A Distributed Fault/Intrusion-Tolerant Sensor Data Storage Scheme Based on Network Coding and Homomorphic Fingerprinting

Rongfei Zeng, Yixin Jiang, Chuang Lin, *Senior Member, IEEE,*
Yanfei Fan, and Xuemin (Sherman) Shen, *Fellow, IEEE*

***Abstract*—Recently, distributed data storage has gained increasing popularity for reliable access to data through redundancy spread over unreliable nodes in wireless sensor networks (WSNs). However, without any protection to guarantee the data integrity and availability, the reliable data storage can not be achieved since sensor nodes are prone to various failures, and attackers may compromise sensor nodes to pollute or destroy the stored data. Therefore, how to design a robust sensor data storage scheme to efficiently guarantee the data integrity and availability becomes a critical issue for distributed sensor storage networks. In this paper, we propose a distributed fault/intrusion-tolerant data storage scheme based on network coding and homomorphic fingerprinting in volatile WSNs environments. For high data availability, the proposed scheme uses network coding to encode the source data and distribute encoded fragments with original data pieces. With secure, compact, and efficient homomorphic fingerprinting, our scheme can fast locate incorrect fragments and then initialize data maintenance. Extensive theoretical analysis and simulative results demonstrate the efficacy and efficiency of the proposed scheme.**

***Index Terms*—Distributed sensor data storage, network coding, homomorphic fingerprinting, data maintenance.**

I. Introduction

Distributed sensor data storage involves storing data reliably on multiple sensor nodes instead of a single source node, so that the original data can be further accessible to any authorized data collectors in wireless sensor networks (WSNs). Compared with the centralized data storage, distributed data storage is of special benefit for the reliable data management in WSNs, where individual sensors are vulnerable to failures and various attacks. Nowadays, distributed sensor data storage is universally applied to various scenarios. For instance, sensor networks are deployed in the remote environments where sensing nodes take measurements and store data on storage nodes over a long period of time. Any authorized data collector may appear at any location to retrieve the useful data from storage nodes [1], [2].

Despite the benefits of reliable data management, distributed data storage is susceptible to various threats to the data availability and integrity in WSNs. In practical, individual nodes are prone to random Byzantine failures, which mean some nodes may behave erroneously or fail to behave consistently [1], [6], [31]. In addition, malicious sensors may deliberately pollute or destroy the stored data by initiating various attacks (e.g., pollution attacks). All these phenomena result in the corruption of data availability and integrity, which correspondingly causes different data to be recovered from different subsets of fragments. Furthermore, if corrupted fragments are not located and updated, the limited resources, such as memory and energy, are abused to store these incorrect fragments or perform computations on them.

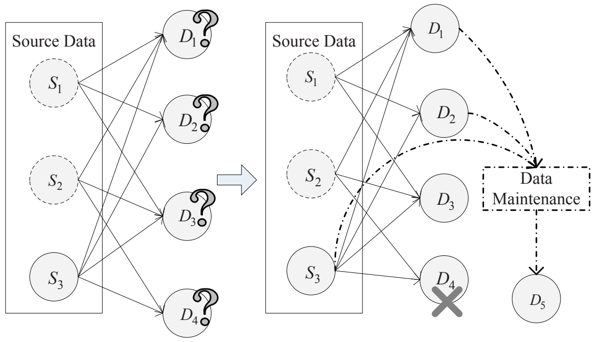


Fig. 1. Sensor data storage with integrity checking and data maintenance

Therefore, we should provide both the data availability and integrity guarantees for sensor data storage in WSNs. For the data integrity, Wang et al. argue that dynamic integrity verification should be utilized to assure data correctness over the period of storage [1]. Corrupted fragments need to be identified by dynamic integrity checking, and then data maintenance is performed to replace corrupted fragments for the future data reconstruction. On the other hand, dynamic integrity verification becomes meaningless without data maintenance. Only if dynamic integrity verification and data maintenance are both provided, can data collectors successfully obtain the stored data. In Fig. 1, we illustrate the integrity verification and data maintenance for data availability and integrity guarantees in sensor data storage. A source node generates four fragments from three original data pieces (or called source data pieces) using coding techniques and then stores encoded fragments at different storages nodes. During the lifetime of fragments, integrity checking is randomly performed to assure the data in-

R. Zeng and C. Lin are with the Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, P. R. China. Email:{zengrf, clin}@csnet1.cs.tsinghua.edu.cn.

Y. Jiang is with the EPRI, China Southern Power Grid Co. Ltd., Guangzhou, 510080, P. R. China. Email:yixin.tsinghua@gmail.com.

Y. Fan and X. Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada. Email:{yfan, xshen}bbcr.uwaterloo.ca.

Digital Object Indentifier 10.1109/TPDS.2011.294

1045-9219/11/$26.00 © 2011 IEEE

tegrity and locate corrupted fragments. In this example, D_4 is identified to be incorrect and then updated with $\{D_1, D_2, S_3\}$, as depicted by the dotted line. Data collectors can recover the original data using any three of $\{D_1, D_2, D_3, D_5\}$.

We first examine the existing data storage schemes. Most previous works focus on data distribution in P2P and wireless networks. For instance, erasure codes (e.g., Reed-Solomon codes [10]), fountain codes (e.g., LT codes [7]), growth codes [5], and priority linear codes [11] have been proposed in the recent years. In addition, some schemes are proposed to provide the integrity guarantee for network coding or erasure codes [26], [27], [29], [30], but the data maintenance issue is not addressed in these schemes. There are also some secure data storage schemes which include the secure data access approach with polynomial-key management, the adaptive polynomial-based data storage scheme, etc. [12]-[19], [31]. However, none of them achieves the requirements of data integrity, high availability, and efficiency at the same time. Recently, Wang et al. [1] propose a dependable and secure sensor data storage scheme with erasure codes and algebraic signature. In summary, distributed sensor data storage with both data integrity and availability guarantees has not been studied so far.

To tackle the above problem, we propose a distributed fault/intrusion-tolerant sensor data storage scheme based on network coding and homomorphic fingerprint. Unlike the traditional store-and-forward mechanism, network coding (NC) [23] allows intermediate nodes to actively code/mix the input packets. This novel information dissemination technique can achieve potential throughput improvement [24], transmission energy minimization [25], delay reduction [21] as well as robustness enhancement [20] in communication networks. Homomorphic fingerprintings proposed for the integrity checking can preserve the algebraic structure of network coding and allow verifiers to rapidly locate corrupted fragments. It has been demonstrated to be secure, compact, and efficient with low bandwidth and computational overheads [8]. Our scheme relies on network coding to encode the original data and distribute encoded fragments with original data pieces and homomorphic fingerprintings to storage nodes. Homomorphic fingerprintings are used to fast identify incorrect fragments and initialize data maintenance. During the data maintenance phase, network coding is also applied to generate alternative fragments using original data pieces and encoded fragments to guarantee the data availability.

The main contributions of this paper are two-fold: 1) To the best of our knowledge, this is the first research work to provide both data integrity assurance and availability guarantee for distributed data storage in WSNs. The data availability and integrity guarantees are achieved with network coding and homomorphic fingerprinting. Network coding provides an efficient and robust paradigm for data distribution and maintenance with low bandwidth and computational overheads, and homomorphic fingerprinting offers an efficient integrity verification mechanism to detect and filter polluted fragments; and 2) Extensive security analysis and performance evaluations demonstrate that our scheme provides an efficient fault/intrusion-tolerant approach for distributed data storage in the volatile WSNs environments, meaning that the proposed scheme is lightweight and resilient against Byzantine failures and malicious node compromising attacks.

The remainder of the paper is organized as follows: Section II introduces the system model, design goals, and preliminary techniques adopted in this paper. In Section III, we propose a distributed sensor data storage scheme with the data integrity and availability guarantees. Security analysis and performance evaluations are presented in Section IV. Section V surveys relate work, followed by the conclusions in Section VI.

II. System Model, Design Goals and Preliminaries

A. System Model

We consider the typical wireless data storage networks consisting of a relatively large number of sensor nodes, which are deployed strategically in the area of interest. Some sensors termed as source nodes can sense the environments, generate data blocks, and distribute them to the local nodes called storage nodes which will store the data for a period of time. The delay-sensitive scenarios are not the focus of this paper. Without loss of generality, we assume that sensor nodes are equipped with sufficient memory to store the sensed data. However, due to the constrained resources, sensors are assumed to have limited power supply and computational capability. In addition, sensor nodes are not tamper-proof, while some basic security mechanisms such as pairwise key and group key establishments [31] are already in place. Finally, each sensor is assumed to have a unique global ID.

