Scheme

1) Principle: The SDMP solution is good for low importance packets, mostly because it is very fast to reconstruct the original message. But if a single piece of the message gets lost, retrieving the whole set of data becomes impossible. On the other hand, if an attacker manages to grasp the "key piece" and several other pieces which content may be retrieved, a large amount of data will leak. Hence we propose to use a (k, n) *Threshold Sharing Scheme* for middle importance packets.

A *Threshold Sharing Scheme* (*TSS*) is an algorithm used so as to share a secret between several participants. Each of the n participants is given a personal *secret share* (or *shadow*) in such a way that any k of them ($1 < k \leq n$) may combine their shares to retrieve the original secret, whereas $k - 1$ participants will not obtain any information about it. There are several existing schemes. Some, for instance, are based on matrix projection [10]. Here we only present a well-known TSS: a variation of *Shamir's Secret Sharing Scheme* [11]. This approach is based on the Lagrange interpolation formula for polynomials. Let p be a prime number and f be a polynomial function of degree $k - 1$ such as:

$$f(x) = (a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + a_0) \bmod(p) \quad (1)$$

The coefficients $a_0, \ldots, a_{k-1}$ are elements over a finite field

$\mathbb{Z}_p$. According to Shamir's original scheme, a_0 is the shared secret, and all other coefficients are chosen at random. If one knows k distinct couples of values (x_i, y_i) with $y_i = f(x_i)$, then the Lagrange linear interpolation formula makes it possible to represent the polynomial function in another way:

$$f(x) = \sum_{i=1}^{k} \left(y_i \prod_{\substack{1 \le j \le k \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \right) \tag{2}$$

and hence to retrieve the coefficients. Note that every operation is made over the finite field $\mathbb{Z}_p$. Practically, it means that additions and subtractions are made modulo p, subtracting b from a is adding b's opposite $(p-b)$ to a, and dividing a into b comes to multiply a with b's inverse in $\mathbb{Z}_p$ (the inverse is c such as $(b \cdot c) \bmod(p) = 1$, it does exist since p is prime).

A basic analogy of this method with geometry would be the following: thanks to Lagrange formula, two points on a plan are enough to determine the 1-degree polynomial function associated with a straight line. Note that three aligned points would also enable us to retrieve the polynomial, but two are enough. In this case, the shared secret is the 0 degree coefficient of the coefficient of the polynomial. Each of the n participants will be given the coordinates of one point of the straight line. A single participant alone is totally enable to retrieve the polynomial (there are an infinity of lines passing by a given point). There must be at least $k = 2$ cooperating participants to obtain the coefficients of the polynomial, and thus to retrieve the shared secret. In the same way, Lagrange interpolation makes it possible to find the polynomial function associated with a parabola from three points, with a hyperbola from four points, *et cætera*.

When applied to network communication, original *Shamir's Secret Sharing Scheme* introduces much overhead. To bypass this a slight modification is realized: in the rest of this article the shared secret is no more constituted only by the 0 degree coefficient (a_0) of the polynomial function. Instead, it embeds the whole set of coefficients (*i.e.* $a_{k-1}, \ldots, a_1$, are part of the original secret, as well as a_0, and are no more chosen at random). As we are working with bytes (values from 0 to 255), we choose p equal to 257 (the lower prime number strictly greater than 255).

2) Message sending: As for the SDMP method, the original packet M is padded (if necessary) and n shares are created before being sent. A set containing any k shares over the n created will be necessary to retrieve the full-length message. So as to enhance the security of the communication, we will choose k equal to n. In that way the recipient node must receive each one of the different shares to achieve the reconstruction process; were the attacker to lack a single share, the attack would remain unsuccessful.

Contrary to the pieces of the SDMP method, the shares created with the TSS method do not result directly from the original message splitting. Instead of being divided into n pieces of equal length, like for SDMP, M is at first divided into several chunks of length k. Let l be the number of chunks created. Each chunk i (over the l chunks) contains the k coefficients $a_{i,k-1}, \ldots, a_{i,1}, a_{i,0}$, which form a polynomial function $f_i(x) = a_{i,k-1}x^{k-1} + \cdots + a_{i,1}x + a_{i,0}$. Each share j (over the n shares) is then created by choosing a unique value x_j for x: the values $f_1(x_j), \ldots, f_l(x_j)$ are computed and concatenated to form a share of length l.

There is no need to "order" the different shares before sending them. But it is necessary to indicate for each share the value x_j that was chosen to compute its $y_i = f_i(x_j)$ values. The minimal number (k) of shares needed to reconstruct the original message M must also be added; otherwise, the receiver node would not know whether it should attempt to retrieve M or whether it lacks some shares. It may also be necessary to indicate the amount of padding bytes that were added to M.

