Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance

QIAN WANG and KUI REN, Illinois Institute of Technology
SHUCHENG YU, University of Arkansas at Little Rock
WENJING LOU, Worcester Polytechnic Institute

Recently, distributed data storage has gained increasing popularity for efficient and robust data management in wireless sensor networks (WSNs). The distributed architecture 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 challenge, in this article we propose a novel dependable and secure data storage scheme with dynamic integrity assurance. 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 techniques of algebraic signature and spot-checking. The proposed scheme enables individual sensors to verify in one protocol execution the correctness of all the pertaining data shares simultaneously in the absence of the original data. Extensive security analysis shows that the proposed scheme has strong resistance against various data pollution attacks. The efficiency of the scheme is demonstrated by experiments on sensor platforms Tmote Sky and iMote2.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection (e.g., firewalls)*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks, network communications*

General Terms: Algorithms, Reliability, Security

Additional Key Words and Phrases: Wireless sensor networks, data storage, integrity check

ACM Reference Format:
Wang, Q., Ren, K., Yu, S., and Lou, W. 2011. Dependable and secure sensor data storage with dynamic integrity assurance. ACM Trans. Sen. Netw. 8, 1, Article 9 (August 2011), 24 pages.
DOI = 10.1145/1993042.1993051 http://doi.acm.org/10.1145/1993042.1993051

A preliminary version of this article was presented at the 28th International Conference on Computer Communications.
This work was supported in part by the U.S. National Science Foundation, under grants CNS-0831963, CNS-0626601, CNS-0716306, CNS-0831628, CNS-0716302, and CNS-1117811.
Authors' addresses: Q. Wang and K. Ren, Department of Electrical and Computer Engineering, Illinois Institute of Technology, 3300 South Federal Street, Chicago, IL 60616; email: {qian, kren}@ece.iit.edu; S. Yu, Department of Computer Science, University of Arkansas at Little Rock, 2801 South University Ave., Little Rock, AR 722-4; email: sxyu@ualr.edu; W. Lou, Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609; email: wjlou@ece.wpi.edu.

DOI 10.1145/1993042.1993051 http://doi.acm.org/10.1145/1993042.1993051

1. INTRODUCTION

Distributed data storage and access has recently found increasing popularity due to many reasons. First, new-generation sensor nodes with significant performance enhancements are available. Such enhancements include energy-efficient storage, greater processing capabilities, and data management abilities. It is now possible to equip sensor devices with energy-efficient storage such as the new-generation flash memory with several gigabytes and low-power consumption [Mathur et al. 2006; Mitra et al. 2005]. Second, distributed data storage has more efficient energy consumption. In the MICAMotes platform,[1] flash memory has less energy efficiency, thereby reducing the energy benefits of local data storage. However, new-generation flash memory has significantly improved its energy efficiency and computation versus communication tradeoff as well. For example, transmitting data over a radio channel consumes 200 times more energy than storing the same amount of data locally on a sensor node [Bhatnagar and Miller 2007]; radio reception uses 500 times more energy than reading the same amount of data from local storage [Mathur et al. 2006]. A measurement study in Desnoyers et al. [2005] showed that equipping the MicaZ[1] platform with NAND flash memory allows storage to be two orders of magnitude cheaper than communication and comparable to computation in cost, which makes local storage and processing more desirable. Last but not least, distributed data storage achieves more robustness. Centralized storage can lead to the single point of failure, and easily attracts attacks. Moreover, it may also cause a 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 in wireless sensor networks (WSNs) 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 [Liu and Ning 2003; Perrig et al. 2002; Ren et al. 2006, 2008, 2009; Wang et al. 2010; Ye et al. 2004]. Some related work [Chessa et al. 2004; Girao et al. 2007; Ma and Tsudik 2007; Pietro et al. 2008; Subbiah and Blough 2005; Subramanian et al. 2007; Zhang et al. 2005] regarding secure distributed data storage can be found in the literature, but none of it satisfies the overall requirements of data confidentiality, dependability, integrity, and efficiency. Pietro et al. [2008] suggested moving the stored data constantly among sensors to increase dependability of one particular data item. Zhang et al. [2005] proposed a secure data access approach by using a polynomial-based key management scheme, where the mobile sinks can retrieve the network data following fixed routes. Subramanian et al. [2007] 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, none of these schemes considered the data dependability and integrity. Rabin [1989] proposed an information dispersal algorithm (IDA) for secure data storage and transmission in distributed systems, where the original information F is dispersed into n by using erasure codes. Chessa et al. [2004] extended the idea of information dispersal in Rabin [1989], and investigated the data storage problem in the context of a redundant residue number system (RRNS). However, the system has to maintain a large library of parameters together with a big set of moduli. [Subbiah and Blough 2005] 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 are required. Table I

[1] `http://www.xbow.com/Products/`

Table I. Comparison of Data Storage Schemes

Property \ Scheme	Ours	Rabin [1989]	Chessa et al. [2004]	Subbiah et al. [2005]	Subramanian [2007]
Confidentiality	√	√	√	√	√
Dependability	√	√	√	√	
Integrity Assurance	√				
Efficiency	√	√			√

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, in order to ensure the data integrity and availability over the entire data lifetime, any unauthorized data modifications or random data corruptions due to malicious attacks and Byzantine failures should be detected as soon as possible. However, this important and unique security issue has been largely overlooked in most existing designs in WSNs.

To address the problem, in this article we propose an efficient and flexible dynamic data integrity checking scheme to verify 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 techniques of erasure coding and secret sharing. 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 and a spot-checking approach, which allow the shareholders 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 shareholders 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 with high probability in one verification operation. Most importantly, the proposed scheme can verify the integrity of aggregated data shares with great efficiency. Through detailed security analysis and experiments on sensor platforms Tmote Sky and iMote2, we show that the proposed scheme is highly effective and efficient and can be well suited for the resource-constrained WSNs.

The rest of the article is organized as follows. Section 2 introduces the system model and attack model, and briefly describes some necessary background for the techniques used in this article. Sections 3 and 4 provide the detailed description of our proposed schemes. Sections 5 and 6 present the security analysis and performance analysis, respectively. Section 7 summarizes related work. Finally, Section 8 concludes the article. The proof of Proposition 5.2 is given in the Appendix.

2. NETWORK AND SECURITY ASSUMPTIONS

2.1. 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. 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 are already in place to provide basic communication security [Blundo et al. 1992]. However, individual sensors are not reliable since they can undergo random Byzantine failures and be compromised due to a lack of tamper-proof hardware.

