Pollution Attack Defense for Coding Based Sensor Storage

Levente Buttyán, László Czap, István Vajda
Budapest University of Technology and Economics
Laboratory of Cryptography and Systems Security (CrySys)
{buttyan,czap,vajda}@crysys.hu

***Abstract*—We present a novel information theoretic approach to make network coding based storage secure against pollution attacks in sensor networks. The approach is based on a new decoding algorithm which makes it possible to find adversarial blocks using one more encoded block than strictly necessary for decoding. Our scheme fits well to the requirements of sensor networks, because it operates with adding very low computational and communication overhead to source and storage nodes, only the collector node needs to perform some additional computation. Our approach does not apply cryptography, hence it works in environments where no pre-shared keys, secure channels or PKI are available, which is often the case in sensor networks.**

I. Introduction

The principle of network coding [1], [2] has emerging applications both in the field of wired and wireless networking [3], [4], as well as in the field of storage systems or peer-to-peer networks [5], [6]. These systems benefit from coding in terms of throughput, efficiency, robustness and fault-tolerance. Here we consider a special application of network coding, a coding based storage system in sensor networks. In this system data is produced by multiple source nodes, and is stored encoded in multiple storage nodes [7], [8], [9], [5]. Storage nodes apply a random linear code on received data. When a collector node needs to reconstruct the original data, it obtains encoded data by downloading the content of some selected storage nodes, and performs decoding. Previous work show the benefits of the encoded storage system [10], [5].

While network coding has advantages in benign environments, its performance may seriously fall in the presence of an adversary [11], [12], [13]. We present a scheme that resists *pollution attacks*, meaning that our scheme makes possible successful decoding even if some compromised nodes store maliciously modified data. The novelty of our scheme is the decoding algorithm that allows the collector node to identify unmodified data blocks using one additional intact block, but without the need to modify the encoding algorithm or adding further overhead to the data.

Contrary to cryptographic integrity protection solutions, we do not require the existence of PKI or secure channels between the source nodes and the collector. Our scheme exploits the special properties of sensor storage systems, hence it provides better applicability in such systems than existing general pollution attack defense techniques, however, its scope of applicability may be smaller. Moreover, our solution is the first practical secure network coding scheme that is applicable when encoded blocks are composed of blocks originating from different sources.

The algorithms presented here are based on the principles introduced in [13]. Here we explore the theoretical limits of the method and we propose a much more efficient decoding algorithm than the one in [13].

The rest of the paper is organized as follows. Section II gives the model of the storage system we assume and defines the adversary as well. Section III describes the decoding algorithm we propose, while the properties of the scheme are analyzed in Section IV. A part of the decoding algorithm, the subspace search is discussed in Section V. In Section VI we investigate the performance compared to other solutions. Section VII gives an overview of related work, while Section VIII concludes the paper.

II. System model

We adopt the model of distributed storage for wireless sensor networks from [5], however our algorithms may be applicable for other storage systems as well. In this system data produced by l independent data sources is stored in n ($n > l$) storage nodes.

We assume that sources transmit equal size data blocks, the l sources have data $M_1, M_2, \ldots M_l$ to send. Data blocks can be parts of a large message, or independent data blocks. We require that the number m of $\mathrm{GF}(q)$ symbols in a data block be sufficiently large, because, as we will see, this is an important security parameter of the system. Storage nodes perform random linear network coding to produce *encoded data blocks* (or shortly encoded blocks). The encoding is performed using a Galois field of size q. Each M_i can be seen as a vector of length m over $\mathrm{GF}(q)$. The encoded block is a random linear combination computed from the data blocks [14]. Formally, to produce an encoded block, the ith storage node selects an encoding vector $\mathbf{e}_i = [e_{i_1}, e_{i_2}, \ldots, e_{i_l}]$ of length l having its elements chosen uniformly at random from $\mathrm{GF}(q)$. The encoded data block is then produced using these values as coefficients: $E_i = \sum_{j=1}^{l} e_{ij} M_j$. An encoded block consists of an encoding vector and the corresponding encoded data: $\mathbf{e}_i || E_i$. Storage nodes select encoding vectors independently for each encoded data block.

An encoded block $\mathbf{e}_i || E_i$ is a linear equation, with coefficients $e_{i_1}, e_{i_2}, \ldots, e_{i_l}$ and variables $M_1, M_2, \ldots, M_l$. We denote such an equation $Z_i = (\mathbf{e}_i, E_i)$. Without confusing

978-0-7695-4049-8/10 $26.00 © 2010 IEEE
DOI 10.1109/SUTC.2010.57

the reader, let us also denote by Z_i the $l+m$ length vector $Z_i = \left[\begin{array}{cc} \mathbf{e}_i, & E_i \end{array} \right]$.