Once complete, each share is then sent through the network *via* a distinct path.

Here is a concrete example of the creation of the shares. We want to send the thirteen following (decimal) bytes over three distinct paths, using Shamir's secret share: 72 101 108 108 111 44 32 119 111 114 108 100 33. Here are the different steps to do so:

1) We have three distinct paths, so we choose $n = 3$. As mentioned earlier, we choose $k = n$, so $k = 3$. We also have $p = 257$ (lowest prime number greater than the maximal byte value, 255).
2) The message has a length of 13 bytes. It should be a multiple of k, so we add two bytes for padding: 0 and 1.
3) We split the padded message into chunks of length k. We obtain four chunks: 72 101 108, 108 111 32, 44 119 111, 114 108 100 and 33 0 1.
4) We get the five following polynomial functions:

$$\begin{cases} f_1(x) = (72x^2 + 101x + 108) \bmod(257) \\ f_2(x) = (108x^2 + 111x + 32) \bmod(257) \\ f_3(x) = (44x^2 + 119x + 111) \bmod(257) \\ f_4(x) = (114x^2 + 108x + 100) \bmod(257) \\ f_5(x) = (33x^2 + 1) \bmod(257) \end{cases} \tag{3}$$

5) We choose n distinct and **non-zero** values for x. For instance, $x_1 = 1$, $x_2 = 3$, and $x_3 = 4$.
6) We compute the shares' content. The first share (s_1) is made of $f_1(x_1)$, $f_2(x_1)$, $f_3(x_1)$, $f_4(x_1)$ and $f_5(x_1)$. We compute and concatenate these values. So the data of s_1 is 24, 251, 17, 65, 34. The other shares s_j are computed in the same way, by using the associated x_j value. Their content is displayed in Table I.

Table I
CONTENT OF THE COMPUTED SHARES

Share	Content				
s_1	24	251	17	65	34
s_2	31	52	93	165	41
s_3	122	148	6	43	15

After inserting a new header containing x_j, k and the number of padding bytes, we get the shares s_j (for s_1,

the header is: 1, 3, 2).
(Note that each share contains data about each one of the bytes from the original message. There is no ordering between the shares, none “comes last”, so the number of padding bytes is the same for every share).

7) Headers of the lower layers are added, and the shares are sent through the network, each one over a distinct path.

3) Message reconstruction: Before attempting to reconstruct the original message M, the recipient node must check that it has received enough shares (*i.e.* that it has received at least k shares). Otherwise the retrieved data will not be consistent. This verification is easy to realize if k has been inserted in a new header for the shares, as explained for the shares' construction. Then, any set of k shares (among all the one received for a message M) may be used with Lagrange linear interpolation formula (see equation 2) so as to get back the coefficients a_i, hence the data of M.

Let us get back to our concrete example. The examination of the k-value field of the shares' headers tells us that three different shares are needed to proceed to the reconstruction. Once the three shares received, we will apply Lagrange formula to the $f_1(x_j)$ values ($1 \leq j \leq 3$). Here is the resulting expression:

$$f_1(x) = \sum_{i=1}^{i=3} \left(f_1(x_i) \prod_{\substack{1 \leq j \leq 3 \\ j \neq i}} \frac{x - x_j}{x_i - x_j} \right) \tag{4}$$

So we have:

$$f_1(x) = (t_{1,1}(x) + t_{1,2}(x) + t_{1,3}(x)) \bmod(257) \tag{5}$$

where

$$\begin{cases} t_{1,1}(x) = f_1(x_1) \cdot \dfrac{x - x_2}{x_1 - x_2} \cdot \dfrac{x - x_3}{x_1 - x_3} = 24 \cdot \dfrac{x-3}{1-3} \cdot \dfrac{x-4}{1-4} \\ t_{1,2}(x) = f_1(x_2) \cdot \dfrac{x - x_1}{x_2 - x_1} \cdot \dfrac{x - x_3}{x_2 - x_3} = 31 \cdot \dfrac{x-1}{3-1} \cdot \dfrac{x-4}{3-4} \\ t_{1,3}(x) = f_1(x_3) \cdot \dfrac{x - x_1}{x_3 - x_1} \cdot \dfrac{x - x_2}{x_3 - x_2} = 122 \cdot \dfrac{x-1}{4-1} \cdot \dfrac{x-3}{4-3} \end{cases} \tag{6}$$

hence:

