Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance

Qian Wang and Kui Ren
Department of ECE
Illinois Institute of Technology
Email: {qwang,kren}@ece.iit.edu

Wenjing Lou
Department of ECE
Worcester Polytechnic Institute
Email: wjlou@ece.wpi.edu

Yanchao Zhang
Department of ECE
New Jersey Institute of Technology
Email: yczhang@njit.edu

***Abstract*—Recently, distributed data storage has gained increasing popularity for efficient and robust data management in wireless sensor networks (WSNs). But the distributed architecture also makes it challenging to build a highly secure and dependable yet lightweight data storage system. On the one hand, sensor data are subject to not only Byzantine failures, but also dynamic pollution attacks, as along the time the adversary may modify/pollute the stored data by compromising individual sensors. On the other hand, the resource-constrained nature of WSNs precludes the applicability of heavyweight security designs. To address the challenges, we propose a novel dependable and secure data storage scheme with dynamic integrity assurance in this paper. Based on the principle of secret sharing and erasure coding, we first propose a hybrid share generation and distribution scheme to achieve reliable and fault-tolerant initial data storage by providing redundancy for original data components. To further dynamically ensure the integrity of the distributed data shares, we then propose an efficient data integrity verification scheme exploiting the technique of algebraic signatures. The proposed scheme enables individual sensors to verify in one protocol execution all the pertaining data shares simultaneously in the absence of the original data. Extensive security and performance analysis shows that the proposed schemes have strong resistance against various attacks and are practical for WSNs.**

I. Introduction

Distributed data storage and access recently have found increasing popularity due to many reasons. First, new-generation sensor nodes with significant performance enhancement are available. Such enhancements include energy-efficient storage, greater processing capabilities, and data management abilities. Energy-efficient storage such as the new-generation flash memory with several gigabytes and low-power consumption is now possible to equip sensor devices [1], [2]. Second, distributed data storage has more efficient energy consumption. In Mica Mote platform [3] flash memory has less energy efficiency, thereby reducing the energy benefits of local data storage. However, new generation flash memory has significantly altered the energy efficiency and *computation vs communication trade-off* as well. For example, transmitting data over radio channel consumes 200 times more energy than storing the same amount of data locally on a sensor node [4]; Radio reception costs 500 times more energy than reading the same amount of data from local storage [2]. A measurement study in [5] shows that equipping the MicaZ [6] platform with NAND flash memory allows storage to be two orders of magnitude cheaper than communication and comparable in cost to computation, which makes local storage and processing more desirable. Last but not least, distributed data storage achieves more robust WSN. Centralized storage can lead to the single point of failure and easily attracts attacks. Centralized storage may also cause performance bottleneck, as all data collection and access have to go through the base station.

To the best of our knowledge, distributed data storage and access security as a fairly new area has received limited attention so far. Previous research on WSN security issues has been focused on network communication security, such as key management, message authentication, secure time synchronization and localization, and intrusion detection [7]–[11]. A few related works [12]–[18] regarding secure distributed data storage can be found in the literature, but none of them satisfies the overall requirements of data confidentiality, dependability, integrity and efficiency. Zhang *et al.* [16] proposed a secure data access approach by using polynomial-based key management scheme, where the mobile sinks can retrieve the network data following the fixed routes. Subramanian *et al.* [18] studied the distributed data storage and retrieval problem in sensor networks and designed an adaptive polynomial-based data storage scheme for efficient data management. However, both of these schemes do not consider the data dependability and integrity. Chessa *et al.* [15] extended the idea of information dispersal in [19] and investigated the data storage problem in the context of Redundant Residue Number System (RRNS). However, whether the space efficiency can be obtained is not clear and the system has to maintain a large library of parameters together with a big set of moduli. Subbiah *et al.* [17] developed a novel combination of XOR secret sharing and replication mechanisms, where each share is managed using replication-based protocols for Byzantine and crash fault tolerance. However, while the computation overhead is reduced drastically, additional servers and storage capacities at each server are required. Table I shows the comparisons between our scheme and some typical data storage schemes with respect to several desired properties.

Data integrity and availability is an important and necessary component of secure data storage for distributed sensor networks. Sensor data are vulnerable to random Byzantine failures as well as data pollution attacks, in which the adversary can modify the data and/or inject polluted data into the storage nodes. These attacks prevent authorized users from recovering the original data information correctly. Therefore, to ensure the data integrity and availability over the entire data lifetime, any unauthorized data modifications or random data corruptions

TABLE I
COMPARISON OF DATA STORAGE SCHEMES

Property \ Scheme	Ours	[19]	[15]	[17]	[18]
Confidentiality	√	√	√	√	√
Dependability	√	√	√	√	
Dynamic Data Integrity Assurance	√				
Efficiency	√	√			√

due to malicious attacks or Byzantine failures should be detected as soon as possible. However, this important and unique issue has been largely overlooked in most existing designs in WSNs.

In this paper, we propose an efficient and flexible dynamic data integrity checking scheme for verifying the consistency of data shares in a distributed manner. In our scheme, the data originating sensor partitions the original data into multiple shares based on the erasure coding and perfect secret sharing techniques. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based techniques, and achieves reliable data storage by providing redundancy for original data components. To ensure data integrity and availability, we utilize algebraic signatures with favorable algebraic properties, which allow the share holders to perform dynamic data integrity checks in a random way with minimum overhead. Since the data originating sensor appends a distinct parity block to each data share, all share holders can verify the distributed data shares independently in each check. A salient feature of our scheme is that the false-negative probability can be reduced to almost zero, thus any unauthorized modifications can be detected in one verification operation. Most importantly, our scheme can verify the integrity of aggregated data shares with great efficiency. We show through detailed analysis that our scheme is highly effective and efficient and can be well-suited for resource-constrained WSNs.

The rest of the paper is organized as follows. Section II introduces the system model, attack model and briefly describes some necessary background for the techniques used in this paper. Section III and IV provide the detailed description of our proposed schemes. Section V analyzes the scheme performance and presents some enhanced techniques. Finally, Section VI concludes the paper.

II. NETWORK AND SECURITY ASSUMPTIONS

A. System Model

We consider a wireless sensor network with a large number of sensor nodes, each of which has a unique ID and may perform different functionalities. These nodes are deployed strategically into areas of interest and continuously sense the environments and (some of them) are equipped with sufficient capacity to store the sensed data locally in a distributed manner for a certain period.

We assume that these nodes have limited power supply, storage space and computational capability. Due to the constrained resources, computationally expensive and energy-intensive operations are not favorable for such systems. In addition, for such an WSN, we also assume that basic security mechanisms such as pairwise key establishment between two neighboring nodes [20] are already in place to provide basic communication security. However, individual sensors are not reliable since they can behave random Byzantine failures and can be compromised due to lack of tamper-proof hardware.