B. Threat Model

Regarding the data availability and integrity, we consider a general and powerful threat model from two aspects: random Byzantine failures and malicious attacks. For Byzantine failures, some nodes may behave erroneously and fail to preserve the data consistency in WSNs. Moreover, a sensor node may be seeped away by ocean current or isolated by other catastrophic surroundings, thus some fragments may be lost or corrupted forever. For malicious attacks, hostile nodes always attempt to compromise and control as many critical storage nodes as possible. Then, the adversary can pollute all the data stored at the compromised node and monitor all its incoming and outgoing messages. In addition, attackers can further make use of compromised nodes to launch a variety of attacks. Note that if a sensor node were fully controlled by an adversary, it can successfully pass the integrity verification. There is no method to detect such attack with the existing schemes [1]. Finally, we assume that all the failures occur with moderate rate [1], [31].

C. Design Goals

The overall design goal is to enable authorized data collectors to correctly recover the original data in distributed sensor data storage. Specifically, we want to achieve the following subgoals: 1) *Data Availability*: Distributed data storage involves storing data reliably and enabling data collectors to retrieve the original information at any time, especially under

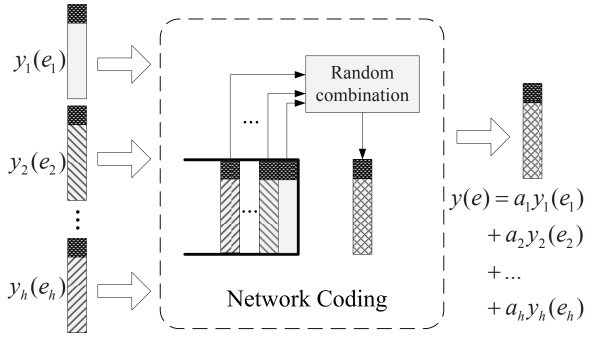


Fig. 2. Network coding at intermediate nodes

the circumstance of random failures and various attacks. Thus, assuring high data availability is one of the most essential and critical requirements for distributed data storage in WSNs; 2) *Data Integrity*: The integrity of fragments must be guaranteed over the period of storage. Otherwise, data collectors cannot correctly reconstruct the original data; and 3) *Efficiency*: The distributed sensor data storage scheme should be efficient in computational and communication overheads so that it is suitable for the inherent resource-constrained nature of WSNs.

D. Preliminaries

Network Coding: Network coding [20], [21] provides an efficient communication paradigm which allows intermediate nodes to mix input messages and make output messages be the mixture of input ones. As shown in Fig. 2, the symbol $y(e) \in \mathbb{F}_{2^q}$ carried on the outgoing edge e of node v can be computed as a linear combination of the symbols $y(e')$ on the incoming edge e', i.e., $y(e) = \sum_{e'} \beta_{e'}(e) y(e')$, where $\overline{\beta(e)} = [\beta_{e'}(e)]$ is called the local encoding vector (LEV), and the element $\beta_{e'}(e)$ are randomly chosen from $\mathbb{F}_{2^q}$ (e.g., q=8 or 16). By induction, we can denote $y(e)$ as a linear combination of the source symbols $x_1, \cdots, x_h$, i.e., $y(e) = \sum_{i=1}^{h} g_i(e) x_i$, where the vector $\overline{g(e)} = [g_1(e), \cdots, g_h(e)]$ is called the global encoding vector (GEV). When a sink node t receives h symbols $y(e_1), \cdots, y(e_h)$, these symbols can be expressed in terms of source symbols as

$$\begin{bmatrix} y(e_1) \\ \vdots \\ y(e_h) \end{bmatrix} = \begin{bmatrix} g_1(e_1) & \ldots & g_h(e_1) \\ \vdots & \ddots & \vdots \\ g_1(e_h) & \cdots & g_h(e_h) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix} = G \begin{bmatrix} x_1 \\ \vdots \\ x_h \end{bmatrix}, \quad (1)$$

where G is called the global encoding matrix (GEM), and the i-th row of the matrix G is the GEV associated with $y(e_i)$. Sink node t can further recover the h source symbols by solving these equations using Gaussian eliminations.

Homomorphic Fingerprinting: Homomorphic fingerprinting is first proposed by Hendricks et al. in [8]. Among various fingerprinting functions, division fingerprinting is widely applied to provide integrity checking due to its fast implementations. Let K be the set of fingerprinting key, the size of which equals the number of monic irreducible polynomials of degree γ, and let $P_{2^q} : K \rightarrow \mathbb{F}_{2^q}[x]$ be a deterministic algorithm that outputs monic irreducible polynomials of prime degree γ with their coefficients in $\mathbb{F}_{2^q}$. The polynomials are chosen with probabilities taken over the choice of input $r \in K$. Then, a fingerprinting function $fp(r, d) : K \times \mathbb{F}_{2^q}^{\delta} \rightarrow \mathbb{F}_{2^q}^{\gamma}$ can be defined as $fp(r, d) : p(x) \leftarrow P_{2^q}(r); return\ (d(x)\ mod\ p(x))$.

There are three important properties for division fingerprinting.

[Property 1] A division fingerprinting function $fp(r,\ d) : K \times \mathbb{F}_{2^q}^{\delta} \rightarrow \mathbb{F}_{2^q}^{\gamma}$ satisfies

$$max\ Pr\{fp(r, d) = fp(r, d') : r \leftarrow K\} \leq \varepsilon, \quad (2)$$

where $d, d' \in \mathbb{F}_{2^q}^{\delta}$, $d \neq d'$ and $\varepsilon \approx \delta / 2^{q\gamma}$. In other words, the probability that the fingerprintings of two different fragments collide is at most ε, with the random selection of r.

[Property 2] A division fingerprinting function $fp(r,\ d) : K \times \mathbb{F}_{2^q}^{\delta} \rightarrow \mathbb{F}_{2^q}^{\gamma}$ is homomorphic, i.e.,

$$fp(r, d) + fp(r, d') = fp(r, d + d'), \quad (3)$$

and

$$b \cdot fp(r, d) = fp(r, b \cdot d), \quad (4)$$

which can be achieved for any $r \in K$, $d, d' \in \mathbb{F}_{2^q}^{\delta}$, and $b \in \mathbb{F}_{2^q}$.

[Property 3] Let $encode$ be a linear network coding with coefficients $b_{ij} \in \mathbb{F}_{2^q} (1 \leq i \leq n, 1 \leq j \leq m)$, and then we can have $[d_1, d_2, \cdots, d_n] \leftarrow encode[s_1, s_2, \cdots, s_m]$. For the division fingerprinting function $fp(r, d) : K \times \mathbb{F}_{2^q}^{\delta} \rightarrow \mathbb{F}_{2^q}^{\gamma}$, the following equation holds:

$$fp(r, d_i) = encode[fp(r, s_1), fp(r, s_2), \cdots, fp(r, s_m)]. \quad (5)$$

E. Notations

In TABLE I, we present the symbols used in this paper.

TABLE I
NOTATIONS OF THIS PAPER

Notations	Descriptions
v, w	Regular sensor nodes
NB_v	The one-hop neighbor set of node v
S_i	The i-th partition of source data
$\overline{V_i}, E_i$	$\overline{V_i}$ is an encoding vector and E_i is the encoded block
$\{\cdot\}_k$	The encrypted data with key k
seq	Sequence number for integrity checking and retrieval
r_j	The key of homomorphic fingerprinting
$\overline{P_i}$	$\overline{P_i}$ is a signature vector stored at a sensor node
K_{UV}	The key shared between users and the data source
k_r, K_{gr}	k_r is a random session key and K_{gr} is a group key
$h(\cdot)$	The public hash function

III. The Proposed Sensor Data Storage Scheme

In this section, we propose a distributed fault/intrusion-tolerant data storage scheme based on network coding and homomorphic fingerprinting to guarantee both the data integrity and availability in WSNs.

A. The Basic Framework

For sensor data storage networks, we first propose a basic framework which consists of four phases:

1) *Data distribution*: After generating the original data, the source node disperses data blocks to the storage nodes in the data distribution phase. It is noteworthy that the delivered data blocks may be replicas or encoded fragments using various coding techniques (e.g., erasure codes and network coding). Moreover, some additional information might be delivered as well for the data availability and integrity guarantees.

2) *Integrity checking*: Any storage node could initialize dynamic integrity verification at any time, aiming to identify corrupted fragments from all data blocks and provide the integrity assurance over the lifetime of data blocks.

3) *Data maintenance*: Once a fragment is identified to be incorrect, the data maintenance must be performed to replace corrupted fragments with correct ones for the future data recovery.

4) *Data recovery*: When an authorized data collector needs to retrieve the original information, it is required to collect enough fragments to reconstruct the original data.