$$\begin{cases} t_{1,1}(x) = 24 \cdot \dfrac{(x-3)(x-4)}{255 \cdot 254} = 24 \cdot 6^{-1} \cdot (x-3)(x-4) \\ t_{1,2}(x) = 31 \cdot \dfrac{(x-1)(x-4)}{2 \cdot 256} = 31 \cdot 255^{-1} \cdot (x-1)(x-4) \\ t_{1,3}(x) = 122 \cdot \dfrac{(x-1)(x-3)}{3 \cdot 1} = 122 \cdot 3^{-1} \cdot (x-1)(x-3) \end{cases} \tag{7}$$

so

$$\begin{cases} t_{1,1}(x) = 24 \cdot 43 \cdot (x^2 + 250x + 12) = 4 \cdot (x^2 + 250x + 12) \\ t_{1,2}(x) = 31 \cdot 128 \cdot (x^2 + 252x + 4) = 113 \cdot (x^2 + 252x + 4) \\ t_{1,3}(x) = 122 \cdot 86 \cdot (x^2 + 253x + 3) = 212 \cdot (x^2 + 253x + 3) \end{cases} \tag{8}$$

and finally we sum up all $t_{1,i}$ to find:

$$f_1(x) = 72x^2 + 101x + 108 \tag{9}$$

The three coefficients ($a_2 = 72$, $a_1 = 101$ and $a_0 = 108$) of this polynomial are the first three bytes of our original message. By applying the same procedure to the other $f_i(x)$ from the shares, we would get the remaining bytes from the initial message. In the end, the only step that would remain would be the removal of the two padding bytes from the three bytes obtained with f_5, to get the full original message.

D. High importance traffic: ciphering

For important packets that embed data expected to remain confidential whatever happens in the network, none of the above solutions is able to provide enough guaranties in terms of security. The only way to ensure that data remains confidential is to use a well-tried strong encryption algorithm. We do not either present or recommend any particular algorithm in this paper. Several algorithms among the many that exist have been specifically designed for WSNs.

III. Comparison of the different methods

A. Confidentiality

While a good encryption algorithm is expected to provide a very good confidentiality level even if most, or all, packets are captured by an attacker, the SDMP and TSS methods are not as efficient to protect data. This is why these methods are proposed for middle and low importance packets.

The security of the SDMP method is even lower than for the *Threshold Sharing Scheme*, especially if the “key” piece is sent in clear text. If captured, this piece will instantly reveal its content to the attacker. Moreover, it would also permit to retrieve the content from other pieces (for instance, if the key piece p_k, and the pieces $p'_{k-1,k}$ and $p'_{k-2,k-1}$ are caught, all of them are easy to retrieve). Lacking signaling may delay the retrieval of data: if the attacker does not know which piece is the key, it may take longer to extract the original message (although there are not enough possible combinations to resist to a “brute force” reconstitution). This is why signaling should be handled with care. Finally, it is worth noticing that single XOR operations between pieces are not a very efficient way to encrypt it, and that it would probably not resist to statistical analysis. Some byte patterns could indeed appear repeatedly in the pieces, or from one piece to another.

Shamir's Secret Sharing Scheme offers a better protection. If we choose k and n such that $k = n$, a single missing share prevents an attacker to get any information about the content of the original message. As a consequence, an attacker must be able to listen over each of the distinct paths that are used to forward the shares. Due to the wireless aspect inhering in WSNs, this condition can be bypassed if the attacker is located close enough to its target. In this case, the attacker is able to intercept all the emitted shares, before they are routed to distinct paths. And note that again, repeated patterns in the original data could produce similar patterns in the share, and provide matter for successful statistical analysis.

B. Complexity

The SDMP method is by far the easiest method to implement. Each piece results directly from the splitting of the original message, and only needs a new header and a single XOR operation before sending.

The secret sharing scheme is also simple, but requires more operations to create the shares. The total number of operations (additions, multiplications and modulos) grows exponentially with k (complexity in $\mathcal{O}(k^2)$). Nevertheless it remains few demanding in resources for low values of k. As the number of available distinct paths for a given node will seldom reach high values, k should remain low, causing the algorithm to remain cheap and fast.

A strong encryption protocol, by contrast, is far more demanding in resources, and more complex to implement. This is the price for a better confidentiality.

C. Overhead

Let l_m be the length of the original message.

With the SDMP method, the message is split into n pieces of length $l_p = \lceil \frac{l_m}{n} \rceil$ (once padded). To each piece we must add the piece number and the padding length l_{pad} in a new header. Signaling must also be taken into account. Out-of-band signaling produces less signaling data, but requires a full packet with lower layers headers. For in-band signaling, we will count four supplementary fields (k, n, and respective masks — see II-B4 and explanations about the signaling for SDMP). So the final length for data, signaling and inserted headers is (in the case of in-band signaling): $n \cdot (l_p + 6) = l_m(+l_{pad}) + 6n$; in other words, $6n$ bytes of overhead are created. Lower layers headers (*e.g.* link layer, network layer) will constitute the greatest part of the overhead: they will be multiplied by n (one per piece to send).