2.2. Adversary Model

We consider a general and powerful adversary model regarding data storage 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 cannot 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. Furthermore, the adversary's movements are unpredictable and untraceable, and can compromise different sets of nodes over different time intervals.

Note that, if the adversary compromises a sensor node and resides there, it can always respond to 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.

2.3. 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.

- *Security*. To enhance data confidentiality and integrity by increasing the attacker's cost, that is, decreasing the its gain on compromising individual sensors.
- *Dependability*. To enhance data availability against both sensor Byzantine failures and sensor compromises, that is, minimizing the effect brought by individual sensor failures and compromises.
- *Dynamic Integrity Assurance*. To ensure that the distributed data shares are correctly stored over their lifetime, so they can finally be used to reconstruct the original data by authorized users.
- *Lightweight*. The scheme design should be lightweight as always in order to fit into the inherent resource-constrained nature of WSNs.

2.4. Notation and Preliminaries

Following is a list of notation used in presenting the idea of the proposed scheme.

— 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 Galois field $\mathcal{GF}(2^q)$ and β_is are distinct elements randomly picked from $\mathcal{GF}(2^q)$. Note that α^i ($i \geq 0$), which is generated from α, is also an element in $\mathcal{GF}(2^q)$. When used in the computation of algebraic signatures, α^i is usually called a symbol which has a length of $q$bits.
— $\overline{S}_i$, $\underline{S}_i$ ($i \in \{1, \ldots, n\}$). $\overline{S}_i$ denotes the shares of original data that generated by Reed Solomon (RS) coding and $\underline{S}_i$ denotes shares of the authorized key (K_{UV}) generated by a secret sharing scheme.
— $sig_{(\alpha,r)}(S)$ ($S \in \{S_1, \ldots, S_n\}$). An r-symbol signature vector $(sig_{\alpha^1}(S), \ldots, sig_{\alpha^r}(S))$ generated 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 RS coding. $\mathcal{P}_i$ denotes the operation of parity calculation based on β_i.

In addition, a sensor node is referred to 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 [Shamir 1979] based on polynomial interpolation, in which m of n shares of a secret are required to reconstruct the secret. Shamir's SS scheme works as follows: the secret C is in field $\mathbb{Z}_p$ (p is prime; $p > n$), and there are n shareholders. All mathematical operations are performed in the finite field $\mathbb{Z}_p$. To distribute C, we select a polynomial $a(x)$ with degree $m - 1$ and generate a share s_i for each shareholder i with $a(x)$: $s_i = C + a_1 i + a_2 i^2 + \ldots + a_{m-1} i^{m-1}$, where s_i is also in $\mathbb{Z}_p$. To reconstruct C, we retrieve m coordinate pairs (i, s_i) of all i in B (B is any subset of the shareholders with $|B| = m$;), and apply the Lagrange interpolation formula: $C = \sum_{i \in B} b_i s_i$, where $b_i = \prod_{j \in B, j \neq i} \frac{j}{j-i}$. Informally, secret sharing schemes which do not reveal any information about the shared secret to unauthorized individuals are called *perfect*. A perfect secret sharing scheme is called *ideal*, if the size of each share is the same as that of 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 [Reed and Solomon 1960] and Rabin's Information Dispersal algorithm [Rabin 1989]. A (k, n)-Reed Solomon code can correct up to $t = \lfloor \frac{n-k+1}{2} \rfloor$ errors, and works as follows, for $q \leq n$. Let $\alpha_0, \ldots, \alpha_{n-1}$ be n distinct elements of $\mathbb{Z}_p$. The RS code of message length k is defined as follows: associate a message $m = \langle m_0, \ldots, m_{k-1} \rangle$ with a polynomial $M(x) = \sum_{j=0}^{k-1} m_j x^j$. The encoding of m is the evaluation of $M(x)$ at the n given points, that is, $E(m) = \langle M(\alpha_0), \ldots, M(\alpha_{n-1}) \rangle$.

Algebraic signature. Algebraic signature (AS) was proposed in Litwin and Schwarz [2004] and Schwarz [2004]. 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 1- or 2-byte words as the elements of the Galois Field $\mathcal{GF}(2^q)$ (e.g., $q = 8, 16$). Assume $l < 2^q - 1$. Let $\overrightarrow{\alpha} = (\alpha^1, \ldots, \alpha^n)$ be a vector consisting of n different nonzero elements in $\mathcal{GF}$. The n-symbol signature $sig_{(n,\alpha)}$ of X based on α is the vector $sig_{(\alpha,n)}(X) = (sig_{\alpha^1}(X), sig_{\alpha^2}(X), \ldots, sig_{\alpha^n}(X))$, where α is called the n-symbol *signature base* and each sig_{α^i} is called a *component signature*. 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 $P_i = \mathcal{P}_i(D_1, \ldots, D_m)$, then $sig_\alpha(P_i) = sig_\alpha(\mathcal{P}_i(D_1, \ldots, D_m))$ $= \mathcal{P}_i(sig_\alpha(D_1), \ldots, sig_\alpha(D_m))$.

Properties I and II demonstrate the security of algebraic signature (the proofs of the properties can be found in Litwin and Schwarz [2004]). Property I gives the best possible behavior of $sig_{(\alpha,r)}$ for changes limited up to r symbols. If a data share is modified up to $k > r$ symbols, property II shows that $sig_{(\alpha,r)}$ still exhibits the low collision probability that is required in a signature scheme. In other words, for an arbitrary signature $sig_{(\alpha,r)}$ of $X = (x_1, \ldots, x_l)$ $(l < 2^q - 1)$, the probability that X is modified but $sig_{(\alpha,r)}$ remains unchanged is 2^{-rq}. When r and q are chosen appropriately, the probability of collision due to any data change is sufficiently low. Property III shows the characteristic homomorphism of algebraic signature when an algebraic signature is applied with erasure and error correcting codes that use only the XOR operation, that is, the signature of the code parity block is equivalent to the code parity of the signatures.

3. 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 WSNs, we discuss two schemes for initial data storage (a basic scheme followed by an enhanced scheme), which we believe will lead us to the final desirable solution.

3.1. The Basic Scheme

Suppose a sensor node v has *data* to be stored locally. To protect the 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 the data, where $h(\cdot)$ denotes a cryptographic hash function.
—*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 of our design here and will not be discussed in this article.

—*Step* 4. Store 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 for 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 undergo 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, when secret sharing is used to manage data, each data share has the same size with the original data block. If n data shares are generated and distributed, the storage and communication overhead are both increased n times. Thus using perfect secret sharing to manage generic data leads to an n-fold increase in storage and communication cost, and is no better than n-fold replication. For a sensor holding a large amount of data, these costs will be very high. This problem becomes worse when the number of nodes increases. Therefore, in terms of space and communication efficiency, the direct use of the secret sharing approach for data storage may not be suitable for resource-constrained wireless sensor networks.