B. Data Distribution

In this subsection, network coding is used to encode the original data and distribute encoded fragments to storage nodes. To provide data integrity and availability guarantees, homomorphic fingerprintings and original data pieces are delivered with encoded fragments as well.

We first introduce two essential data structures: $flag_i$ and $list_i$. Let $flag_i$ be the number of times that the original data piece hold by node i is stored by other storage nodes. $list_i$, initially set to *NULL*, records the ID of storage node which contains the same original data piece with node i. Then, the procedure of our data distribution can be described as follows:

1) Source node v calculates the keyed hash of data and encrypts $<data, k_r, h(data, k_r)>$ with key K_{UV}, i.e., *DATA*= $\{data, h(data, k_r), k_r\}_{K_{UV}}$, where K_{UV} is shared between the source node v and authorized data collectors. It can be seen that the confidentiality of data is guaranteed by the encryption.

2) Let *DATA*=$< S_1, S_2, \cdots, S_m >$, and node v encodes *DATA* into $n (m \leq n \leq 2m)$ fragments with random linear network coding as

$$\begin{bmatrix} a_{11} & \dots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \times \begin{bmatrix} S_1 \\ \vdots \\ S_m \end{bmatrix} = \begin{bmatrix} E_1 \\ \vdots \\ E_n \end{bmatrix},$$
$$\overline{V_i} = [a_{i1}, a_{i2}, \cdots, a_{im}], \overline{D_i} = [\overline{V_i}, E_i], \tag{6}$$

where the elements of $\overline{V_i}$ are randomly picked from $\mathbb{F}_{2^q}$.

3) The source node v randomly selects n local storage nodes to form the set NB_v. For each neighbor $w \in NB_v$, node v randomly picks the key r_i from $\mathbb{F}_{2^q}$ and computes the homomorphic fingerprinting of each original data piece as

$$fp(r_i, S_j) : p(x) \leftarrow P_{2^q}(r_i);\ return(S_j(x)\ mod\ p(x)), \tag{7}$$

where $j = 1, \cdots, m$ and $i, k = 1, \cdots, n$. Then, we can get the fingerprinting vector

$$\overline{P_i} = [fp(r_i, S_1), fp(r_i, S_2), \cdots, fp(r_i, S_m)] \tag{8}$$

4) Let $a = \lceil n/2 \rceil$. Node v randomly selects $S_{z1}, S_{z2}, \cdots,$ S_{za} from the original data blocks $\{S_1, S_2, \cdots, S_m\}$, while other data blocks $S_j, j \notin \{z1, z2, \cdots, za\}$ are deleted. Then, node v randomly assigns $S_k, k = z1, z2, \cdots, za$ to storage nodes, guaranteeing that each original piece cannot be delivered to more than two storage nodes. In addition, node v needs to set $flag_i$ and $list_i$ according to the definitions.

5) Node v distributes $\{v, seq, \overline{D_i}, r_i, \overline{P_i}, S_{zi}, flag_i, list_i\}$ to each neighbor $w_i \in NB_v$ and then deletes all the data associated with seq.

In summary, the original data is divided into m blocks which are further encoded into n fragments using network coding. Subsequently, node v distributes each fragment with homomorphic fingerprintings and an original data piece to a storage node for the future data retrieve.

C. Integrity Verification

Fast fingerprinting verification can be initiated by any storage node at any time. Suppose that node w_i wants to verify the integrity of encoded fragments. It first broadcasts a challenge $\{w_i, seq, r_i\}_{K_{gr}}$. On receiving the challenge, each storage node $w_j \in NB_v$ computes the fingerprinting of E_j associated with seq and then responds an acknowledgement $\{w_j, seq, fp(r_i, E_j), \overline{V_j}\}_{K_{gr}}$. Once all the responses are obtained, the verifier w_i constructs a new vector as

$$[c_1, c_2, \cdots, c_m] = b_1\overline{V_1} + b_2\overline{V_2} + \cdots + b_n\overline{V_n}, \tag{9}$$

where $b_j, j = 1, \cdots, n$ are randomly picked from $\mathbb{F}_{2^q}$. Node w_i can verify the integrity of all the encoded fragments in batch by checking the following equation:

$$\begin{aligned} & c_1 fp(r_i, S_1) + c_2 fp(r_i, S_2) + \cdots + c_m fp(r_i, S_m) \\ = \ & b_1 fp(r_i, E_1) + b_2 fp(r_i, E_2) + \cdots + b_n fp(r_i, E_n). \end{aligned} \tag{10}$$

This is because

$$\begin{aligned} & c_1 fp(r_i, S_1) + c_2 fp(r_i, S_2) + \cdots + c_m fp(r_i, S_m) \\ = \ & fp(r_i, [c_1, c_2, \cdots, c_m][S_1, S_2, \cdots, S_m]^T) \\ = \ & fp\left(r_i, [b_1, \cdots, b_n]\begin{bmatrix} \overline{V_1} \\ \vdots \\ \overline{V_n} \end{bmatrix}[S_1, \cdots, S_m]^T\right) \\ = \ & fp\left(r_i, [b_1, \cdots, b_n]\begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}\left[\begin{bmatrix} S_1 \\ \vdots \\ S_m \end{bmatrix}\right]\right) \\ = \ & fp(r_i, [b_1, b_2, \cdots, b_n][E_1, E_2, \cdots, E_n]^T) \\ = \ & b_1 fp(r_i, E_1) + b_2 fp(r_i, E_2) + \cdots + b_n fp(r_i, E_n). \end{aligned} \tag{11}$$

If Eq. (10) holds, all the fragments $\overline{D_1}, \overline{D_2}, \cdots, \overline{D_n}$ are correct. Otherwise, some of them are polluted. We can further locate the corrupted fragments from Eq. (12) or using binary verification [26].

$$\begin{bmatrix} fp(r_i, E_1) \\ \vdots \\ fp(r_i, E_n) \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} fp(r_i, S_1) \\ \vdots \\ fp(r_i, S_m) \end{bmatrix} \tag{12}$$

It can be seen that our fingerprinting verification approach can not only efficiently check the integrity of multiple aggregated fragments in batch, but also rapidly locate corrupted fragments for the future data maintenance.

D. Data Maintenance

Once a fragment is identified to be incorrect, it is required to be updated to guarantee the data availability. In this subsection, we propose a novel data maintenance scheme to efficiently update corrupted fragments using encoded fragments and original data pieces. The procedure is detailed as follows:

1) Suppose that the fragment $\overline{D_x}$ stored at node x is incorrect. For a sensor node y with $list_y = x$, its $flag_y$ and $list_y$ are set to 0 and *NULL*, respectively.

2) When n is odd, node y picks another storage node w_i with $flag_i = 0$. In other words, node y should only choose the storage node holding the original data block which is unique in other storage nodes. The node y also needs to check the integrity of $\overline{D_i}$ stored at node w_i according to the Subsection C.

3) Subsequently, node w_i updates the incorrect fragment $\overline{D_x}$. Since node w_i has additional original piece S_k, and the fragment $\overline{D_i} = [\overline{V_i}, E_i]$ can be expressed as:

$$\begin{aligned} E_i &= a_{i1}S_1 + \cdots + a_{ik}S_k + \cdots + a_{im}S_m, \\ \overline{V_i} &= [a_{i1}, \cdots, a_{i(k-1)}, a_{ik}, a_{i(k+1)}, \cdots, a_{im}], \end{aligned} \tag{13}$$

a new fragment $\overline{D'_x} = [\overline{V'_x}, E'_x]$ can be constructed as

$$\begin{aligned} E'_i &= a_{i1}S_1 + \cdots + bS_k + \cdots + a_{im}S_m, \\ \overline{V'_i} &= [a_{i1}, \cdots, a_{i(k-1)}, b, a_{i(k+1)}, \cdots, a_{im}], \end{aligned} \tag{14}$$

where b is randomly picked from $\mathbb{F}_{2^q}$. In addition, we need to set $flag_i = 1, flag_x = 1, list_x = i, list_i = x$.

4) When n is even, there is no additional sensor node with $flag_i = 0$. Node y updates the incorrect fragment $\overline{D_x}$ according to the Step (3).

5) Finally, nodes w_i or y distribute $\{v, seq, \overline{D'_x}, r_i, \overline{P_i}, S_{zi}, flag_x, list_x\}_{K_{gr}}$ to node x and delete all the data associated with $\overline{D'_x}$.

E. Data Recovery

Considering the characteristics of data maintenance, we present an optimized data recovery algorithm to reduce the computation complexity. As shown in Algorithm 1 (line 04), the data blocks can be easily recovered in low computational overheads. Moreover, as the order of equations decreases, the computational overheads of solving these equations are significantly reduced. The complexity analysis of Algorithm 1 is given in Section IV.