In a benign environment, the collector node obtains the original message by downloading l equations from l randomly selected storage nodes and by simply solving a system of linear equations (s.l.e.). As coefficients are chosen randomly, according to [15], the linear independence of equations holds with probability

$$p_q = \prod_{i=1}^{l} \left(1 - \frac{1}{q^i}\right),$$

which is sufficiently large for a properly chosen field size q. E.g. if $q = 2^8, l = 10$, then $p_q = 0.996$. Our decoding algorithm described later first finds a set of unmodified (*intact*) equations then computes the solution of this intact s.l.e.

The encoded data block is the unit in which the collector may download data. We do not distinguish random transmission errors and malicious modifications of the transmitted data. However, common error detection and error correction techniques applied in the lower layer against channel errors are assumed to ensure the reception of unmodified blocks as well. The collector node applies our decoding algorithm to recover the data sent by the source nodes in a way similar to decoding of rateless codes.

We further assume that the source nodes and the collector do not share a key, neither have the possibility to establish a secure channel.

A. Adversary

We consider an adversary who compromises some selected storage nodes, and reads and modifies the contents of them. We do not limit the number of nodes the adversary may compromise, but note that at least $l+1$ intact nodes are required anyway for successful recovery. By compromising more nodes the adversary can also increase the error probability of the decoding. For detailed analysis we refer to Section IV. We assume the adversary can not compromise sources, but she may compromise the communication links between the sources and the selected storage nodes. It gives more possibility to the adversary, but does not influence the effect of the attack. The adversary has no information about the set of nodes from which the collector downloads data.

This model of adversary is realistic in practice, because contrary to source nodes, storage nodes are exposed to attacks for an extended period of time.

III. Attack-resilient decoding

The novelty of our scheme is the algorithm of decoding that assures the attack-resilient property, which is achieved by using one more encoded block than strictly necessary for data reconstruction. The order and numbering of blocks are not relevant, so we number the equations from the collector's point of view. Let Z_i^* denote the ith block (equation) received by the collector, either it is from a compromised node or not. The collector can reconstruct the original blocks if it is able to collect l linearly independent equations, and if it can be sure, that all of these l equations are intact.

The main difficulty of the decoding is to find the intact equations among the received equations. We show, how the $(l+1)$-st intact equation is eligible to find the l intact equations in the received set of encoded equations.

Here we explain the principle of our algorithm. An encoding vector and the corresponding encoded data form a vector from the $l+m$ dimensional vector space. Each block, either intact or adversarial, is a vector from that space. However, intact and adversarial equations can be distinguished upon whether they belong to a specific subspace. Decoding exploits the fact that intact equations span an l dimensional subspace in the $l+m$ dimensional vector space, because all intact equations are linear combinations of the same l base equation, which we get by tagging the ith data block with the ith unit vector as encoding vector. Each intact equation belongs to that specific subspace, and each element of this subspace can be treated as an intact equation. We take advantage of the fact that it is unlikely that $l+1$ vectors taken from the $l+m$ dimensional space fall into the same l dimensional subspace if they are not all intact equations. Hence, when $l+1$ equations that belong to the same l dimensional subspace are found, it is reasonable to assume that all these equations are intact. Random coding and the random order of downloading equations together assure this property. We further exploit that the $(l+1)$-st intact equation is a linear combination of the formerly received l equations. We use this idea to find the intact equations in a set that contains both intact and adversarial equations.

The pseudo-code of the decoding algorithm is presented as Algorithm 1. The algorithm finds a set S that contains l intact equations. Equations are received while the rank of matrix $\mathbf{Z}$, formed by the received equations, increases (lines 5-8). The $(l+1)$-st intact equation certainly does not increase the rank, because it is a linear combination of the first l intact equations. If the last received equation does not increase the rank, we can assume that it is the $(l+1)$-st intact equation received. In this case, the linear combination of all received equations (coefficient vector λ) that produces the last equation of the already received equations is computed (line 9). Set S is formed of the equations that have non-zero coefficient in this linear combination (lines 10-14). If S has less than l elements, the last received equation is not innovative and it is dropped (line 4), and the operation continues (loop of lines 3-15). If S has exactly l elements, we can assume that the last received equation is the $(l+1)$-st intact equation, and S contains the first l intact equations received. In this case we return this set as an intact s.l.e. (lines 16-18). Otherwise, we cannot say anything about which equations are intact, because intact and adversarial equations together span the subspace that the last received equation belongs to, and its dimension is larger than l. In this case we start searching the proper subspace in the set of received equations (lines 18-21). We describe the algorithm of this subspace search later in Section V, but we note that it is unlikely that an intact equation falls into a subspace that is not spanned by intact equations only, hence this case happens

rarely.