3.2. 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, that is, $\{data, h(data, k_r)\}_{k_r}$ 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 $M(x) = D_1 + D_2x + \ldots + D_m x^{m-1}$, where $< \{data, h(data, k_r)\}_{k_r} >: = D_1 \| \ldots \| D_m$. v further obtains n $\overline{S}_j$s with each $\overline{S}_j = M(\alpha^j)$ $(1 \leq j \leq n)$, where $n \leq 2^q - 1$.

—*Step* 3. v then employs an (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 $\{v, seqno, \overline{S}_i, \underline{S}_i\}_{K_{vw_i}}$ to w_i, where K_{vw_i} is the pairwise key shared between v and w_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 the 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 fewer 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 $n * |\mathsf{DATA}|$.

4. DYNAMIC DATA INTEGRITY ASSURANCE

In mission-critical applications, the availability of the stored data must be guaranteed over the whole data lifetime. One of the key issue related to this is to detect any unauthorized data modification and/or data corruption. In reality, after data shares are distributed among the neighbor nodes, these shares may be modified illegally once the corresponding storage sensor is compromised or it undergoes Byzantine failures. If such illegal modifications and data corruptions 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.

4.1. The Basic Schemes

We start with some basic solutions aiming to provide integrity assurance of the sensor data.

Scheme-I. One straightforward approach of ensuring the integrity of data or data shares is to use "fingerprints", where the hash of the data shares are computed and stored as a hash vector $\overrightarrow{\mathbf{H}}$ at the data originating sensor υ before the share distribution. Later on, the data originating sensor υ can check the data integrity by requesting each share to be returned. Upon receiving all the shares from the shareholders, υ can calculate $H(S_1), \ldots, H(S_n)$ and check $\overrightarrow{\mathbf{H}} \overset{?}{=} \big(H(S_1), \ldots, H(S_n)\big)$, where $H(\cdot)$ denotes the hash operation (e.g., SHA-1, MD5). However, this approach has a number of serious problems: (i) it is single point of failure. If υ fails to function correctly, no one else can perform the integrity check. (ii) It is not secure, as υ will collect all the original 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 storage and communication overhead become high when the sensed data is aggregated. To verify *data*, n hash values need to be stored in the data-originating sensor. The communication overhead is also $n * |S_i|$ as n shares have to be returned for verification. The overhead problem becomes much worse in the case of data aggregation, that is, many more hash values need to be stored and many more shares need to be returned to verify multiple data blocks.

Scheme-II. A variant approach could be to ask the shareholders to return the fingerprints instead of the original share. However, the storage overhead is the same as for the above scheme. Moreover, this method suffers from a severe drawback: a compromised shareholder could simply precompute and store the fingerprint $H(S_i)$ instead of the genuine data. When required to return the fingerprint, it could respond to the verifier with $H(S_i)$, the use of which will always pass the integrity checks. This problem can be solved using keyed fingerprinting [Bellare et al. 1996], where a different secret key is employed to generate the fingerprint for each integrity check. The purpose of constructing keyed hash functions is to use cryptographic hash functions in conjunction with a key for building secure message authentication functions. One common approach is to key the function's *initial variable* (IV). The fixed and known IV, as defined by the original function, is replaced by a random and secret value known only to the parties [Bellare et al. 1996]. Using the keyed IV approach, any iterated hash construction (e.g., MD5, SHA-1) can be associated to a family of keyed functions. However, the use of keying hash functions for integrity checking would result in an additional overhead in both computation and storage, since the verifier (data-originating sensor) must precompute and store all the keyed fingerprints for data shares it plans to check. In addition, the number of verifications allowed to be performed in this solution is prelimited by the number of secret keys held by the verifiers. After these keys are exhausted, the aforementioned problem still exists.

Another technique to safeguard against modified data shares is verifiable secret sharing (VSS). Different from the classical secret sharing schemes, VSS includes auxiliary information that allows shareholders to verify their shares as consistent. We give a commonly used example of VSS scheme proposed by Feldman [1987], which is constructed based on Shamir's [1979] secret sharing scheme. Assume a cyclic group $\mathbb{G}$ of prime order p and a generator g of $\mathbb{G}$ are chosen publicly as system parameters. The scheme is executed as follows: (i) the data-originating sensor v generates shares $P(1), \ldots, P(n)$ and distributes each shareholder one value, where $P(x) = S + a_1 x + \ldots + a_t x^t$ is a random polynomial of degree t and $P(0) = S$ is the original data share; (ii) v computes and distributes commitments to the coefficients of P, that is, $c_0 = g^S, c_1 = g^{a_1}, \ldots, c_t = g^{a_t}$. Then any shareholder can verify the integrity of its own share: shareholder i that holds $u = P(i)$ checks if the following equation holds: $g^u = c_0 c_1^{i^1} \cdots c_t^{i^t} = \prod_{j=0}^{t} g^{a_j i^j} = g^{\sum_{j=0}^{t} a_j i^j} = g^{P(i)}$. It is obvious that VSS allows each shareholder to perform an integrity check for its own data share. However, it cannot tell the status of the other shares corresponding to the same original data block as each shareholder only has one share.

4.2. The Algebraic Signature-Based Dynamic Checking Scheme

To achieve dynamic data integrity assurance with a lightweight design, we propose to use algebraic signatures for our purpose [Litwin and Schwarz 2004; Schwarz and Miller 2006]. The idea is as follows: for each data share, the data originating sensor generates a distinct parity and appends the parity to the data share. During the integrity verification process, each shareholder can act as a verifier to validate the integrity of the stored data shares as long as they have the corresponding parities. The nice properties of the proposed scheme are as follows: (i) any unauthorized modifications can be almost detected in one integrity check; (ii) a fast integrity check can be realized when data shares are aggregated on shareholders; (iii) the shareholders 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 the dynamic data integrity checking scheme is presented in the following subsections.

4.2.1. 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 $S_i = \underline{S}_i || \overline{S}_i$ $(i \in \{1, \ldots, n\})$. 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 $\mathcal{GF}(2^q)$.