F. A Concrete Example

In Fig. 3, we present a concrete example to illustrate the proposed scheme. For simplicity, some parameters, such as $\overline{V_i}$, the ID of source node, seq, $flag$, and $list$, are omitted in the example.

In the data distribution phase, the source node v generates the original data, divides it into four pieces, and then encodes them into five fragments $\{E_1, E_2, E_3, E_4, E_5\}$ using network coding. Subsequently, these fragments are individually distributed to storage nodes with the homomorphic fingerprintings and original pieces.

Algorithm 1 The Optimized Data Recovery

```
00 Function DataRecovery($Msg\ \overline{D_1}, \cdots, \overline{D_n}$)
01   do {
02       Result=$Select()$;
          // Select two fragments $\overline{D_i}$ and $\overline{D_j}$, which have
          // just one different coefficient. Assume that
          // $E_i = a_{i1}S_1 + \cdots + a_{ik}S_k + \cdots + a_{im}S_m$ and
          // $E_j = a_{i1}S_1 + \cdots + bS_k + \cdots + a_{im}S_m$
03       if (Result==$TRUE$) {
04         Get $S_k = (E_i - E_j)/(a_{ik} - b)$;
05         $count$++;
06         Delete $\overline{D_j}$;
07       }
08   }while(Result==$TRUE$)
09   $Gaussian\_elimination()$;
      // Solve $(n - count)$ linear equations
10   return ($Msg\ S_1, \cdots, S_m$);
```

Consider node w_5 wants to check the integrity of encoded fragments. It broadcasts a challenge $\{r_5\}_{K_{gr}}$ to storage nodes. On receiving the challenge, each node computes the fingerprinting and responds $\{fp(r_5, E_j)\}_{K_{gr}}$. Once the acknowledgements are obtained, node w_5 checks Eq. (10). Unfortunately, fragment E_4 is identified to be incorrect and should be updated for the data availability.

Another node w_3 initializes the data maintenance, since only w_3 contains the original piece S_4, and other original pieces are stored twice. Node w_3 constructs a new fragment with its own GEV. Specifically, w_3 chooses a random number 6 as the coefficient of S_4 and then generates an alternative fragment E_6.

When a data collector needs to retrieve the source data, it has to collect four fragments. If E_3 and E_6 are both obtained, the data collector can quickly obtain the original data piece S_4. Subsequently, the data collector can use fragments $\{E_1, E_2, E_5\}$ to recover other original pieces using Gaussian eliminations. Note that the more times the data maintenance is performed, the more efficiently the data collector reconstructs the original data.

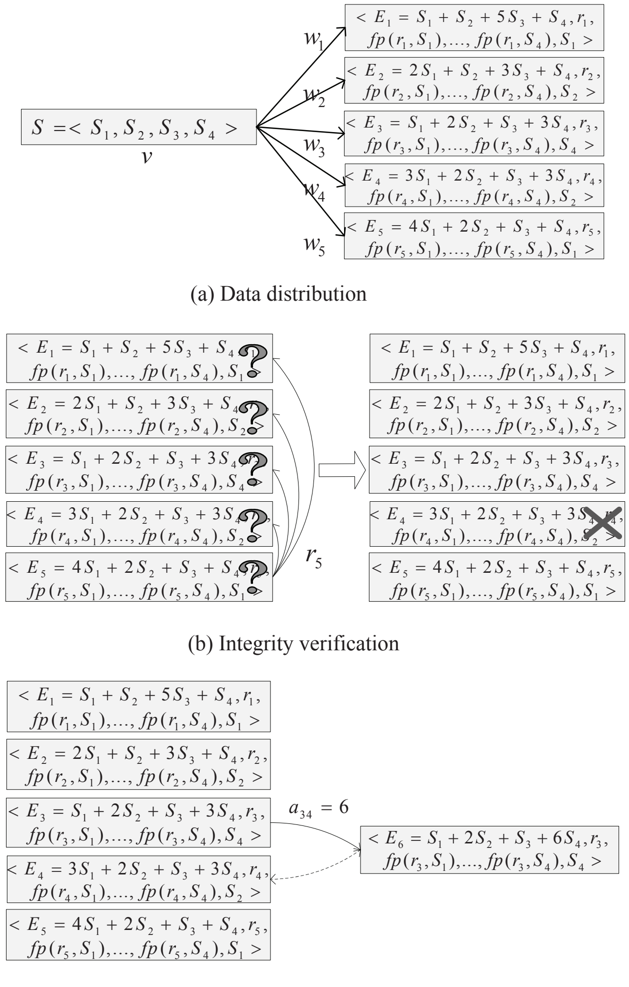
In summary, the proposed scheme provides a lightweight and efficient distributed sensor data storage with integrity and availability guarantees by exploiting network coding and homomorphic fingerprinting.

IV. Performance and Security Analysis

In this section, we present the performance evaluations and security analysis of the proposed scheme in terms of the recoverable probability, data integrity, data availability, communication overhead, computation cost, etc.

A. Recoverable Probability

During the data maintenance, a new fragment is generated to replace the incorrect one. Although the data maintenance


S =< S1, S2, S3, S4 >
v
w1
w2
w3
w4
w5
< E1 = S1 + S2 + 5S3 + S4, r1, fp(r1, S1),..., fp(r1, S4), S1 >
< E2 = 2S1 + S2 + 3S3 + S4, r2, fp(r2, S1),..., fp(r2, S4), S2 >
< E3 = S1 + 2S2 + S3 + 3S4, r3, fp(r3, S1),..., fp(r3, S4), S4 >
< E4 = 3S1 + 2S2 + S3 + 3S4, r4, fp(r4, S1),..., fp(r4, S4), S2 >
< E5 = 4S1 + 2S2 + S3 + S4, r5, fp(r5, S1),..., fp(r5, S4), S1 >
r5
a34 = 6
< E6 = S1 + 2S2 + S3 + 6S4, r3, fp(r3, S1),..., fp(r3, S4), S4 >

(a) Data distribution

(b) Integrity verification

(c) Data maintenance

Fig. 3. An instance of distributed sensor data storage

scheme is performed, we should recover the source data with the new GEM G. In the following, we analyze the recoverable probability, which is defined as the probability that source data blocks can be reconstructed with the GEM G, to demonstrate the feasibility of the proposed scheme.

[Theorem 1] For the one-time data maintenance that only one fragment is maintained, the recoverable probability becomes

$$P(G) = \frac{C_{n-2}^{m} + 2C_{n-1}^{m-1} + C_{n-2}^{m-2} p_{m-1}}{C_{n-2}^{m} + 2C_{n-1}^{m-1} + C_{n-2}^{m-2}} p_m$$

where $p_m = (\frac{1}{q})^{\frac{m(m+1)}{2}} \prod_{i=1}^{m}(q^i - 1)$.

[Proof] Without loss of generality, let the fragment $\overline{D_n}$ be updated by $\overline{D_{n-1}}$ during the data maintenance, and the new GEM G can be denoted as

$$G = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(n-1)1} & a_{(n-1)2} & \cdots & a_{(n-1)m} \\ a_{n1} & a_{n2} & \cdots & b \end{bmatrix} = \begin{bmatrix} \overline{a_1} \\ \vdots \\ \overline{a_{n-1}} \\ \overline{a_n} \end{bmatrix}. \quad (15)$$

Let $P(G)$ be the recoverable probability, then we have

$$P(G) = \frac{C_{n-2}^{m} P(A) + 2C_{n-1}^{m-1} P(B) + C_{n-2}^{m-2} P(C)}{C_{n-2}^{m} + 2C_{n-1}^{m-1} + C_{n-2}^{m-2}}. \quad (16)$$

In Eq. (16), A is a matrix with m vectors randomly picked from $\{\overline{a_1}, \overline{a_2}, \cdots, \overline{a_{n-2}}\}$. B is composed of $(m-1)$ vectors randomly selected from $\{\overline{a_1}, \overline{a_2}, \cdots, \overline{a_{n-2}}\}$ and one vector selected from $\{\overline{a_{n-1}}, \overline{a_n}\}$. C is a matrix with vectors $\{\overline{a_{n-1}}, \overline{a_n}\}$ and the other $(m-2)$ vectors randomly selected from $\{\overline{a_1}, \overline{a_2}, \cdots, \overline{a_{n-2}}\}$. Thus, the invertible probability of these three matrixes can be respectively considered as follows:

Case 1: For matrix A, all the elements are randomly selected from $GF(q)$ (In this section, $GF(q)$ denotes $\mathbb{F}_{2^{q'}}$, where $q = 2^{q'}$). Consider column i of the transposed matrix A^T as the image of the i-th basis vector in $GF^m(q)$. For the first column, we may choose any non-zero vector in $GF^m(q)$, meaning that there are $q^m - 1$ choices for the first column. For the second column, we may choose any vector that is not in the 1-dimensional subspace spanned by the first column. Since q elements exist in the subspace, there are $q^m - q$ choices. By induction, the number of possible invertible matrix is

$$(q^m - 1)(q^m - q) \cdots (q^m - q^{m-1}) = q^{\frac{m(m-1)}{2}} \prod_{i=1}^{m}(q^i - 1). \quad (17)$$

Therefore, the invertible probability of A is

$$P(A) = p_m = \left(\frac{1}{q}\right)^{\frac{m(m+1)}{2}} \prod_{i=1}^{m}(q^i - 1). \quad (18)$$

Case 2: Since all the elements of matrix B are randomly selected from $GF(q)$, this case is similar to Case 1, and we can have $P(A) = P(B)$.