B. Adversary Model

We consider a general and powerful adversary model regarding data storage and retrieval security and dependability. More specifically, we consider an adversary with both passive and active capabilities:

• The adversary is interested in modifying or polluting the data stored at the storage sensors without being detected. Once a storage sensor is compromised, an "active" adversary can not only read the stored data but also pollute it by modifying or introducing its own fraudulent data. Furthermore, the adversary aims to remain stealthy in order to periodically or occasionally visit the network and harvest potentially valuable data.

• The adversary seeks to compromise as many storage sensors as possible and as long as it remains in control of that node, it reads all of the memory or storage contents and monitors all incoming and outgoing communication.

Note that if the adversary compromises a sensor node and resides there, it can always respond the "verifier" with the correct data and successfully pass the periodic data integrity checks. In fact, there is no way to detect such a compromised sensor if it is fully controlled by the adversary and behaves properly all the time.

C. Design Goal

Our goal is to provide various mechanisms for ensuring and maintaining the security and dependability of sensed network data under the aforementioned adversary model. Specifically, we have the following goals: (i) *Security*: To enhance data confidentiality and integrity by increasing the attacker's cost, i.e., decreasing the gain on compromising individual sensors. (ii) *Dependability*: To enhance data availability against both sensor Byzantine failures and sensor compromises, i.e., minimizing the effect brought by individual sensor failures and compromises. (iii) *Dynamic Integrity Assurance*: We should be able to ensure the distributed data shares are correctly stored over the lifetime, thus can finally be used to reconstruct the original data by authorized users. (iv) *Lightweight*: The scheme design should be lightweight as always in order to fit into the inherent resource-constrained nature of WSNs.

D. Notation and Preliminaries

• v, w, NB_v: v and w are regular sensor nodes. NB_v is the set of one-hop neighbors of v.

• D_i ($i \in \{1, \ldots, m\}$): partitioned blocks of the original data with equal size.

• q, $seqno$: q denotes the bit length of a symbol. $seqno$ is the sequence number used for future data integrity check or retrieval.

• α, β_i ($i \in \{1, \ldots, n\}$): α is a primitive element in $GF(2^q)$ and β_is are distinct elements randomly picked from $GF(2^q)$.

- $\overline{S}_i$, $\underline{S}_i$ $(i \in \{1, \ldots, n\})$: $\overline{S}_i$ denotes the shares of original data that generated by RS coding and $\underline{S}_i$ denotes shares of the authorized key (K_{UV}) generated by secret sharing scheme.
- $sig_{(\alpha,r)}(S)$ $(S \in \{S_1, \ldots, S_n\})$: a r-symbol signature vector $(sig_{\alpha^1}(S), \ldots, sig_{\alpha^r}(S))$ based on data share S.
- P_i, $\mathcal{P}_i$ $(i \in \{1, \ldots, n\})$: P_i is the parity block generated based on all data shares using erasure coding. $\mathcal{P}_i$ denotes the operation of parity calculation based on β_i.

In addition, a sensor node is referred as a *share holder* if data shares have been stored on it and any node can be a potential *verifier* if it holds a parity P_i with its corresponding secret β_i.

We now introduce some necessary cryptographic background for our proposed scheme:

Secret Sharing Shamir proposed an (m, n) Secret Sharing (SS) scheme [21] based on polynomial interpolation, in which m of n shares of a secret are required to reconstruct the secret.

Erasure Code An (k, n) erasure code encodes a block of data into n fragments, each has $1/k$ the size of the original block and any k fragments can be used to reconstruct the original data block. Examples are Reed Solomon (RS) codes [22] and Rabin's Information Dispersal Algorithm [19].

Algebraic Signature Algebraic Signature (AS) is proposed by Litwin *et al.* in [23], [24]. An algebraic signature of a string X of l symbols, $X = (x_1, \ldots, x_l)$, is itself a symbol defined as $sig_\alpha(X) = \sum_{i=1}^{l} x_i \alpha^{i-1}$. The symbols x_i can be one-byte or 2-byte words as the elements of the Galois Field $GF(2^q)(e.g., q = 8, 16)$. We now list three important properties for algebraic signatures:

- *Property* I: Given the string length $l < 2^q - 1$, then $sig_{(\alpha,r)}$ can detect any changes up to r symbols.
- *Property* II: Given the string length $l < 2^q - 1$, then $sig_{(\alpha,r)}$ of two different shares collide with probability of 2^{-rq}.
- *Property* III: Assume $P_i = \mathcal{P}_i(D_1, D_2, \ldots, D_m)$, then $sig_\alpha(\mathcal{P}_i(D_1, \ldots, D_m)) = \mathcal{P}_i(sig_\alpha(D_1), \ldots, sig_\alpha(D_m))$.

III. Dependable Initial Data Storage

To guarantee the security of the stored data, sensor nodes must encrypt the data for confidentiality. Thus, only authorized user can obtain the access privilege and decrypt the data information. In addition, as sensors may exhibit Byzantine behaviors and are attractive for attacks, data dependability should also be ensured to avoid single point of failure. To address these problems and achieve a lightweight design in resource-constrained WSN, we discuss two schemes for initial data storage (a basic scheme followed by an enhanced scheme), which we believe would lead us to the final desirable solution.

A. The basic Scheme

In the basic scheme, suppose a sensor node v has $data$ to be stored locally. To protect $data$, it can perform the following operations to ensure the data integrity and confidentiality:

- *Step* 1: Generate a random session key k_r and compute the keyed hash value $h(data, k_r)$ of $data$.
- *Step* 2: Encrypt $data$, $h(data, k_r)$ with k_r and obtain $\{data, h(data, k_r)\}_{k_r}$.
- *Step* 3: Encrypt k_r using the key K_{UV} shared between the authorized users and itself. This key can be either symmetric or asymmetric depending on the chosen user access control mechanism, which is independent to our design here and will not be discussed in this paper.
- *Step* 4: Store $\mathsf{DATA} =< \{data, h(data, k_r)\}_{k_r}, \{k_r\}_{K_{UV}} >$ and destroy k_r.

DATA will be fed back to the user when required. An authorized user will be able to decrypt the original $data$ with K_{UV} and ensure its integrity by checking $h(data, k_r)$.

Discussion This basic approach only provides the least protection over the data security and dependability. It cannot stand sensor Byzantine failures or data losses due to sensor compromises. One straightforward way to enhance data dependability is to replicate the data and distribute the replicas to the neighbors. If some nodes are compromised or behave Byzantine failures, the data can still be correctly recovered from the other nodes that store the replicas. However, the simple replication approach incurs a very high scheme overhead. Since n replicas need to be distributed and stored, both the communication overhead and the storage overhead are $n * |\mathsf{DATA}|$.