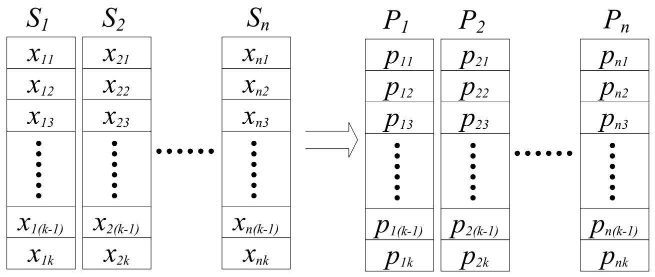
S_1
S_2
S_n
P_1
P_2
P_n
x_{11} x_{12} x_{13} $x_{1(k-1)}$ x_{1k}
x_{21} x_{22} x_{23} $x_{2(k-1)}$ x_{2k}
x_{n1} x_{n2} x_{n3} $x_{n(k-1)}$ x_{nk}
p_{11} p_{12} p_{13} $p_{1(k-1)}$ p_{1k}
p_{21} p_{22} p_{23} $p_{2(k-1)}$ p_{2k}
p_{n1} p_{n2} p_{n3} $p_{n(k-1)}$ p_{nk}

Fig. 1. Parity generation from data shares.

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

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

where $n \leq 2^q - 1$. Notice that β_j ($j = 1, \ldots, n$) are distinct elements randomly picked from $\mathcal{GF}(2^q)$. Figure 1 shows the parity generation from data shares.

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

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

where *seqno* is the data sequence number used for future data integrity check or retrieval. Then, *data*, S_is and P_is are erased. Note P_i is generated based on its corresponding β_i, and, after parity generation for the remaining data blocks, β_is are erased.

4.2.2. Dynamic Data Integrity Checking. At a later time, suppose a shareholder 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 shareholders,

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

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 shareholders that store $\{seqno, S_i\}$ ($i \in \{1, \ldots, n\}$) compute an r-symbol algebraic signature based on α,

$$sig_{(\alpha,r)}(S_i) = (sig_{\alpha^1}(S_i), sig_{\alpha^2}(S_i), \ldots, sig_{\alpha^r}(S_i)),$$

and respond to w_i with $\left\{sig_{(\alpha,r)}(S_i)\right\}_{K_{w_j,w_i}}$, where $\{j : j \in [1, n], j \neq i\}$.

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

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

Equation (2) holds because

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

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, that is, 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 5, the false-negative probability can be reduced to almost zero.

4.2.3. Fast Integrity Check for Multiple Data Blocks. Data shares belonging to different 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 computational overhead, for example, if the sensor node v has another data block *data′* to be stored locally. Similarly, v follows the *hybrid share generation scheme* to generate shares $S'_1, \ldots, S'_n$, and distributes them to the preselected n neighbors that stores the data shares of *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 shareholders (for simplicity, we assume $r = 1$ here). Upon receiving the message, each node acts as follows.

— Assume $S'_i = (x'_{i1}, x'_{i2}, \ldots, x'_{ik})$. Thus the concatenation of S_i and S'_i can be written as

$$\begin{aligned} S_i || S'_i &= (x_{i1}, x_{i2}, \ldots, x_{ik}, x'_{i1}, x'_{i2}, \ldots, x'_{ik}) \\ &= (x_{i1}, x_{i2}, \ldots, x_{ik}, x_{i(k+1)}, x_{i(k+2)}, \ldots, x_{i(2k)}). \end{aligned}$$

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 $sig_\alpha(S_i||S'_i)$s for $(i = 1, \ldots, n)$ are returned to the verifier. Note all signatures are encrypted using pairwise keys.

— Assume this verifier holds P_i and P'_i $(i \in \{1, \ldots, n\})$, where P_i and P'_i are parities of *data* and *data′* based on the same secret β_i, respectively. Upon receiving all responses from the other shareholders, 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 storage sensors, the proposed checking scheme can verify the integrity of different data blocks. This property can be easily generalized to more or any combination of data shares for arbitrary 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 communication overhead.

4.2.4. Spot Checking. The above approach provides deterministic data integrity assurance straightforwardly as the verification covers all the data shares. Thus the time consumption for the integrity check is proportional to the number of blocks required to be checked. To verify the data integrity, the verifier can also adopt a probabilistic spot-checking approach [Ateniese et al. 2007], that is, requesting the algebraic signature of a number of randomly selected symbols to be returned. The spot-checking approach can provide probabilistic integrity assurance of the data blocks. Specifically, in the challenge message, the verifier specifies the index of the symbols to be checked in the data shares. All the shareholders compute the signatures based on the specified symbols and return them to the verifier. On the other side, the verifier can easily compute the parity of the received signatures and compare the result to the signature of the parity, which is computed based on the the symbols of the same index in the parity. By sampling the data blocks randomly for integrity check, it suffices to challenge a small number of symbols in each data share to achieve detection with high probability (the performance analysis of the spot-checking approach is shown in Section 5.2.3). Note that, due to the use of spot checking, the number of possible signatures is much higher than the actual size of the data blocks. Thus, to correctly answer all possible challenges, a rational adversary by doing the right thing simply keeps the original data on the nodes rather than all possible signatures.

4.3. Multiparty Data Integrity Checking

Recall that in the integrity checking process, the signatures are encrypted using pairwise keys shared between the verifier and the other shareholders. If another shareholder wants to perform an independent integrity check, it has to ask other shareholders to resend the signatures again. In this section, we eliminate this restriction by employing the broadcast-based nature of wireless communication. In the modified scheme, all the shareholders (except the verifier itself) respond to verifier w_i with $\{sig_{(\alpha,r)}(S_i)\}$ in a broadcast manner, that is, $\{sig_{(\alpha,r)}(S_i)\}$ are sent openly, not encrypted. Now these signatures 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, that is, $\{w_i, seqno, \alpha, r, sig_{(\alpha,r)}(S_i)\}$. Therefore, after the challenge-response process, the other shareholders can obtain $sig_{(\alpha,r)}(S_i)$ $(i = 1, \dots, n)$ and act as a verifier in each check process.

This approach not only saves computational overhead (encryption and decryption) but also offers verification "diversity" since it enables efficient multiparty verification by all parties sharing their signature information with one another. In addition, the verification does not reveal much information about the original data shares since signatures only carry a few bytes of information. It would take thousands of randomly selected queries to retrieve sufficient signatures to be able to solve for the original data shares [Schwarz and Miller 2006]. Now we consider the worst case: if an adversary luckily obtains the signatures of S_i based on k different *signature bases*, it first constructs a system of linear equations. The adversary can break the data share since the number of equations is k and the number of unknowns is k, that is, $(x_{i1}, x_{i2}, \dots, x_{ik})$. To further enhance the security strength of the scheme, we develop a modification of the original protocol to allow each shareholder to safeguard the exposed signatures with a minimum of overhead. Different from the basic scheme, shareholders 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 will present more details of this technique in the following.