Case 3: For the invertible probability of matrix C, the calculation method is similar to Case 1. Let the last column be spanned by the former $(m-1)$ columns, i.e.,

$$x_1 \overline{c_1}^T + x_2 \overline{c_2}^T + \cdots + x_{m-1} \overline{c_{m-1}}^T = \overline{c_m}^T. \quad (19)$$

Considering the first $(m-1)$ elements of each vector, we have

$$x_1 \begin{bmatrix} c_{11} \\ c_{12} \\ \vdots \\ c_{1(m-1)} \end{bmatrix} + \cdots + (x_{m-1} - 1) \begin{bmatrix} c_{(m-1)1} \\ c_{(m-1)2} \\ \vdots \\ c_{(m-1)(m-1)} \end{bmatrix} = 0. \quad (20)$$

If the vectors in Eq. (20) are linear independent, there is only one choice for $\overline{c_m}^T$ spanned by the vectors $\{\overline{c_1}^T, \overline{c_2}^T, \cdots, \overline{c_{m-1}}^T\}$. Otherwise, the number of possible $\overline{c_m}^T$ is q. Then, the average number of possible $\overline{c_m}^T$ is

$$E(N) = p_{m-1} \times 1 + (1 - p_{m-1}) \times q, \quad (21)$$

where p_{m-1} is the probability that vectors in Eq. (20) are linear independent. Thus, the number of possible invertible C^T is

$$\begin{aligned} & (q^m - 1) \cdots (q^m - q^{m-2})(q - E(N)) \\ & = q^{\frac{(m-2)(m-1)}{2}} (q - E(N)) \prod_{i=2}^{m}(q^i - 1). \end{aligned} \quad (22)$$

TABLE II
THE NUMERICAL RESULTS OF INVERTIBLE PROBABILITY

	$q = 2^8$ (one byte)	$q = 2^{16}$ (two bytes)	$q = 2^{24}$ (three bytes)	$q = 2^{32}$ (four bytes)	$q = 2^{40}$ (five bytes)
$m = 4$	0.995667	0.999983	1.000000	1.000000	1.000000
$m = 5$	0.995487	0.999982	1.000000	1.000000	1.000000
$m = 6$	0.995297	0.999982	1.000000	1.000000	1.000000
$m = 7$	0.995136	0.999981	1.000000	1.000000	1.000000
$m = 8$	0.994996	0.999981	1.000000	1.000000	1.000000

Then, we can get

$$P(C) = \left(\frac{1}{q}\right)^{\frac{m(m+1)}{2}} (q - E(N)) \prod_{i=2}^{m} (q^i - 1). \tag{23}$$

In addition, the relationship among $P(A)$, $P(B)$, and $P(C)$ is

$$P(C) = \frac{q - E(N)}{q - 1} P(A) = p_{m-1} P(A). \tag{24}$$

Thus, the recoverable probability is

$$P(G) = \frac{C_{n-2}^{m} + 2C_{n-1}^{m-1} + C_{n-2}^{m-2} p_{m-1}}{C_{n-2}^{m} + 2C_{n-1}^{m-1} + C_{n-2}^{m-2}} p_m,$$

where $p_m = (\frac{1}{q})^{\frac{m(m+1)}{2}} \prod_{i=1}^{m} (q^i - 1)$. ■

We present the numerical results of Theorem 1 in Table II. The following results are achieved with the parameter setting $n = 10$. It can be seen that the recoverable probability is almost not impacted by the data maintenance. Moreover, recoverable probability is an increase function of q and a decrease function of m.

Though Theorem 1 demonstrates that the data maintenance has little impact on the recoverable probability, we still consider the worst case that data maintenance is performed for infinite times and have the following corollary.

[Corollary 1] For the data maintenance performed for infinite times, the recoverable probability satisfies

$$\begin{cases} P(G) > W_1(\frac{n}{2}, m)/q^{\frac{n(m+1)}{2}} & n \text{ is even}, \\ P(G) > W_2(\frac{n+1}{2}, m)/q^{\frac{(n+1)(m+1)}{2}-1} & \text{Otherwise}. \end{cases}$$

where $W_1(x, y) = \prod_{i=1}^{x} W_1(1, y - x + i)^{C_{x-1}^{i-1}} \prod_{i=1}^{x} \prod_{j=0}^{x-i} Q(i, i + j + y - x)^{C_{x-i}^{j}}$, $Q(x, y) = (q^y - q^{2x-2})(q^{y-1} - q^{2x-2})(q - 1)q^{1-xy}$, $W_2(x, y) = W_1(x - 1, y)(q^y - q^{2(x-1)})$, $W_1(1, y) = (q^y - 1)(q - 1)$, and $W_2(1, y) = (q^y - 1)$.

[Proof] In the worst scenario, matrix M has the maximal number of paired vectors, among which each pair has only one different coefficient. The recoverable probability satisfies

$$P(G) > P(M) = \frac{W}{Enumeration\ of\ M}, \tag{25}$$

where W is the enumeration of invertible M. In the following, we focus on calculating W. From Theorem 1, we have the recurrence relation

$$\begin{aligned} W_n(m) \ = \ & W_{n-1}(m)\{q - [W_{n-1}(m-1)q^{-\frac{mn-2}{2}} + \\ & (1 - W_{n-1}(m-1)q^{-\frac{mn-2}{2}})q]\}, \end{aligned} \tag{26}$$

where n is even and denotes the number of vectors in the invertible M, and m denotes the dimension of vectors. In addition, we have $W_{n-1}(m-1) = W_{n-2}(m-1)(q^{m-2} - q^{n-2})$. Thus, we can further get

$$\begin{aligned} W_n(m) \ = \ & W_{n-2}(m)W_{n-2}(m-1) \times \\ & (q^{m-2} - q^{n-2})(q-1)q^{\frac{mn-2}{2}}. \end{aligned} \tag{27}$$

This recurrence relation can also be denoted as

$$W_1(x, y) = W_1(x-1, y)W_1(x-1, y-1)Q(x, y), \tag{28}$$

where $x = n/2$ and $y = m$. By deduction, we have

$$W_1(x,y) = \prod_{i=1}^{x} W_1(1, y-x+i)^{C_{x-1}^{i-1}} \prod_{i=1}^{x} \prod_{j=0}^{x-i} Q(i, i+j+y-x)^{C_{x-i}^{j}}$$

where $Q(x, y) = (q^y - q^{2x-2})(q^{y-1} - q^{2x-2})(q - 1)q^{1-xy}$. When n is odd, we can similarly get

$$W_2(x, y) = W_1(x - 1, y)(q^y - q^{2(x-1)}). \qquad ■$$

In the following, we also evaluate the proposed data maintenance scheme using the commutation system toolkit of Matlab. We analyze the recoverable probability with the increasing frequency of data maintenance. In each experiment, we perform 5000 trails, and the presented results are the average of these 5000 trails. In experiment 1, we study the recoverable probability with different finite field sizes and data maintenance times. From Fig. 4, we can see that the recoverable probability decreases with the increase of data maintenance times and approaches to a constant. However, the recoverable probability is still high when data maintenance is performed for infinite times. Moreover, the recoverable probability is high with small finite field size. When the size of finite field is $q = 2^8$, we can reconstruct the source data with probability one. In the second experiment, we study the impacts of parameter n on the recoverable probability. From Fig. 5 and Fig. 6, we can see $P(G) = 1$ with small n. Although the increase of n positively impacts the invertible probability, we do not need to set large n to achieve high recoverable probability, which also enhances the scalability of the proposed scheme. Therefore, the proposed scheme is practical and scalable in terms of data maintenance.