Algorithm 1 Decoding

```
1: download Z_1^*
2: let i = 1; Z = [ Z_1^* ]; S = {}
3: while |S| < l do
4:    let Z' = Z; S = {}
5:    repeat
6:       download Z_{i+1}^*
7:       let i = i + 1; Z = Z'; Z' = [ Z^T  Z_i^{*T} ]^T
8:    until rank(Z) = rank(Z')
9:    let λ = linsolve(xZ = Z_i)
10:   for j = 1 to length(λ) do
11:      if λ_j ≠ 0 then
12:         let S = S ∪ Z_{j,1...l+m}
13:      end if
14:   end for
15: end while
16: if |S| == l then
17:    return S
18: else
19:    S = subspace_search(Z)
20:    return S
21: end if
```

IV. Analysis

First, we investigate the performance of the decoding algorithm assuming no subspace search is required, and analyze the subspace search algorithm separately.

a) Security.: Let us assume for a moment that the collector downloads at most l adversarial blocks. Later we discuss the cases when this assumption does not hold. The adversary is successful, if decoding gives an erroneous result, which happens if the returned set S contains adversarial equations. From the operation of the algorithm it follows that this can only happen if the last received equation belongs to an l dimensional subspace spanned by l not only intact equations. Let us denote by $\langle I\rangle$ the subspace spanned by intact equations and $\langle R\rangle$ the subspace spanned by received equations at the collector. Figure 1 illustrates the situation.

First, we investigate the case when the last received equation is intact. The error probability in this case is the probability that this intact equation belongs to an l dimensional subspace of $\langle R\rangle$. We show that this probability is 0. The last received equation belongs to $\langle I\rangle$ for sure, thus decoding is erroneous, if the last received intact equation falls into the intersection of $\langle R\rangle$ and $\langle I\rangle$. This intersection is a subspace of the l dimensional subspace of intact equations. Consequently, the last received equation is a linear combination of the intact equations received, thus set S contains intact equations only. If S contains less than l equations the last equation is not innovative and is dropped, otherwise decoding is successful, hence the error probability is 0 when the last equation is intact.

We now consider the case, when the last received equation is not intact. Erroneous result can occur, if the last equation belongs to an l dimensional subspace of $\langle I\rangle \cup \langle R\rangle$. We investigate the probability of this event. Each l size subset of the received equations spans such a subspace. Recall that the number t' of adversarial equations received is at most l. If the adversary does not know intact equations, adversarial blocks can be treated as random vectors. In the worst case situation, l intact and $l-1$ adversarial blocks were received before the lth adversarial block arrives. The probability that the last adversarial blocks belongs to a specific l dimensional subspace is $1/q^m$. If all subspaces were independent, the probability that it does not belong to any would be $(1-1/q^m)^{\binom{r}{l}}$, where r denotes the number of received blocks. Hence an upper bound for the error probability is:

$$P_{\text{err}} \leq 1 - \left(1 - \frac{1}{q^m}\right)^{\binom{r}{l}}.$$

This error probability is practically very small. As an example, consider $q = 2^8$, that is each byte forms exactly one symbol, let the size m of the encoded data be small, e.g. $m = 64$, the number of data blocks $l = 100$, and assume that the adversary successfully inserts 99 adversarial blocks. In this case, the error probability is less than 2^{-300}.

If the adversary can attain intact equations before constructing adversarial blocks, she can enforce incorrect decoding by selecting adversarial blocks from a subspace that contains the subspace of eavesdropped equations. Even in this case, due to the randomness of coefficients, and to the random order of downloading, at least one dimension of the subspace is unknown, thus the error probability can not be larger than $1 - \left(1 - \frac{1}{q}\right)^{\binom{r}{l}}$. This can be made arbitrarily small by using a large field size q. Note that this error probability does not depend on the computational strength of the adversary.

b) Computational cost.: The computational cost of the decoding is solving two s.l.e.'s in $\mathrm{GF}(q)$. One s.l.e. is solved to identify intact equations (line 9), and another to reconstruct original data when a set of l intact equations is obtained. Note, that the computation of the rank does not require additional effort, because the Gaussian elimination can be performed on a newly received equation at once. If it results in a zero vector, the s.l.e. $\lambda[\mathbf{Z}] = [Z_i^*]$ is solved, otherwise the next equation is received and processed similarly. If not all received equations