Recall that, in the 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* $(\gamma_1, \gamma_2, \gamma_3, \dots, \gamma_n)^T$, where all γ_i $(i = 1, \dots, n)$ are picked from $\mathcal{GF}(2^q)$ independently. By multiplying the matrix by this perturbation vector, we obtain another symbol vector

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

Here, $(\theta_1, \theta_2, \theta_3, \ldots, \theta_n)^T$ is called the *counteracting vector*. We have

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

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 shareholder i. When required to return a signature to the verifier, shareholder 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.

Discussion. 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. This is because the symbol x_{i1} is fully blinded by γ_i. Assume $q = 8$, which means the fist 8 bits of $\underline{S_i}$ are blinded. To reconstruct $\{k_r\}_{K_{UV}}$, the adversary needs to solve for m shares and guess $8m$ bits, which is infeasible. For example, if $m = 10$, The complexity to break $\{k_r\}_{K_{UV}}$ is thus $O(2^{80})$. Therefore, even if K_{UV} is known to the adversary, it is almost impossible to break the session key k_r.

4.4. Data Maintenance

In our data storage scheme, the original data block is first partitioned into n shares of equal size using Reed Solomon coding before distribution. Once any unauthorized data modification is detected, to repair the polluted shares, the basic idea of using mobile sinks [Zhang et al. 2005] can be applied. The network controller can periodically dispatch mobile sinks (MSs) to collect data shares, and perform error correction. In this article we only focus on the secure data storage scheme with dynamic data integrity check. The topic of data maintenance is out of the scope here.

5. SECURITY ANALYSIS

In this section, we present the security analysis of the proposed schemes. Our security analysis focuses on the adversary model defined in Section 2.2. Further discussion on other attacks is also presented.

5.1. Security and Dependability of Initial Data Storage

The security proof in Shamir [1979] and Reed and Solomon [1960] 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/compromised nodes reveal no information about the encoded data. In practice, requirements may vary depending on different applications, and thus the threshold m can be adjusted to accommodate the special needs.

5.2. Security of Dynamic Integrity Assurance

5.2.1. Random Modification of 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.

Table II. Probability of 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 first analyze the case that the adversary modifies data shares randomly 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, the parity of signatures can be considered as the signature of a *signature page* based on β, that is, $\sum_{i=1}^{n} \beta^{i-1} sig_\alpha(S_i)$. We have the following proposition.

PROPOSITION 5.1. *Assuming 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 [Litwin and Schwarz 2004]. □

Next, we study the probability of a false negative result where the data has been modified randomly but all the checking equations hold, that is, $sig_{(\alpha^t)}(P_i) = \sum_{j=1}^{n} \beta_i^{j-1} sig_{\alpha^t}(S_j)$, where $(t = 1, \ldots, r)$ and $(i = 1, \ldots, n)$. Suppose n_c nodes are misbehaving due to the possible compromise or Byzantine failure. In the following analysis, we do not limit the value of n_c, that is, $n_c \leq n$.

PROPOSITION 5.2. *Assume the number of shares $n < 2^q - 1$, $\mathcal{P}_i(sig_{(\alpha^t)}(S_1), \ldots, sig_{(\alpha^t)}(S_n)) = \sum_{j=1}^{n} \beta_i^{j-1} sig_{\alpha^t}(S_j)$, 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. See the Appendix. □

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, which 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 during the check.

Table II shows some numeric results of Pr. For example, assume the symbol length is $q = 8$ and $n_c = 20$. 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 computational and communication overhead, since all the shareholders 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, the increase of the number of r offers little gain in reducing the false-negative probability. In our dynamic checking scheme, we distribute to each shareholder a distinct parity P and its corresponding secret β, so all the share holders can receive $sig_{(\alpha,r)}(S_i)$s and act as independent verifiers. The results show that the false-negative probability is reduced significantly as p increases. This multiparty verification can tolerate large number of unavailable sensors (verifiers) and reduce the false-negative probability to an acceptable extent.

5.2.2. Targeted Modification of Data Shares. We have shown that, if an adversary modifies the data share randomly, the proposed data integrity checking scheme can successfully detect the attack in one verification. However, by observing the system for a long time, the adversary may find out the set of challenge number αs. Then it is targeted to modify data such that the polluted data share has the same signatures as the original ones.

First, we analyze how this attack can be carried out. Considering 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 $\mathcal{GF}(2^q)$ in a random way. For this reason, the adversary has to guess which α will be used in the future. As analyzed, using one signature ($r = 1$) is sufficient for each multiparty integrity check. Thus α does not have to be a primitive element now; it can be any element in $\mathcal{GF}(2^q)$. In practice, as long as q is appropriately selected, for example, $q = 8$ or 16, the probability of passing successive checks is very small, which makes the attack unfeasible.

5.2.3. Spot Checking. In this section, we analyze the performance of the spot-checking strategy on selected symbols instead of all in terms of the detection probability of data modification. Following the same assumptions as in Section 5.2.1, we further assume the adversary randomly modifies z symbols out of the total l symbols at data storage nodes. Let r be the number of different symbols for which the verifier asks for check in a challenge. Let X be a discrete random variable that is defined to be the number of symbols chosen by the verifier that matches the symbols modified by the adversary. We first analyze the matching probability that at least one of the

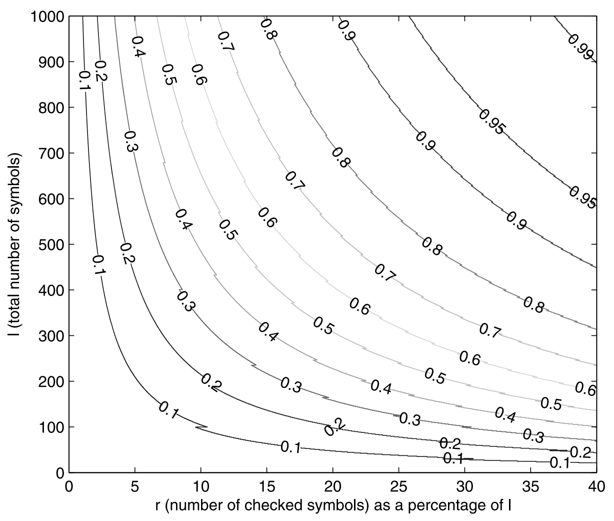

Fig. 2. The detection probability of data modification P_d $(z/l = 1\%)$.

symbols picked by the user matches one of the symbols modified by the adversary [Ateniese et al. 2007]:

$$\begin{aligned} P_m &= 1 - P\{X = 0\} \\ &= 1 - \prod_{i=0}^{r-1}(1 - \min\{\frac{z}{l-i}, 1\}) \\ &\geq 1 - (\frac{l-z}{l})^r. \end{aligned}$$

Obviously, if none of the specified r symbols are modified, the adversary avoids the detection in a data integrity check. Assume there are p independent verifiers; according to Proposition 5.2, the false negative probability is P_r. It follows that the probability of data modification detection is $P_d = P_m \cdot (1 - P_r)$.