B. Data Availability

This subsection compares the proposed scheme with some influential works in terms of the data availability. We define the data availability Ava as the probability that the original data can be accessible to authorized data collectors, i.e.,

$$Ava = Pr\{The\ original\ data\ can\ be\ accessible\}.$$

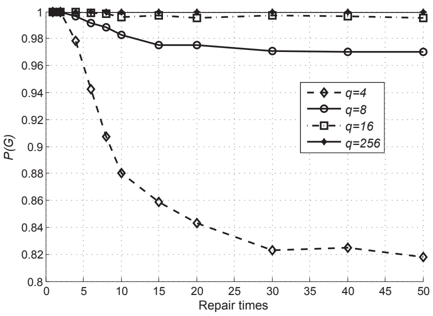


Fig. 4. The $P(G)$ with different field sizes and repair times

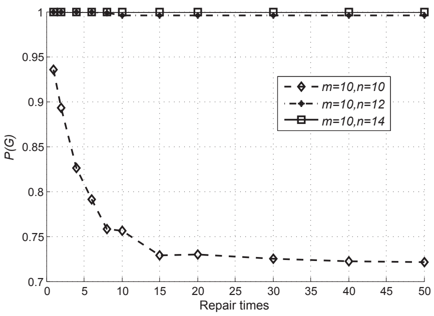


Fig. 5. The $P(G)$ with different repair times and $m = 10$

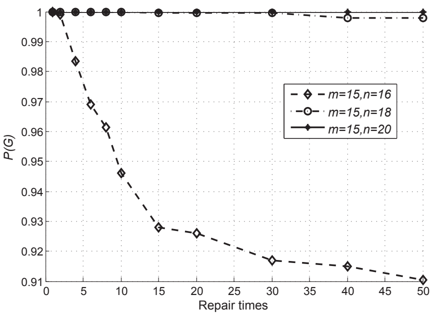


Fig. 6. The $P(G)$ with different repair times and $m = 15$

In [1], (m, n)-erasure codes and algebraic signature are applied to the data dispersion and integrity assurance, respectively. In this scheme, the availability can be denoted as:

$$Ava_0 = \sum_{i=m}^{n} C_n^i (1 - p_{att})^i p_{att}^{n-i}, \qquad (29)$$

where p_{att} is the probability of node failures caused by Byzantine failures and node compromising attacks. In the hybrid strategy [4], erasure codes are used to encode and distribute fragments, and the original data is maintained by a storage node; thus, the availability can be computed as

$$Ava_1 = (1 - p_{att}) + p_{att}(\sum_{i=m}^{n} C_n^i (1 - p_{att})^i p_{att}^{n-i}). \qquad (30)$$

In our scheme, the data availability is a constant, i.e., $Ava_2 = 1$, because corrupted fragments can always be updated from correct ones and source data pieces.

Fig. 7 shows the numerical results of these schemes with parameter settings: $m = 10$ and $n = 7$. The data availabilities of erasure codes [1] and the hybrid strategy [4] decrease as the probability of node failures increases. In addition, the hybrid strategy has better performance than erasure codes [1] due to the maintenance of the original data at a storage node. From Fig. 7, it is fair to say that our scheme outperforms the other two counterparts in terms of the data availability.

C. Data Integrity

In the proposed scheme, the data integrity is assured by homomorphic fingerprintings which can efficiently filter incorrect fragments with high probability. However, corrupted fragments may still pass fingerprinting verifications in the extreme situations defined as false negative. In this subsection, we analyze the probability of false negative.

Two cases lead to the false negative. The fingerprinting of the corrupted fragment may be identical with that of correct one. Furthermore, although the fingerprintings of corrupted fragments are changed, the expression $b_1 fp(r_i, E_1) + b_2 fp(r_i, E_2) + \cdots + b_n fp(r_i, E_n)$ may still remain the same. Thus, we denote the probability of false negative as $Pr = Pr_1 + Pr_2$, where Pr_1 and Pr_2 denote the probability of false negative in the above cases, respectively.

Case 1: Let the number of compromised storage node be n_c. Similar to [1], the probability of false negative can be denoted as

$$Pr_1 = \frac{\sum_{i=1}^{n_c} C_{n_c}^i \varepsilon^i}{\sum_{i=1}^{n_c} C_{n_c}^i} = \frac{(1 + \varepsilon)^{n_c} - 1}{2^{n_c} - 1}, \qquad (31)$$

where ε is the probability that fingerprintings of two different fragments collide, and $\varepsilon \approx \delta / 2^{q\gamma}$ for finite field $\mathbb{F}_{2^q}$.

Case 2: In this case, the fingerprintings of corrupted fragments are changed, however, the right hand of equal mark in Eq. (10) still remains the same. During the integrity verification, we randomly choose a vector $[b_1, b_2, \cdots, b_n]$ and check Eq. (10). In addition, the probability of false negative decreases with the increase of checking times. Let the number of checking be n_v. If $n_v \geq n_c$, the false negative will never happen. Otherwise, the probability of false negative is about

$$Pr_2 = (1 - Pr_1) \sum_{i=n_v+1}^{n_c} \frac{C_{n_c}^i \cdot (2^q)^{2i-n_v}}{C_{n_c}^2 + \cdots + C_{n_c}^{n_c}}. \qquad (32)$$

D. Communication Overhead

This subsection analyzes the communication overhead of the proposed scheme. We assume that ID and seq are denoted with 1 byte. In addition, we also ignore the influence of encryption on the size of plaintext, which depends on particular encryption algorithms [1]. In our scheme, the communication overheads of data distribution, integrity checking, and data maintenance can be denoted as $Bw_d = Size(D_i) + Size(S_i) + Size(P_i) + Size(r_i) + 3$, $Bw_{ic} = 2 + Size(r_i) + (n - 1)(2 + Size(fp(r_i, E_j)) + Size(V_j))$, and $Bw_m = Bw_d$, respectively. In practice, the overheads of encoding coefficients and the constants can be omitted [21], [22]. Then, the communication overheads can be simplified to $Bw_d \approx Size(D_i) + Size(S_i) + m \cdot Size(fp(r_i, S_j))$ and $Bw_{ic} \approx Size(r_i) + (n - 1)(Size(fp(r_i, E_j)))$. The main parts of communication overheads are $Size(D_i)$, $Size(S_i)$, $Size(r_i)$, and $Size(fp(r_i, E_j)$. For the latter two elements, $Size(fp(r_i, E_j) = \gamma \ll Size(S_i)$ and $Size(r_i) \ll Size(S_i)$ according to the definition in Section II. The former two elements approximately equal to $1/m$ of the original data size, which is acceptable in the resource-constrained WSNs. Therefore, our scheme is feasible for WSNs in terms of communication overheads.

In Wang et al.'s scheme [1], the communication overheads of data distribution and dynamic integrity verification can

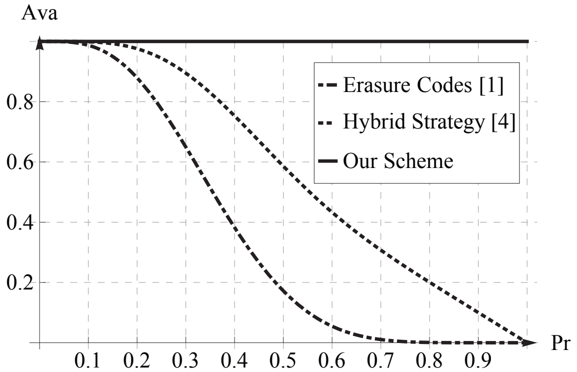


Fig. 7. The comparison of data availability

be separately denoted as $Bw'_d \approx Size(S_i) + Size(P'_i) \approx 2Size(S_i)$ and $Bw'_{ic} \approx (n-1)\nu(nSize(P'_i))$, where ν is the number of checking for each dynamic integrity verification. Compared with Wang et al.'s scheme, our scheme has the comparative communication overheads in the data distribution and integrity checking.

E. Computation Cost

As a critical component of the proposed scheme, homomorphic fingerprintings can be computed with add operations and table lookups, contributing little computational overhead to sensor nodes. In [8], homomorphic fingerprintings gain $4x$ throughputs per second, compared with Gladman's implementation of SHA-1. This improvement is achieved by sacrificing the memory usage which is not a primary concern in sensor data storage. Therefore, homomorphic fingerprinting is efficient and lightweight.

We also present the total computational overheads of the proposed scheme. The notations of cryptographic operations are summarized in Table III. In the data distribution, the source node requires a hashing and a symmetric-key encryption, and it also computes the encoded fragments and homomorphic fingerprintings. Thus, the computation cost at the source node is $Hash^1 + SymEncr^2 + FPop^m + NCop_m^n$, and the computation cost at each storage node is $SymDecr^1$. In the dynamic integrity checking, the verifier requires encrypting the challenge, decrypting the responses, and checking whether Eq. (10) is satisfied. Each storage node needs to generate the fingerprinting. Consequently, the computational overheads at the verifier and each storage node are separately $SymEncr^1 + IntCh^1 + SymDecr^{n-1}$ and $FPop^1 + SymEncr^1 + SymDecr^1$. For the data maintenance, the computational overhead for the involved node is $SymEncr^1$ or $SymDecr^1$.