To further enhance data security and dependability, we may resort to the secret sharing technique. In a (k, n)-threshold secret sharing scheme, any combination of less than k secret shares reveals no information regarding the secret. On the other hand, no more than k out of n shares are required to fully recover the secret. Obviously, the first property can be used to enhance data security, while the second is good for data dependability. However, using perfect secret sharing to manage generic data leads to n-fold increase in storage overhead, and is no better than n-fold replication. Therefore, the solution may not be feasible in resource-constrained WSNs.

B. An Enhanced Scheme based on Erasure Coding and Secret Sharing

To deal with the limitations in the secret sharing-based approach, we now propose to integrate the technique of erasure coding as it can achieve optimal space efficiency. We thus propose a hybrid scheme as follows, which takes advantage of both erasure coding and secret sharing techniques:

- *Step* 1: v calculates DATA according to the basic scheme and further divides the DATA into two parts, i.e., $\{data, h(data, k_r)\}_{k_r}$ and $\{k_r\}_{K_{UV}}$.
- *Step* 2: v then encodes $< \{data, h(data, k_r)\}_{k_r} >$ into n fragments by employing a (m, n) RS code. That is, v constructs $M(x) = D_1 + D_2 x + \ldots + D_m x^{m-1}$, where $< \{data, h(data, k_r)\}_{k_r} >: = \{D_1 \| \ldots \| D_m\}$. v further obtains n $\overline{S}_j$s with each $\overline{S}_j = M(\alpha^j)$ $(1 \le j \le n)$, where $n \le 2^q - 1$.
- *Step* 3: v then employs a (m, n) SS scheme to obtain n shares of $\{k_r\}_{K_{UV}}$ denoted as $\underline{S}_1, \ldots, \underline{S}_n$.
- *Step* 4: v randomly selects n neighbors from NB_v. For each neighbor w_i $(i \in \{1, \ldots, n\})$, v distributes $\{v, seqno, \overline{S}_i, \underline{S}_i\}_{K_{vw_i}}$. The original $data$ is erased.

Now, an authorized user can recover DATA by reconstructing $< \{data, h(data, k_r)\}_{k_r} >$ from $\overline{S}_j$s based on RS code and $\{k_r\}_{K_{UV}}$ from $\underline{S}_i$s based on secret sharing. In both cases, any m out of n shares are sufficient. Obviously, the proposed hybrid scheme offers the same level of security and dependability as the secret sharing-based scheme. Compromising less than m sensors will not damage data security and dependability. Moreover, the hybrid scheme is much more efficient in terms of communication and storage overheads. In fact, these two overheads are approximately now both $\frac{n}{m} * |\mathsf{DATA}|$ as compared to $n * |\mathsf{DATA}|$.

IV. Ensuring Dynamic Data Integrity

In a mission-critical application, the security and availability of the stored data must be guaranteed over the whole data lifetime. One of the key issue related to this is to be able to detect any unauthorized data modification possibly due to sensor compromise and/or random Byzantine failures [25]. In fact, after data shares are distributed among the neighbor nodes, these shares may be modified illegally once the corresponding storage sensor is compromised or it behaves Byzantine failures. If such illegal modifications cannot be detected, the user, when retrieving the data, will not be able to correctly recover the original data due to the share pollution. Therefore, it is critical to enable dynamic data integrity check throughout the whole data lifetime. The challenge then is *how to achieve dynamic data integrity check without the existence of the original data and in a lightweight and secure manner.*

A. The Hash-based Scheme

One straightforward approach of ensuring data or data shares integrity is to use fingerprints, where the hash of these shares are stored as a hash vector $\vec{\mathbf{H}}$ at the data originating sensor v before the distribution of data shares. Later on, the data originating sensor v can check the data integrity by requesting each share to be returned. Upon receiving all the shares from the share holders, v can calculate $H(S_1), \ldots, H(S_n)$ and check $\vec{\mathbf{H}} \overset{?}{=} (H(S_1), \ldots, H(S_n))$, where $H(\cdot)$ denotes the hash operation.

However, this approach has a number of serious problems: (i) It is single point of failure. If the data originating sensor fails to function correctly, no one else can perform the integrity check; (ii) It is not secure, as the data originating sensor will collect all the shares for each integrity check. If it is compromised, the attacker will be able to recover the original data by leveraging this operation; (iii) The use of standard fingerprints (e.g., SHA-1, MD5) introduces both high computational and storage overhead. In the verification process, the overall computation overhead involves with n hash operations. In terms of storage overhead, additional n hash values need to be stored in the data originating sensor. The communication overhead is also $n * |S_i|$ as the n shares have to be returned for verification. The overhead problem becomes much worse in the case of data aggregation, where much more hash values need to be computed and stored for future integrity check.

A variant approach could be to ask the share holders to return the fingerprints instead of the original share. However, the storage and computation overhead are almost the same as the above scheme. Moreover, this method suffers from a severe drawback: a malicious or compromised share holder could simply pre-compute and store the fingerprint $H(S_i)$ instead of the genuine data. When required to return the fingerprint, it can respond to the verifier with $H(S_i)$ which will always pass the integrity checks. This problem can be solved using keyed-fingerprinting [26], where a different secret key is employed to generate the fingerprint for each integrity check. However, this would result in an additional overhead in both computation and storage, since the verifier must pre-compute and store all the keyed-fingerprints for data shares it plans to check. In addition, the security level of this scheme is limited by the number of secret keys held by the verifiers and/or share holders. After these keys are exhausted, the aforementioned problem still cannot be eliminated.

Another technique to safeguard against modified data shares is verifiable secret sharing (VSS). With verifiable secret sharing schemes, it is possible for each share holder to perform an integrity verification for the data shares it stored. However, such techniques further increase the computation time during the encoding and decoding of data [17]. Moreover, VSS can only allow the share holder to verify their own shares, it cannot tell the status of the other shares corresponding to the same original data block.

B. The Algebraic Signature-based Dynamic Checking Scheme

In our scheme, the data originating sensor generates a distinct parity based on all data shares and appends the parity to the data share. Once a share holder initiates an integrity verification, other share holders can also act as a verifier to validate the integrity of the stored data shares as long as they have the corresponding parities. Our proposed scheme is based on algebraic signature technique [23], [25], where data shares can be verified without the presence of the original data. The nice properties of our scheme are: (i) any unauthorized modifications can be almost detected in one integrity check; (ii) fast integrity check can be realized when data shares are aggregating on share holders; (iii) the share holders have no ability to create valid signatures that are internally consistent for all data shares (including both the original or polluted ones). The details of our scheme are as follows:

1) Parities generation and distribution: Suppose a sensor node v has $data$ to be stored locally. v first follows the *hybrid share generation scheme* to obtain n shares $S_1, \ldots, S_n$, where each share $S_i = \overline{S}_i || \underline{S}_i$ $(i \in \{1, \ldots, n\})$. Let $(x_{i1}, x_{i2}, \ldots, x_{ik})$ denote the data share S_i , where $k \leq 2^q - 1$. Note all these symbols are elements of a Galois field $GF(2^q)$.