Figures 2, 3, and 4 plot P_d for different values of l, r, z when we set $q = 8$, $n_c = 10$ and $p = 5$. The results show that, if more than a fraction of the data file is modified, then it suffices to challenge for a small number of symbols in order to achieve detection with high probability.

5.2.4. 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 shareholders 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 computational and communication overhead, we use only one signature base ($r = 1$) in a multiparty integrity check; thus n signatures from n verifiers are returned for each verification. 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.

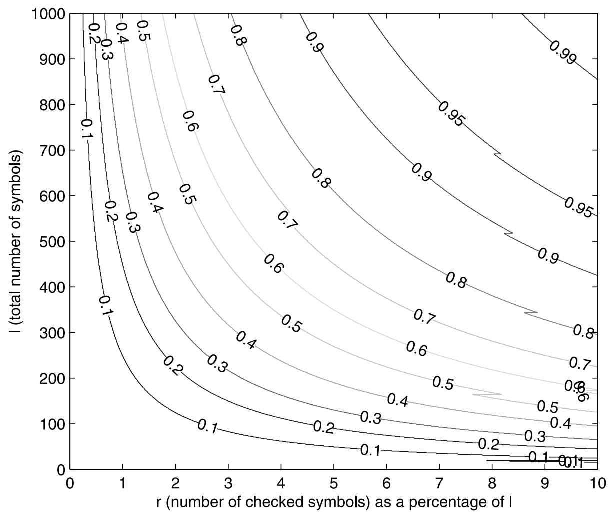

Fig. 3. The detection probability of data modification P_d (z/l = 5%).

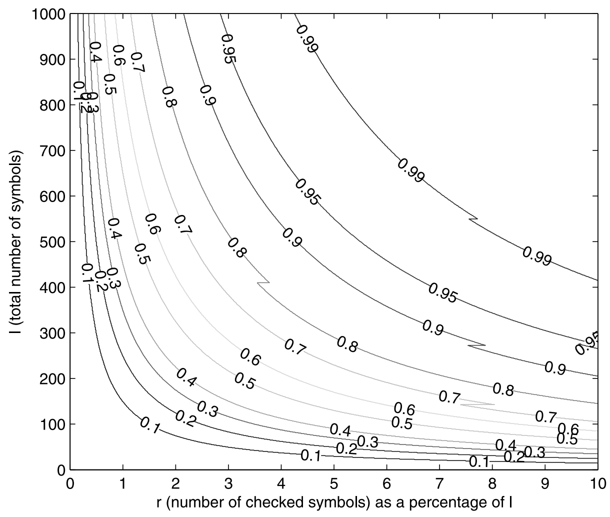

Fig. 4. The detection probability of data modification P_d (z/l = 10%).

6. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of our dynamic data integrity checking scheme in terms of storage, computational, and communication overheads. In our scheme, sensor data is generated and distributedly stored, and retrieved and decrypted by authorized users. Since sensor nodes are usually resource constrained, they may not be able to efficiently execute expensive operations, which may become the bottleneck of the scheme. Thus, our evaluation focuses on the efficiency of the scheme. In the following section, we first present the numeric results. Then, we present our implementation results on real sensor platforms. The notation of cryptographic operations is summarized in Table III.

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

6.1. Initial Data Storage

We first consider the computational overhead. Before the share generation process, the data-originating node v computes one keyed hash value $h(data, k_r)$ and performs two symmetric-key encryptions $\{data, h(data, k_r)\}_{k_r}$ and $\{k_r\}_{K_{UV}}$, respectively. The data share generation requires two sets of polynomial evaluations. v first encodes $\{data, h(data, k_r)\}_{k_r}$ into n fragments using an (m, n)-RS code, where m denotes the number of partitioned data blocks and n denotes the number of selected neighbors (i.e., shareholders). Assuming each partitioned data block D_i contains c symbols, there are totally $n \cdot c$ polynomial evaluations in this step. Then v employs an (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 shareholder. Let l denote the size of $data||k_r$; the total computation cost at the data-originating node is hence $Hash_l^1 + SymEncr^2 + PolyEval_m^{n\cdot(c+1)} + AlgSigGen_n^n + ParGen_k^n$. Correspondingly, the cost at each shareholder 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 $\mathcal{GF}(2^q)$ $(q = 8$ or $16)$. After share generation, the source node v distributes $\{v, seqno, S_i, \beta_i, P_i, \gamma_i, \theta_i\}_{K_{vw_i}}$ to neighbor i, where we assume S_i of *data* contains k symbols. There are a total n such messages. Thus, to store the data, the communication overhead during the distribution process is approximately $n \cdot (2k + 5) \cdot q$ bits. Obviously, it requires $(2k + 5) \cdot q$ bits of storage overhead to keep the distributed information at each shareholder.

6.2. Dynamic Data Integrity Check

Consider a shareholder w who initiates a data integrity check to verify the integrity of the data. It broadcasts a challenge message $\{w, seqno, \alpha, r\}$ to all the shareholders. 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 shareholder needs to compute a 1-symbol algebraic signature and return it to the check initiator. Thus, for each shareholder (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 $(i = 1, \dots, n)$, each shareholder can act as a verifier to check the integrity of the data. It is clear that in this step the computational cost at each node is $ParGen_k^1 + AlgSigGen_k^1$.

6.3. Implementation

In our implementation, we choose Tmote Sky and iMote2 as the target sensor platforms. We use SHA-1 as the one-way hash function and AES (supported by CC2420 Radio module of the motes) as the the data encryption algorithm. Our implementation shows that it takes about 0.06ms for SHA-1 to execute one hash operation and 0.4ms for AES to encrypt 64 bytes data. Since addition is done with XOR and fast in $\mathcal{GF}(2^q)$,