TABLE III
THE NOTATIONS OF CRYPTOGRAPHIC OPERATIONS

Notations	Explanations
$Hash^t$	t hash operations
$SymEncr^t$	t symmetric-key encryption operations
$SymDecr^t$	t symmetric-key decryption operations
$FPop^t$	t fingerprinting generations
$NCop_m^n$	Encoding m blocks into n fragments using NC
$IntCh^t$	t checkings of Eq. (10)

F. Complexity Analysis of Algorithm 1

In this subsection, we evaluate Algorithm 1 from the aspect of computation complexity, since this algorithm does not require much additional memory and the analysis of space complexity is neglected in this paper. For the computation, the entire algorithm can be divided into two parts. The first part (line 01-08) is to find pairwise vectors and to compute the corresponding original data pieces. The computation complexity is $O(m(n^2-n)/2)$ in terms of comparison operations. The computation cost of this part can be neglected, comparing with multiplication operations in the second part. The second part (line 09) is to get other source data pieces by solving linear equations, and the computation complexity is $O(C_{n/2}^{m-n/2}(m-n/2)^3)$ in terms of multiplication operations. Algorithm 1 reduces many computation costs, comparing to the traditional data reconstruction algorithm with the computation complexity of $O(C_n^m m^3)$. Therefore, Algorithm 1 is efficient and scalable.

G. Scalability

We discuss the scalability of the proposed scheme in this subsection. In terms of performance, data maintenance has almost no impact on the recoverable probability, and we can always reconstruct the source data blocks after the data maintenance. In addition, this performance can be achieved without large $n\,(m \leq n \leq 2m)$. In terms of costs, both the computation costs and communication overheads are acceptable for the resource-constraint WSNs. Moreover, the computation complexity of Algorithm 1 is drastically reduced by the proposed scheme. Finally, each source data only involves n storage nodes instead of the total storage nodes, thus it is much scalable to the large-scale networks. Therefore, the proposed scheme satisfies the requirement of scalability for the distributed sensor data storage networks.

H. Comparisons with Other Schemes

We compare our scheme with other impressive works qualitatively, as shown in TABLE IV. The schemes in [2], [4], [26] provides various techniques (e.g., erasure codes and regenerating codes) in data distribution to optimize the performance such as memory usage, repair bandwidth, etc. However, the data availability and integrity guarantees are not included. Wang et al. use erasure codes and algebraic signature to provide lightweight integrity assurance in [1]. Compared with these schemes, our scheme provides a distributed fault/intrusion-tolerant sensor data storage with both data availability assurance and integrity guarantee.

V. RELATED WORK

Efficient data distribution has been extensively studied in the recent years. Erasure codes (e.g., Reed-Solomon codes [10]) are widely applied to distributed data storage in P2P and WSNs, since erasure codes can achieve much higher reliability compared to the replication scheme with the same number of storage nodes [1]. Fountain codes, such as LT codes [7] and Tornado codes [32], can efficiently reduce the computation costs of encoding and decoding. Kamra et al. [5] propose growth codes to increase the data persistence in WSNs, so that the sensed data is more likely to reach the sink node. Lin et al. [11] propose an efficient distributed storage scheme for WSNs. In this scheme, priority random linear codes are introduced to have different data in different priorities, making critical data have higher opportunity to survive node failures than the data of less importance.

In addition, many literatures focus on securing data distribution. Yu et al. propose a RSA-based homomorphic signature scheme to provide the integrity guarantee for network coding [26]. However, Yun et al. demonstrate that Yu et al.'s scheme does not satisfy the required homomorphic property in [33]. Some schemes (e.g., the homomorphic hashing scheme, the MAC-based scheme, the dynamic-identity based signature scheme, and the efficient subspace authentication [27], [28], [29], [30]) have been proposed against pollution attacks for network coding and erasure codes. Y. Fan et al. [34] and P. Zhang et al. [35] propose the novel schemes to secure network coding against traffic analysis attacks and eavesdropping attacks, respectively. But these schemes can not be directly applied to sensor data storage, since they only focus on the integrity and privacy assurances in the data distribution, not including the data availability guarantee.

In distributed data storage networks, the data maintenance is also a critical topic. Rodrigues et al. [4] compare the traditional data replication with erasure codes in the bandwidth-reliability tradeoff space and propose a hybrid scheme which can efficiently update corrupted data blocks with less bandwidth requirements. Dimakis et al. [3] theoretically introduce regenerating codes to improve the efficiency of data repair using large fragments. Pietro et al. [13] consider the data availability problem in sensor data storage and further propose an efficient data management scheme. As an elegant solution to ubiquitous data distribution and collection, Wang et al. [22] propose the partial network coding technique to enable efficient storage replacement for distributed sensor data storage. All the schemes mainly focus on the data maintenance of corrupted fragments or bandwidth requirements. How to detect incorrect fragments is not addressed, meaning that the data integrity is not guaranteed in these schemes.

Several schemes are proposed to secure distributed data storage in WSNs [12]-[19]. Based on the polynomial-based key management, Zhang et al. [16] present a secure data access approach, where sinks can retrieve data following a fixed routing. Zeng et al. [31] apply network coding and matrix decomposition in the key pre-distribution to secure data management in vulnerable sensor data storage networks. In [17], the combination of XOR secret sharing and replication are introduced to build a secure and fault-tolerant data storage system in collaborative working environments. Wang et al. [1] present an impressive distributed data storage scheme with the integrity assurance in WSNs. They utilize erasure codes and algebraic signature to achieve the integrity assurance.

In summary, sensor data storage with both data integrity assurance and availability guarantee has been overlooked in the existing schemes, and it is critical to consider network coding and homomorphic fingerprinting for designing a distributed fault/intrusion-tolerant data storage scheme in WSNs.

TABLE IV
THE COMPARISONS OF DISTRIBUTED DATA STORAGE SCHEMES

	Ours	[1]	[2]	[4]	[8]	[22]
Integrity verification	√	√			√	
Error location	√					
Data maintenance	√		√	√		√
Efficiency	√	√				√

VI. CONCLUSION

In this paper, we have studied the problem of data availability and integrity guarantees in distributed sensor data storage networks. Based on networking coding and homomorphic fingerprintings, we have proposed a fault/intrusion-tolerant data storage scheme for volatile WSNs environments, aiming to provide both high data availability and integrity assurances. Extensive theoretical analysis and simulative evaluations demonstrate that the proposed scheme outperforms other counterparts.

When the rate of node failures and data corruptions becomes extremely high, an efficient distributed error correction algorithm should be designed. Moreover, we can utilize the spatial and temporal information redundancy for data maintenance in WSNs.

ACKNOWLEDGMENT

This work is supported by the National Grand Fundamental Research 973 Program of China (No. 2010CB328105, No. 2011CB302703, and No. 2009CB320504), the National Natural Science Foundation of China (No. 60932003, No. 61071065, No. 60970101, No. 60872055, No. 61020106002, and No. 60834004), and the National Natural Science Foundation of China A3 Program (No. 61161140320).

REFERENCES

[1] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and secure sensor data storage with dynamic integrity and assurance," in *Proc. of IEEE INFOCOM*, 2009.

[2] A. G. Dimakis and K. Ramchandran, "Network coding for distributed storage in wireless networks," *Networked Sensing Information and Control, Signals and Communication Series, V. Saligrama, Springer Verlag*, 2008.

[3] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage system," in *Proc. of IEEE INFOCOM*, 2007.

[4] R. Rodrigues and B. Liskov, "High availability in DHTs: erasure coding vs. replication," in *Proc. of IEEE IPTPS*, 2005.

[5] A. Kamra and V. Misra, "Growth codes: maximizing sensor network data persistence," in *Proc. of ACM SIGCOMM*, 2006.

[6] Y. Lin, B. Liang, and B. Li, "Data persisitence in large-scale sensor networks with decentralized fountain codes," in *Proc. of IEEE INFOCOM*, 2007.

[7] M. Luby, "LT codes," in *Proc. of IEEE Foundations of Computer Science (FOCS)*, 2002.

[8] J. Hendricks, G. R. Ganger, and M. K. Reiter, "Verifying distributed erasure-coded data," in *Proc. of ACM PODC*, 2007.

[9] A. Z. Broder, "Some applications of Robin's fingerprinting method," *Sequences II: Methods in Communications, Security, and Computer Science*, pp. 143-152, 1993.

[10] I. S. Reed and G. Solomon, “Rolynomial codes over certain finite fields,” *Journal of SIAM*, 1960.