Based on the n data shares $S_1, \ldots, S_n$, v generates n parities

$$\begin{aligned} P_j &= \sum_{i=1}^{n} \beta_j^{i-1} S_i \\ &= (\sum_{i=1}^{n} \beta_j^{i-1} x_{i1}, \sum_{i=1}^{n} \beta_j^{i-1} x_{i2}, \ldots, \sum_{i=1}^{n} \beta_j^{i-1} x_{ik}) \\ &= (p_{j1}, p_{j2}, \ldots, p_{jk}) \qquad \text{for } j \in \{1, \ldots, n\}, \end{aligned} \tag{1}$$

where $n \leq 2^q - 1$. Notice that β_j $(j = 1, \ldots, n)$ are distinct elements randomly picked from $GF(2^q)$.

After parity generation, v randomly chooses n neighbor nodes $w_1, \dots, w_n$ from NB_v. For each neighbor w_i ($i \in \{1, \dots, n\}$), v distributes a data share together with a distinct parity, i.e.,

$$\{v, seqno, S_i, P_i, \beta_i\}_{K_{vw_i}},$$

where $seqno$ is the data sequence number used for future data integrity check or retrieval. Then, $data$, S_is, P_is and β_is are erased. Note P_i is generated based on the corresponding β_i.

2) Dynamic data integrity Check: At a later time, suppose a share holder w_i who holds $\{P_i, \beta_i\}$ wants to check whether the data item $data$ corresponding to $seqno$ is being correctly stored. w_i will launch a data integrity check by broadcasting a challenge message to all the other data share holders

$$\{w_i, seqno, \alpha, r\},$$

where α is a primitive number randomly picked from $GF(2^q)$ and r is the required number of signatures such that $r << 2^q - 1$.

Upon receiving the challenge, all the other share holders that store $\{seqno, S_i\}$ ($i \in \{1, \dots, n\}$) will compute a r-symbol algebraic signature $sig_{(\alpha,r)}(S_i) = (sig_{\alpha^1}(S_i), \dots, sig_{\alpha^r}(S_i))$ and respond to w_i with $\{sig_{(\alpha,r)}(S_i)\}$ in a broadcast manner. Note all $\{sig_{(\alpha,r)}(S_i)\}$s are sent openly, not encrypted.

After obtaining all $sig_{(\alpha,r)}(S_i)$ ($i = 1, \dots, n$), w_i can verify the data integrity by checking

$$\begin{aligned} sig_{(\alpha^t)}(P_i) &\overset{?}{=} \mathcal{P}_i(sig_{(\alpha^t)}(S_1), \dots, sig_{(\alpha^t)}(S_n)) \\ &= \sum_{j=1}^{n} \beta_i^{j-1} sig_{\alpha^t}(S_j), \quad t = 1, \dots, r. \end{aligned} \tag{2}$$

That is because

$$\begin{aligned} sig_{(\alpha)}(P_i) &= \sum_{j=1}^{k} \alpha^{j-1} p_{ij} \\ &= \sum_{j=1}^{k} \alpha^{j-1} \sum_{m=1}^{n} \beta_i^{m-1} x_{mj} \\ &= \sum_{m=1}^{n} \beta_i^{m-1} \sum_{j=1}^{k} \alpha^{j-1} x_{mj} \\ &= \sum_{m=1}^{n} \beta_i^{m-1} sig_{\alpha}(S_m) \\ &= \sum_{j=1}^{n} \beta_i^{j-1} sig_{\alpha}(S_j) \end{aligned} \tag{3}$$

Equation 3 shows that the signature of parity block should be equal to the parity of signatures of the **original** data block. There are totally r such equations. If any of these r equations does not hold, we can conclude that the $data$ has been modified due to either malicious attacks or Byzantine faults, i.e., there is no *false positive result* (the data has not been modified but the test says it has). However, if all the checking equations hold, a *false negative result* (the data has been modified but the test says it has not) is possible. As will be shown in section V, the false-negative probability can be reduced to almost zero in our scheme.

The motivation of using unencrypted $\{sig_{(\alpha,r)}(S_i)\}$ is due to the broadcast-based nature of wireless communication. Therefore, these signature information could be considered free and employed by other potential verifiers who possess $\{P_i, \beta_i\}$ ($i \in \{1, \dots, n\}$) to perform an independent integrity check. To make it work, the initiator w_i itself should include signatures of its own data share in the published challenge, i.e., $\{w_i, seqno, \alpha, r, sig_{(\alpha,r)}(S_i)\}$. Therefore, all share holders can obtain $sig_{(\alpha,r)}(S_i)$ ($i = 1, \dots, n$) and act as a verifier in each check process. As will be shown, using more parities instead of one can prevent multiple sensors from colluding to fabricate consistent data or signatures that match a compromised parity. In addition to be useful for collusion resistance, it can also tolerate certain number of unavailable verifiers and enhance the probability of success in detecting any unauthorized modification in one check. However, this configuration suffers from a drawback: sufficient number of exposed signatures could lead to the breaking of data shares. To address this problem, we propose a perturbation-based scheme in section V.

3) Fast Integrity Check for Data Aggregation: In many scenarios, data shares of different original data blocks may aggregate on the storage sensors. An important advantage of the proposed dynamic checking scheme over existing approaches is that it can efficiently verify the integrity of the aggregated data shares with minimum communication and computation overhead. For example, if the sensor node v has $data^{'}$ to be stored locally. Similarly, v will follow the *hybrid share generation scheme* to generate shares $S_1^{'}, \dots, S_n^{'}$, and distribute them to the pre-selected n neighbors that stores data shares for $data$. To verify that the data shares corresponding to $data$ and $data^{'}$ are both correctly stored, a verifier can initiate an integrity check by broadcasting a challenge to all share holders (For simplicity, we assume $r = 1$). Upon receiving the message, each node acts as follows:

- Assume $S_i^{'} = (x_{i1}^{'}, x_{i2}^{'}, \dots, x_{ik}^{'})$, thus

$$\begin{aligned} S_i || S_i^{'} &= (x_{i1}, x_{i2}, \dots, x_{ik}, x_{i1}^{'}, x_{i2}^{'}, \dots, x_{ik}^{'}) \\ &= (x_{i1}, x_{i2}, \dots, x_{ik}, x_{i(k+1)}, x_{i(k+2)}, \dots, x_{i(2k)}) \end{aligned}$$

The signature of $S_i || S_i^{'}$ can be computed as

$$\begin{aligned} sig_{\alpha}(S_i || S_i^{'}) &= \sum_{j=1}^{k} \alpha^{j-1} x_{ij} + \sum_{j=k+1}^{2k} \alpha^{j-1} x_{ij} \\ &= sig_{\alpha}(S_i) + \alpha^k \sum_{j=1}^{k} \alpha^{j-1} x_{ij}^{'} \\ &= sig_{\alpha}(S_i) + \alpha^k sig_{\alpha}(S_i^{'}). \end{aligned}$$

Then $sig_{\alpha}(S_i || S_i^{'})$ ($i = 1, \dots, n$) is returned to the verifier.