Table IV. Performance Comparison Under Different Sensor Platforms

	Tmote Sky (8MHz)		iMote2 (13MHz)	
Metric\Parameter	m = 10, n = 20	m = 20, n = 30	m = 10, n = 20	m = 20, n = 30
Coefficient matrix gen. time (ms)	4.7	15.7	0.47	1.4
RS coding time (ms)	135.0	271.0	8.90	17.8

the expensive operations in *PolyEval*, *ParGen*, and *AlgSigGen* are multiplications. In our experiment, we implement all multiplications by a power of α by successive lookups to a single multiplication-by-α table as in Schwarz and Miller [2006]. Note that (i) the operations in *ParGen* and *AlgSigGen* include the generation of coefficient matrix (e.g., matrix that consists of powers of β or α) and polynomial evaluations; (ii) the operations in RS coding are matrix multiplications, which consists of multiple polynomial evaluations. As RS coding involves the most number of multiplication operations, we focus on the performance of RS coding on the sensor platforms. Assume $q = 8$ and let $c = 100$ be the number of symbols per partitioned block. Table IV compares the computation time of coefficient matrix generation and RS coding on Tmote Sky running at 8MHz and iMote2 running at 13MHz. Our results show that these operations can be efficiently implemented, especially on high-end sensor platforms. When running at higher frequencies, for $m = 20$ and $n = 30$, iMote2 consumes 7.3ms at 104MHz, 3.6ms at 208MHz and 1.8ms at 416MHz to execute RS codings. However, as the number of data block increases (i.e., more sensed data needs to be encoded and stored), the RS coding may be expensive in terms of time cost and energy consumption for low-end sensor nodes.

7. RELATED WORK AND DISCUSSION

[Rabin 1989] proposed an information dispersal algorithm (IDA) for secure storage and transmission of data files in distributed systems, where the original information F is dispersed into n pieces or locations by using erasure codes. The salient point of this idea is that each piece is of size $|F|/m$; thus the file can be reconstructed from any m pieces. Apparently the IDA, as compared with Shamir [1979] algorithm for sharing secrets, can achieve data reliability and space efficiency in both storage and transmission. Due to these desirable properties, IDA has found its potential applications to secure and fault-tolerant storage and transmission of information. [Chessa et al. 2004] further extended this idea and investigated the storage problem in the context of a redundant residue number system (RRNS) for encoding information, which aims to provide a dependable and secure data storage to mobile wireless networks. The idea of using number theoretic constructs for information dispersal can be traced back to Asmuth and Blakley [1982], who proposed an algorithm based on the Chinese Remainder theorem. The main idea of RRNS is to partition the data file into records and encode each record separately as $(h+r)$-tuples of data residues using $h+r$ moduli. Then the residues are distributed among the mobiles in the network. To recover the original information, one is required to know at least h residues and of the correspondence moduli. Data dependability is ensured since data can be reconstructed in the presence of up to $s \leq r$ residue erasures, combined with up to $\lfloor \frac{r-s}{2} \rfloor$ corrupted residues. It has been shown in Chessa et al. [2004] that RRNS and IDA have comparable performance in terms of code efficiency and complexity; however, whether space efficiency can be obtained remains unclear. Another severe drawback of this approach is that, to reduce the computational overhead in the encoding and decoding process, the system has to maintain a large library of parameter values together with a big set of moduli.

Subbiah and Blough [2005] developed a novel combination of XOR secret sharing and replication mechanisms, which aimed to reduce the computation overheads introduced by perfect sharing schemes and achieve speeds comparable to standard encryption schemes. This scheme uses XOR secret sharing for confidentiality and manages each share using replication-based protocols for Byzantine and crash fault tolerance. However, while the computational overheads are reduced drastically, additional servers and storage capacities at each server are required. Moreover, by using perfect secret sharing to manage generic data leads to an n-fold increase in storage overhead, and is no better than n-fold replication.

Subramanian et al. [2007] studied the distributed data storage and retrieval problem in sensor networks and designed an adaptive polynomial-based (APB) data storage scheme for efficient data management. In the APB scheme, each storage1 sensor (with ID v) is given a share $f(v, y)$ of a t-degree polynomial $f(x, y)$ for generating data encryption keys in different phases, while an authorized user equipped with a share $f(x, i)$ and a target sensor ID can obtain the data decryption keys for phase i. Different from the basic polynomial-based scheme [Blundo et al. 1992], the APB scheme gives each node or user the perturbed share instead of the original share. The APB scheme can provide high scalability and flexibility, and hence is the most suitable for real applications. However, it cannot guarantee the data dependability and integrity, that is, if the data blocks are modified or polluted, it cannot tell whether the data has been altered or not. In earlier sections, we have demonstrated that our techniques have many desirable properties and advantages over these schemes. Table I shows that, compared to our scheme, none of these data storage schemes provides all the required properties.

Portions of the work presented in this article have previously appeared as an extended abstract [Wang et al. 2009]. This article revises that article a lot (in particular the scheme description part and the security analysis section) and adds more technical details. In addition, we discuss and analyze a new spot-checking approach when it is combined with our algebraic signature-based dynamic checking scheme. The spot-checking scheme prevents the adversary from precomputing and storing all possible signatures of the data shares. We also realize our design on the real sensor platforms. The implementation results are presented in our performance analysis section, including both a low-end sensor (e.g., Tmote Sky) and a high-end sensor (e.g., iMote2) running at different frequencies.

8. CONCLUSION

In this article, 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 and a spot-checking method. In contrast to the existing approaches, more desirable properties and advantages are achieved in our scheme. Furthermore, through detailed performance and security analysis and experiments on real sensor platforms, we show that the proposed scheme is highly secure and efficient, and thus can be implemented in the current generation of sensor networks.

APPENDIX: PROOF OF PROPOSITION 5.2

PROOF. We first consider the simplest scenario where $r = 1$ and $p = 1$, that is, where 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, that is, 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 $(sig_{(\alpha)}(S_1), \ldots, sig_{(\alpha)}(S_n))$ 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

$$\Pr_1 = \frac{C_{n_c}^{1}(2^{-q})^1 + \cdots + C_{n_c}^{n_c}(2^{-q})^{n_c}}{C_{n_c}^{1} + \cdots + C_{n_c}^{n_c}} = \frac{(1+2^{-q})^{n_c} - 1}{2^{n_c} - 1}.$$

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 5.1, 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

$$\Pr_2 = (1 - \Pr_1)2^{-q}.$$

The probability of a false-negative result with $r = 1$ and $p = 1$ is thus $\Pr = \Pr_1 + \Pr_2$.

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 the 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

$$\Pr_1 = \frac{(1+2^{-rq})^{n_c} - 1}{2^{n_c} - 1}.$$

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

$$\Pr_2 = (1 - \Pr_1)2^{-q}.$$

Furthermore, we can explore the scenario where $r = 1$ and $p > 1$. When $r = 1$, there is only one signature page; thus $\Pr_1 = \frac{(1+2^{-q})^{n_c}-1}{2^{n_c}-1}$. According to Proposition 5.1, if p different parities are used, the collision probability of a *page* is 2^{-pq}. Therefore, $\Pr_2 = (1 - \Pr_1)(2^{-q})^p$.