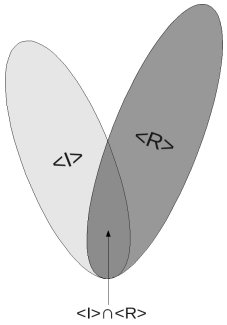

Fig. 1. Illustration of subspaces

are innovative, and thus line 9 of the decoding algorithm runs several times, the computational effort does not increase notably as there is no need to restart the Gaussian elimination after dropping the last equation.

c) Communication overhead.: The algorithm eventually stops when the $(l+1)$-st intact equation is received. It means one encoded block overhead, as l encoded blocks are needed in any case for successful decoding. If we take encoded blocks as the unit of coding, it can be proven that no attack detection or recovery from attack is possible using fewer blocks; in this sense our scheme is optimal.

Theorem 1. *Assuming the described encoding algorithm, no attack detection or recovery is possible using less than $l+1$ intact encoded blocks.*

Proof: Having l' intact encoding vectors, they are considered as rows of a matrix G of size $l \times l'$. This matrix determines an (l', l) linear code applied to the transmitted data $M = [M_1, M_2, \dots M_l]$, which is considered as a vector with l elements. It is known [16] that no error detection or correction is possible, if the Hamming distance of the code is less than 2. The linear code G (assuming independent rows) has Hamming distance more than 1 only if $l' > l$, that is at least $l+1$ intact encoding vectors are required to form such a matrix G that corresponds to a code with Hamming distance 2. ■

This property has importance, because in sensor networks the communication has high cost. The overhead of our scheme is thus one additional block, the size of this additional block is $(l+m)\log_2 q$.

d) Error probability.: The main constraint of our scheme is the limitation on the number of adversarial blocks the collector may download. If we allow the adversary to compromise more than l nodes, $l+1$ adversarial equations may be considered as an intact s.l.e. by the collector. In this case it can not be assured that the collector does not download more than l adversarial equations. This introduces an error probability

$$P_{error} \leq \frac{\binom{t}{l+1}}{\binom{n-t}{l+1} + \binom{t}{l+1}},$$

where n is the number of storage nodes, and t is the number of compromised nodes. There are at most $\binom{t}{l+1}$ seemingly intact s.l.e.'s ($l+1$ size set of adversarial equations indicated as intact when decoding) and exactly $\binom{n-t}{l+1}$ intact s.l.e's. The error probability is the ratio between the number of seemingly intact s.l.e.'s and the number of all intact and seemingly intact s.l.e.'s. This error probability is practically small, e.g. if $n = 50$, $l = 15$, $t = 20$, then $P_{error} \approx 3.3 \cdot 10^{-5}$.

This error probability can be reduced to 0 if the decoded data contains authentication information, or with the assumption $t < n/2$. In the latter case, the following method can be applied: It can be exploited that the data sent by the source satisfies at least $n/2$ equations, while adversarial data satisfies at most t equations in the system. Using the decoding algorithm the collector can find a seemingly intact set, and then substitutes the result into further equations until all together more than $n/2$ equations are satisfied. If all equations are downloaded, but the obtained result fails to satisfy $n/2$ of them, the result is erroneous and all equations that this result satisfies are surely adversarial. After dropping the adversarial equations, the process can restart until the correct data is obtained.

V. Subspace-search

e) Probability of occurrence.: We now discuss the cases when computing the subspace fails, hence search is required during decoding (the decoding algorithm runs line 19). This process is needed if set S contains more than l equations. This can happen, if the last received equation belongs to a more than l dimensional subspace spanned by the already received equations. As we have seen already, this can only happen, if the last received equation is adversarial. If d equations are already received, they span $\binom{d}{\delta}$ different δ dimensional subspaces. Applying the same considerations as above, the probability that an adversarial equation falls into a given δ dimensional subspace is

$$P_{\text{cl}} = \frac{1}{q^{l+m-\delta}}.$$

This probability can be made arbitrarily small by choosing a sufficiently large field.