- Assume this verifier holds P_i and $P_i^{'}$ ($i \in \{1, \dots, n\}$), where P_i and $P_i^{'}$ are parities for $data$ and $data^{'}$ based on the same secret β_i, respectively. Upon receiving all responses from share holders, similarly the verifier generates signatures of $P_i || P_i^{'}$ by $sig_{\alpha}(P_i || P_i^{'}) = sig_{\alpha}(P_i) + \alpha^k sig_{\alpha}(P_i^{'})$. Now the checking equation can be written as

$$sig_{(\alpha)}(P_i || P_i^{'}) \overset{?}{=} \sum_{j=1}^{n} \beta_i^{j-1} sig_{\alpha}(S_j || S_j^{'}).$$

Hence, when data shares are aggregating on sensors, our proposed checking scheme can speed up the verification of different data blocks in the checking process. Actually, this property can be easily generalized to more or any combination of data shares for arbitrary original data blocks. Note that no matter how many data shares or parities are concatenated, the signature is only a symbol of length q, thus using this scheme allows us to check a group of data blocks in a more efficient way while introducing a minimum of overhead.

C. Data Maintenance

In our data storage scheme, the original data block is first partitioned into n shares of equal size using Reed-solomon coding. Based on these shares, additional n parities of the same size are generated. By distributing each neighbor a distinct data share with a parity, we form another reliability group with error correcting capabilities. Once any unauthorized data modification is detected, to repair the polluted shares, the basic idea of using Mobile Sink [16] can be applied. The network controller can periodically dispatch mobile sinks (MSs) to collect data shares and/or parities, and perform error correction. Due to the space limitation, in this paper we only focus on the secure data storage scheme with dynamic data integrity check. The issue of data maintenance will be addressed in detail in our future work.

V. Scheme Analysis and Enhancements

In this section, we analyze our proposed data storage scheme in terms of security and efficiency. Our security analysis focuses on the adversary model defined in section II. Further discussion on other attacks are also discussed. Based on the analysis, some enhanced techniques are presented.

A. Security and Dependability of Initial Data Storage

The security proof in [21] and [22] ensures that our *hybrid share generation scheme* for initial data storage is unconditionally secure and $(m-1)$-collusion resistant. That is, up to $(m-1)$ colluding nodes reveal no information on the encoded data. In practice, requirements may vary depending on different applications, thus the threshold m can be adjusted to accommodate the special needs.

B. Security of Dynamic Integrity Assurance

1) Modifying Data Shares: After an adversary has compromised a set of nodes, it would like to modify the data shares in order to prevent authorized users from recovering the original data blocks correctly.

We first analyze the case that the adversary modifies data shares *accidentally* and study the probability of a *false negative result* in the process of data integrity checks. In the following analysis, we call $(sig_{(\alpha)}(S_1), \ldots, sig_{(\alpha)}(S_n))$ a *n-symbol signature page*. Since parity calculations and algebraic signature calculations are isomorphic, thus the parity of signatures can be considered as the signature of a *signature page* based on β, i.e., $\sum_{i=1}^{n} \beta^{i-1} sig_{\alpha}(S_i)$. We have the following proposition:

***Proposition** 1:* Assume the number of shares $n < 2^q - 1$, then the parity P of two different *n-symbol signature pages* collide with probability 2^{-q}. If p different parities $P_1, \ldots, P_p$ are used, the collision probability (of all p parities) can be further reduced to 2^{-pq}.

Proof The proof of this proposition is similar to the proof of *Property* II [23]. ■

Next, we study the probability of a *false negative result* where the data has been modified *accidentally* but all the checking equations hold, i.e., $sig_{(\alpha^t)}(P_i) = \sum_{j=1}^{n} \beta_i^{j-1} sig_{\alpha^t}(S_j)$, where $(t = 1, \ldots, r)$ and $(i = 1, \ldots, n)$. Suppose n_c nodes are compromised, in the following analysis, we do not limit the value of n_c, i.e., $n_c \geq m$ is also considered.

***Proposition** 2:* Assume the number of shares $n < 2^q - 1$, $\mathcal{P}_i(sig_{(\alpha^t)}(S_1), sig_{(\alpha^t)}(S_2), \ldots, sig_{(\alpha^t)}(S_n)) = \sum_{j=1}^{n} \beta_i^{j-1} sig_{\alpha^t}(S_j)$, where $(t = 1, \ldots, r)$ and $(i = 1, \ldots, p)$, $p \leq n$. Then the probability of a *false negative result* is

$$\mathrm{Pr} = \mathrm{Pr}_1 + \mathrm{Pr}_2, \tag{4}$$

where $\mathrm{Pr}_1 = \frac{(1+2^{-rq})^{n_c}-1}{2^{n_c}-1}$ and $\mathrm{Pr}_2 = (1-\mathrm{Pr}_1)2^{-pq}$.

Proof We first consider the simplest scenario where $r = 1$ and $p = 1$, i.e., only one signature is calculated for each data share and only one parity is generated for all the data shares. Without loss of generality, we denote the signature base by α and the **secret** for parity generation by β, respectively. The checking equations can be rewritten as

$$\begin{aligned} sig_{(\alpha)}(P) &\overset{?}{=} \mathcal{P}(sig_{(\alpha)}(S_1), sig_{(\alpha)}(S_2), \ldots, sig_{(\alpha)}(S_n)) \\ &= \sum_{i=1}^{n} \beta^{i-1} sig_{\alpha}(S_i) \end{aligned}$$

As discussed above, a *false negative result*is that the data has been modified but the integrity test says it has not, i.e., the parity of the *n-symbol signature page* does not change. An analysis of the checking equation reveals two possible cases:

Case 1 The data has been modified, but the *signature page* $(sig_{(\alpha)}(S_1), sig_{(\alpha)}(S_2), \ldots, sig_{(\alpha)}(S_n))$ remains the same. Hence the parity of the *page* will not change. We denote the probability of this case by Pr_1. According to *Property* II, the probability that S_i $(i \in \{1, \ldots, n\})$ is modified but $sig_{(\alpha)}(S_i)$ remains unchanged is 2^{-q}. It is also reasonable to assume that $sig_{\alpha}(S_1), \ldots, sig_{\alpha}(S_n)$ are mutually independent. Let $C_{n_c}^{k}(2^{-q})^k$ denote the probability that k shares are modified but the *signature page* remains the same, we have

$$\mathrm{Pr}_1 = \frac{C_{n_c}^{1}(2^{-q})^1 + \cdots + C_{n_c}^{n_c}(2^{-q})^{n_c}}{C_{n_c}^{1} + \cdots + C_{n_c}^{n_c}} = \frac{(1+2^{-q})^{n_c}-1}{2^{n_c}-1}.$$

Case 2 The data has been modified, the *signature page* will change **for sure**, but the parity of the *page* will remain the same. We denote the probability of this case by Pr_2. A *page* can be considered as a "data share" with n symbols, thus the parity of a *page* is actually a signature based on the secret β. According to proposition 1, the probability that the *n-symbol* *signature page* is modified but the parity of the *signature page* remains unchanged is 2^{-q}, thus we have

$$\mathrm{Pr}_2 \;=\; (1-\mathrm{Pr}_1)2^{-q}.$$

The probability of a *false negative result* with $r = 1$ and $p = 1$ is thus $\mathrm{Pr} = \mathrm{Pr}_1 + \mathrm{Pr}_2$.

Now we analyze a more complex scenario where $r > 1$ and $p = 1$. Similarly, there are two possible cases:

Case 1 The data has been modified, but all the r *signature pages* will remain the same. To facilitate analysis, we denote the r *signature pages* by a so-called *compound signature page* $(sig_{(\alpha,r)}(S_1)^T, sig_{(\alpha,r)}(S_2)^T, \ldots, sig_{(\alpha,r)}(S_n)^T)$, where $sig_{(\alpha,r)}(S_i)^T = (sig_{\alpha^1}(S_i), \ldots, sig_{\alpha^r}(S_i))^T$.

According to *Property* II, the probability that S_i is modified but $sig_{(\alpha,r)}(S_i)$ remains unchanged is 2^{-rq}. Thus

$$\mathrm{Pr}_1 \;=\; \frac{(1+2^{-rq})^{n_c}-1}{2^{n_c}-1}.$$

Case 2 The data has been modified, the *compound signature page* $(sig_{(\alpha,r)}(S_1)^T, \ldots, sig_{(\alpha,r)}(S_n)^T)$ will change **for sure**, but the parity of the *page* will remain the same. Similarly, Pr_2 can be calculated as

$$\mathrm{Pr}_2 \;=\; (1-\mathrm{Pr}_1)2^{-q}.$$

Further more, we can explore the scenario where $r = 1$ and $p > 1$. When $r = 1$, there is only one *signature page*, thus $\mathrm{Pr}_1 = \frac{(1+2^{-q})^{n_c}-1}{2^{n_c}-1}$. According to proposition 1, if p different parities are used, the collision probability of a *page* is 2^{-pq}. Therefore, $\mathrm{Pr}_2 = (1-\mathrm{Pr}_1)(2^{-q})^p$.

Base on the discussions above, it is easy to see that the probability of a *false negative result* with $r > 1$ and $p > 1$ is

$$\mathrm{Pr} \;=\; \mathrm{Pr}_1 + \mathrm{Pr}_2.$$

where $\mathrm{Pr}_1 = \frac{(1+2^{-rq})^{n_c}-1}{2^{n_c}-1}$ and $\mathrm{Pr}_2 = (1-\mathrm{Pr}_1)2^{-pq}$. ■

After compromising n_c nodes, the adversary can modify any data share preloaded to these nodes. Any node that possesses a pair $\{P_j, \beta_j\}$ can launch a data integrity check. Consider the probability of a false negative result Pr, it is a sum of two probabilities. For fixed n_c and q, Pr is determined by the number of signatures and the number of parities used in the check.

Table II shows some numeric results of Pr. For example, let us assume the symbol length is $q = 8$ and the number of compromised nodes is $n_c = 20$. It is shown that by increasing r and p simultaneously, the false-negative probability can be reduced significantly. However, using more signatures will incur more computation and communication overhead, since all the share holders have to compute r different signatures $(sig_{\alpha^1}(S), sig_{\alpha^2}(S), \ldots, sig_{\alpha^r}(S))$ and return them to the verifier. Observe that for a specific p, further increasing the number of r offers little gain in reducing the false-negative probability. Also recall that in our dynamic checking scheme, we distribute each share holder a distinct parity P and with its corresponding secret β, so that once a sensor initiates an integrity check, all the share holders can receive $sig_{(\alpha,r)}(S_i)$s and act as a verifier. That is, n parities are employed in each verification process. This configuration can tolerate large number of unavailable sensors (verifiers) and reduce the false-negative probability to an acceptable extent.

TABLE II
PROBABILITY OF A FALSE NEGATIVE RESULT ($q = 8$).

n_c		$r = 1$	$r = 2$	$r = 3$
10	p = 1	3.9450e-003	3.9064e-003	3.9063e-003
	p = 2	5.4121e-005	1.5408e-005	1.5259e-005
	p = 3	3.8922e-005	2.0877e-007	6.0187e-008
20	p = 1	3.9063e-003	3.9063e-003	3.9063e-003
	p = 2	1.5336e-005	1.5336e-005	1.5336e-005
	p = 3	1.3694e-007	5.9896e-008	5.9606e-008
30	p = 1	3.9063e-003	3.9063e-003	3.9063e-003
	p = 2	1.5259e-005	1.5259e-005	1.5259e-005
	p = 3	5.9720e-008	5.9605e-008	5.9605e-008

We have shown that, if an adversary modifies the data share *accidentally*, our data integrity checking scheme can successfully detect the attack in one verification. In addition, using only 1-symbol signature ($r = 1$) in each multi-parity verification is sufficient to meet our design goal. However, by observing the system for a long time, the adversary may find out the set of challenge number αs. Then, it may attempt to modify data *deliberately* such that the polluted data share has the *same* signatures as the original ones.