This is quite similar for *Shamir's Secret Sharing Scheme*: we basically have n shares of length $\lceil \frac{l_m}{k} \rceil$ (once the message is padded, and with $k = n$), to which we add a new header containing k and the padding length. The final length for data and inserted headers is: $l_m(+l_{pad}) + 2n$. Overhead is slightly lower than for the SDMP method, but the lower layers headers are again multiplied by n, so the difference for the two methods is insignificant.

Encryption algorithms often add new headers and initialization vectors which end up in consequent overhead. On the other hand, the packets are not split into pieces or shares, which prevent the replication of lower layers headers. Therefore, the method which adds the greatest overhead will depend on the value of n and on the amount of overhead that the chosen encryption algorithm will create for each packet.

D. Fault resistance

The SDMP method offers no fault resistance. The pieces are created through a kind of chained encryption process. If one link of the chain is missing, *i.e.* if one piece is lost, all the following pieces in the chain become useless and can not get unencrypted.

This is even worse with the *Threshold Sharing Scheme*: since we chose $k = n$, should a single share get lost, it would cause every other share to become worthless. Remember that this is the principle of the secret sharing: with $k-1$ participants, no information can be obtained about the shared message. This method introduces a very interesting mechanism for fault tolerance, however, when k is lower than n. In this case, all the data is retrieved even if any $n-k$ shares are lost; but this is beyond the scope of our analysis.

IV. Conclusion

If sensor nodes in a WSN have to send packets with different levels of importance, it may be worth to avoid to use systematically heavy encryption algorithms. We propose in this paper a solution using a traffic shaper to determine the degree of criticalness of each packet to send. According to this degree and to the number of available distinct paths between the sender and the target node, one of three securing methods is used: the *Securing Data based on Multi-Path routing* method, *Shamir's Secret Sharing Scheme*, or strong cryptography. We have detailed the operation and provided concrete examples for the first two methods, before comparing which confidentiality guaranties, which complexity, which additional overhead and which fault resistance come with each one of the three methods (SDMP, SSSS, and regular encryption).

Future works include simulating the solution over the *ns-3* simulator, in order to evaluate the gain in performance and in energy provided by this model.

References

[1] S. Alouneh, A. Agarwal, and A. En-Nouaary, "A novel path protection scheme for MPLS networks using multi-path routing," *Computer Networks*, vol. 53, no. 9, pp. 1530–1545, Jun. 2009.

[2] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network Magazine*, vol. 13, no. 6, pp. 24–38, Dec. 1999.

[3] Y. Mao, "A feedback-based multipath approach for secure data collection in wireless sensor networks," *Ubiquitous Computing and Communication Journal*, vol. 5, no. 2, pp. 27–32, Jun. 2010.

[4] H. Alzaid, E. Foo, and J. G. Nieto, "Secure data aggregation in wireless sensor network: a survey," in *Proceedings of the 6th Australasian Information Security Conference (AISC'08)*, vol. 81, Wollongong, NSW, Australia, Jun. 2008, pp. 93–105.

[5] M. A. Simplicio, Jr, B. T. de Oliveira, P. S. L. M. Barreto, C. B. Margi, T. C. M. B. Carvalho, and M. Naslund, "Comparison of authenticated-encryption schemes in wireless sensor networks," in *Proceedings of the 36th Annual IEEE Conference on Local Computer Networks*, Bonn, Germany, Oct. 2011, pp. 454–461.

[6] P. Ballarini, L. Mokdad, and Q. Monnet, "Modeling tools for detecting DoS attacks in WSNs," *Security and Communication Networks*, 2013, not published yet.

[7] S. K. Singh, M. P. Singh, and D. K. Singh, "A survey on network security and attack defense mechanism for wireless sensor networks," *International Journal of Computer Trends and Technology*, May 2011.

[8] E. Stavrou and A. Pitsillides, "A survey on secure multipath routing protocols in WSNs," *Computer Networks*, vol. 54, no. 13, pp. 2215–2238, Sep. 2010.

[9] J. Ben-Othman and L. Mokdad, "Enhancing data security in ad hoc networks on based multipath routing," *Journal of Parallel and Distributed Computing*, vol. 70, no. 3, pp. 309–316, Mar. 2010.

[10] K. Wang, X. Zou, and Y. Sui, "A multiple secret sharing scheme based on matrix projection," in *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09)*, Seattle, WA, USA, Jul. 2009, pp. 400–405.

[11] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.