A powerful adversary who can attain several intact equations can make this probability higher in the following way. She chooses a δ dimensional subspace from the $l+m$ dimensional space and selects the adversarial equations from that subspace. In this case, the dimension of the subspace that received equations span (the rank of matrix $\mathbf{Z}$) can not be higher than $l+\delta$. This attack has significance, if $\delta \geq l$ and the intersection of the subspace of intact equations and the subspace that adversarial equations span is not trivial, because, as the algorithm drops all further adversarial equations as irrelevant equations, the number of adversarial equations in $\mathbf{Z}$ may not increase above δ, if $\delta < l$. In the following, we assume $\delta \geq l$. If the dimension of the intersection is α, the rank eventually stops to increase after receiving $l+\delta-\alpha$ equations, because the dimension of the subspace that received equations span can not be larger than $l+\delta-\alpha$. This implies that if there are already $l+\delta-\alpha$ equations received, a newly received equation eventually falls into their spanned subspace, thus we cannot gain any information of which equations are intact by solving the s.l.e. of line 9, and a search of the subspace is required. It follows, that the attack can force the search with the highest probability if δ is small and α is large. Of course, $\delta \geq l$ and $\alpha \leq l$ must hold. If $\alpha = l$ and $\delta = l+1$ ($\delta = l$ would mean that the adversary inserts intact equations) and there is at least one adversarial equation received, subspace search is eventually required.

To perform such an attack the adversary has to be powerful enough to be able to eavesdrop at least α intact blocks and combine their contents with her own $\delta-\alpha$ equations to produce the adversarial equations.

Of course, δ is also bounded by the dimension of the space the equations are selected from, because even if all adversarial and intact equations are linearly independent, the

dimension of the spanned subspace can not be larger than $l+m$, that is $l+\delta-\alpha \leq l+m$. Consequently, if there are $l+m$ already received equations, the next received equation eventually falls into their spanned subspace without revealing any information about which equations are intact, and the search of the subspace is required. It means that if the number of adversarial equations in $\mathbf{Z}$ reaches m, intact equations can be found only by searching. However, the number of blocks is usually much less then the size of a block, thus $l \ll m$ holds, while the adversary is limited to insert l blocks.

Note, that the subspace search process does not effect the security property of the scheme. The adversary may enforce the collector to perform the subspace search, but she still can not compromise the decoded data. It influences only the computational effort the collector needs for successful decoding.

f) Algorithm.: An algorithm similar to the recovery process proposed in [13] can be applied for subspace search. The algorithm finds the $l+1$ intact equations by exhaustive search among all received equations. This is done by applying the same principle as before, that is $l+1$ equations that belong to the same l dimensional subspace are treated as intact equations. This can be done by checking all possible $l+1$ size subsets of the received d equations, whether the rank of the matrix $\mathbf{Z}$ of size $(l+1)\times(l+m)$ they form is l. If $\mathrm{rank}(\mathbf{Z})=l$, the search is ready, $\mathbf{Z}$ contains intact equations only. If no such subset is found, further equations are needed.

Algorithm 2 gives the pseudo-code of subspace search. Equations are received until the search succeeds (line 4). All possible $l+1$ size subsets of equations are checked whether they form an l dimensional subspace (lines 6-11). In each iteration only selections containing the last received equation are checked to avoid checking the same subset multiple times. If no such subset is found, a new iteration is performed.

Algorithm 2 Subspace-search

```
1: let i = number of already received equations
2: while true do
3:    let i = i+1
4:    download Z_i^*
5:    let Z' = [ Z^T,  Z_i^{*T} ]^T
6:    for every possible selection S' of l rows of Z do
7:       let S'_i = [ S'^T,  Z_i^{*T} ]^T
8:       if rank(S'_i) = l then
9:          let S = {all rows of S'}
10:         return S
11:      end if
12:   end for
13: end while
```

g) Searching complexity.: Due to the exhaustive search, the process is computationally expensive. Assume that when searching starts, $\mathbf{Z}$ has d rows, and the algorithm stops when D equations are received. The last iteration of the algorithm is not executed fully, on average the half of the subsets needs to be investigated until the intact set is found. From this it follows that subspace search requires

$$\frac{1}{2}\binom{D}{l}+\sum_{i=d}^{D-1}\binom{i}{l} \tag{1}$$

rank computations. The function of (1) is exponential in D, but there are strong reasons to assume that the subspace search problem is computationally hard, and consequently it is presumably theoretically impossible to find an efficient algorithm. The following consideration supports this intuition. Let us first formalize the problem:

Problem 1. *Given a matrix $\mathbf{Z}$ of size $d\times(l+m)$, with $d>\mathrm{rank}(\mathbf{Z})\geq l+1$, does there exist a vector x having exactly $l+1$ non-zero elements such that $x\mathbf{Z}=\mathbf{0}$?*

Here d corresponds to the number of received equations. If x exists, $l+1$ rows of $\mathbf{Z}$ are found that span an l dimensional subspace, in other words, these rows form a matrix with rank l, so the non-zero elements of x indicate intact equations in $\mathbf{Z}$. The problem refers to one iteration of the algorithm.