First, we analyze how this attack can be carried out. Consider the process of calculating signatures, a data share S can be written as a sum of two special symbol vectors $(x_1, x_2, \ldots, x_k) = (x_1, x_2, \ldots, x_n, 0, \ldots, 0) + (0, \ldots, 0, x_{n+1}, x_{n+2}, \ldots, x_k)$. The n-symbol algebraic signature $sig_{(\alpha,n)}$ of $(x_1, x_2, \ldots, x_n, 0, \ldots, 0)$ is

$$\begin{pmatrix} sig_\alpha \\ sig_{\alpha^2} \\ \vdots \\ sig_{\alpha^n} \end{pmatrix} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \ldots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & \ldots & (\alpha^2)^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^n & (\alpha^n)^2 & \ldots & (\alpha^n)^{n-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}$$

The square matrix is of Vandermonde type, thus for each n-symbol signature, there is one and only one symbol vector $(x_1, x_2, \ldots, x_n, 0, \ldots, 0)$ corresponding to it. Now consider the symbol vector of form $(0, \ldots, 0, x_{n+1}, x_{n+2}, \ldots, x_k)$, we denote its $sig_{(\alpha,n)}$ by $\widetilde{s}$. Assume the $sig_{(\alpha,n)}$ of the original share $(x_1, x_2, \ldots, x_k)$ is s. The adversary can first randomly choose a vector of the form $(0, \ldots, 0, x'_{n+1}, x'_{n+2}, \ldots, x'_k)$, then it can compute the vector $(x'_1, x'_2, \ldots, x'_n, 0, \ldots, 0)$ that has $(s - \widetilde{s})$ as its signature. It is obvious that the symbol vector $(x'_1, \ldots, x'_n, x'_{n+1}, \ldots, x'_k)$ has signature s, which is the same as the original data share! Notice that $n < k$, hence the polluted data generated by the adversary can possibly pass $k-1$ data integrity checks. This attack, however, does not work because in our scheme, the verifier picks α from $GF(2^q)$ in a random way. Due to this reason, the adversary has to guess which α will be used in the future. As analyzed, using one signature is sufficient for each check, thus α does not have to be a primitive element now, it can be any element in $GF(2^q)$.

Let us suppose the adversary guesses that $\alpha_1, \ldots, \alpha_{k-1}$ are most likely to be used in the forthcoming integrity checks, it may modify data shares *deliberately* in order to deceive the verifier for at most $k-1$ times. Because the number of possible α is $2^q - 1$, so the probability that the polluted share can successfully pass t checks is $(\frac{k-1}{2^q-1})^t$. In practice, as long as q is appropriately selected, e.g., $q = 8$ or 16, the probability is very small, which makes the attack unfeasible.

2) Breaking data shares by eavesdropping: Recall that in the course of integrity check, when all the share holders return $sig_{(\alpha,r)}(S_i)$ to the check initiator, these responses are sent openly, not encrypted. This configuration not only saves computation overhead (encryption and decryption) but also offers verification "diversity" since it enables share holders to realize multi-verification by sharing their signature information with each other. However, this technique suffers from a drawback: Consider an algebraic signature of a data share $sig_\alpha(S_i) = \sum_{j=1}^{k} x_{ij}\alpha^{j-1}$, the adversary can construct a system of linear equations. By eavesdropping, an adversary only needs to find out signatures based on k different αs for this data share. The number of equations is k and the number of unknowns is k, i.e., $(x_{i1}, x_{i2}, \ldots, x_{ik})$. Therefore, if enough of algebraic signatures are collected, the adversary can break the data share by solving a system of linear equations.

To address this problem, we develop a modification of the original protocol to allow each share holder to safeguard the exposed signatures with a minimum of overhead. Different from the basic scheme, share holders do not give verifiers the original signature but the perturbed one, which is a sum of the original signature and a perturbation symbol. Thus, by blinding each signature information with perturbation, the adversary cannot capture the original signatures directly. On the other hand, after receiving these perturbed signatures, the verifier computes parities based on these signatures without needing to decrypt them. To further explain the above idea, we present more details of this technique.

Recall that in parity generation phase, β_js are chosen as **secrets** by the data originating sensor to generate parities. The first step under this scheme is to construct a Vandermonde matrix using these secrets. Next, the originator constructs a random *perturbation vector* $(\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_n)^T$, where all γ_i $(i = 1, \ldots, n)$ are picked from $GF(2^q)$ independently. By multiplying the matrix by this perturbation vector, we obtain another symbol vector

$$\begin{pmatrix} 1 & \beta_1 & \beta_1^2 & \ldots & \beta_1^{n-1} \\ 1 & \beta_2 & \beta_2^2 & \ldots & \beta_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \beta_n & (\beta_n)^2 & \ldots & (\beta_n)^{n-1} \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_n \end{pmatrix} = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \\ \theta_n \end{pmatrix}$$

Here, $(\theta_1, \theta_2, \theta_3, \ldots, \theta_n)^T$ is called *counteracting vector*. Note that

$$\begin{aligned} \sum_{i=1}^{n} \beta_j^{i-1}(sig_\alpha(S_i) + \gamma_i) &= \sum_{i=1}^{n} \beta_j^{i-1} sig_\alpha(S_i) + \sum_{i=1}^{n} \beta_j^{i-1}\gamma_i \\ &= \sum_{i=1}^{n} \beta_j^{i-1} sig_\alpha(S_i) + \theta_j. \end{aligned}$$

This property is exploited in our design of the integrity checking scheme. In the parity distribution phase, a piece of additional information $\{\gamma_i, \theta_i\}_{K_{vw_i}}$ $(i \in \{1, \ldots, n\})$ is distributed to share holder i. When required to return a signature to the verifier, share holder i blinds $sig_\alpha(S_i)$ by adding a perturbation γ_i. On the verifier side, to uncover parity of the original signatures, it can first compute parity of the perturbed signatures based on $\beta \in \{\beta_1, \ldots, \beta_n\}$ and then subtract the corresponding $\theta \in \{\theta_1, \ldots, \theta_n\}$ from it. It is clear that this configuration has no effect on the validity of the checking results. Furthermore, the above operation ensures that the adversary cannot capture the original signatures by eavesdropping and obtain the original data share by solving a system of linear equations (symbol x_{i1} is fully blinded by γ_i). Therefore, to break the data shares by eavesdropping is unfeasible.

3) Collusion Attacks: Share holders may collaborate to modify data or even make up signatures as long as they are internally consistent. In other words, a verifier receiving signatures can verify that the parity matches the data, but it cannot tell whether some of sensors collude to provide fake signatures for a compromised parity. This attack, however, does not work because in our dynamic checking scheme, all share holders will participate in each checking process to act as a verifier. To make up signatures consistent with n different parities, the colluding sensors have to find out all the **secret** βs, and attempt to construct a set of fake signatures that go with them. In practice, to reduce computation and communication overhead, we use only one α $(r = 1)$ in each integrity check, thus n signatures are returned for n-parity verification, i.e., the number of signatures is equivalent to the number of parities. Based on the above analysis, it is quite evident that using n parities can detect any changes up to n signatures. Therefore, our scheme can prevent multiple sensors from colluding to fabricate consistent signatures.