[11] Y. Lin, B. Liang, and B. Li, “Priority random linera codes in distributed storage systems,” in *Proc. of IEEE ICDCS*, 2007.

[12] J. Girao, D. Westhoff, E. Mykletun, and T. Araki, “Tinypeds: tiny persistent encrypted data storage in asynchronous wireless sensor networks,” *Elsevier Ad Hoc Networks*, vol. 5, no. 7, pp. 1073-1089, 2007.

[13] R. D. Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, “Catch me (if you can): data survival in unattended sensor networks,” in *Proc. of IEEE PerCom*, 2008.

[14] D. Ma and G. Tsudik, “Forward-secure sequential aggregate authentication,” in *Proc. of IEEE Symposium on Security and Privacy*, 2007.

[15] S. Chessa, R. D. Pietro, and P. Maestrini, “Dependable and secure data storage in wireless ad hoc networks: an assessment of DS2,” in *Proc. of WONS*, 2004.

[16] W. Zhang, H. Song, S. Zhu, and G. Cao, “Least privilege and privilege deprivation: towards tolerating mobile sink compromises in wireless sensor networks,” in *Proc. of ACM MOBIHOC*, 2005.

[17] A. Subbiah and D. M. Blough, “An approach for fault tolerant and secure data storage in collaborative work environments,” in *Proc. of 2005 International Workshop on Storage Security and Survivability*, 2005.

[18] N. Subramanian, C. Yang, and W. Zhang, “Securing distributed data storage and retrieval in sensor networks,” in *Proc. of IEEE PerCom*, 2007.

[19] M. O. Rabin, “Efficient dispersal of information for security, load balancing, and fault tolerance,” *Journal of the ACM*, vol. 36, no. 2, pp. 335-348, 1989.

[20] C. Fragouli, J. Y. Boudec, and J. Widmer, “Network coding: an instant primer,” *ACM SIGCOMM Communication Review*, 2008.

[21] P. A. Chou and Y. Wu, “Network coding for the Internet and wireless networks,” *IEEE Signal Processing Magazine*, 2007.

[22] D. Wang, Q. Zhang, and J. Liu, “Partial network coding: theory and application for continuous sensor data collection,” in *Proc. of IEEE IWQoS*, 2006.

[23] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, “Network information flow,” *IEEE Trans. on Information Theory*, vol. 46, no. 4, pp. 1204-1216, 2000.

[24] Z. Li, B. Li, and L. C. Lau, “On achieving maximum multicast throughput in undirected networks,” *IEEE Trans. on Information Theory*, vol. 52, no. 6, pp. 2467-2485, 2006.

[25] Y. Wu, P. Chou, and S. Kung, “Minimum-energy multicast in mobile ad hoc networks using network coding,” *IEEE Trans. on Communications*, vol. 53, no. 11, pp. 1906-1918, 2005.

[26] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, “An efficient signature-based scheme for securing network coding against pollution attacks,” in *Proc. of IEEE INFOCOM*, 2008.

[27] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan, “An efficient scheme for securing xor network coding against pollution attacks,” in *Proc. of IEEE INFOCOM*, 2009.

[28] S. Agrawal and D. Boneh, “Homomorphic MACs: MAC-based integrity for network coding,” *Springer LNCS*, 2009.

[29] P. Zhang, Y. Jiang, C. Lin, H. Yao, A. Wasef, and X. Shen, “Padding for Orthogonality: efficient subspace authentication for network coding,” in *Proc. of IEEE INFOCOM*, 2011.

[30] Y. Jiang, H. Zhu, M. Shi, X. Shen, and C. Lin, “An efficient dynamic-identity based signature scheme for secure network coding,” *Elsevier Ad Hoc Networks*, vol. 54, no. 1, pp. 28-40, 2010.

[31] R. Zeng, Y. Jiang, C. Lin, Y. Fan, and X. Shen, “A scalable and robust key pre-distribution scheme with network coding for sensor data storage networks,” *ELSEVIER Computer Networks*, vol. 55, no. 10, pp. 2534-2544, 2011

[32] M. Luby, M. Mizenmacher, M. A. Ahokrollahi, and D. Spielman, “Efficient erasure correcting codes,” *IEEE Trans. on Informantion Theory*, vol. 47, pp. 569-584, 2001.

[33] A. Yun, J. Cheon, and Y. Kim, “On homomorphic signatures for network coding,” *IEEE Transaction on Computers*, vol. 59, no. 9, pp. 1295-1296, 2010.

[34] Y. Fan, Y. Jiang, H. Zhu, and X. Shen, “An efficient privacy-preserving scheme against traffic analysis attacks in network coding,” in *Proc. of IEEE INFOCOM*, 2009.

[35] P. Zhang, Y. Jiang, C. Lin, Y. Fan, and X. Shen, “P-Coding: secure network coding against eavesdropping attacks,” in *Proc. of IEEE INFOCOM*, 2010.

Rongfei Zeng received a B.S. degree (2002) from Northeastern University (China) in computer science and technology. Currently, he is a Ph.D. candidate at Computer Science Department, Tsinghua University, China. His current research interests include wireless network security, smart grid, and performance evaluations.

Yixin Jiang is an associate professor in Tsinghua University. In 2007-2009, he was a Post Doctorial Fellow with University of Waterloo. He received the Ph.D. degree (2006) from Department of Computer Science and Technology, Tsinghua University, China. In 2005, he was a Visiting Scholar with the Department of Computer Sciences, Hong Kong Baptist University. In 2009, he was a Visiting Scholar with the Department of Computer Science and Engineering, the Chinese University of Hong Kong. He has served as the Technical Program Committee (TPC) member for main network conferences, such as IEEE ICCCN, IEEE GLOBECOM, IEEE ICC, IEEE WCNC, etc. He is a member of IEEE CISTC. His current research interests include network coding, clouding computing, security and privacy in wireless communication and mobile computing. He has received Excellent Backbone Talents Fund Award, Outstanding Doctoral Graduate Award, and Excellent Doctoral Thesis Award of Tsinghua University.

Chuang Lin (IEEE SM’04) is a professor of the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He received the Ph.D. degree in Computer Science from the Tsinghua University in 1994. His current research interests include computer networks, performance evaluation, network security analysis, and Petri net theory and its applications. He has published more than 300 papers in research journals and IEEE conference proceedings in these areas and has published three books. Professor Lin is a member of ACM Council, a senior member of the IEEE and the Chinese Delegate in TC6 of IFIP. He serves as the Technical Program Vice Chair, the 10th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS 2004); the General Chair, ACM SIGCOMM Asia workshop 2005; the Associate Editor, IEEE Transactions on Vehicular Technology; the Area Editor, Journal of Computer Networks; and the Area Editor, Journal of Parallel and Distributed Computing.

Yanfei Fan received the M.Eng. degree (2005) from Tsinghua University, China, and the B.Eng. degree (2002) from Beijing University of Posts and Telecommunications, China, all in Computer Science. He is currently pursuing his Ph.D. degree in the Department of Electrical and Computer Engineering at University of Waterloo, Canada. His research interests include network coding, security in wireless communication and mobile computing.

Xuemin (Sherman) Shen (IEEE M'97-SM'02-F'09) received the B.Sc. (1982) degree from Dalian Maritime University (China) and the M.Sc. (1987) and Ph.D. degrees (1990) from Rutgers University, New Jersey (USA), all in electrical engineering. He is a University Research Chair Professor, Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on mobility and resource management in interconnected wireless/wired networks, UWB wireless communications networks, wireless network security, wireless body area networks and vehicular ad hoc and sensor networks. He is a co-author of three books, and has published more than 400 papers and book chapters in wireless communications and networks, control and filtering. He is a Distinguished Lecturer of IEEE Communications Society. He serves as the Tutorial Chair for IEEE ICC'08, the Technical Program Committee Chair for IEEE Globecom'07, the General Co-Chair for Chinacom'07 and QShine'06, the Founding Chair for IEEE Communications Society Technical Committee on P2P Communications and Networking. He also serves as a Founding Area Editor for IEEE Transactions on Wireless Communications; Editor-in-Chief for Peer-to-Peer Networking and Application; Associate Editor for IEEE Transactions on Vehicular Technology; KICS/IEEE Journal of Communications and Networks, Computer Networks; ACM/Wireless Networks; and Wireless Communications and Mobile Computing (Wiley), etc. He has also served as Guest Editor for IEEE JSAC, IEEE Wireless Communications, IEEE Communications Magazine, and ACM Mobile Networks and Applications, etc. He received the Excellent Graduate Supervision Award in 2006, and the Outstanding Performance Award in 2004 and 2008 from the University of Waterloo, the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada, and the Distinguished Performance Award in 2002 and 2007 from the Faculty of Engineering, University of Waterloo. He is a registered Professional Engineer of Ontario, Canada.