From the theory of linear algebra, it is known that the equation $x\mathbf{Z}=\mathbf{0}$ has many solutions, more precisely, x belongs to a subspace of dimension $\beta=d-\mathrm{rank}(\mathbf{Z})-1$. We consider a basis $\mathbf{B}$ of that subspace, consisting of β linearly independent vectors. Let $\mathbf{B}$ be a matrix of size $\beta\times d$. As $\mathbf{B}$ is a basis of the solution of the equation $\mathbf{Z}x=\mathbf{0}$, any linear combination of rows of $\mathbf{B}$ results in a vector x containing the coefficients for the equations, for which the corresponding linear combination results the zero vector. We are interested in the case when vector x has exactly $l+1$ non-zero elements, consequently Problem 1 is equivalent to the question: Does there exist a vector c such that $c\mathbf{B}$ has exactly $l+1$ nonzero elements?

This is the problem of finding the minimum distance of a linear code that is known to be NP-complete [17], [18], [19].

h) Accelerated computation.: Although the above reasoning implies that we can not hope to find a polynomial-time algorithm for the subspace search problem, a considerable acceleration of our proposed algorithm is possible.

The algorithm essentially checks for each possible $l+1$ size selection of rows of $\mathbf{Z}$, whether the formed matrix has full rank. The size of the matrix the computation is performed on is $(l+1)\times(l+m)$. Assuming a large field and a large m, this may take considerable time. It is possible to get the same result while working with smaller matrices in the following way.

We have seen, that we can equivalently investigate the number of zeros in the product $x=c\mathbf{B}$. To decide if it is possible to have $d-l-1$ zeros in x we check for each $d-l-1$ size selection of elements of x whether they can result all zero for any vector c. Formally, let π be a $d-l-1$ size selection of d elements. Let us select the columns of matrix $\mathbf{B}$ according to the same selection: $\mathbf{B}_\pi$. If the linear equation

$$c\mathbf{B}_\pi=0 \tag{2}$$

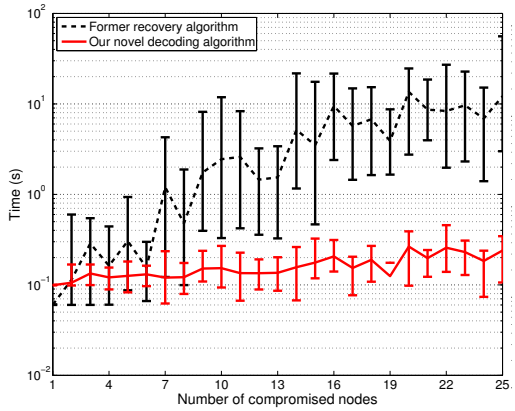

Fig. 2. Computation times of our decoding algorithm and the former recovery algorithm of [13]

has a non-trivial solution c_π, vector $x = c_\pi \mathbf{B}$ has at most $l+1$ non-zero elements. If equation (2) does not have non-trivial solution for any possible selection π, the iteration is not successful, further equations are needed. It is known from linear algebra that equation 2 has non-trivial solution, if $\mathrm{rank}(\mathbf{B}_\pi) < \beta$. This method requires the same number $\binom{d}{d-l-1} = \binom{d}{l+1}$ of matrix rank computations as before, but the size of matrix $\mathbf{B}_\pi$ is only $\beta \times (d-l-1)$, which substantially accelerates computation. The computation of matrix $\mathbf{B}$ is efficient, it is equivalent to solving a s.l.e.

VI. Performance

i) Computation.: Although the subspace search is computationally expensive, it does not affect the performance of the scheme when the adversary has limited capabilities, and despite the high complexity it remains in the domain of feasibility in many other practical cases as well. For comparison we implemented both the recovery algorithm presented in [13] and the novel decoding algorithm. The complexity of the novel algorithm is polynomial in the majority of the cases, and still remains in the domain of feasibility when the former algorithm becomes computationally expensive. Figure 2 shows an example result of the comparison. The computation was performed on a 2.6 GHz desktop computer and in Matlab environment. The parameter setting of the presented figure is $n = 100$, $l = 10$, $q = 2^8$, $m = 1000$, and a worst-case adversary was assumed: $\delta = l+1$, $\alpha = l$. It is clear from the figure that the novel algorithm operates with smaller deviation and remains feasible in the case of stronger attacks as well.