C. Efficiency

We now assess the performance of the proposed distributed data storage scheme. The notation of cryptographic operations is summarized in table III.

1) Initial Data Storage: In terms of computational overhead, before the share generation process the data source node v has to compute one keyed hash value $h(data, k_r)$ and perform two symmetric-key encryptions $\{data, h(data, k_r)\}_{k_r}$ and $\{k_r\}_{K_{UV}}$. The data share generation requires two sets of polynomial evaluations. v first encodes $\{data, h(data, k_r)\}_{k_r}$ into n fragments by employing a (m, n) RS code, where m denotes the number of partitioned data blocks and n denotes the number of selected neighbors (i.e., share holders). Assume each partitioned data block D_i contains c symbols, there are totally $n \cdot c$ polynomial evaluations in this step. Then v employs a (m, n) SS scheme to obtain n shares of $\{k_r\}_{K_{UV}}$. The construction for the *counteracting vector* can be considered as n signature generations with vector size of n. Finally, a parity based on all shares will be generated for each share holder. Let l denote the size of $data||k_r$, the total computation cost at the data source node is hence

TABLE III
NOTATION SUMMARY OF CRYPTOGRAPHIC OPERATIONS.

$Hash_l^t$	t hash operations with input size of l.
$SymEncr^t$	t symmetric-key encryption operations.
$SymDecr^t$	t symmetric-key decryption operations.
$PolyEval_m^t$	t polynomial evaluations with polynomial of degree m.
$ParGen_k^t$	t parity generations with vector of size k.
$AlgSigGen_k^t$	t algebraic signature generations with vector of size k.

$Hash_l^1 + SymEncr^2 + PolyEval_m^{n\cdot(c+1)} + AlgSigGen_n^n + ParGen_k^n$. Correspondingly, the cost at each share holder is only $SymDecr^1$.

Now let us estimate the storage and communication overhead during the initial data distribution stage, assume we use a Galois field $GF(2^q)$ where $q = 8$ or 16. After share generation, the source node v will delete $data$ and distribute $\{v, seqno, S_i, \beta_i, P_i, \gamma_i, \theta_i\}_{K_{vw_i}}$ to neighbor i, where we assume S_i contains k symbols. In fact, there are total n such messages. Thus, the communication overhead during the distribution process is approximately $n \cdot (2k+5) \cdot q$ bits. Obviously, it requires $(2k+5)\cdot q$ bits storage overhead to keep the distributed information at each share holder.

2) Dynamic Data Integrity Check: Consider a share holder w who initiates a data integrity check to verify the integrity of $data$. It will broadcast a challenge message $\{w, seqno, \alpha, r\}$ to all the share holders. In addition, a 1-symbol algebraic signature based on its own data share is generated and included in the challenge. Hence, the communication overhead involved in this broadcast message is $5 \cdot q$ bits. Upon receiving the challenge, each share holder needs to compute a 1-symbol algebraic signature and return it to the check initiator. Thus, for each share holder (including the initiator) the computational cost is just $AlgSigGen_k^1$. The communication overhead involved in every response message $sig_\alpha(S_i)$ is q bits. After obtaining all $sig_\alpha(S_i)$s $(i = 1, \ldots, n)$, each share holder can act as a verifier to check the integrity of $data$. It is clear that in this step the computational cost at each node is $ParGen_k^1 + AlgSigGen_k^1$.

VI. CONCLUSION

In this paper, we propose a secure and dependable data storage scheme with dynamic integrity assurance in wireless sensor networks. We utilize perfect secret sharing and erasure coding in the initial data storage process to guarantee data confidentiality and dependability. To ensure the integrity of data shares, an efficient dynamic data integrity checking scheme is constructed based on the principle of algebraic signatures. In contrast to the existing approaches, more desirable properties and advantages are achieved in our scheme. Furthermore, through detailed performance and security analysis, we show that our scheme is highly secure and efficient, thus can be implemented in the current generation of sensor networks.

ACKNOWLEDGMENT

This work was supported in part by the US National Science Foundation under grant CNS-0831963, CNS-0626601, CNS-0716306, CNS-0831628 and CNS-0716302.

REFERENCES

[1] A. Mitra, A. Banerjee, W. Najjar, D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos, "High-performance low power sensor platforms featuring gigabyte scale storage," in *Proc. MobiQuitous*, 2005.

[2] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, "Capsule: An energy-optimized object storage system for memory-constrained sensor devices," in *Proc. ACM Sensys*, 2006.

[3] "Mica motes," http://www.xbow.com/Products/.

[4] N. Bhatnagar and E. L. Miller, "Designing a secure reliable file system for sensor networks," in *Proc. 2007 ACM workshop on Storage security and survivability*, 2007.

[5] P. Desnoyers, D. Ganesan, and P. Shenoy, "Tsar: a two tier sensor storage architecture using interval skip graphs," in *Proc. ACM Sensys*, 2005.

[6] "Mica z," http://www.xbow.com/Products/.

[7] F. Ye, H. Luo, S. Lu, and L. Zhang, "Stastical en-route filtering of injected false data in sensor networks," in *Proc. IEEE INFOCOM*, 2004.

[8] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler, "Spins: Security protocols for sensor networks," *ACM Wireless Networks*, Sep. 2002.

[9] K. Ren, K. Zeng, and W. Lou, "A new approach for random key pre-distribution in large-scale wireless sensor networks," *Journal of Wireless Communication and Mobile Computing*, vol. 6, no. 3, pp. 307–318, 2006.

[10] K. Ren, W. Lou, and Y. Zhang, "Leds: Providing location-aware end-to-end data security in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 5, pp. 585–598, May 2008.

[11] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," in *Proc. 2003 ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[12] J. Girao, D. Westhoff, E. Mykletun, and T. Araki, "Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks," *Ad Hoc Networks, Elsevier*, 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. IEEE PerCom*, 2008.

[14] D. Ma and G. Tsudik, "Forward-secure sequential aggregate authentication," in *Proc. 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. 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. 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. 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. 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, April 1989.

[20] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," *Advances in Cryptology (CRYPTO'92), LNCS*, vol. 740, pp. 471–486, 1993.

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

[22] S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *SIAM Journal of Applied Mathematics*, vol. 8, pp. 300–304, 1960.

[23] W. Litwin and T. Schwarz, "Algebraic signature for scalable distributed data structure," in *Proc. ICDE*, 2004.

[24] T. Schwarz, "Verification of parity data in large scale storage systems," in *Proc. PDPTA*, 2004.

[25] T. Schwarz and E. Miller, "Store, forget and check: Using algebraic signature to check remotely administered storage," in *Proc. ICDCS*, 2006.

[26] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in *Proc. Advances in Cryptology (CRYPTO'96)*, 1996.