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

$$\Pr = \Pr_1 + \Pr_2,$$

where $\Pr_1 = \frac{(1+2^{-rq})^{n_c}-1}{2^{n_c}-1}$ and $\Pr_2 = (1 - \Pr_1)2^{-pq}$. □

ACKNOWLEDGMENTS

The authors would like to thank Hanfei Zhao for his contribution on the implementation. The authors would also like to thank the anonymous reviewers for their valuable suggestions that helped to improve the quality of this article.

REFERENCES

ASMUTH, C. AND BLAKLEY, G. 1982. Pooling, splitting, and restituting information to overcome total failure of some channels of communication. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society, Los Alamitos, CA, 156.

ATENIESE, G., BURNS, R., CURTMOLA, R., HERRING, J., KISSNER, L., PETERSON, Z., AND SONG, D. 2007. Provable data possession at untrusted stores. In *Proceedings of the ACM Conference on Computer and Communications Security*. ACM Press, New York, NY, 598–609.

BELLARE, M., CANETTI, R., AND KRAWCZYK, H. 1996. Keying hash functions for message authentication. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*. Springer-Verlag, Berlin, Germany, 1–15.

BHATNAGAR, N. AND MILLER, E. L. 2007. Designing a secure reliable file system for sensor networks. In *Proceedings of the ACM Workshop on Storage Security and Survivability*. ACM Press, New York, NY, 19–24.

BLUNDO, C., SANTIS, A. D., HERZBERG, A., KUTTEN, S., VACCARO, U., AND YUNG, M. 1992. Perfectly-secure key distribution for dynamic conferences. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*. Vol. 740. Springer-Verlag, Berlin, Germany, 471–486.

CHESSA, S., PIETRO, R. D., AND MAESTRINI, P. 2004. Dependable and secure data storage in wireless ad hoc networks: an assessment of ds2. In *Proceedings of the 1st IFIP TC6 Working Conference Wireless On-Demand Network Systems.* Springer, Berlin, Germany, 184–198.

DESNOYERS, P., GANESAN, D., AND SHENOY, P. 2005. Tsar: A two tier sensor storage architecture using interval skip graphs. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*. ACM Press, New York, NY, 39–50.

FELDMAN, P. 1987. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, Los Alamitos, CA, 427–438.

GIRAO, J., WESTHOFF, D., MYKLETUN, E., AND ARAKI, T. 2007. Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks. *Ad Hoc Netw. 5,* 7, 1073–1089.

LITWIN, W. AND SCHWARZ, T. 2004. Algebraic signature for scalable distributed data structure. In *Proceedings of the 20th International Conference on Data Engineering*. IEEE Computer Society Press, Los Alamitos, CA.

LIU, D. AND NING, P. 2003. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proceedings of the Network and Distributed System Security Symposium*. 263–276.

MA, D. AND TSUDIK, G. 2007. Forward-secure sequential aggregate authentication. In *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, Los Alamitos, CA, 86–91.

MATHUR, G., DESNOYERS, P., GANESAN, D., AND SHENOY, P. 2006. Capsule: An energy-optimized object storage system for memory-constrained sensor devices. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. ACM Press, New York, NY, 195–208.

MITRA, A., BANERJEE, A., NAJJAR, W., ZEINALIPOUR-YAZTI, D., KALOGERAKI, V., AND GUNOPULOS, D. 2005. High-performance low power sensor platforms featuring gigabyte scale storage. In *Proceedings of the 3rd International Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks*. 1–10.

PERRIG, A., SZEWCZYK, R., TYGAR, J., WEN, V., AND CULLER, D. 2002. SPINS: Security protocols for sensor networks. *ACM Wirel. Netw. 8,* 5, 521–234.

PIETRO, R. D., MANCINI, L. V., SORIENTE, C., SPOGNARDI, A., AND TSUDIK, G. 2008. Catch me (if you can): Data survival in unattended sensor networks. In *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications*. IEEE Computer Society Press, Los Alamitos, CA, 185–194.

RABIN, M. O. 1989. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM 36,* 2, 335–348.

REED, S. AND SOLOMON, G. 1960. Polynomial codes over certain finite fields. *SIAM J. Appl. Math. 8,* 2, 300–304.

REN, K., ZENG, K., AND LOU, W. 2006. A new approach for random key pre-distribution in large-scale wireless sensor networks. *J. Wirel. Comm. Mobile Comp. 6,* 3, 307–318.

REN, K., LOU, W., AND ZHANG, Y. 2008. Leds: Providing location-aware end-to-end data security in wireless sensor networks. *IEEE Trans. Mobile Comp. 7,* 5, 585–598.

REN, K., LOU, W., YU, S., AND ZHANG, Y. 2009. Multi-user broadcast authentication in wireless sensor networks. *IEEE Trans. Vehic. Tech. 58,* 8, 4554–4564.

SCHWARZ, T. J. E. 2004. Verification of parity data in large scale storage systems. In *Proceedings of the Conference on Parallel and Distributed Processing Techniques and Applications*. 508–514.

SCHWARZ, T. J. E. AND MILLER, E. L. 2006. Store, forget and check: Using algebraic signature to check remotely administered storage. In *Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*. 12.

SHAMIR, A. 1979. How to share a secret. *Comm. ACM 22,* 11, 612–613.

SUBBIAH, A. AND BLOUGH, D. M. 2005. An approach for fault tolerant and secure data storage in collaborative work environments. In *Proceedings of the ACM Workshop on Storage Security and Survivability*. ACM Press, New York, NY, 84–93.

SUBRAMANIAN, N., YANG, C., AND ZHANG, W. 2007. Securing distributed data storage and retrieval in sensor networks. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications*. 191–200.

WANG, C., CHEN, J., AND SUN, Y. 2010. Sensor network localization using kernel spectral regression. *J. Wirel. Comm. Mobile Comp. 10,* 8, 1045–1054.

WANG, Q., REN, K., LOU, W., AND ZHANG, Y. 2009. Dependable and secure sensor data storage with dynamic integrity assurance. In *Proceedings of the IEEE International Conference on Computer Communications*. 954–962.

YE, F., LUO, H., LU, S., AND ZHANG, L. 2004. Stastical en-route filtering of injected false data in sensor networks. In *Proceedings of the IEEE International Conference on Computer Communications*. 839–850.

ZHANG, W., SONG, H., ZHU, S., AND CAO, G. 2005. Least privilege and privilege deprivation: Towards tolerating mobile sink compromises in wireless sensor networks. In *Proceedings of the 6th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing*. ACM Press, New York, NY, 378–389.

Received March 2010; revised September 2010; accepted November 2010