j) Communication.: We present a comparison of the communication overhead of our scheme and an alternate scheme using digital signatures. However, note that using digital signatures introduces additional overhead, that we do not consider here. For the digital signature scheme, we assume that the size in bits of the signature is s, and that on average p ($p > 1$) equations need to be downloaded to get an intact equation. Source nodes send their raw data to γ storage nodes out of the n available nodes, where $\gamma = 5(n/l)\ln(l)$ to ensure successful decoding [5]. The overall overhead is then $l\gamma s + lps$ bytes, where the first term corresponds to the overhead of transmitting raw data to the storage nodes, while the second term refers to the additionally downloaded bytes when reconstructing data.

Our encoding scheme requires one more intact equation to download, so the overhead is pL, where L is the bitlength of one packet: $L = (l+m)\log_2 q$. From this it follows, that the overhead of our coding scheme is less than that of digital signatures if $L < l\left(s + \frac{\gamma s}{p}\right)$. E.g. if $s = 40$ bytes, $n = 50$, $l = 15$, $p = 1.8$, our scheme performs better than the signature based solution if $L < 15600$ bytes. The typical size of sensor data is much smaller than this value. In larger systems this threshold further increases. Furthermore, in our scheme source nodes do not have any overhead, only the collector node does, but that node is often a powerful base station.

VII. Related work

Securing network coding based systems has two main approaches: information theoretic and cryptographic. A summary of network coding related security threats and solutions is available in [20]. First we show some information theoretic results. In [21] a rate-optimal code is introduced for securing network coding. In that scheme redundant bits are added to the sent data, and decoding relies either on a secure channel or on publicly known information. The authors of [21] assume a more general adversary, and allow adversarial blocks to be encoded together with intact ones. Our scheme is less general, instead it is tailored for the requirements of sensor networks. It does not need secure channels or public information, and contrary to the encoding of [21] it can be applied when there are more than one sources. The same holds for the error correction encoding scheme of [22]

The most similar approaches to the one presented here are [23] and [13]. However, [23] proposes only an attack detection scheme. We introduce a more efficient decoding algorithm than in [13], and give a more in-depth analysis of the problem.

The other possible approach is to apply cryptography for securing coding based systems. Cryptographic functions applied to encoded data need to have homomorphic property, which means that a hash value or a digital signature of an encoded block can be computed from the corresponding values computed on raw data. The verification of an encoded block is similar to the uncoded case. The advantage of cryptographic solutions is that blocks can be verified one by one, while our scheme can verify blocks only when decoding becomes possible.

In the case of homomorphic hash function, the collector computes the hash value of the encoded block from the hashes of raw data received through a secure channel from the sources. It also computes the hash of the received block, and if the two values match, the block is verified. Homomorphic hash functions are proposed in [24] and in [25]. The drawback of these schemes is that they are computationally expensive,

and they require a secure channel between the sources and the collector. We do not assume such secure channel.

For the homomorphic digital signatures [26], [27], [28], [29], [30] the source (or intermediate nodes) can construct a valid signature on the encoded block using the signatures of the raw data without accessing the private key. The verification of the signature is possible for the collector in the usual way, assuming a PKI is available. As we have seen, digital signatures require per block overhead and additional computations at both the source and the collector. Moreover, if there are multiple sources in the system - as in our applied model - all source have to use the same signing key, that introduces a difficult key management problem.

VIII. Summary

We proposed a novel decoding algorithm for coding based sensor storage that is able to correctly decode original data even if some encoded blocks are maliciously modified or inserted by an adversary. Our scheme does not require any modification at the sources or storage nodes, instead the collector uses one additional block by decoding. The decoding algorithm has linear complexity in most cases, however a powerful adversary has a chance to force more costly operations at the collector. We investigated the probability and the computational complexity of this latter case. By carefully selecting the parameters, the probability of computationally hard operations can be made sufficiently small in practice ($\leq \frac{1}{q^x}$, where q is the field size of network coding and x is at least 1).

The security analysis shows that, with proper parameters, the scheme is at least as secure as cryptographic approaches. Not using cryptography, our scheme does not rely on the existence of a PKI or a secure channel.

Our scheme is developed applicable in sensor networks in the first place, but other storage systems like peer-to-peer content sharing may also benefit from it. Performance analysis results confirms that the scheme is applicable in practice. This method is the first attempt that gives practical security solution for a multi-source network coding scenario.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° ICT-225186 (WSAN4CIP, www.wsan4cip.eu).

In addition, the work of Levente Buttyán was partially supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow." *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[3] C. Fragouli, J.-Y. L. Boudec, and J. Widmer, "Network coding: an instant primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, 2006.

[4] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *IEEE/ACM Trans. Netw.*, vol. 216, no. 3, pp. 497–510, 2008.

[5] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous access to distributed data in large-scale sensor networks through decentralized erasure codes," in *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN)*. Piscataway, NJ, USA: IEEE, 2005, p. 15.

[6] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 4, March 2005, pp. 2235–2245.

[7] A. G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Distributed data storage in sensor networks using decentralized erasure codes," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, November 2004.

[8] ——, "Distributed foutain codes for networked storage," in *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toulouse, France, 2006.

[9] ——, "Decentralized erasure codes for distributed networked storage," *IEEE Transactions on Information Theory Netw.*, vol. 52, pp. 2809–2816, Jun. 2006.

[10] S. Acedański, S. Deb, M. Médard, and R. Kötter, "How good is random linear coding based distributed networked storage," in *Proceedings of the Network Coding Workshop (NetCod)*, 2005.

[11] J. Dong, R. Curtmola, and C. Nita-Rotaru, "Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks," in *WiSec '09: Proceedings of the second ACM conference on Wireless network security*. New York, NY, USA: ACM, 2009, pp. 111–122.

[12] C. Gkantsidis and P. Rodriguez, "Cooperative security for network coding file distribution," in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2006.

[13] L. Buttyán, L. Czap, and I. Vajda, "Securing coding based distributed storage in wireless sensor networks," in *Proceedings of the IEEE Workshop on Wireless and Sensor Network Security (WSNS)*, Atlanta, USA, 2008.

[14] T. Ho, R. Kötter, M. Médard, D. R. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proceedings of the IEEE Information Theory Symposium (ISIT)*, June 2003.

[15] C. Cooper, "On the rank of random matrices," *Random Structures and Algorithms*, vol. 16, p. 2000, 2000.

[16] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 26, no. 2, pp. 147–161, Apr. 1950.

[17] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 384–386, May 1978.

[18] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1757–1766, Nov. 1997.

[19] N. Courtois, "Efficient zero-knowledge authentication based on a linear algebra problem minrank," in *Proceedings of ASIACRYPT*, 2001, pp. 402–421.

[20] L. Lima, J. Vilela, P. Oliveira, J. Barros, I. Filiz, X. Guo, J. Morton, B. Sturmfels, M. Mungan, J. Ramasco *et al.*, "Network Coding Security: Attacks and Countermeasures," 2008.

[21] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Médard, "Resilient network coding in the presence of byzantine adversaries," in *Proceedings of the Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Anchorage, Alaska, USA, 2007, pp. 616–624.

[22] S. Jaggi, M. Langberg, T. Ho, and M. Effros, "Correction of adversarial errors in networks," in *Proceedings of International Symposium on Information Theory (ISIT)*, 2005, pp. 1455–1459.

[23] T. Ho, B. Leong, R. Kötter, M. Médard, M. Effros, and D. R. Karger, "Byzantine modification detection in multicast networks using randomized network coding," in *Proceedings of the 2004 IEEE International Symposium on Information Theory (ISIT)*, Jun. 2004.

[24] M. N. Krohn, M. J. Freedman, and D. Mazieres, "On-the-fly verification of rateless erasure codes for efficient content distribution," in *Proceedings of 2004 IEEE Symposium on Security and Privacy*, 2004, pp. 226–240.

[25] D. L. G. Filho, P. Sérgio, and L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," IACR ePrint archive, 2006.
[26] K. E. Lauter, D. Charles X, and K. Jain, "Signatures for network coding," in *Proceedings of the 40th Annual Conference on Information Sciences and Systems (CISS '06)*, Mar. 2006.
[27] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient signature-based scheme for securing network coding against pollution attacks," in *Proceedings of the Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2008.
[28] F. Zhao, T. Kalker, M. Médard, and K. J. Han, "Signatures for content distribution with network coding," in *Proceedings of 2007 IEEE International Symposium on Information Theory (ISIT '07)*, Jun. 2007.
[29] D. Boneh, D. Freeman, J. Katz, and B. Waters, *Signing a Linear Subspace: Signature Schemes for Network Coding*. Springer, Mar. 2009, vol. 5443/2009, pp. 68–87.
[30] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, "An efficient signature-based scheme for securing XOR network coding against pollution attacks," in *Proceedings of the Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2009.